TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ACHE	Adaptive Control of Home Environments
AST	Adaptation Strategy Table
BMS	Building Management System
CARA	Context Aware Real-time Assistant
CASAS	Centre for Advantace Stadies in Adaptative Systems
CBR	Case Based Reasoning
CC/PP	Composite Capabilities/ Preference Profile
CLM	CASAS Lightweight Middleware
CMF	Context Management Framework
CML	Context Modelling Language
CML	Context Modeling Language
COBRA	Common Object Request Broker Architecture
$context_{new}$	Nouveau contexte
$context_{old}$	Ancien contexte
CSCP	Comprehensive Structured Context Profiles
Dis	Function de distance
DL	Description Logics
DSOA	Device Service Oriented Architecture
EIB	European Installation Bus

ESB	Entreprise Service Bus
GTSH	Gator Tech Smart House
ID	Identificateur Unique
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
KNN	K-Nearest Neighbors
KNX	Konnex
KP	Knowledge Processors
LCS	Least Common Subsumer
log	Function Logarithmique
Matrix-Q	Matrice Q-learning
MavHome	Managing an Adaptive Versatile Home
MDP	Markov Decision Processes
MVC	Model View Controller
MVCA	Model View Controller Adapter
OMG	Object Management Group
ORM	Object-Role Modelling
OSGI	Open Services Gateway initiative
OWL	Web Ontology Language
PACE	Pervasive, Autonomic, Context-aware Environments

QL-CBR	Q-Learning- Case Based Reasoning
r	Récompense
RDF	Resource Description Framework
RDFS	RDF Schema
Result-Q	Résultat Q-learning
RL	Reinforcement Learning
S2CAS	Smart Space Context Aware System
SI	Système d'Information
SIAC	Système Intelligent Adapté au Contexte
SIB	Semantic Information Broker
sim	Similarity function
SOA	Service Oriented Architecture
SOCAM	Service Oriented Context- Aware Middleware
SOUPA	Standard Ontology for Ubiquitous and Pervasive Applications
SSAP	Smart Space Acces Protocol
SSL	Secure Sockets Layer
STI	Systèmes (ou services) de Transport Intelligents
SVM	Support Vector Machines
TD	Temporal Differential

UML	Unified Modeling Language
VAR	Variance
w	Weight of a contextual variable
W3C	World Wide Web Consortium
XCS	eXtended Classifier System
XML	EXtensible Markup Language
XMPP	eXtensible Messaging and Presence Protocol

INTRODUCTION

0.1 Contexte de la recherche

Ces dernières décennies ont été marquées par un progrès hyper croissant dans les technologies de l'informatique et des réseaux sans fils. Bien entendu, les réseaux deviennent de plus en plus hétérogènes, mobiles et dynamique. Les dispositifs deviennent de plus en plus miniatures, volatiles, et intelligents (Smartphones, Tablettes etc.). Les technologies mobiles (4G, Bluetooth, géolocalisation) ont bouleversé le style et la manière de la vie humaine.

Ces avancées remarquables qui sont intégrées dans notre vie quotidienne de façon invisible, ont contribué à améliorer et à progresser le niveau de vie humaine. Assurément, ils ont donné le pouvoir d'interagir avec le monde d'une manière naturelle et facile, fournissant ainsi des bienfaits pour la société dans son ensemble.

De plus, cette panoplie d'équipements, de technologies et de réseaux a donné naissance à un nouvel environnement connu par l'environnement ubiquitaire. Ce domaine, combinant à la fois l'informatique distribuée et l'informatique mobile, permet d'accomplir de façon intelligente des tâches quotidiennes dans différents domaines et servir également les utilisateurs dans leurs propres activités.

Dans cette situation, le système se voit ajouter et manipuler des informations d'analyse et de perception sur l'environnement et de ses utilisateurs, en tenant compte de leurs préférences ainsi que de leurs activités. Ceci implique que la réaction et la réponse du système dépend fortement du contexte environnemental. Évidemment, le contexte se voit, d'ailleurs, comme la source de données la plus importante d'un système intelligent sur lequel se base l'ensemble des traitements. D'un autre sens, il se voit comme un ensemble des informations nécessaires à la description d'une situation déclenchant par la suite une adaptation de service.

En revanche, une exploitation efficace pour le contexte joue un rôle primordial pour mettre en place un système intelligent adaptable au changement.

Un tel environnement se trouve aux conflits de plusieurs domaines tels que l'intelligente ambiante, les capteurs/les actionneurs, l'intelligence artificielle, la sensibilité au contexte et l'adaptation des services (Mozer and all., 2004). Principalement, il se caractérise par la faculté de raisonner selon la situation présente pour produire un service ou une action selon la demande. De ce fait, un système intelligent doit se doter de mécanismes de raisonnement et d'adaptation permettant à la fois de raisonner et d'ajuster le service selon la situation actuelle.

0.1.1 Intelligence ambiante

Le terme intelligence ambiante a été introduit pour la première fois en 1998 par la société Philips dans le cadre du projet «Vision of the Future». Dans ce projet, Philips mène une réflexion et une analyse prospective en interne sur l'évolution de l'électronique grand public.

Dans la littérature, il existe plusieurs définitions pour le terme intelligence ambiante. Selon l'ISTAG (Information Societies Technology Advisory Group) (Ducatel et al., 2001), l'intelligence ambiante consiste à créer des environnements capables de prendre en compte les caractéristiques de chaque usager, de s'adapter et de répondre intelligemment à ses besoins spécifiques, d'agir de manière non intrusive et le plus souvent invisible, de permettre à l'usager d'accéder aux services de la façon la plus naturelle et intuitive. Selon Reignier (Reignier et al., 2007), l'intelligence ambiante est un paradigme résultant de l'intersection de l'informatique ubiquitaire et de l'intelligence artificielle. L'intelligence ambiante trouve donc son origine dans la vision de l'informatique ubiquitaire dans laquelle, les ordinateurs imprègnent l'environnement quotidien, tout en étant, transparents à l'utilisateur, mais en y ajoutant la notion d'intelligence. C'est à dire, ayant la faculté d'agir et de s'adapter dynamiquement en fonction du contexte. À partir de ces définitions, on peut conclure que le paradigme de l'intelligence ambiante est une composition de trois mondes : monde physique,

monde numérique et un monde modélisé. L'ensemble a pour objectif de créer un environnement intelligent permettant d'améliorer la qualité de vie de ses utilisateurs. Ainsi le fonctionnement d'un environnement ambiant doit reposer sur un système capable d'assurer un contrôle total sur l'espace intelligent en étant capable de (Figure 0.1):

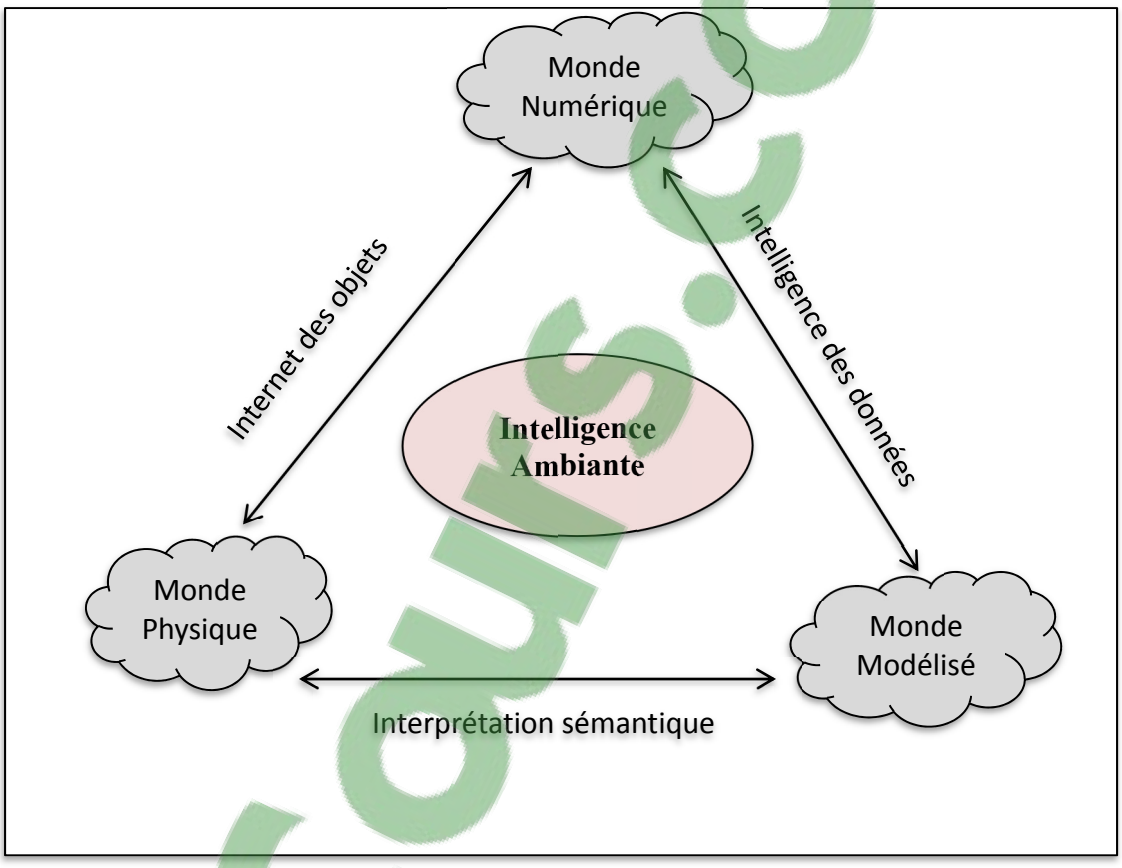


Figure 0.1 Paradigme d'intelligence ambiante

- percevoir les évènements générés par les différentes entités du monde physique (personne, un équipement ou une application). Ces dernières sont interconnectées grâce à différentes technologies (internet des objets) et communiquent via un monde numérique ;
- collecter et interpréter les informations à partir de sources hétérogènes pour déduire l'intention humaine du monde modélisé ;
- réagir en conséquence pour fournir les services souhaités de manière appropriée et transparente.

0.1.2 Sensibilité au contexte

Un système intelligent doit percevoir le contexte de l'environnement pour interagir adéquatement et naturellement avec l'utilisateur. Un système est dit sensible au contexte s'il est capable de capter en permanence le contexte courant et d'en tenir compte pour mieux répondre aux besoins et aux attentes des utilisateurs (Figure 0.2). Tout d'abord, le contexte est capturé par des équipements spécialisés (caméras, microphones, capteurs biométriques). Les données du contexte sont en général perçues dans leurs formes brutes. Ce qui nécessite des mécanismes de description sémantique du contexte afin de le rendre compréhensible et intelligible par le système. Ces mécanismes se basent essentiellement sur des méta-données décrites dans des langages sémantiques avec un haut niveau d'abstraction des connaissances permettant ainsi de décortiquer le vrai sens du contexte.

La sensibilité au contexte et une caractéristique délicate et indispensable pour un système intelligent. Effectivement, elle reflète la capacité de réagir proprement suite aux modifications du contexte et de changer le comportement de façon dynamique pour aboutir à des services conformes aux attentes des utilisateurs.

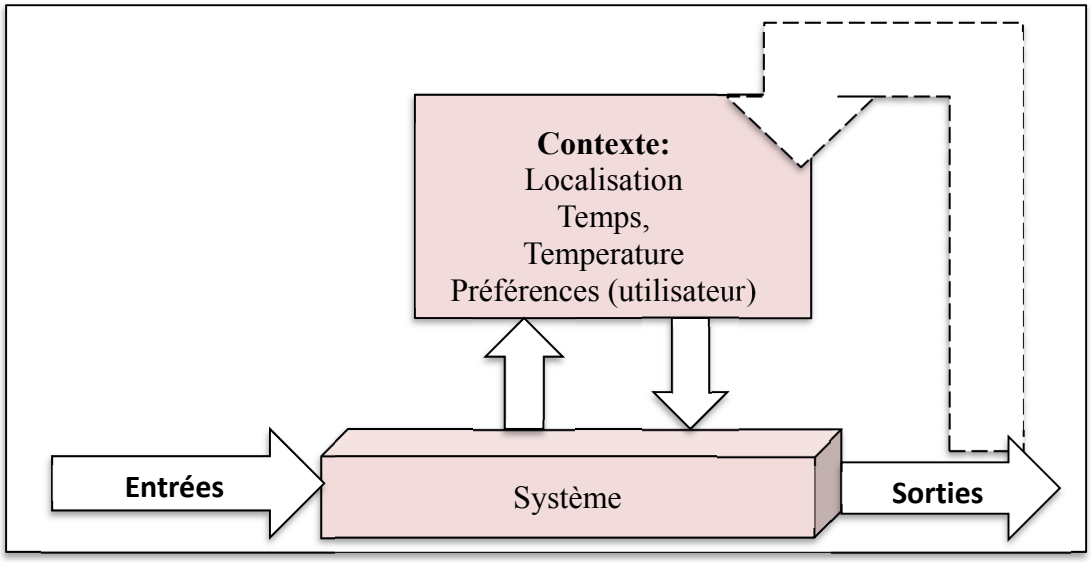


Figure 0.2 Système tenant compte du contexte

0.1.3 Auto-adaptation

Au sein de ce paysage, des mécanismes d'auto-adaptation sont alors nécessaires afin d'assurer une continuité de service et répondre aux attentes des utilisateurs en fonction de leurs contextes. L'intégration des mécanismes d'auto-adaptation dans un système intelligent constitue la clé pour gérer les changements de contexte et permettent d'augmenter le degré d'autonomie, de robustesse et de flexibilité du système. L'autonomie reflète la capacité du système à s'adapter seul aux changements sans aucune intervention humaine, alors que la robustesse reflète la capacité du système à assurer la continuité du service en cas de pannes et à s'adapter constamment au contexte d'utilisation. La flexibilité, quant à elle, reflète la souplesse du système dans la façon d'opérer l'adaptation qui doit rester complètement transparente pour l'utilisateur. Essentiellement, l'auto-adaptation se base sur la prise en compte des habitudes des utilisateurs ainsi que les changements de l'environnement. Mettre en oeuvre un tel mécanisme se base sur deux processus : auto-apprentissage et la prise de décision. L'auto-apprentissage consiste à identifier le processus des changements contextuels associés aux services exécutés. Il se base aussi sur l'historique contextuel afin de sélectionner la solution optimale (service désiré). Ce processus permet dans le fond, d'anticiper les changements afin de mieux s'adapter aux nouvelles situations. La prise de décision, quant à elle, permet de déterminer le service le plus approprié parmi les services candidats qui fournissent des fonctionnalités similaires ou proches. Le but de la sélection est de répondre constamment aux besoins des utilisateurs et d'améliorer aussi la qualité du service fournie.

0.2 Problématique

Les nouvelles technologies liées à l'intelligence ambiante apportent de nouvelles opportunités à l'utilisateur. Les systèmes intelligents adaptés au contexte (SIAC) se caractérisent non seulement par l'hétérogénéité des dispositifs, mais également par la mobilité et l'interaction continue des utilisateurs.

À notre sens, quatre propriétés majeures caractérisent les SIAC: distribution, dynamisme, hétérogénéité et auto-adaptation. Plus précisément :

- **Distribution** les applications d'un système intelligent sont fréquemment distribuées. Souvent cette distribution est très large puisqu'elle peut aller de capteurs embarqués jusqu'aux dispositifs des utilisateurs. Par ailleurs, les architectures proposées sont souvent complexes et ne suivent pas forcément la topologie client-serveur classique. Ce qui rend l'évolution et la gestion des systèmes plus délicates ;
- **Dynamisme** ce nouveau défi a un gros impact sur la conception des SIAC. En effet, un système intelligent n'est plus conçu comme un seul dispositif fixe, mais, comme un ensemble de dispositifs, pouvant apparaitre, disparaitre ou évoluer au cours du temps. La complexité engendrée par le dynamisme est grande et s'applique à plusieurs niveaux comme la conception, le développement et l'exécution ;
- **Hétérogénéité** un autre défi est de prendre en compte l'importance de la grande diversité des technologies et des services mis en oeuvre au sein d'un espace intelligent. Un développeur doit alors comprendre et savoir utiliser un grand nombre de technologies différentes, notamment, pour traiter la distribution. Les systèmes SIAC doivent affronter ce problème et savoir gérer aussi l'interaction entre l'utilisateur et le monde physique. Évidemment, cette gestion devra être qualifiée par sa transparence vis à vis de l'utilisateur ;
- **Auto-adaptabilité** c'est une exigence importante dans la conception des SIAC permettant ainsi d'adapter les services exécutés sans aucune interaction entre l'utilisateur et le système. Un système est dit auto-adaptable lorsqu'il peut anticiper l'action à exécuter. Cela veut dire, avoir la capacité de répondre aux besoins des utilisateurs sans procurer des changements environnementaux. Mettre en œuvre une telle adaptation nécessite une étude sur les stratégies d'adaptation ainsi que les mécanismes utilisés, en particulier, les machines d'apprentissage, les techniques de raisonnements et de prise de décisions.

On peut rajouter deux propriétés également très structurantes : l'interopérabilité, et la scalabilité. Il n'est illusoire de stocker l'interopérabilité et/ou l'hétérogénéité des systèmes en

terme temporel. Certainement, les systèmes ainsi que les dispositifs deviennent de plus en plus sophistiqués et complexes du côté technologique. Face à ce progrès, les systèmes SIAC doivent être conçus de façon évolutif, capable d'accroitre leurs capacités de traitement et de continuer la production des services en cas de forte demande de nouveaux dispositifs.

Les différentes problématiques discutées ci-dessus sont étudiées depuis de nombreuses années. Cependant, les récents changements dans le monde informatique réduisent l'efficacité des solutions actuelles, les rendant même obsolètes dans certains cas. De nombreuses architectures ont récemment vu le jour afin de faire face aux nouvelles difficultés liées aux développements des systèmes intelligents. Toutefois, la mise en place d'un tel système adapté aux changements environnementaux de manière efficace reste à nos jours un problème crucial.

D'un autre côté, un environnement intelligent fait face à un dynamisme d'entités, en particulier, à une variabilité dans les comportements, les préférences ou mêmes aux déplacements liés aux utilisateurs. Par conséquent, l'offre des services risque d'augmenter en fonction des changements d'où s'impose la nécessité d'exploiter un processus d'adaptation aux conditions opérationnelles environnementales. La notion d'adaptation est ainsi un caractère recherché et souhaitée dans un système intelligent. L'adaptation peut se définir comme étant un processus intelligent, permettant d'analyser le contexte reçu et d'agir en conséquence. Une des problématiques fondamentales des environnements intelligents est de savoir comment garantir une autonomie dans l'adaptation et/ou la prise de décisions sur les services. D'une autre manière, quel processus adopter dans un SIAC permettant d'aboutir à des décisions fiables sur les services en se basant sur des observations contextuelles?

Toutefois, afin d'être capable de traiter la problématique d'adaptation, il est nécessaire avant de bien éclaircir la notion du contexte, puisqu'elle représente l'information cruciale auxquelles se base plusieurs mécanismes. Le contexte est un terme assez défini, vu son large spectre de compréhension et d'utilisation. Généralement, les systèmes intelligents se réfèrent à toutes données collectées, soit physiquement (par équipements), soit logiquement (par

utilisateurs), et seront considérées par la suite, comme une entrée d'information principale au système. La formalisation et la catégorisation d'un contexte varie en fonction de l'intérêt porté par les systèmes intelligents. Bien évidemment, dans notre cas, nous optons pour une formalisation contextuelle reliée au caractère dynamique des utilisateurs, et orientée pour réaliser l'adaptation des services.

Partant de ces faits, les SIAC se doivent d'être sensible au contexte, muni des modules permettant non seulement de détecter tous changements liés aux utilisateurs mais aussi de s'adapter et d'ajuster les services d'une manière efficace et flexible pour la satisfaction des utilisateurs. En dépit des efforts de développements significatifs reliés à la conception des environnements intelligents. Il reste toujours, jusqu'à date, des problèmes à se poser liés, soit à la façon dont le contexte est identifié et formalisé pour le système, soit aux mécanismes d'adaptation adoptés et qui souffrent toujours de manque de fiabilité et de souplesse. En effet, à la différence des systèmes intelligents traditionnels, le nouveau SIAC, doit désormais être flexible, dynamique, capable de s'évoluer facilement, pour mieux répondre aux besoins des utilisateurs. Il doit être également en mesure de produire des services capables de s'adapter parfaitement à la hauteur des attentes des utilisateurs.

Les algorithmes d'apprentissages existants dans la littérature spécifique à une auto-adaptation sont nombreux (De lemos et al., 2013; Degeler et al., 2013; Fleury et al., 2010; Gu et al., 2010; Hsu et al., 2007; Miraoui et al., 2014). Toutefois, la plupart de ces algorithmes souffrent encore des problèmes liés 1) soit aux caractère dynamique de l'environnement qui entraine une évolution ou modification au niveau des variables présentes en entrée ou en sortie du système, 2) Soit à la production des services inappropriés aux utilisateurs, 3) Soit au temps de convergence de l'algorithme ce qui entraine un apprentissage assez long.

Ainsi, nous pouvons résumer notre problématique à un problème de conception d'un SIAC flexible, sensible au contexte, muni de mécanismes d'adaptation efficace, capable de s'ajuster au caractère dynamique de l'environnement et de ses utilisateurs pour assurer une exécution transparente, continue et efficace des services.

0.3 Objectifs de la recherche

Dans cette recherche, nous nous intéressons aux problèmes d'adaptation de services, vis-à-vis, le contexte au sein d'un espace intelligent. Notre objectif en premier lieu, sera donc, de fournir un cadre conceptuel pour le développement d'un système intelligent adaptable au contexte. Un des principes fondamentaux de notre travail est de séparer clairement les différents traitements du système intelligent pour aboutir à une meilleure performance.

Tout d'abord, avant de commencer la conception architecturale, nous visons à identifier les principaux indicateurs de qualité pour un SIAC. Par la suite, une architecture générale d'un SIAC sera proposée sous forme de couches. Chaque couche est constituée de modules nécessaires pour un traitement dédié fournissant ainsi une tâche à la couche suivante. L'ensemble des couches fonctionne de façon cohérente pour assurer la production des services adaptés aux besoins.

Par la suite, nous nous intéressant à concevoir un modèle logiciel, favorisant la séparation totale entre les différentes préoccupations du système. Il en résulte une architecture modulaire, homogène, extensible, permettant de prendre en compte la dynamicité et l'évolutivité de l'environnement. Nous nous concentrons également à spécifier pour chaque module l'ensemble des mécanismes nécessaires pour la perception, le raisonnement et l'action. L'objectif de cette architecture consiste 1) à faciliter l'intégration d'un nouveau module dans un environnement ambiant. 2) de pouvoir accéder à un module spécifique, sans devoir passer par d'autres. 3) de faciliter la modification à chaud de nombreux processus internes.

Notre dernier objectif consiste à améliorer un algorithme d'apprentissage pour le rendre plus adapté aux contraintes spécifiques des systèmes intelligents. Pour ce faire, nous proposons

d’hybrider deux algorithmes d’apprentissage. Le premier est un apprentissage par renforcement réalisé par l’algorithme Q-learning. Ce genre d’apprentissage auto-organisant nous permet, dans une certaine mesure, d’ajuster les services avant d’être exécutés dans l’environnement dépendamment des attentes des utilisateurs. Le deuxième fait appel à la conception d’un CBR ayant pour but d’améliorer la phase d’auto-adaptation en offrant ainsi des services adéquats, selon la demande des utilisateurs.

0.4 Méthodologie

Dans cette thèse, nous nous intéressons à la problématique d’adaptation des services dans le cadre d'un système intelligent sensible au contexte. Nous abordons le sujet selon deux facettes différentes. La première est au niveau architectural, elle touche beaucoup plus l'organisation logique des différents modules du système et les relations entre eux, et cherche à assurer la maintenabilité et l’efficacité du système. Notant que une architecture maintenable est une architecture simple, compréhensible, facile à corriger. La deuxième est au niveau de la mise en oeuvre et de l’exécution du processus de raisonnement pour assurer l’adaptation des services, Elle touche plus la façon dont le système doit procéder pour adapter un service selon les attentes de l’utilisateur.

Le point de départ de cette recherche est d’identifier les principales métriques de qualité pour un système intelligent. Au cours de notre recherche, on a trouvé que la conception d’une architecture efficace exige une architecture modulaire, scalable et flexible au changement. La fonctionnalité est aussi une autre exigence importante pour la conception d’un SIAC. Nous avons déduit que la fonctionnalité d’un SIAC signifie, non seulement, que le système soit sensible au contexte mais aussi auto-adaptable aux changements. Pour mener à bien notre projet de doctorat, nous avons adopté une méthode de type de recherche appliquée, dont les grandes étapes sont montrées dans la figure 0.3 :

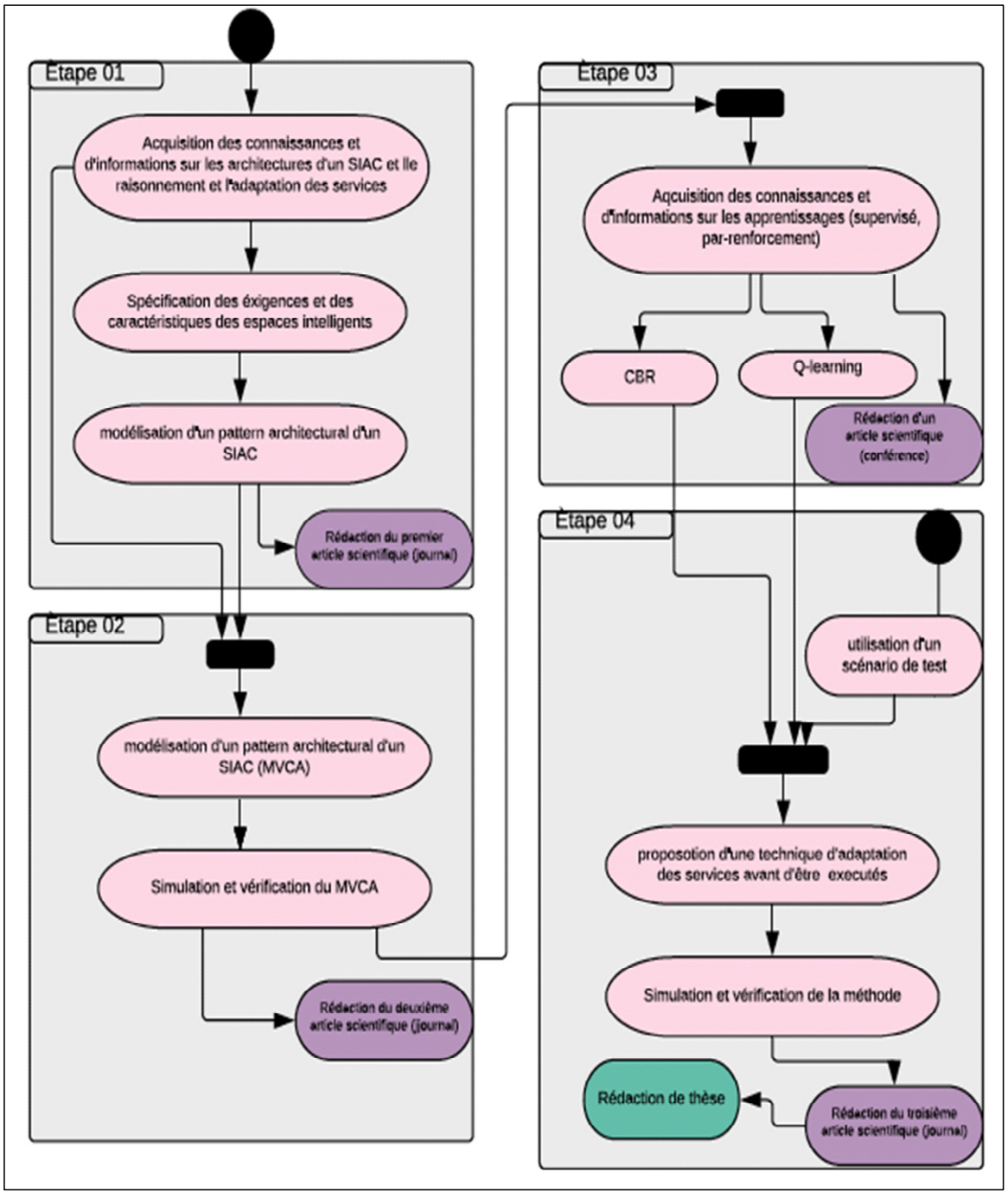


Figure 0.3 Méthodologie et structure de travail

0.4.1 Définition d'une architecture de qualité pour un SIAC

Mesurer la qualité d'un logiciel consiste alors à déterminer son adéquation par rapport aux objectifs de départ. Dans la première étape, nous nous intéressons alors à définir précisément quels sont les principaux indicateurs à utiliser pour avoir une architecture efficace d'un SIAC capable de s'adapter avec l'environnement et de produire des services adéquats au utilisateurs.

Par la suite, à partir de la base des métriques élaborées, nous avons procédé à la modélisation d'une architecture générique pour un SIAC proposant ainsi un modèle complet capable d'acquérir des données contextuelles et d'assurer des nouveaux services à la demande des utilisateurs. Ce modèle contient toutes les entités nécessaires au : suivi de la réception du contexte, l'interprétation, le raisonnement sur le contexte, l'adaptation et l'exécution du service. Il s'appuie sur des mécanismes de modélisation, de raisonnement et d'auto-adaptation pour s'ajuster à de nouvelles situations. À cet environnement, on associe aussi l'utilisateur, qui est représenté par des caractéristiques telles que son identité et ses préférences.

0.4.2 Proposition d'un pattern architectural (MVCA)

Dans la deuxième étape, nous nous concentrons sur l'aspect logiciel du système intelligent. Alors, nous proposons un pattern logiciel capable d'assurer une sensibilité au contexte, un raisonnement et une adaptation afin de répondre efficacement aux attentes des utilisateurs dans un environnement à large échelle. Nous proposons alors une structure issue du pattern MVC qui a prouvé son efficacité dans plusieurs domaines, sur laquelle on rajoute un autre composant dédié à l'adaptation. L'objectif de notre proposition est de séparer les fonctionnalités du système dans le but de réaliser une architecture modulaire, flexible et scalable en cas de forte demande de services. Nous commençons par analyser et identifier les principaux modules nécessaires à la conception d'un système intelligent ainsi que leurs interactions. À partir de cette analyse, nous précisons pour chaque module les principales préoccupations à réaliser pour arriver à adapter les services vis à vis le changement du

contexte. Notons que notre structure est munie d'un module additionnel nommé Adaptateur comportant tous les processus de raisonnement et/ou d'apprentissage nécessaires pour réaliser une certaine adaptation.

0.4.3 Revue et choix des méthodes d'apprentissages

Dans la troisième étape, nous nous intéressons à l'étude des méthodes de raisonnement et d'apprentissage pour réaliser l'adaptation qui aura lieu dans la prochaine étape. Une étude est réalisée dans le domaine de l'apprentissage automatique, en particulier, l'apprentissage par renforcement et l'apprentissage supervisé (Belaidouni et Moeiz, 2016). Nous réalisons dans ce travail une hybridation entre les deux types d'apprentissage pour aboutir à une adaptation efficace de service dans un espace intelligent. Cette hybridation fait appel aux algorithmes : Q-learning et CBR. Q-learning permet d'atteindre un service voulu par une succession d'essais. Autrement dit, il va découvrir par une suite de tentatives et d'erreurs ce qu'il convient de faire comme service en différentes situations jusqu'à aboutir au service désiré. Le CBR quant à lui, permet de s'adapter à un nouveau contexte en utilisant un ensemble de contextes déjà résolus. Le but de cette hybridation est de surmonter les inconvénients du Q-learning pour faire ressortir une nouvelle approche capable d'acquérir le contexte et de produire le service adéquat.

0.4.4 Revue et application de l'adaptation dynamique

Dans la quatrième étape, nous nous concentrons beaucoup plus sur le processus d'adaptation. Nous proposons un algorithme d'adaptation muni d'un mécanisme de raisonnement et d'apprentissage pour être en mesure d'ajuster des services selon le contexte de l'utilisateur et de son environnement.

Notre algorithme est une hybridation entre le Q-learning et le CBR. D'abord, le Q-learning permet de sélectionner pour un nouveau contexte, le service le plus bénéfique, suite à une séquence de services exécutés pour différentes situations. Dans chaque séquence, le système évalue le service soit par une récompense positive (le cas d'un service adapté), soit par une

autre négative (service inadapté). Cependant, cette procédure converge moins rapidement ce qui rend le temps d'exécution assez élevé. Pour remédier à cet inconvénient, nous incorporons le CBR, non seulement pour réduire le temps de convergence, mais aussi pour effectuer une adaptation dynamique des services.

0.5 Originalité des travaux proposés et contributions

Cette thèse s'inscrit dans le domaine de l'informatique ambiante et les espaces intelligents. Notre objectif consiste à proposer un système SIAC capable de répondre aux différents défis posés par l'environnement à intelligence ambiante. Ces environnements se caractérisent notamment par l'ouverture, l'hétérogénéité, l'incertitude et la dynamicité des entités qui les constituent. Ces caractéristiques imposent alors des défis scientifiques importants pour la mise en oeuvre de systèmes intelligents. Les principaux défis soulevés sont : l'intelligence, la sensibilité au contexte, l'auto-adaptation, l'interopérabilité, la transparence et la scalabilité.

Dans cette thèse, nous considérons que l'architecture MVC demeure la base de la construction d'un système SIAC en raison de l'abstraction apportée dans la représentation des entités de l'environnement ambiant et du couplage faible qu'elle permet entre ces différentes représentations. En effet, les entités hétérogènes ne sont représentées que par leurs fonctionnalités encapsulées et qui sont indépendantes des technologies d'implémentation. Ainsi, le paradigme orienté module garantit l'interopérabilité, une forte cohésion et un faible couplage nécessaires à la construction dynamique d'applications dans un environnement ouvert et hétérogène.

Cependant, afin de pouvoir appliquer correctement l'adaptation des services dans le cadre d'un système SIAC, il est nécessaire d'identifier des mécanismes adéquats pour l'adaptation des services permettant de réaliser l'ajustement des services dépendamment du contexte.

Le présent travail se focalise à modéliser une structure modulaire, extensible, permettant l'adaptation dynamique de services dans un espace intelligent. Ainsi, plusieurs contributions vont ressortir à la fin de ce travail à savoir :

1. définition d'un modèle de qualité pour identifier les mesures indispensables conduisant à une architecture flexible, sensible au contexte pour offrir des services aux utilisateurs ;
2. proposition d'une structure générique d'un système intelligent auto-adaptable ;
3. décomposition de l'ensemble des traitements effectués dans un SIAC en quatre composants nécessaires pour suivre le contexte à partir de la détection à la sélection du service approprié ;
4. proposition d'une classification du contexte capturé pour une meilleure adaptation de service ;
5. adaptation de services selon le changement contextuel.

0.5.1 Définition d'un modèle de qualité

La première contribution a porté sur la proposition d'un modèle des exigences. Ce modèle est riche en termes de métriques de qualité qui sont indispensables pour un SIAC car il englobe l'ensemble des identificateurs de qualité pour une architecture d'un environnement ambiant. Ces métriques concernent trois caractéristiques principales à savoir : fonctionnalité, réutilisabilité et maintenabilité. Chaque caractéristique dépend fortement d'autres mesures de façon directe ou indirecte. Tout d'abord, un SIAC est dit fonctionnel, si et seulement si, il est sensible au contexte, auto-adaptable et modulaire. Ces aspects essentiels procurent au système la capacité de raisonnement pour la reconnaissance du contexte, l'adaptation et la prise de bonnes décisions, afin d'assurer la pérennité des services par rapport à l'évolution de l'environnement intelligent et de ses utilisateurs. Par ailleurs, ces trois aspects s'appuient sur d'autres exigences nécessaires pour le bon fonctionnement d'un SIAC. Tout d'abord, un tel système doit assurer un haut niveau d'abstraction du contexte. Ensuite, un système doit être en mesure de changer la valeur du contexte selon le changement de l'environnement intelligent et de ses usagers, et d'adapter aussi le service suite à ces changements. Ces aspects

essentiels procurent au SIAC une certaine efficacité dans le traitement et la réponse aux demandes des utilisateurs.

0.5.2 Proposition d'un système générique auto-adaptatif

La deuxième contribution a porté sur la proposition d'une architecture générique pour un système SIAC. Le système décrit, ainsi, notre vision du système dans un environnement ambiant. Cette infrastructure est construite suivant un modèle multicouche afin de supporter un système adaptatif capable d'effectuer la perception, le raisonnement, l'adaptation, la sélection et l'invocation des services appropriés lorsqu'un évènement se produit dans un environnement intelligent.

Plus précisément, l'architecture est structurée en quatre couches principales à savoir : la couche physique, la couche contexte, la couche adaptation et la couche application. L'ensemble repose sur plusieurs modules interconnectés et engagés dans un processus de collaboration afin de réaliser les objectifs spécifiés lors de la détection d'un nouveau contexte. Ainsi, ce système qui est sensible au contexte, réagit à chaque détection du contexte, et fournit aux utilisateurs concernés des services appropriés.

0.5.3 Proposition d'une architecture logicielle MVCA pour un SIAC

La troisième contribution a porté sur la proposition d'un pattern architectural nommé par MVCA. Ce Framework qui se base sur le pattern MVC permet de mieux gérer le contexte, de contrôler l'adaptation, la sélection et l'invocation de services afin de garantir une continuité de services face aux pannes imprévisibles qui peuvent survenir dans un environnement ambiant. En effet, le modèle MVC offre une séparation claire des préoccupations au sein d'une application permettant de réaliser des responsabilités uniques de forte cohésion et de faible couplage. De plus, cette structure permet de gérer les services suivant une architecture flexible et tolérante aux pannes avec une prise en compte du contexte courant.

L'objectif de cette architecture consiste à augmenter les chances d'exécution des services adaptables convenables aux attentes des utilisateurs dans un environnement dynamique et incertain tout en lui garantissant une meilleure qualité de service. Afin d'atteindre cet objectif, notre architecture repose sur quatre modules : (1) Contrôleur, (2) Modèle Contextuel, (3) Adaptateur et (4) Vue. Chaque modèle repose sur des mécanismes dédiés pour effectuer un traitement spécifique. Le Contrôleur est le premier module de l'architecture. Il est responsable principalement à capter tous contextes ou changements au niveau du contexte. Il s'occupe également à interpréter le contexte pour avoir une donnée plus abstraite et compréhensible par les autres modules. Le Modèle Contextuel représente la base de stockage du système. Il est responsable à enregistrer les différents contextes traités dans le système avec leurs services correspondants pour les transmettes, par la suite, à l'Adaptateur en cas de besoin. L'Adaptateur représente le noyau de l'architecture. Il est muni de différents processus pour l'apprentissage et/ou la prise de décision pour bien mener la sélection et l'adaptation des services. La Vue, quant à elle, représente le dernier module responsable à exécuter le service reçu par l'adaptateur.

0.5.4 Réalisation d'un apprentissage renforcé et supervisé dans un cadre intelligent

La quatrième contribution a porté sur l'exploitation d'un apprentissage par renforcement et un autre supervisé. Nous nous intéresserons essentiellement à les appliquer sur un contexte environnemental porté sur le comportement d'un utilisateur, sa localisation ou ses préférences. Cela concerne leur capacité à déterminer pour chaque situation, quel service sera proposé à l'utilisateur. Le principe de l'apprentissage par renforcement repose, en premier lieu, sur une interaction itérée du système intelligent avec l'environnement, sous la forme de l'exécution à chaque instant t d'un service S_t depuis le contexte courant C_t qui conduit à une nouvelle situation avec une récompense r. Alors, la sélection du service sera, petit à petit, améliorée en se basant sur une fonction de valeur permettant, ainsi, d'associer à chaque situation possible une estimation de se situer sur cet état en fonction de l'objectif visé. La grande limite potentielle de ce genre d'apprentissage provient du grand nombre de cycles nécessaires avant d'aboutir à un service adéquat, ce qui, nécessite un temps de plus en phase

d'exécution. Le raisonnement par cas (CBR) est particulièrement adapté, non seulement, pour réduire le temps de convergence de l'algorithme par renforcement, mais, aussi de proposer une adaptation des services avant leurs exécutions. Dans notre thèse, nous optons pour une combinaison entre l'algorithme Q-learning et le CBR, qui permet, d'effectuer une adaptation de service basant sur le contexte environnemental.

0.5.5 Adaptation des services par l'algorithme QL-CBR

La quatrième et dernière contribution a porté sur la proposition d'un modèle d'adaptation de services. Ce modèle constitue une étape primordiale et nécessaire à l'exécution de services. A l'instar de récents travaux (Colman et all.,2014), nous ferons l'usage d'un algorithme dit « QL-CBR », capable d'apprendre de façon optimale à partir des différents contextes pour proposer et adapter par la suite le service concerné. Le modèle proposé utilise à la fois un apprentissage par renforcement (Q-learning) et autre supervisé (CBR). Il comprend trois phases principales : (1) apprentissage ; (2) récupération des cas similaires et (3) adaptation de services. La phase d'apprentissage est réalisée par un agent qui consiste à associer un service prédéfini au contexte, capturé par le biais d'une suite d'essai-erreurs basée sur des récompenses indicatrices de la qualité de ses décisions. La phase de récupération est introduite par le CBR. Elle consiste à récupérer les cas les plus similaires au nouveau contexte en se basant sur la bibliothèque WordNet (Fellbaum, 1994). Elle se base essentiellement sur les résultats issus de la phase précédente. La phase d'adaptation de service consiste, quant à elle, à ajuster les paramètres du service pour mieux les adaptées au changement contextuel.

Afin d'évaluer la performance de notre algorithme. Nous avons mené plusieurs tests pour étudier l'impact de l'algorithme proposé dans l'adaptation des services. Notre simulation est réalisée sur une base de 150 cas. Chaque cas est une combinaison de différents attributs permettant de spécifier le contexte actuel de l'utilisateur. Alors, nous avons itéré notre algorithme de 100 à 1000 fois. À chaque itération, nous avons mesuré le nombre de services bien adapté, le nombre de service mal adapté et le nombre de service par défaut. Sachant que,

un service par défaut est un service généré par une situation planifiée et un service adapté est un service généré d'une situation non planifiée. Dans ce cas, un service adapté se considère soit : bien adapté dans le cas où il convient aux attentes de l'utilisateur, ou mal adapté dans le cas contraire où il ne convient pas aux attentes de l'utilisateur. Pour ce faire, nous avons varié le facteur d'exploration de 0.1 à 1. Nous constatons qu'une valeur de 1 est plus avantageuse pour l'algorithme. Les résultats obtenus sont comparés à l'algorithme CBR.

La figure 0.4 montre clairement que le taux des services bien adapté et nettement meilleur à celui des services mal adapté. En outre, cette figure montre que le taux des services bien adapté augmente avec le nombre d'itération. Cela s'explique par le fait que l'algorithme QL-CBR permet d'ajuster plus de services lorsque le nombre d'itération est élevé. Nous constatons aussi que le nombre de services exécuté augmente proportionnellement avec le nombre d'itération jusqu'à une certaine limite, où le nombre des services exécuté se stabilise. Ce seuil est d'environ 800 itérations.

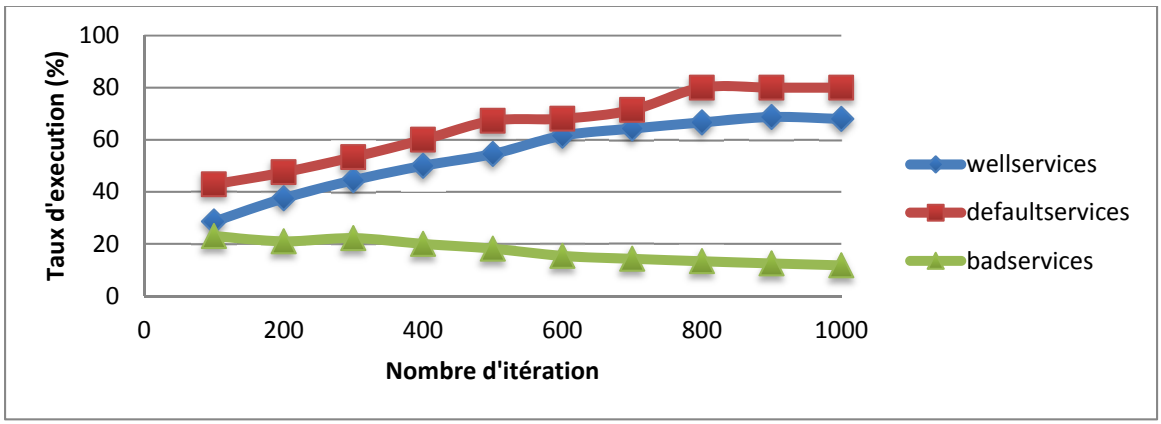


Figure 0.4 Résultat d'adaptation par QL-CBR

La figure 0.5 montre que le taux des services par défaut d'un algorithme CBR est assez élevé en comparent avec ceux qui ont subi une adaptation. Aussi, le taux de services adapté est réduit, par apport à celui du QL-CBR. Cela explique que l'hybridation entre le Q-learning et le CBR permet d'exploiter plus les services pour mieux les adaptés.

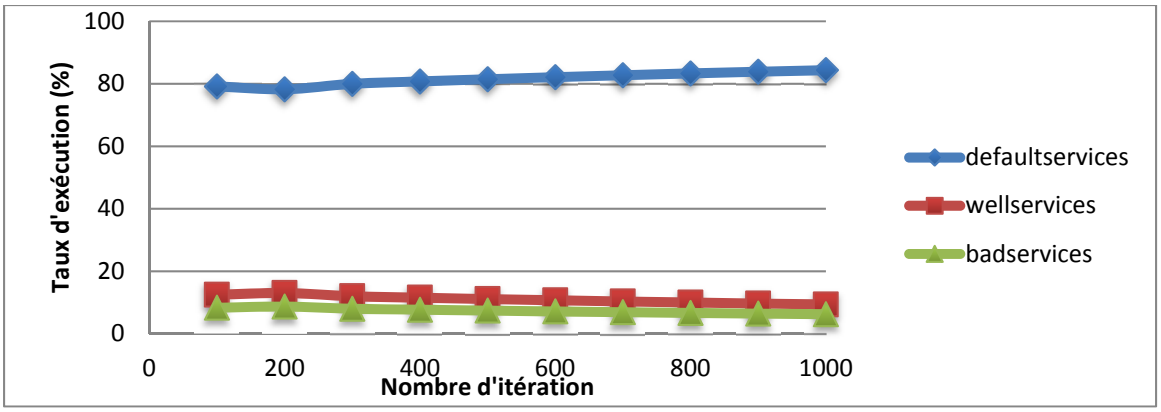


Figure 0.5 Résultat d'adaptation par CBR

Dans ce test, nous avons aussi mesuré la variation du temps d'exécution de l'algorithme hybride (QL-CBR) et celui du CBR (Figure 0.6). Nous constatons que le temps d'exécution des services reçu par les deux algorithmes augmente proportionnellement avec le nombre d'itération. De plus, l'algorithme CBR fourni un temps d'exécution assez faible comparant avec celui du QL-CBR. Cela s'explique par le fait que le processus d'apprentissage appliqué par l'algorithme de Q-learning prend plus de temps lorsque il exploite les états pour aboutir à une meilleure récompense. En effet, lorsque l'algorithme Q-learning exécute un service, il reçoit une récompense qui peut être positive ou négative, selon la nouvelle situation. Dans le cas où la récompense est négative, l'algorithme doit exploiter d'autre situation jusqu'au trouver la meilleure situation qui mène au meilleur service.

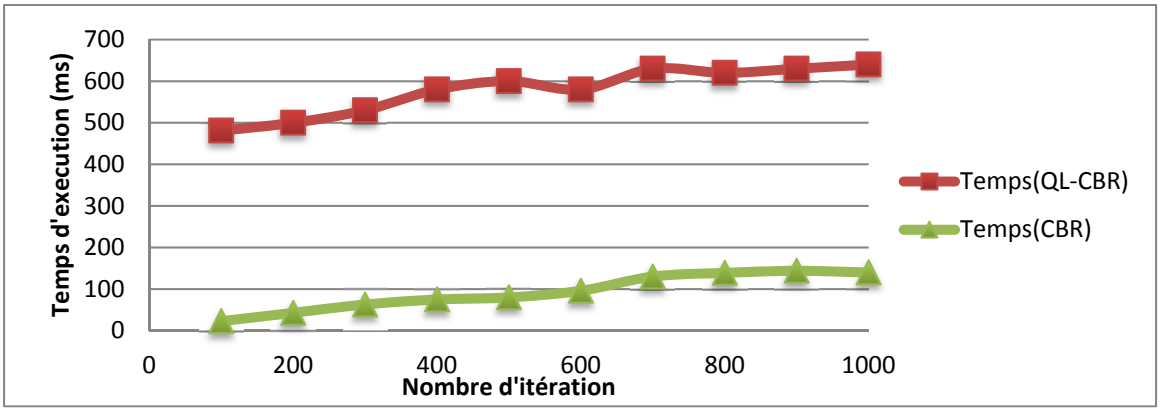


Figure 0.6 Temps d'exécution par itération

0.6 Organisation de la thèse

Cette thèse est ainsi organisée :

1. Le premier chapitre a pour objectif de présenter et de définir les systèmes intelligents. La première section présente l'environnement intelligent et sa composition. La deuxième section décrit la notion du contexte ainsi que la sensibilité au contexte. La troisième aborde la notion d'adaptation dans un tel environnement ainsi que les techniques de raisonnement utilisées. Enfin la dernière section présente l'architecture d'un ensemble de systèmes intelligents avec leurs différentes technologies pour la perception du contexte et l'adaptation du service.

Les trois chapitres qui suivent sont des travaux acceptés/publiés/soumis.

2. Le deuxième chapitre est un article qui a été publié dans le journal « International Journal of Advanced Studies in Computer Science and Engineering » (IJASCSE) :
Belaidouni Somia, Moeiz Miraoui, and Chakib Tadj. « Towards an Efficient Smart Space Design. » International Journal of Advanced Studies in Computer Science and Engineering, IJASCSE, Volume 5, Issue 1, 2016.

Cet article, a été initié par une revue de la littérature sur les architectures des systèmes intelligents sensibles aux contextes, pour être en mesure d'identifier les principales métriques de qualité pour une architecture capable d'assurer des services adéquats suivant le dynamisme de l'environnement. Un modèle architectural générique était réalisé pour un système intelligent intégrant différents processus homogènes capables d'acquérir le contexte, de l'interpréter et de le traiter pour savoir quel service produire à l'utilisateur.

3. Le troisième chapitre est un article qui a été publié dans le journal « Journal of Computer Science » :

Belaidouni Somia, Moeiz Miraoui and Chakib Tadj. « Using MVCA to Improve Architecture Modularity of Smart Spaces. » Journal of Computer Science, Volume 13, Issue 10, (2017): 697-707.

Dans cet article, nous avons abordé différentes structures architecturales spécifiques pour des systèmes intelligents. Nous avons aussi analysé leurs points forts et points faibles. Par la suite, nous avons présenté une architecture par composants qui vise à décomposer toutes les préoccupations dans des modules isolés pour aboutir à une architecture modulaire, flexible, facile à reconfigurer. Il s'agit plus précisément de présenter l'architecture globale et de détailler l'implantation des composants majeurs.

4. Le quatrième chapitre est un article qui a été soumis dans « Journal of Intelligent Systems » (JISYS) le 10 avril 2018 :
Belaidouni Somia, Moeiz Miraoui, Chakib Tadj, « QL-CBR: Hybrid Approach for Adapting Context-Aware Services. »

Dans cet article, nous abordons le problème d'adaptation dans les systèmes intelligents. Nous avons présenté dans un premier temps les différentes méthodes de raisonnement et d'adaptation utilisées. Par la suite, nous avons discuté des deux types d'apprentissages. Le premier est le Q-learning qui est un apprentissage par renforcement. Le deuxième est le CBR qui est apprentissage supervisé. Notre contribution est d'effectuer une hybridation entre ces deux algorithmes pour aboutir à une meilleure adaptation.

Enfin, le dernier chapitre synthétise les idées principales de notre proposition. Nous rappelons à cet effet les points principaux de notre contribution et nous décrivons les perspectives envisagées pour de futurs travaux.

CHAPITRE 1

REVUE DE LA LITTÉRATURE

1.1 Introduction

L'informatique ubiquitaire. Il se définit comme tout espace autour des utilisateurs, constitué de dispositifs et de fonctions interconnectés, capable de réagir intelligemment pour garantir en toute circonstance les services attendus avec la valeur attendue, tel qu'un immeuble intelligent, une maison intelligente ou même une cité intelligente. Un tel espace est une composition de différents dispositifs qui s'interrogent sous différentes techniques pour offrir des services selon les attentes des utilisateurs. Toutefois, afin de concevoir un système intelligent (SI) évoluant dans un environnement intelligent, il est évident de bien comprendre quelles sont les caractéristiques et les exigences auquel il devra faire face.

Les thématiques choisies dans cette thèse ont pour objectif de résoudre notre problématique de recherche située dans un cadre pluridisciplinaire. Le domaine de l'intelligence ambiante couvre l'aspect global de notre problématique de recherche découpée en plusieurs sous-domaines tels, le contexte et la sensibilité au contexte, l'architecture et l'auto-adaptation.

Ce chapitre constitue l'état de l'art à propos de trois axes de recherche. Nous commençons par aborder notre premier axe en présentant, dans un premier temps, quelques définitions essentielles sur l'environnement intelligent, puis, dans un second temps, la composition et l'architecture de ce dernier. Par la suite, nous introduisons notre deuxième axe de recherche mettant l'accent, plus particulièrement, sur le contexte et la sensibilité au contexte. À la fin, nous détaillons notre troisième axe de recherche concernant l'auto-adaptation. Dans cette section, nous aborderons les types d'adaptations qui peuvent être réalisées dans un système intelligent. Nous exploitons aussi les types de raisonnement et d'apprentissage. Vers la fin de cette section, nous présenterons quelques systèmes adaptatifs assez connus dans le monde des systèmes intelligents.

1.2 Présentation d'espace intelligent

Un environnement intelligent ou un espace intelligent est une superficie limitée (une chambre, une maison, un immeuble, voire une métropole) capable d'acquérir et d'appliquer des connaissances sur son propre contexte entre les différentes entités (utilisateurs et/ou dispositifs) pour fournir un service adéquat. Cet espace résultant du phénomène de l'intelligence ambiante est capable de récolter l'information physique via des capteurs spécialisés. Cette information sera traitée par la suite et partagée par des applications ambiantes pour but de déclencher des actions souhaitées. La figure 1.1 révèle la composition générale d'un espace intelligent exploitant ainsi des objets communicants (capteurs, actionneurs, terminaux numériques, équipements intelligents, etc.) d'un environnement connecté avec ces utilisateurs. Ce genre d'espace constitue des espaces physiques et numériques riches et offrant une multitude de services intelligents.

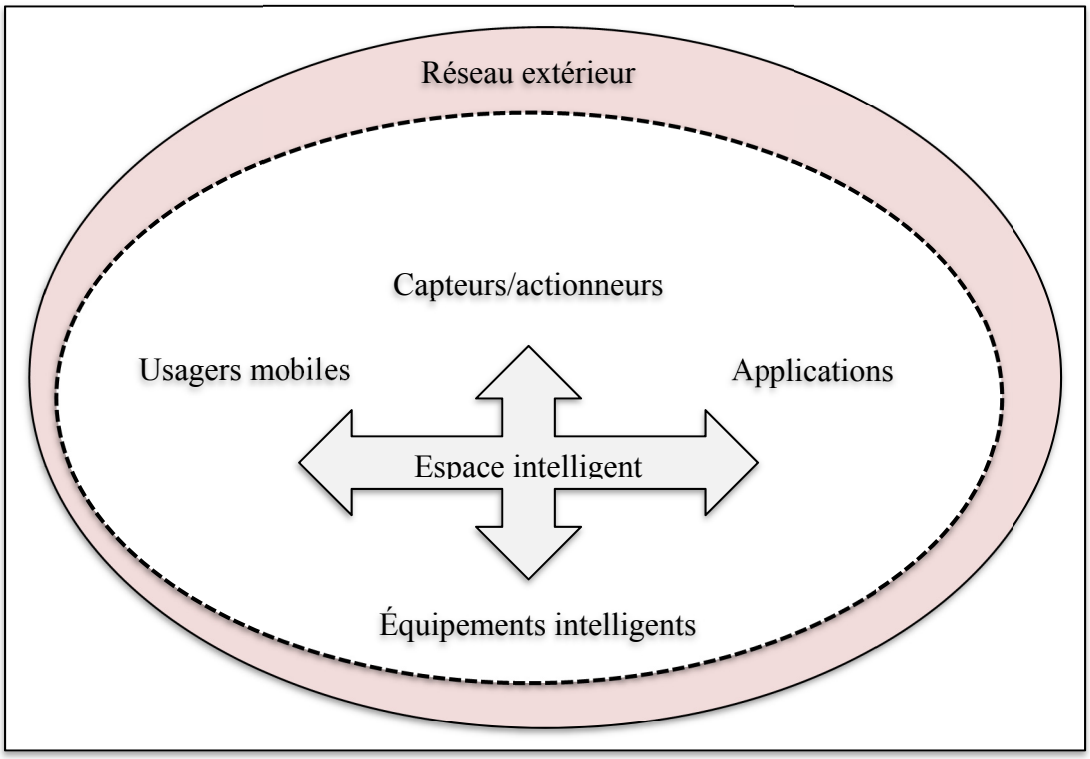


Figure 1.1 Composition d'un espace intelligent

Un système à intelligence ambiante peut offrir alors une multitude de services réactifs ou proactifs permettant d'améliorer la qualité de vie et l'état physique, mental, et le bienêtre social des usagers (Sabri, 2013). Ces services peuvent être de plusieurs types : assistance à domicile, assistance à la mobilité, confort, sécurité, surveillance, etc.

1.2.1 Les caractéristiques d'un espace intelligent

Un espace intelligent consiste en un énorme réseau d'entités physiques (capteurs, actionneurs, équipements, etc.) et virtuelles (services d'informations divers, composants logiciels, etc.) hétérogènes interconnectés et qui interagissent avec l'humain. Les applications d'un espace intelligent évoluent dans un environnements fortement contraints : hétérogène, ouvert, dynamique, incertains et multi-échelles. Ces caractéristiques imposent ainsi des défis scientifiques considérables pour le développement, le déploiement, l'exécution et la maintenance de ces applications. La figure 1.2 illustre les différentes caractéristiques d'un environnement intelligent qui permettent de mieux comprendre l'espace utilisé afin de mieux répondre aux défis exigés.

1.2.1.1 Hétérogénéité

L'hétérogénéité est l'une des caractéristiques majeures des environnements intelligents. Cette hétérogénéité est due, d'une part, aux diverses technologies des dispositifs intégrés dans l'espace, telles que les technologies des capteurs et des actionneurs embarqués, d'autre part, à la diversité des technologies de communication filaires et sans fil. Cette diversité s'applique aussi au niveau des composants logiciels (e.g. systèmes d'exploitation et langages de programmation). Les informations échangées entre les objets communicants sont également hétérogènes et se diffèrent non seulement dans la nature des messages (e.g. textes, images, flux audio) mais aussi dans le format (e.g : doc, Jbeg, RIFF). Chaque objet communicant, à savoir, le type de dispositif et sa localisation a pour mission de préparer le traitement du contexte et d'offrir le service attendu.

1.2.1.2 Ouverture et distributivité

L'ouverture d'un environnement intelligent à des dispositifs électroniques variés dans un contexte évoluant rapidement est aussi une caractéristique importante d'un espace intelligent. Ces dispositifs, considérés comme des ressources pour l'utilisateur, ne se limitent pas à un seul endroit, mais sont souvent distribués dans plusieurs lieux. Ils sont répartis et interconnectés via des réseaux filaires et/ou sans fil pour se communiquer de façon continue, sans prendre en considération l'état de l'environnement.

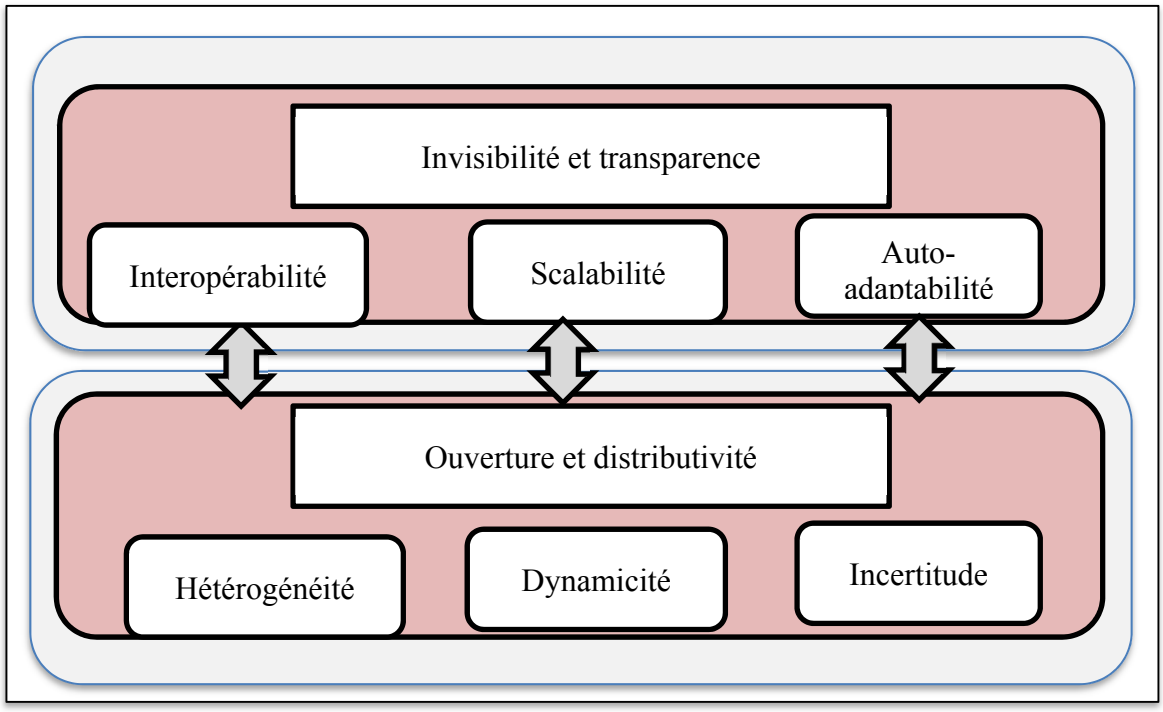


Figure 1.2 Caractéristiques et défis d'un espace intelligent

1.2.1.3 Dynamicité et incertitude

Un environnement intelligent se considère aussi comme très dynamique et incertain. La nature dynamique est due aux changements fréquents de l'environnement, alors que, la nature incertaine est due à l'apparition aléatoire et/ou imprévue de ces changements.

Le changement constant de l'environnement se traduit soit par la mobilité de l'utilisateur (e.g changement de localisation), soit par, l'apparition et la disparition d'équipements comme conséquences aux interactions entre l'utilisateur et les équipements électroniques/physiques [43].

1.2.1.4 Interopérabilité

C'est le tout premier et le plus important attribut à envisager dans la création d'un espace intelligent. L'interopérabilité permet de traiter principalement les problèmes liés à l'hétérogénéité des environnements intelligents. Elle définit la possibilité de comprendre et de communiquer entre les différents dispositifs de marques distinctes. Elle concerne d'une part, l'incompatibilité des moyens de communication, et d'autre part, les différents modèles traitant la donnée. Les intergiciels (middleware) offrent un support permettant de cacher l'hétérogénéité tout en facilitant l'intégration des différentes entités de l'environnement pour assurer leurs communications.

1.2.1.5 Scalabilité

Dans un espace intelligent, la scalabilité démontre la capacité de s'adapter aux fortes demandes des entités communicantes. Un tel environnement est exposé à une perspective d'évolutivité matérielle (e.g. l'ajout des nouveaux dispositifs) ou logicielle (e.g. l'extensibilité des systèmes d'exploitation). Par conséquent, le système doit être en mesure d'accepter cette montée de charge et de réagir correctement aux demandes accrues de services. L'aspect de scalabilité est un facteur qui doit être étudié de prêt pour garantir un bon fonctionnement du système tout en répondant aux attentes des utilisateurs.

1.2.1.6 Invisibilité et transparence

Un environnement intelligent ambiant doit fournir une certaine transparence en rendant toute la technologie sous-jacente invisible à l'utilisateur vis à vis le lieu et le contexte actuel. Par conséquent, l'utilisateur aura l'accès à l'information d'une manière intuitive et plus

transparente. Un environnement intelligent exige aussi un minimum d'intervention humaine qui va être remplacée, d'un autre côté, par des mécanismes d'auto-adaptation, rendant ainsi, le système plus autonome et adaptable aux changements.

1.2.1.7 L'adaptabilité et la pro-activité

L'adaptation au contexte est un caractère indispensable dans un système intelligent pour une utilisation nomade. Cette adaptation nécessite la modélisation des informations contextuelles d'une façon abstraite. Elle se compose de différents mécanismes permettant d´écrire le comportement du système dans un contexte particulier et de l'apparition et/ou l'évolution des nouvelles technologies mobiles afin de réagir efficacement pour répondre aux attentes. L'adaptation se manifeste grâce à l'apprentissage. Ce dernier est une phase importante qui dépend fortement d'une suite de contextes déjà établis. Il faut donc que le système prenne en compte l'historique de l'apprentissage pour être en mesure de proposer des services à exécuter.

Notre thèse s'inscrit dans ce contexte et vise à répondre aux défis des systèmes intelligents dans le cadre d'une architecture orientée adaptation de service.

1.2.2 Architecture générique d'un espace intelligent

La structure générale d'un espace intelligent se compose de trois éléments principaux : le premier élément représente le monde physique. Il comporte les différentes entités qui peuvent exister dans un espace intelligent (e.g. utilisateurs, dispositifs intelligents ou équipement électroniques). Le deuxième élément représente l'interface. C'est le dispositif qui permet d'accéder aux différents capteurs, actionneurs ou mêmes dispositifs. Le rôle primordial des interfaces et de permettre l'échange de données entre les différentes entités, ainsi que l'exécution des services suite à la demande des utilisateurs. La plupart des interfaces dans un espace intelligent sont reliés à des capteurs ou des actionneurs.
Les capteurs sont des dispositifs disséminés dans l'environnement pour collecter l'information qui circule. C'est une source principale d'information contextuelle qui entraine

l'invocation et l'exécution des services pour l'utilisateur. Tandis que les actionneurs, sont des dispositifs permettant d'actionner des services dans l'environnement ou commander des équipements à distance. Le dernier élément de l'architecture représente le système intelligent qui regroupe toutes les modalités de traitements et de gestion du contexte pour assurer le bon fonctionnement du système. La figure 1.3 illustre les principales entités existant dans un espace intelligent.

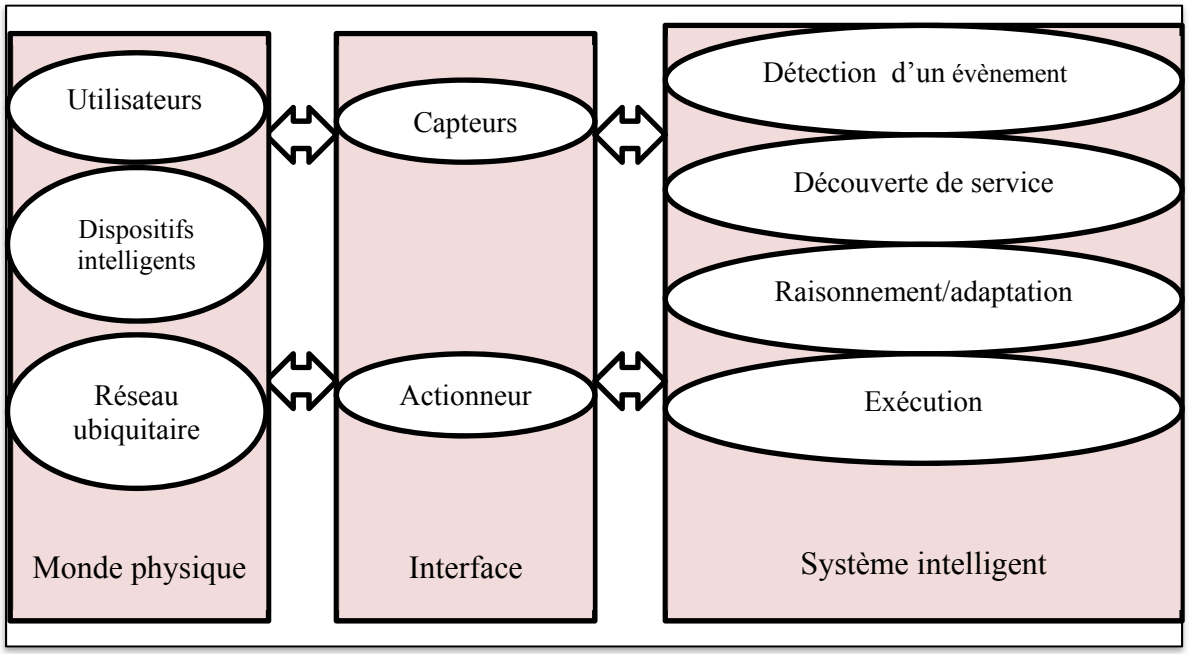


Figure 1.3 Architecture générique d'un espace intelligent

1.2.3 Fondements d'un espace intelligent

Un environnement intelligent se base sur différentes technologies utilisées dans les systèmes informatiques. Les capteurs, le web sémantique et les middlewares sont parmi les entités les plus importantes à réaliser pour assurer une exécution continue et transparente pour les services.

1.2.3.1 Capteurs

Dans un espace intelligent, les capteurs sont des composants distribués au tour d'un point d'intérêt ayant pour mission de collecter et de transmettre les données environnementales de manière autonome. Généralement, les capteurs sont de trois catégories : capteurs physiques, capteurs virtuelles ou capteurs manuels. L'ensemble des capteurs forment un réseau, de façon que chacun ait la possibilité de communiquer avec un autre, de même réseau, via des protocoles de communications de haut niveau.

1.2.3.2 Web sémantique

Le web sémantique est un mouvement provenant de l'organisme W3C (World Wide Web Consortium) qui favorise le développement des méthodes d'échanges de données web. C'est une technologie visant à ajouter des métadonnées aux données contenues dans le web. Cette solution ajoute des descriptions variées à l'ensemble de données ordinaires de différentes sources et transforme par la suite le monde chaotique du web en une bibliothèque organisée, de données précises et définies. Cette technique est prometteuse dans un espace intelligent notamment pour résoudre les problèmes d'interopérabilités entre les systèmes et les services d'origines différentes. Dans ce cadre, plusieurs langages ont été développés pour la représentation des métadonnées. On trouve essentiellement Ressource Description Framework (RDF), RDF Schéma (RDFS) et Web Ontology Language (OWL 2). Chacun de ces langages précités se base sur le précédent, et possède également une syntaxe en XML qui est utilisée dans les documents Web.

1.2.3.3 Middleware

C'est une couche d'abstraction située entre les applications et les systèmes d'opérations. Essentiellement, elle est organisée en deux couches essentielles: la couche inférieure et la couche supérieure. La couche inférieure est connue par l'entreprise service bus (ESB), elle est en contact avec les capteurs et les actionneurs. Son premier but est d'assurer une communication entre les applications hétérogènes qui n'ont pas été conçu pour fonctionner

ensemble. Elle fournit également une représentation standard pour les différentes données reçues (par des capteur) ou à exécuter (par des actionneurs). Elle permet d'unifier les différentes interfaces responsables dans le traitement des services, soit à la découverte de service, l'enregistrement du service ou la consommation du service. La couche supérieure, quant à elle, permet de gérer, en premier ordre, les données contextuelles de bases reçues par la couche inférieure. Elle participe par la suite à les sauvegarder dans un stockage persistant, donnant ainsi la possibilité de les modifier ou de les interroger via des procédures spécifiques. La couche supérieure s'occupe également de mieux gérer les différentes requêtes effectuées par les applications. Elle se caractérise essentiellement par l'aspect d'inférence qui est un point essentiel et indispensable dans un espace intelligent. Cet aspect englobe donc toutes activités à entreprendre soit, pour analyser, décider, ou même, recommander un raisonnement adéquat. Comme le montre la figure 1.4, le middleware utilise d'un côté les services de base d'un système d'exploitation pour fournir des fonctions de communication de haut niveau. D'un autre côté, les applications utilisent les fonctions du middleware pour se communiquer entre eux de façon transparente.

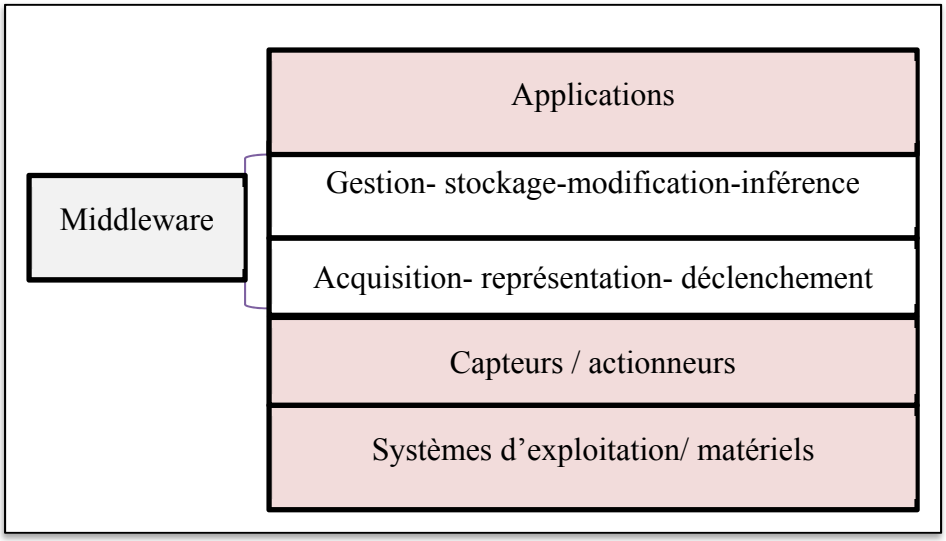


Figure 1.4 Emplacement d'un middleware

1.2.4 Champs d'applications

Aujourd'hui, les applications intelligentes se trouvent dans différents domaines et rendent plusieurs services dans la vie quotidienne. La domotique, par exemple, est un univers plein de nouvelles technologies et techniques de mises en œuvre au sein d'un habitat pour intégrer le confort et la sécurité des habitants. Cet environnement comprend différents dispositifs domestiques, reliés par un réseau et contrôlés par des utilisateurs. Les équipements se caractérisent par un certain type, ainsi qu'un mode de fonctionnement. Ils peuvent fournir un nouveau contexte à n'importe quel moment, comme, ils peuvent exécuter un service pour l'utilisateur. Les utilisateurs, quant à eux, peuvent interagir avec les équipements pour exprimer leurs besoins, ou réagir suite à un évènement ou un service exécuté (Hess, 2003). Les systèmes domotiques doivent être capables d'identifier tous les équipements (anciens/nouveaux) de l'environnement, de contrôler les différentes interactions réalisées entre eux, ou avec l'utilisateur, et surtout d'être sensible à la variation du contexte ainsi que les changements qui s'y produisent pour mieux répondre aux besoins.

Le domaine de transport intègre également les technologies d'informations et de communications connu sous le nom de systèmes (ou services) de transport intelligents (STI). Ces applications ayant la faculté de récolter, stocker, traiter et distribuer des informations relatives à l'état de l'infrastructure et aux mouvements des voitures ou des personnes. Les STI sont employés pour améliorer les performances des services de transport offerts sur le plan environnemental, social et économique.

L'assistance à la personne est un domaine concernant le développement des systèmes spécialisés à s'occuper des personnes âgées, tout en offrant des services d'aides pour vivre d'une manière digne et indépendante (Gu, 2005). Ces applications sont destinées aux personnes âgées et/ou atteintes d'un handicap, pour faciliter leurs tâches quotidiennes : soit pour réaliser, coordonner ou même rappeler des activités précises. Elles sont munies d'interface conviviale, simple à utiliser par tous les concernés, peu importe leurs compétences en informatique. L'avantage de telles applications est de minimiser

l'intervention de tiers dans l'assistance des personnes déficientes et aussi, pour augmenter leurs autonomies et leurs confiances en soi.

Modèle concret

EasyLiving est un modèle concret d'un espace intelligent. En effet, c'est un projet développé au centre de recherche de Microsoft par les auteurs (Brumitt, 2000) pour objectif de bâtir une architecture avec diverses technologies pour un environnement intelligent capable d'acquérir différents équipements et interagir avec plusieurs utilisateurs de façon aisée, et transparente. Pour ce faire, les auteurs identifient d'abord les capteurs qui se présentent comme des dispositifs ayant la possibilité de rassembler des informations de l'extérieur désignant par exemple, la localisation des utilisateurs, le climat ou d'autres dispositifs. De plus, les auteurs prennent en considération l'hétérogénéité des équipements et établissent des applications supportant et facilitant l'interaction entre les utilisateurs et les machines via des interfaces adaptées, facile à manipuler. Généralement, Cette architecture est basée sur un modèle géométrique composé d'un serveur central qui englobe l'ensemble des logiciels nécessaires pour le fonctionnement du projet. Ce serveur est également développé avec une capacité de se localiser, se mettre à jour ou même modifier ses données à distance. L'ensemble des composants du système est relié et supervisé par un middleware utilisant des services de géolocalisations, d'annuaires et de communications asynchrones et assurant la coordination, l'exécution et l'adaptation des services sensibles au contexte pour but de répondre aux attentes des utilisateurs.

1.3 Contexte et définitions

La définition générale d'un contexte selon les dictionnaires s'exprime comme suit : « ensemble de circonstances sur lesquelles se produit un évènement ». En intelligence ambiante, un contexte se réfère à chaque donnée considérée pertinente pour décrire ou modéliser un environnement, sur lequel un service sera déployé et exécuté.

Il existe autant de définitions de la notion de contexte que d'équipes de recherche dans le domaine de l'informatique ubiquitaire. La réalité prouve la difficulté de se limiter par une seule définition de contexte qui sera valable pour tous les systèmes intelligents dû à la nature et l'objectif de chaque projet. De ce fait, plusieurs chercheurs ont mis l'accent sur cette notion et introduit différentes définitions.

Schilit et Theimer en 1994 définissent ainsi le contexte comme toutes données relatives aux changements de l'environnement physique, de l'utilisateur et des ressources de calcul. Ils prévoient que l'étude d'un contexte revient à répondre aux questions «Où suis-je ?», «Avec qui suis-je ?», «Quelles sont les ressources disponibles aux environs ?». Pascoe en 1998, prolonge ce concept et définit le contexte comme un sous-ensemble d'états d'intérêts physiques et conceptuels à une entité particulière.

Dey (Dey, 2001) propose une définition généralisée du contexte : « Le contexte est toute information qui peut être utilisée afin de caractériser la situation d'une entité. Une entité peut être une personne, une place ou un objet que l'on considère pertinent. L'auteur (Zimmermann, 2003) décrit le contexte comme un ensemble d'informations collectées, décrivant un élément qui peut être classé en cinq catégories : individus, localisations, temporelles, tâches et relations.

Brezillon en 2002 considère qu'il n'existe pas un contexte sans contexte. En 2004, une analyse faite par Brezillon et son équipe (Brezillon et al., 2004) conduisant ainsi à conclure que la plupart des définitions du contexte sont une réponse aux six questions majeures : «Qui?»« Quoi» « Ou?» « Quand?» «Pourquoi?» et «comment?».

Miraoui et Tadj en 2007 proposent une définition du contexte orientée service comme étant «Toute information dont le changement de sa valeur déclenche un service ou entraine un changement de sa forme ». Cette définition du contexte délimite les informations pertinentes nécessaires pour offrir les services appropriés à un utilisateur.

Malgré le nombre important de définitions existantes dans la littérature et la grande similarité entre certaines, le mot contexte reste jusqu'à nos jours un cadre général. D'une manière générale à la fois simple et précise, nous pouvons considérer un contexte comme un ensemble d'informations nommées essentielles, ayant la possibilité d'influencer sur une entité appartenant à un environnement pour fournir un service adéquat aux attentes de l'utilisateur (figure 1.5).

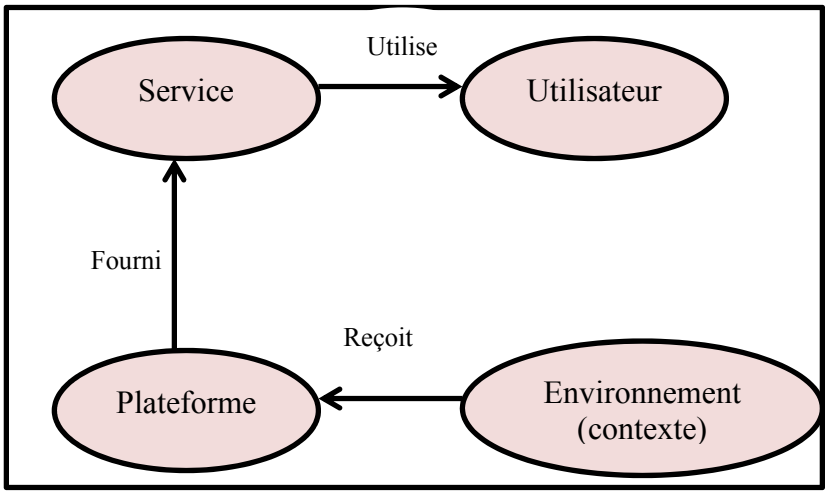


Figure 1.5 Vue d'ensemble d'un contexte

1.3.1 Catégorisation du contexte

À des fins d'adaptation, les informations contextuelles dans un environnement intelligent doivent être collectées, interprétées et présentées au système pour être utilisées dans d'autres traitements. Vu l'hétérogénéité et la diversité des informations reçues, il est préférable de faire une catégorisation, permettant ainsi de bien identifier la nature du contexte pour faciliter l'opération d'adaptation. Nous présentons par la suite la plupart des catégorisations utilisées dans différents projets traitant la notion du contexte. Schilit et son équipe en 1994 (Schilit et al., 1994) proposent de catégoriser le contexte en deux dimensions : le contexte primaire et le contexte secondaire.

Le contexte primaire reflète toute information concernant la localisation, l'identité, le temps et l'activité de l'utilisateur ; le contexte secondaire contient toute informations pouvant être

déduites du contexte primaire (par exemple, de la localisation, on peut déduire les personnes à proximité). (Petrelli et al., 2000), se basent sur les travaux de Schilit et Chen, et introduisent deux nouvelles dimensions du contexte : le contexte matériel (dispositifs et plateformes existants) ; et le contexte social concernant l'individu et ses relations (être seul ou pas, qui sont les autres, etc.). Dans les travaux de (Derntl and Hummel, 2005), le contexte est composé de cinq dimensions : la dimension physique (ressources physiques); la dimension numérique (ressources numériques) ; la dimension dispositifs (logiciels, matériels, réseau, dispositifs portables) ; la dimension apprenant ; et le contexte spécifique du domaine d'application. (Yazid Benazzouz, 2011) a catégorisé les sources des informations contextuelles comme : les préférences de l'utilisateur, l'historique de ses habitudes, son environnement comme la température ambiante ou sa localisation géographique et son environnement système tel que le cadre matériel, les applications et réseaux dans lesquels fonctionne le système.

Dans ce qui suit, nous décrivons en détail les catégories du contexte les plus utilisées dans le domaine de l'apprentissage et l'auto-adaptation. La figure 1.6 illustre les principales catégories d'un contexte.

1.3.1.1 Contexte temporel

Le contexte temporel représente des données relatives aux temps (heure, jour, mois ou année). Ce genre de donnée peut être déployé pour déclencher un service ou une action désirée. Selon Soylo et son équipe (Soylu et al., 2009), le contexte temporal caractérise tous les éléments du contexte auquel ils sont liés à un moment donné au moment de la perception. Prenant le cas d'un utilisateur qui retourne à sa maison en retard de 30min, par apport, à ses habitudes dû à un accident en route. Dans ce cas, le système prévoit ce retard pour retarder aussi les services qui lui sont fournis.

Les formes les plus courantes pour un contexte temporel sont : l'heure (le fuseau horaire, l'heure actuelle d'apprentissage, etc.), la date (jour, mois, année,) et la période (le début d'un

service, la fin d'un service, etc.), la durée d'un service ou la durée d'une ressource (15min, 1h, etc) et aussi les relations temporelles (avant, après, au moment, pendant, etc). Le contexte temporel se trouve sous deux types : sporadique ou périodique. Un contexte sporadique reflète les évènements occasionnels qui se répètent de temps en temps ou même une seule fois, tandis que le contexte périodique décrit les évènements qui se produisent régulièrement et/ou qui peuvent être prévisibles.

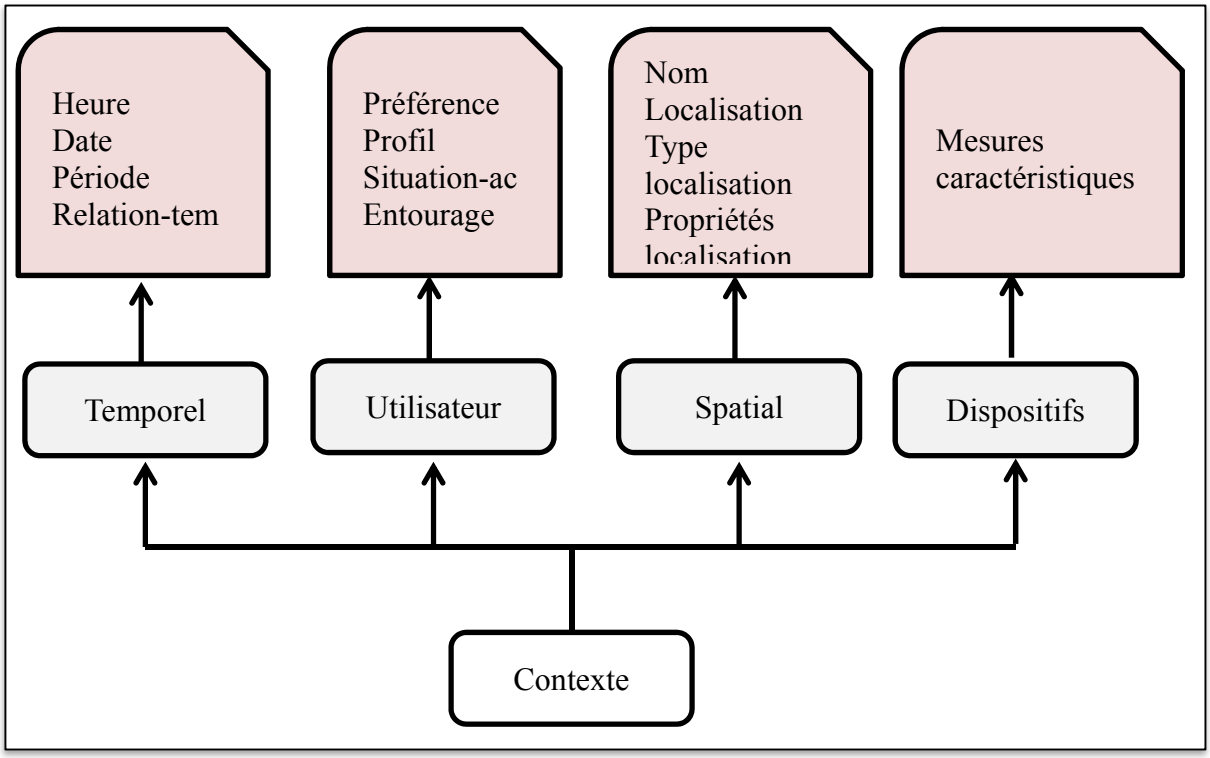


Figure 1.6 Catégorisation du contexte

1.3.1.2 Contexte utilisateur

Le contexte utilisateur est le genre de contexte qui peut englober le plus de données. Effectivement, le système doit se procurer de toutes données concernant ses utilisateurs. Cette donnée peut concerner le profil, les préférences, la situation actuelle et même l'entourage. Le contexte utilisateur de chaque entité peut être soit des données individuelles

décrivant l'entité elle-même, soit des données sociales décrivant le milieu et les alentours de cette entité.

Ces données peuvent être classées essentiellement en deux ensembles : un ensemble de données statiques qui regroupe des informations générales sur l'utilisateur non changeable, telles que le nom, le prénom, la date de naissance, la langue maternelle, etc. et un ensemble de données dynamiques qui varie en fonction de différents domaines d'application, tels que les préférences, les connaissances, les objectifs, les compétences, etc. Une gestion des profils joue un rôle important dans un système d'apprentissage pour adapter les ressources aux spécificités de chaque utilisateur (Brusilovsky,1996) (Rety et al., 2003).

1.3.1.3 Contexte spatial

La localisation de l'utilisateur est un facteur d'une grande importance pour être en mesure de proposer des services adaptés à l'environnement où il se situe (Soylu et al., 2009). Souvent, la localisation est considérée comme des coordonnées physiques par rapport à un système géographique. Elle peut être exprimée sous différentes façons : relative (à côté de, près de, loin de, etc.), nom de la localisation (maison, bureau, université, etc.), type de localisation (dynamique, fixe, public, privé, etc.), et propriété de la localisation (niveau de bruit, niveau de luminosité, confort, etc.)(Dobson, 2005).

1.3.1.4 Contexte dispositif

Le contexte dispositif permet de mesurer les caractéristiques des dispositifs (De Groot and van Welie, 2002). Cette mesure est nécessaire de la connaitre, à priori, afin d'être capable d'adapter les services selon la technologie mobile. Par exemple, si un cours à distance sera donné en vidéo, il ne sera pas adapté à des appareils mobiles sans écran. Ce type de contexte envisage également l'hétérogénéité entre les mobiles et manifeste dans le calcul de leurs capacités et ressources. Dans un cas d'itinérance, il prend en considération les nouveaux mobiles rencontrés ainsi que leurs distinctions en ressources.

1.3.2 Sensibilité au contexte

La sensibilité au contexte ou encore la prise en compte du contexte est une approche prometteuse et indispensable dans un environnement intelligent. Elle est introduite pour la première fois par (Schilit and Theimer, 1994) dans leurs travaux autour d'un système de localisation. Ils ont défini un système sensible au contexte s'il a l'aptitude d'interpréter les informations issues du contexte et d'adapter sa réponse en fonction du contexte d'utilisation.

Selon (Brown, 1995) une application sensible au contexte doit extraire de l'information ou effectuer des actions en fonction du contexte utilisateur détecté par des capteurs. (Salber et al., 1998) définit la sensibilité au contexte comme étant la capacité d'un système à agir avec des données provenant du contexte. D'après (Abowd et al., 1998), un système est sensible au contexte s'il utilise ce contexte pour mettre à disposition de l'utilisateur des informations ou des services qui lui sont utiles.

Autour des années 2000, Chen et Kotz ont également donné deux définitions pour la sensibilité au contexte:
La sensibilité active : toute application ayant la capacité de s'adapter automatiquement au contexte découvert par le changement du comportement de l'application ;
La sensibilité passive : toute application ayant la capacité de rendre un contexte pertinent pour une utilisation ultérieure.

Dey et Abowd (Dey et Abowd, 2001) envisage un système sensible au contexte s'il utilise un contexte ayant pour but de fournir un service ou une infirmation selon les besoins de l'utilisateur. Cette définition a été adoptée par plusieurs chercheurs dans le domaine de l'informatique ubiquitaire. Miraoui (Miraoui, 2009) considère un système sensible au contexte s'il possède la capacité de changer automatiquement les formes de ses services ou de déclencher un service à la suite d'un changement au niveau d'une information ou un ensemble d'informations caractérisant le service.

Dans le cadre de notre travail nous avons repris la définition proposée par Dey et abowd et nous avons mis l'accent sur l'aspect contextuel pour répondre aux besoins des utilisateurs. En effet, la sensibilité au contexte exige une capacité de traiter le contexte reçu pour produire un service désiré. Ceci signifie que l'application doit être en mesure de capter des informations (sous forme numérique), de les déchiffrer en chiffres pour les transformer en données symboliques et compréhensibles de haute abstraction.

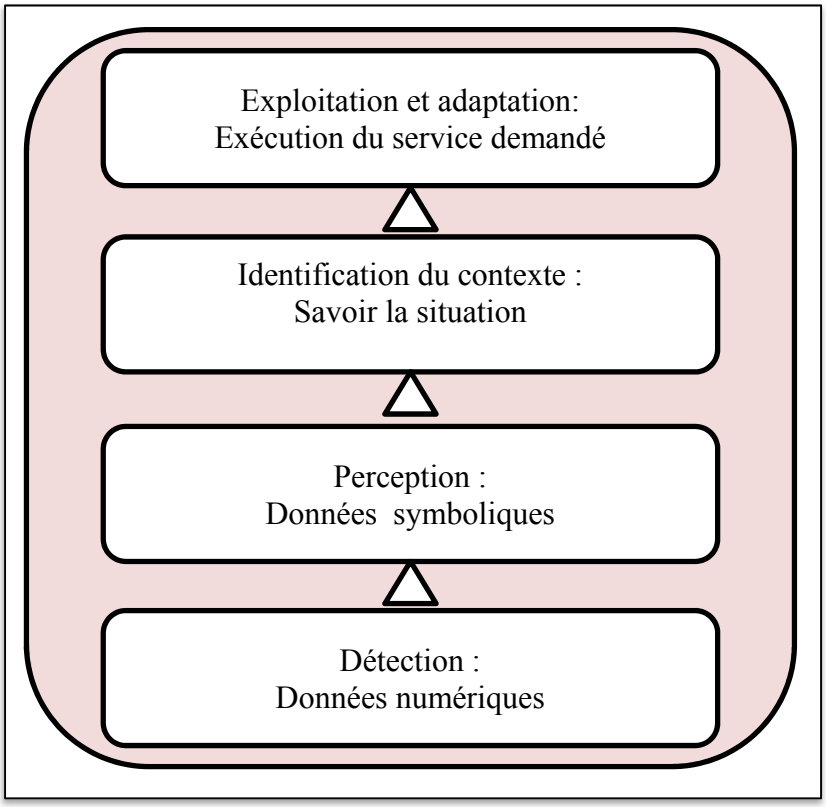


Figure 1.7 Infrastructure d'un modèle de contexte

Une fois le contexte capté, il sera bien identifié par le système et parviendra à déclencher ou à adapter un service. Autrement dit, l'application doit être en mesure de réagir en cas d'un changement de contexte et de s'adapter selon la nouvelle situation. La figure 1.7 illustre les différentes phases passées dans une application sensible au contexte.

1.3.3 Modélisation du contexte

Pour les systèmes ubiquitaires qui doivent fonctionner de façon correcte et performante, il est indispensable de construire un contexte bien structuré et ordonné afin qu'il soit exploité de façon efficace et simple. La modélisation du contexte a pour objectif de présenter une idée, un phénomène, une situation du monde réelle sous forme de descriptions simplifiées (Buthpitiya et al., 2012). Précisément, modéliser une situation, c'est la présenter en données abstraites pour exprimer un contexte donné.

Généralement, elle est accompagnée d'un développement de système de gestion de contexte approprié pour la collecte, le raisonnement et le traitement des données. Un contexte reçoit des informations de plusieurs sources de nature hétérogènes. Elles peuvent être reçues via des capteurs, entrées utilisateurs, données systèmes ou même interaction avec d'autres systèmes.

Par conséquent, ces données fournissent des caractères distincts. Ils existent celles qui sont plus fiables, telles que les données utilisateurs mais obsolète et changeables rapidement. D'autres, ils sont dynamiques et moins fiables comme celles reçues par des capteurs. Ainsi le besoin de prendre en considération ses contraintes pour donner une modélisation correcte et efficace, devient une nécessité pour le bon fonctionnement d'un système ubiquitaire.

Dans la suite, nous présentons un aperçu sur les principales techniques de modélisation de contexte :

1.3.3.1 Modélisation par mots clés

La modélisation du contexte par mots clés est l'une des premières méthodes utilisées dans la description des préférences des utilisateurs. En effet, elle est beaucoup utilisée dans les systèmes de navigation ou de recherche d'information. Son principe consiste à associer un ensemble de mots clés pondérés à des utilisateurs. Ces mots clés sont des données numériques, décrivant l'intérêt et le profil des utilisateurs dans un domaine d'application (Schilit et al., 1994).

Cette technique souffre de plusieurs inconvénients. Le plus important consiste à donner une représentation que pour des informations d'intérêt et de préférences et son inaptitude à modéliser les dépendances et les interactions entre les informations du contexte. Cette modélisation parait simple à réaliser et permet d'un côté le partage des données contextuelles. Toutefois, son utilisation est limitée sur des concepts simples. Les concepts les plus compliqués ainsi que les relations entre eux ne peuvent pas être modéliser par mots clés se qui rond cette modélisation insuffisante et inflexible (Strang and Linnhoff-Popien, 2004).

1.3.3.2 Modélisation par langages à balises

Les langages à balises tel que XML sont une autre façon pour représenter un contexte. C'est une approche permettant de décrire les données sous forme de balises, contenant des attributs associés aux contenus. En particulier, cette structure permet de décrire des données de profiles. Son principe se fond sur la norme générique des langages à balises SGML (Standart Generalized Markup Language) qui définit la super classe de tous les langages à balises.

Le standard CC/PP (Composite Capabilities/ Preference Profile) est un exemple de modélisation de contexte par langages à balises. Il se base sur RDF (Ressources Description Framework) qui est développé par W3C, l'organisme de normalisation des technologies WEB. RDF est un format standard pour l'échange de données sur le WEB. Il fournit un modèle de graphe destiné à décrire des ressources web et leurs métadonnées. Le Framework CC/PP combiné avec RDF, permet de structurer les profiles des usagers sous forme d'attributs associés à des valeurs et accessible en deux niveaux de hiérarchies. Toutefois, cette représentation souffre d'un manque de structuration pour le traitement des profils complexes.

Un deuxième standard CSCP (Comprehensive Structured Context Profiles) surpasse les points faibles du CC/PP. Toujours, en se basant sur RDF, le standard CSCP offre une expression à plusieurs niveaux de hiérarchies sous forme de sessions de profiles. Une session

de profile décrit toutes les informations contextuelles concernant l'usager, le dispositif ainsi que le réseau pendant une session mobile.

1.3.3.3 Modèle graphique

Le modèle graphique définit une autre modélisation pour le contexte. Des exemples de telles techniques sont les diagrammes UML et les graphes contextuels. Les diagrammes d'UML de OMG (Object Management Group) sont les plus répandus et les plus utilisés pour modéliser un contexte. Principalement, les attributs représentent les concepts, et les associations sont dédiées pour définir les relations entre ces différents concepts.
CML ou Context Modeling Language est un exemple de ce modèle graphique. Le modèle est proposé par (Henricksen et Indulska, 2006) offre une structure permettant, en premier, de capturer les données contextuelles suivi de leurs ressources de provenance (statique, par capteurs, ou des données dérivées), de modéliser les relations entre les différents types de données, puis, de sauvegarder un historique sur les informations contextuelles. Cependant, CML est un modèle non standardisé qui n'est pas fourni avec un historique.

1.3.3.4 Modèle basé sur la logique

La logique est un phénomène qui se trouve dans chaque processus de raisonnement ou d'inférence. Théoriquement, une logique est un ensemble de règles d'intelligence artificielle appliquées sur des expressions ou des faits pour décider ou déduire d'autres faits. Les modèles basés sur la logique se servent, en premier ordre, de la logique et de l'algèbre booléenne afin d'être capable de formaliser les données contextuelles sous forme de faits, des expressions ou des règles. Ils peuvent également subir des modifications tel les ajouts, la suppression, la mise à jour. L'auteur (McCarthy, 1994) et son groupe de recherche ont introduit le premier modèle basé sur la logique en définissant le contexte comme étant une entité mathématique avec des propriétés issues de l'intelligence artificielle. Ensuite, ils ont spécifié des formalisations de base, composées d'axiomes et permettant d'avoir des phénomènes logiques. Les projets Mobile GALA, sont parmi les projets qui s'intéressent aux modèles logiques. Ils se sont développés par les auteurs (Shiva, 2005) et (Ranganathan and

Campbell 2003) dont le principe est de présenter le contexte sous un ensemble composé de quatre prédicats de premier ordre de la façon suivante : Contexte = (<ContextType>, <Subject>, <Relater>, <Object>), sachant que (<Subject>, <Relater>, <Object>) sont reliés soit par des prépositions, des verbes ou des opérateurs de comparaison (=, >, ou <).

1.3.3.5 Modèles basés sur des ontologies

La conception sous forme d'ontologies représente le modèle le plus puissant comparé aux modèles décris précédemment. C'est une approche robuste, bien structurée, possédant des techniques de raisonnement fiables utilisées par différents outils. Une ontologie est une description sémantique, structurée et formelle, représentant des concepts d'un domaine et de leurs inter-relations (Uschold and Grüninger, 1996). Elle se fond en quatre entités importantes (Eunhoe et Jaeyoung, 2008): l'environnement, la plateforme, l'usager et le service. Les ontologies permettent de concevoir différents contextes avec leurs interactions de manière uniforme. Elles sont souvent associées à des moteurs d'inférence pour raisonner sur les informations de contexte selon des règles d'inférence. Elles possèdent également des outils de vérification de cohérence pour les informations représentées.

Principalement, l'analyse évaluative se fond sur deux questions majeures :

1. De qu'elle façon les connaissances (les usagers, l'environnement, les dispositifs et les relations entre eux) sont présentées dans l'approche ;
2. De qu'elle façon sera le raisonnement au sein des connaissances reçus.

Les modèles OWL et DAM+AIL sont proposés par (McGuinness, 2002) et (Zuo, 2003) respectivement. Ils représentent les premières approches utilisant les ontologies spécialement pour la modélisation du Web sémantique en se basant sur XML et RDF.

(Strang et al.,) ont aussi proposé le modèle CoOL qui est structuré en deux types d'ontologies. Le premier est le noyau de CoOL. Il se base sur les langages d'ontologies OWL et DAM+AIL. Le second est l'intégration de CoOL. C'est un ensemble de graphes et de

protocoles permettant et facilitant le fonctionnement du noyau de CoOL dans différents intergiciels.

SOUPA est une autre approche introduite par (Chen, 2004) qui se base sur RDF et OWL pour fournir une représentation contextuelle sémantique adaptée au raisonnement et au partage de données.

SOCAM est un intergiciel adapté par (Gu, 2005) pour le développement d'applications sensibles au contexte. En effet, il offre une infrastructure composée de plusieurs niveaux, tout en permettant, la représentation et la classification des données contextuelles, le raisonnement ainsi que le partage de données.

1.3.4 Techniques du traitement du contexte

Dans cette section, nous allons présenter les principales techniques qui ont pour but de décrire un contexte capté à partir de sources diverses, pour arriver à un niveau particulier, compréhensible par les autres modules du système.

1.3.4.1 Raisonnement

Le raisonnement est un processus primordial dans les applications sensibles au contexte. En effet, cet aspect accroit l'intelligence du système tout en procurant la capacité de reconnaitre le contexte et l'interpréter à un niveau d'abstraction plus élevé afin de le rendre compréhensible par le système. Un raisonnement peut s'effectuer via des axiomes, qui sont des hypothèses (propositions ou règles) valides. Ensuite, en fonction des relations entre ces axiomes, de nouvelles connaissances sont déduites via le raisonnement.

1.3.4.2 Fusion de données

La fusion de donnée constitue le processus de combinaison et d'association de données contextuelles provenant de différentes sources. Effectivement, elle permet de construire et

d'intégrer de nouvelles données qui ne sont pas nécessairement une interprétation à des données de base. Prenons l'exemple de deux données de contexte provenant de deux différentes sources, la première est « la vielle est malade ». La deuxième dit que : « la vielle s'appelle Vanessa ». Donc la nouvelle donnée déduite est « Vanessa est malade».

1.3.4.3 Reconnaissance

La reconnaissance du contexte est une autre approche de traitement. Elle se fond sur l'utilisation des techniques d'apprentissage automatiques sur des données provenant généralement de capteurs pour être en mesure de décrire et de connaitre le contexte correspondant. Le principe de ces techniques est exploité comme suit : en premier, l'apprentissage se fait sur l'historique des données contextuelles stockées dans des fichiers. Ensuite, le contexte voulu sera recherché parmi les modèles d'apprentissages existants. À la fin du processus, le contexte sera identifié.

1.3.4.4 Prédiction

Prédire un contexte est une autre alternative permettant d'anticiper un futur contexte en se basant sur le contexte observable actuel et les contextes passés. Cette méthode et très utile dans les environnements de prévision d'actions tels que les environnements intelligents. La prédiction peut être achevée par des outils de classifications qui permettent de faire un accord entre les états observables ainsi que les états prédis et leurs services associés, par exemple : « la personne accède à son bureau le matin, elle ouvre son ordinateur », alors, l'état « ouvre son ordinateur » est un contexte prédis à chaque fois que l'état observable sera « la personne accède à son bureau le matin ». La figure 1.8 résume l'ensemble des types utilisés pour la modélisation et le traitement du contexte

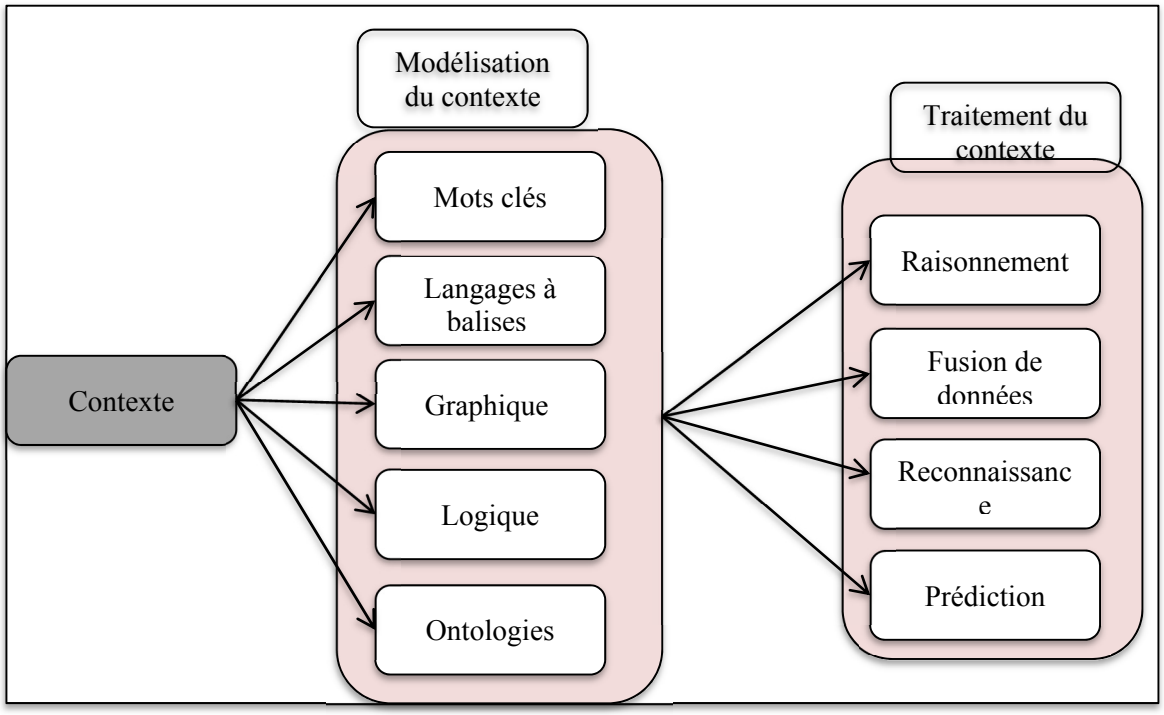


Figure 1.8 Types de modélisation et de traitement d'un contexte

1.4 Auto- adaptation

1.4.1 Concepts D'adaptation

L'adaptation dans un espace intelligent peut être classée en deux types : statique et dynamique dépendamment de l'implication de l'utilisateur. Dans le cas où l'adaptation de service se fait de manière transparente à l'utilisateur, on parlera alors d'une adaptation dynamique. Dans le cas contraire, où l'utilisateur intervient pour adapter le service selon ses préférences, on parlera donc d'une adaptation statique (Figure 1.9). Dans notre thèse, nous optons pour un système adaptatif permettant de réaliser une adaptation dynamique basée sur l'information provenant de l'utilisateur ou de l'environnement.

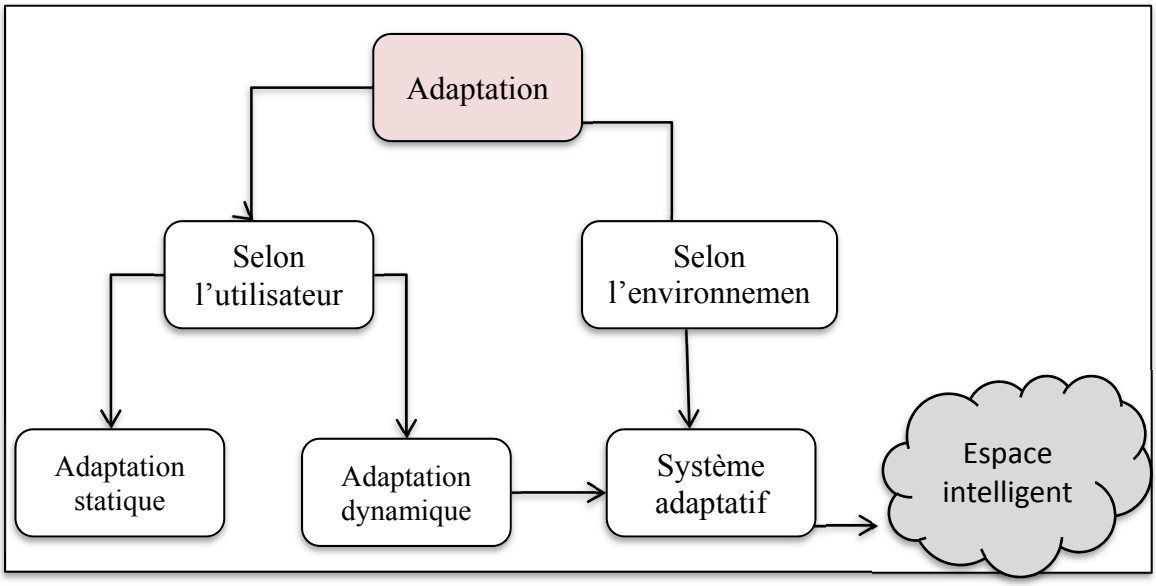


Figure 1.9 Types d'adaptation

Généralement, l'adaptation se présente sous trois concepts principaux : la personnalisation, la recommandation et la reconfiguration. Un système intelligent doit être capable de prendre des décisions sur l'exécution des services. Le processus de raisonnement est un élément essentiel pour effectuer un comportement intelligent.

Toutefois, ce processus est interchangeable et peut effectuer différentes tâches suivant l'objectif initial. Nous allons aborder, par la suite, les différentes formes d'adaptation utilisées dans divers systèmes :

1.4.1.1 Personnalisation

La personnalisation est considérée comme une forme d'adaptation. Son principe vise à adapter le système afin qu'il réagisse en fonction du contexte utilisateur et de ses préférences. L'auteur (Ivar, 2006) définisse la personnalisation comme étant un processus d'adaptation de services selon les besoins des utilisateurs qui peuvent être reliés soit à la présentation ou au fonctionnement du service lui-même. De plus, il prévoit que la gestion de l'information nécessaire pour l'adaptation du service représente la clé d'une personnalisation. Selon (McBurney, 2008), la personnalisation se considère comme une adaptation basée sur les

préférences des utilisateurs qui seront fournies par des mécanismes de surveillances et d'apprentissage. (Papadopoulou et al. 2008) distinguent la personnalisation comme étant le processus de création, de maintien et d'application des préférences utilisateurs pour la prise de décision.

1.4.1.2 Recommandation

L'idée principale de la recommandation est d'enregistrer les préférences des utilisateurs pour les recommander lorsqu'elles semblent intéressantes à produire ou répondre à un service désiré. Ce processus d'adaptation est considéré comme un cas particulier de la personnalisation puisqu'il est bâti également sur les préférences et les besoins des utilisateurs. Les auteurs (Pignotti et al., 2004) considèrent que la compréhension du contexte dans toutes ses formes (profils, préférences, historiques, emplacements, dates et heures) reste le facteur essentiel pour élaborer ce type d'adaptation. Les auteurs (Mettouris, Cristoph et Papado, 2014) introduisent les systèmes ubiquitaires de recommandation qui se basent sur l'utilisation de méthodes de recommandation sophistiquées au lieu d'offrir des méthodes anciennes basées sur le mode de requête ou de recherche dans les répertoires des systèmes.

1.4.1.3 Reconfiguration

La reconfiguration est également une autre forme d'adaptation. Sa particularité réside dans le fait qu'elle se base essentiellement sur les informations du système (réseaux, capteurs, applications). Cette forme est adoptée dans plusieurs travaux. Par exemple, les auteurs (A.Giovanni et al., 2005) adoptent cette stratégie dans une maison pour reconfigurer les services selon les habitudes des nouveaux utilisateurs ainsi que l'état des capteurs existants. (Cremene et al., 2004) représentent les systèmes auto-adaptatifs comme des composants logiciels et définissent ainsi la reconfiguration automatique des services dans la capacité de faire un ajout, une suppression ou un remplacement des composants constituant le système.

Après avoir étudier le principe des formes d'adaptation, on peut conclure que ces derniers dépendent essentiellement de deux catégories de ressources : des ressources utilisateurs et/ou

des ressources systèmes. La figure 1.10 schématise le positionnement des trois concepts d'adaptation.

Nous notons que la recommandation fera un sous-groupe des systèmes de personnalisation dont l'objectif est d'anticiper le choix de l'utilisateur selon des paramètres spécifiques. De plus, ces deux formes d'adaptation se basent essentiellement sur l'environnement utilisateur. Toutefois, la reconfiguration apparait comme une forme isolé des autres formes et qui tient également compte de l'environnement système. Par conséquent, les trois formes d'adaptation coopèrent entre elles pour assurer une auto-adaptation du système entier, dont le but est de produire un service homogène aux attentes des utilisateurs.

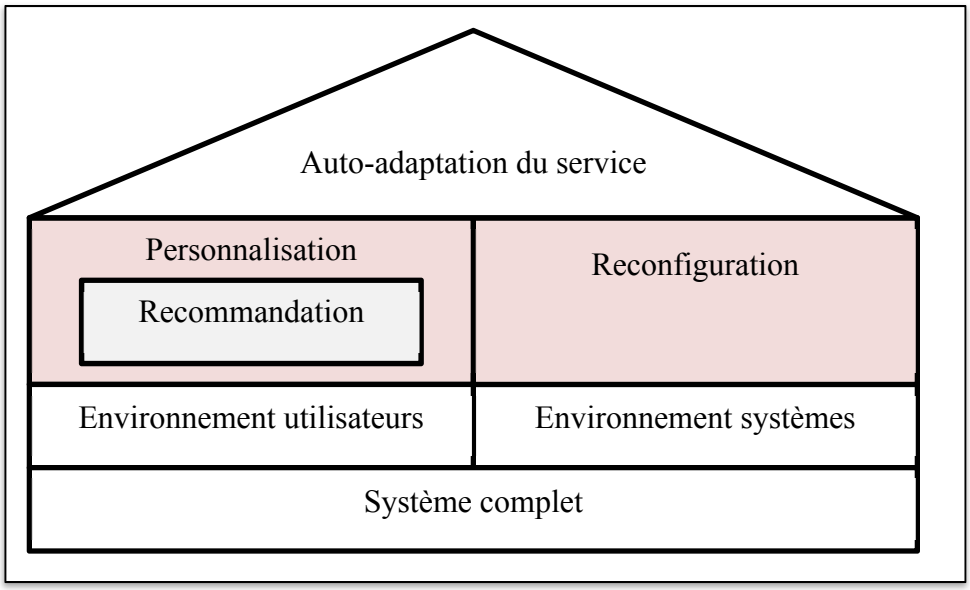


Figure 1.10 Relation entre les formes d'adaptation

1.4.2 Catégories d'adaptation

Les systèmes auto-adaptatifs sont conçus spécialement pour répondre aux vives attentes des utilisateurs au sein d'un environnement intelligent. Ils se caractérisent par la sensibilité au contexte, la capacité de se modifier en fonction du contexte et de s'ajuster automatiquement dépendamment de l'environnement et des besoins des utilisateurs. Les auteurs (Laddaga et

al., 2001), (McKinley , Sadjadi et Kasten, 2004), (Salehie et Tahvildari, 2009), (Lemos, Giese, Müller et Shaw, 2013), (Macías-Escrivá et Haber, 2013) se sont intéressés à la modélisation des systèmes auto-adaptables, de plus ils ont donné lieu à des classifications différentes pour la notion d'adaptation. Généralement, le caractère de l'auto-adaptation peut exister sous quatre formes qui sont : la raison, le temps, le niveau, le contrôle et la technique (Figure 1. 11). Par la suite, nous aborderons chaque forme en détail :

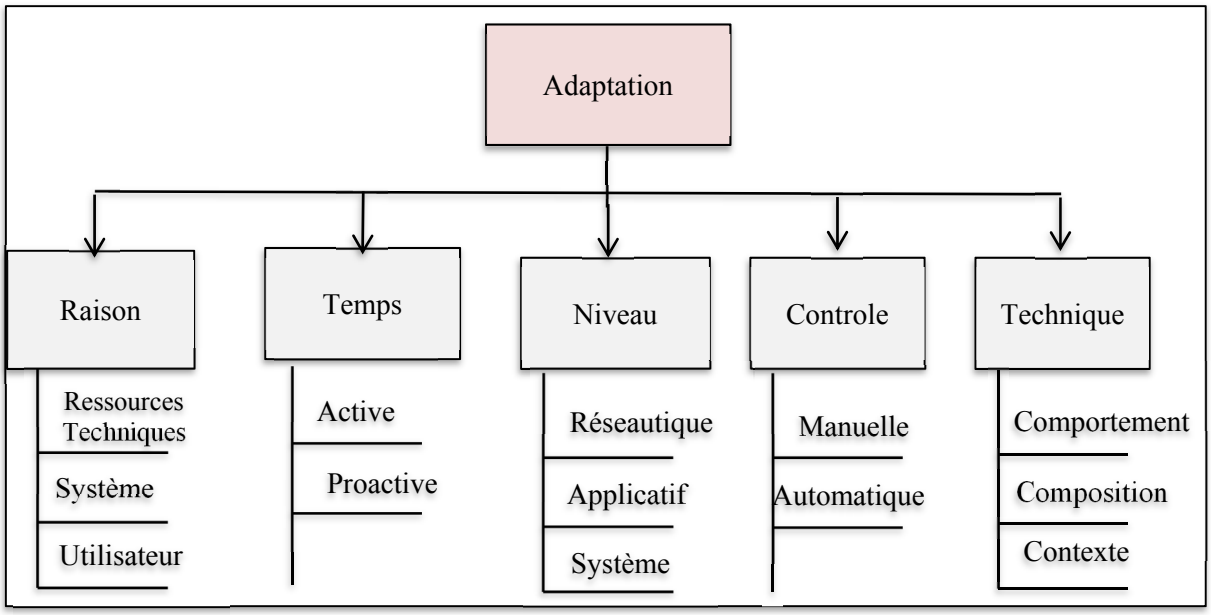


Figure 1.11 Catégories d'adaptation

1.4.2.1 Raison

Globalement, chaque changement entraine une adaptation. Toutefois, choisir la raison du changement auquel devra s'adapter le système reste une question fréquemment posée dans chaque système auto- adaptable. La réponse à cette question permet d'identifier le niveau de modification pour être en mesure de choisir le mécanisme d'adaptation le plus adéquat à la situation. Les raisons majeurs de chaque changement dans un espace intelligent peuvent être de trois niveaux : (1) des changements au niveau des ressources techniques tel que la panne d'un matériel ou la déconnexion d'un réseau, (2) des changements dans l'environnement comme l'apparition d'un nouvel utilisateur ou la variation d'un contexte, (3) des

changements entrainés par l'utilisateur, par exemple la composition de nouvelles préférences ou des nouveaux profils.

1.4.2.2 Temps

La notion du temps est un facteur crucial dans l'adaptation du système. Il sert à désigner le moment où le processus d'adaptation doit se déclencher. Il existe deux façons (basées sur le critère temporel) autorisant ainsi une adaptation de services : réactive et proactive.
La première est une ancienne technique utilisée jusqu'à date dans des systèmes intelligents. Son principe permet de lancer le processus d'adaptation à la fin de l'évènement (action) provoquant un besoin d'une adaptation. La deuxième est une nouvelle technique, beaucoup appréciée dans les environnements intelligents. L'adaptation proactive a pour particularité d'être un processus qui planifie et anticipe l'action à produire, tout en basant, sur des mécanismes d'intelligences artificielles telles que les méthodes d'apprentissages et de prédictions.

1.4.2.3 Niveau

Un système ambiant est une foule d'applications et de systèmes distribués capable d'interagir entre eux via des différents réseaux sans fils. Les niveaux d'adaptation au sein d'un tel système sont de trois. (1) niveau réseautique pour objectif de connecter les différents composants des applications et des systèmes permettant l'interaction multiple pour accomplir leurs tâches. (2) niveau applicatif en modifiant soit la composition des applications, soit le comportement des applications. (3) niveau système, en adaptant les services selon le contexte détecté.

1.4.2.4 Contrôle

L'adaptation d'un système nécessite un contrôle soit manuelle, soit automatique. Le contrôle automatique est réalisé par l'utilisateur. Celui-ci, doit avoir une connaissance sur l'application ainsi que son fonctionnement pour savoir prendre les bonnes décisions. Le

contrôle automatique est effectué par l'application, elle-même, sans l'intervention de l'utilisateur. Cependant, l'application doit se disposer de mécanismes décisionnels capables de se modifier en suivant le changement des situations.

1.4.2.5 Technique

Le dernier paramètre essentiel du processus d'adaptation est défini par la nature technique adaptée par le système. Ce paramètre montre le genre d'ajustement appliqué sur l'ensemble des composants du système. Dans cette dimension, on peut distinguer trois catégories : (1) le comportement, en modifiant la manière dont les composant de l'application seront réalisés. (2) de composition, en modifiant la structure ou les fonctionnalités constituant l'application. (3) de contexte, en changeant le contexte, lui-même, tout en appuyant sur les situations ainsi que les besoins des utilisateurs.

Dans cette thèse, nous nous basons essentiellement sur une adaptation proactive, automatique, fondé sur le comportement de l'utilisateur et le contexte environnemental, pour être en mesure, d'ajuster et modifier un service pour des raisons venant de l'utilisateur et de son entourage.

1.4.3 Processus de raisonnement

Le processus de raisonnement se présente comme étant le cœur d'un système intelligent. Effectivement, il détient la logique sur laquelle est bâtie le raisonnement du système. Cette unité se compose des modules qui réagissent pour prendre des décisions et exécuter par la suite des actions. Les modules participants dans ce processus sont des modules de reconnaissance, d'apprentissage ou des modules de prise de décisions. Toutefois, ce processus est propre à chaque système et peut différer d'une application à une autre, dépendamment des besoins des utilisateurs. La figure 1.12 révèle les principaux processus de raisonnement et d'adaptation qui peuvent exister dans un système auto-adaptable. La suite de cette section, nous présenterons les modules de raisonnement précités en détail :

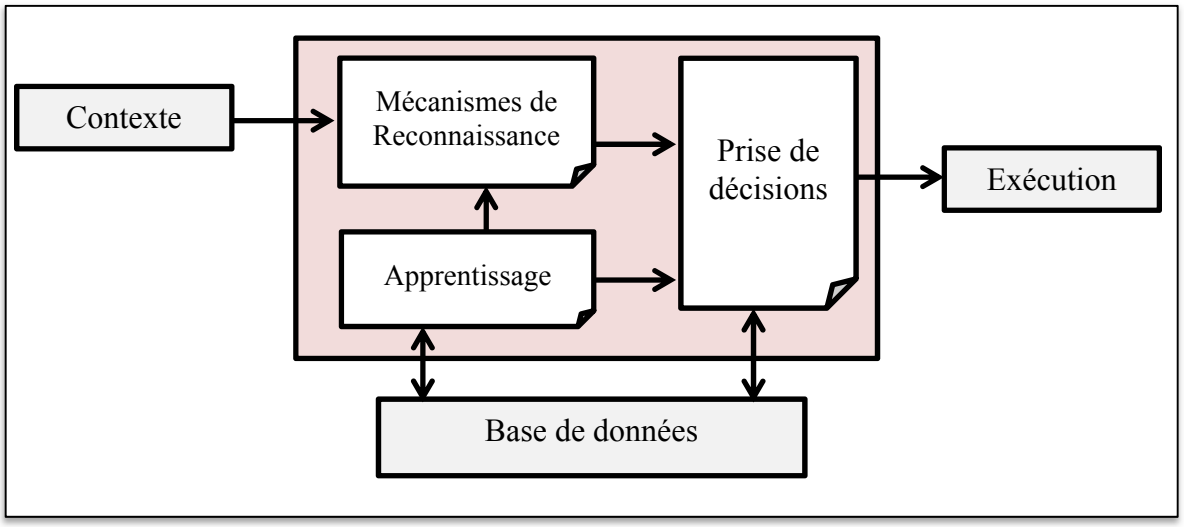


Figure 1.12 Processus de raisonnement

1.4.3.1 Processus de reconnaissance

C'est un processus qui prend en entrée le contexte environnemental, sur lequel, il applique des activités de reconnaissance pour mieux identifier le contexte et le transformer à un niveau plus haut. À la fin du traitement, il obtient la même information reçue en entrée, mais, sous une façon plus abstraite. Cette dernière sera, par la suite, envoyée au processus de prise de décision. Il faut noter que la présence de ce genre de processus n'est pas obligatoire dans chaque système intelligent. Toutefois, son absence causera des complications au niveau de l'identification du contexte et également pour la prise de décision.

1.4.3.2 Processus d'apprentissage

L'intérêt de cette unité est d'assurer l'automatisation du système intelligent dans différentes situations. Dans l'ensemble, elle utilise des algorithmes dédiés pour l'apprentissage qui permettent, dans une certaine mesure, de déduire des réactions apprises par le système qui seront envoyées au processus de prises de décisions. Par ailleurs, ce processus se manifeste par sa capacité de changer de comportement et de recommencer un autre dans le cas des nouvelles situations contextuelles.

1.4.3.3 Processus de prise de décision

La prise de décision est la dernière étape avant l'exécution du service. Elle est formée d'une base d'algorithmes de différents domaines tels que l'intelligence artificielle, l'automatisation, la logique de raisonnement et la théorie d'optimisation qui sont utilisés pour trancher une décision exécutable. En outre, Les décisions prises peuvent se présenter soit de manière courte, soit de manière continue. Cependant, chaque décision prise sera envoyée au processus d'exécution pour être présentée comme une décision finale à exécuter par le système de sorte qu'elle ne peut être révisée et/ou changée par la suite. Dans le cas où l'environnement entourant détecte et signale une erreur, le processus de décision sera rétabli de nouveau.

1.4.4 Mécanismes de raisonnement et d'adaptation de services

La mise en œuvre d'une adaptation de service au sein d'un espace ambiant nécessite des mécanismes spécialisés dans le but de personnaliser, recommander, anticiper, reconfigurer, etc. Le principe général de ces mécanismes est constitué de trois phases essentielles (Figure 1.13) :

Observation : avant toute modification, la phase d'observation identifie la situation causant ainsi, une adaptation et détermine, par la suite, le moment pour déclencher le processus d'adaptation ;

Décision : lorsqu'une adaptation se lance, la décision lui désigne les éléments susceptibles à un changement ;

Action : représente la dernière étape qui vise à exécuter les commandes sélectionnées pour le processus d'adaptation de service.

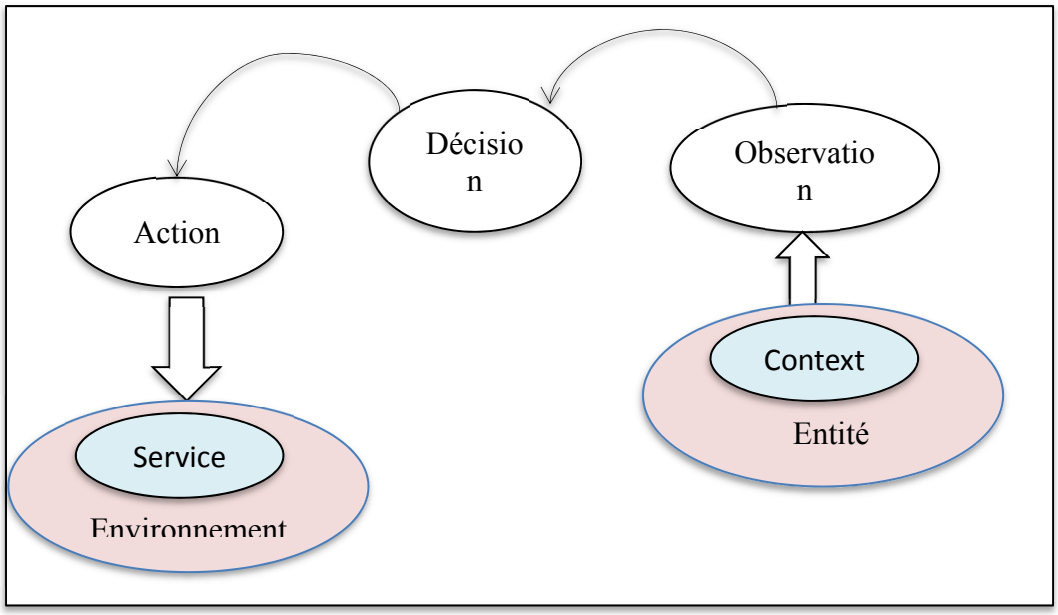


Figure 1.13 Processus général d'une adaptation

Nous présentons maintenant quelques travaux qui ont trait à l'étude de l'adaptation des services dans un espace intelligent :

1.4.4.1 Apprentissage

Les auteurs (Miraoui, Rtimi, Cherif et Tadj, 2014) proposent deux techniques d'apprentissage pour l'adaptation des services sensibles au contexte dans un salon intelligent. Ce mécanisme identifie, en premier, tous les cas possibles des contextes produits dans cet espace. Par la suite, il fournit toutes les possibilités de configurations reliées aux équipements et dépendent du contexte. L'utilisation des méthodes d'apprentissage et de prédiction tels que les réseaux de neurones et les réseaux bayésiens permettent au système d'anticiper la configuration la plus probable du dispositif selon le contexte détecté dans l'environnement.

1.4.4.2 Modules d'intelligence

Le projet réalisé par (Sanchez et Tercero, 2010) porte sur la définition d'une maison intelligente, intégrant ainsi des nouveaux modules d'intelligences réalisant un processus

d'apprentissage et de raisonnement sur un contexte au sein d'une maison. Cette maison intelligente se compose d'un ensemble d'équipements hétérogènes et se caractérise par une complexité dans le système informatisé et une diversité dans le contrôle et la supervision des entités. Par conséquent, les auteurs proposent une coopération entre les entités de telle façon que chaque client envoie des informations caractérisant l'évènement tourné ainsi que sa réponse via des outils spéciaux. Ensuite, l'ensemble des caractéristiques reçues sera analysé et traité par des mécanismes d'apprentissage et de raisonnement pour être servis au futur.

1.4.4.3 Services auto-adaptables

Les auteurs (Salehie et Tahvildari, 2009) proposent un système adaptable basé sur un service qui s'adapte tout seul sans l'intervention humaine. Plus précisément, le service doit être en mesure de s'ajuster selon son contexte opérationnel ainsi que les conditions des ressources offrant ce service.

Ce phénomène d'adaptation est réalisé par une politique de gestion permettant ainsi de définir de façon sémantique un ensemble de règles et de comportement pour chaque service offert. Cet ensemble de règles est capable de préciser ,à la fois, les situations où le service sera considéré comme sensible au contexte, contrôlé par des ressources ou utilisé par d'autres services.

1.4.4.4 Apprentissage et forage de données

Les auteurs (Frey et al., 2013) s'intéressent à une adaptation basée sur une combinaison entre des méthodes d'apprentissage et de forage de données. De tels mécanismes font l'objet d'apprentissage et de mémorisation des interactions habituelles des utilisateurs avec les différents équipements de la maison. Ainsi, ses méthodes peuvent être utilisées pour prédire de nouvelles réactions quotidiennes de l'utilisateur, de reconnaitre des modèles périodiques de comportement, et de différencier entre des évènements normaux et anormaux en proposant une adaptation conforme au système. En plus, les auteurs ont pensé implémenter plusieurs algorithmes d'apprentissages (réseaux de neurones artificiels, règles de logique

flou, techniques de classification, etc.) pour tirer, le plus de cas, de reconnaissances et de prédiction.

1.4.4.5 Logique floue

Les auteurs dans (Vainio, Valtonen et Vanhala, 2008) développent un système pour une maison intelligente, assurant à la fois une adaptation proactive avec un contrôle basé sur la logique flou. Le principe de cette approche se fond sur les contextes ainsi que les actions déroulées par les habitants de cette maison. Le processus d'apprentissage utilisé sauvegarde toute information contextuelle qui sera ensuite divisée en sous groupe en implémentant les méthodes de la logique flou, de telle façon que, chaque sous groupe soit référencé à un détecteur, actionneur avec une mesure temporelle. De plus, le système utilise des règles de bases et en déduit d'autres, via le processus de la logique utilisé. Des mises à jour sont également sur l'ensemble de données et de règles pour la cohérence du système.

1.4.4.6 Apprentissage par renforcement

Les auteurs (Boytsov et Zaslavsky, 2010) dans cet article s'intéressent à implémenter une adaptation proactive basée sur un apprentissage par renforcement dans un environnement intelligent sensible au contexte. Ils utilisent comme point fort le principe de la théorie de l'espace contextuel. Ce principe a pour objectif d'offrir une représentation spatiale multi-dimension pour modéliser des contextes sensibles et de traiter également des situations de raisonnement en focalisant sur le problème d'incertitude et de manque de fiabilité des capteurs. La nouveauté dans ce modèle est la possibilité d'intégrer un processus d'adaptation proactif. Pour ce faire, les auteurs ajoutent la modélisation des actionneurs dans le modèle comme étant un espace isolé des capteurs qui sera ensuite une source pour réaliser une adaptation à travers l'apprentissage renforcé.

Dans notre thèse, nous optons pour une adaptation par apprentissage. Toutefois, notre approche est couplée d'un apprentissage par renforcement et d'un autre supervisé, afin d'assurer une meilleure adaptation.

1.5 Architecture des systèmes adaptatifs

La conception d'un système intelligent dépend de plusieurs critères liés à la condition de l'environnement, le type et la localisation des capteurs, le nombre d'utilisateur et la nature des dispositifs utilisés. Tous ces facteurs précités jouent un rôle important pour implémenter un système performant capable de relier entre l'espace physique et les applications pour fournir des services adaptables sensibles aux contextes. Jusqu'à date, plusieurs systèmes (Framework, middlewares ou architectures) ont été proposés sous différentes contraintes et architectures pour un seul but : répondre aux besoins des utilisateurs en fournissant des services conformes aux demandes utilisateurs d'une manière transparente.

Les systèmes adaptatifs sont abordés par plusieurs auteurs sous différentes définitions :
(Burn et al, 2009) défini un système auto-adaptable s'il possède la capacité de prendre des décisions autonomes pour changer son comportement et s'adapter par la suite conformément au contexte. Les auteurs dans (Weyns et al., 2012) le représentent comme étant des boucles fermées, qui se changent de façon continue en suivant leurs feed-back. Les auteurs Salehie et Tahvildari (Salehie et Tahvildari, 2011) adoptent un mécanisme d'évaluation de comportement effectué sur le système auto-adaptable ayant pour objectif de montrer l'aptitude d'accomplir ses tâches et de les comparer en se basant sur les critères de performances.

Dans cette section, nous passons en revue les différentes architectures des systèmes adaptatifs, pertinents dans le monde des systèmes intelligents. Ces modèles sont réalisés suivant certains aspects en relation directe avec la perception du contexte, l'adaptation et le déclenchement du service.

1.5.1 MavHome

MavHome est un projet réalisé par (Cook, 2003), autour d'une maison avec un objectif de produire un environnement intelligent basé sur des agents rationnels. L'intérêt de cette recherche est de créer une maison de confort pour ses habitants avec un minimum

d'opérations réalisées. Les agents imbriqués jouent un rôle crucial dans l'implémentation du système. En effet, Ils sont responsables à choisir les actions à entreprendre tout en maximisant le degré de performance (Figure 1.14).

L'architecture de MavHome se base essentiellement sur l'emploi d'agents répartis en quatre couches :

Couche physique : la première couche qui reflète tout ce qui est matériel dans la maison.

Couche de communication : facilite et assure la communication des requêtes entre les agents.

Couche information : cette couche stocke toute les connaissances importantes pour la prise des décisions telle les mises à jour ou les prédictions.

Couche décision : c'est la couche supérieure. Elle est basée sur l'information provenant de la couche inférieure. Cette couche est responsable dans la sélection et l'exécution des actions par les agents.

1.5.2 COBRA Project

COBRA (Context Broker Architecture) est une architecture développée par (Chen, 2004) pour supporter des systèmes sensibles au contexte dans un espace intelligent. Cette architecture se base principalement sur un agent intelligent nommé par *Context Broker* (CB) qui est une entité centrale conservant un model contextuel responsable dans le partage et la protection des données entre toutes les entités existantes dans l'espace intelligent. La conception d'un CB se fond sur quatre fonctionnalités essentielles (Figure 1.15):

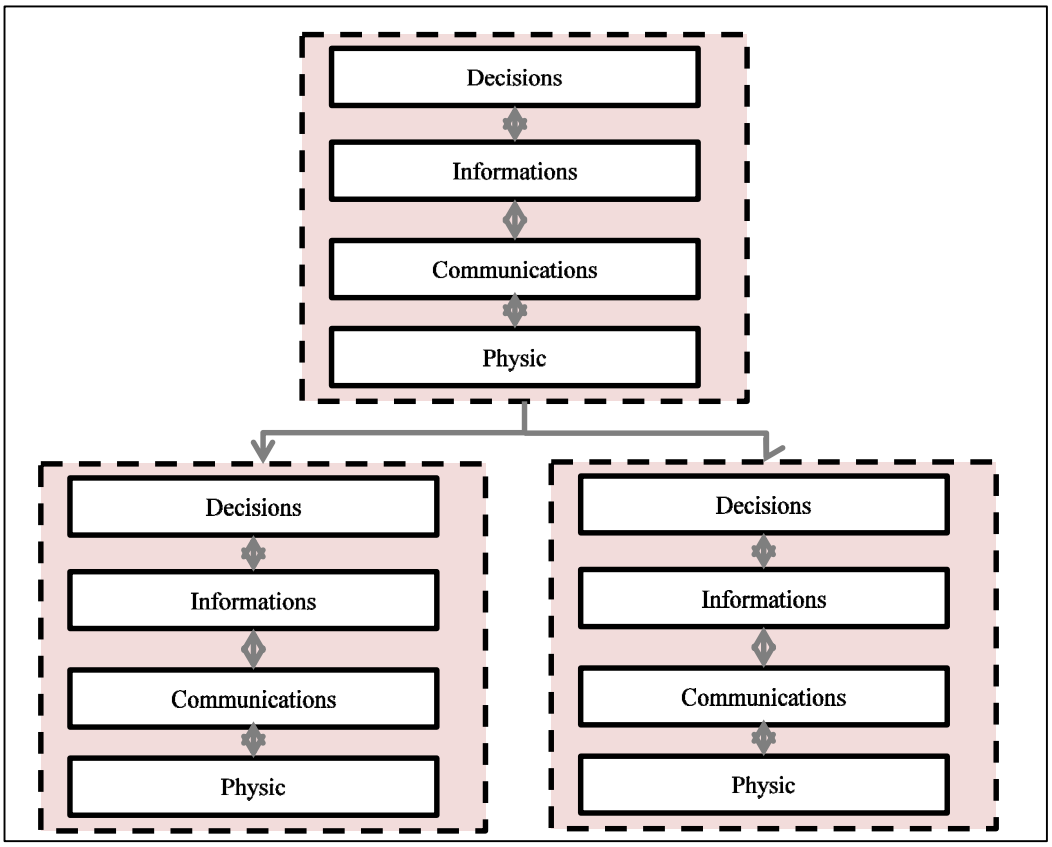


Figure 1.14 Architecture de MavHome
Tirée de Cook (2003)

- La base d'information du contexte : c'est un composant basé sur des ontologies, et désigné spécialement pour la gestion et le stockage des données contextuelles. En plus, il fournit un ensemble d'API permettant l'accès aux contextes ;
- Le module de raisonnement basé sur des ontologies et munie d'une logique de raisonnement capable de raisonner sur des données contextuelles, ou d'inférer de nouvelles règles utilisables dans le système ;
- Le module d'acquisition composé de procédures et de fonctions responsables dans l'acquisition du contexte ;
- Le module de gestion du contexte se compose d'un ensemble d'inférences et de règles permettant de prendre des décisions concernant le partage du contexte entre les différentes entités.

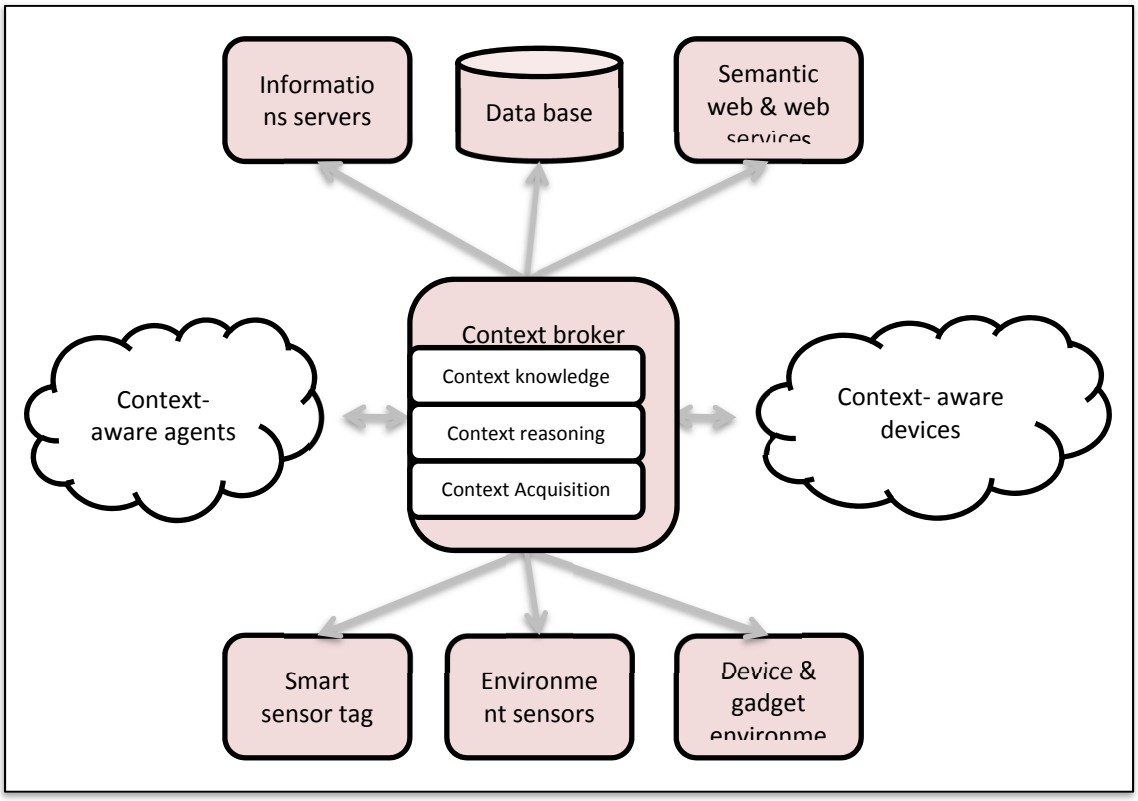


Figure 1.15 Architecture de COBRA
Tirée de Chen (2004)

1.5.3 SOCAM Project

Service Oriented Context- Aware Middleware (SOCAM) est un Middleware développé par (Gu, 2005) pour la conception d'un système sensible au contexte basé sur des services au sein d'un environnement ubiquitaire. Cette infrastructure est capable de transférer des données physiques captées vers des données sémantiques accédées et partagées par des services sensibles au contexte (Figure 1.16).

L'architecture de SOCAM se repose sur cinq éléments fondamentaux :

- Un fournisseur du contexte pour acquérir les données contextuelles auprès des capteurs physiques et les transférer, par la suite, en données OWL pour être utilisées par d'autres composants ;
- Un interpréteur du contexte fournissant des services de raisonnement logique pour le traitement des données contextuelles ;

- Une base de données contextuelles spécialement conçue pour rassembler les ontologies du contexte ainsi que l'historique contextuel ;
- Les services sensibles au contexte représentent les applications et les services obtenus après un traitement contextuel. Ces services se caractérisent par leurs capacités de s'adapter dans un espace dynamique.

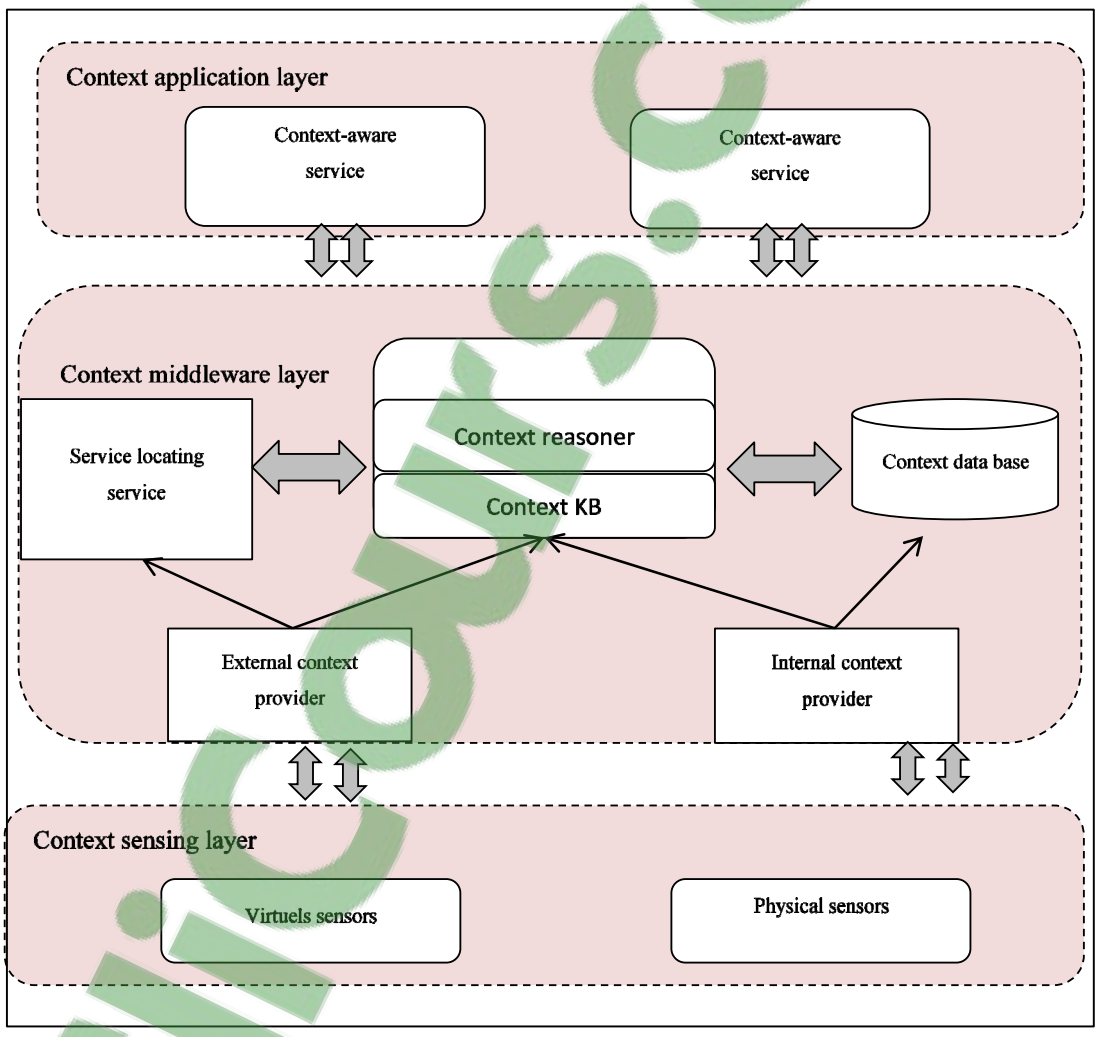


Figure 1.16 Architecture de SOCAM
Tirée de Gu (2005)

1.5.4 GTSH Project

Gator Tech Smart House (GTSH) est un projet issu par (Helal, 2005) ayant pour un but de produire un espace intelligent programmable et générique, capable d'évoluer et d'intégrer d'ailleurs de nouveaux composants avec une assurance de services. Le projet est réalisé sur une maison qui réagit, à la fois, comme un espace exécutable évolutif muni de capteurs, d'actionneurs, et de différents dispositifs intelligents. Cet espace maintient également une librairie de logiciels contenant un middleware général, des protocoles de communications et des processus pour la réception, le traitement et l'exécution de services.

L'unité centrale de cette maison réside dans la conception du middleware qui assure la coordination entre les différentes entités. Ce projet est implémenté en six couches (Figure 1.17):

Couche physique : c'est la couche qui comporte tous les équipements, que ce soit des dispositifs de maison (TV, lampes, frigidaires, sonnettes), des capteurs ou actionneurs (air climatisé, thermostat de chauffage, système de sécurité) ou même de nouvelles technologies importantes pour cet espace (systèmes de télé-déverrouillage) ;

Couche et plateforme de capteurs : cette couche est responsable à communiquer tout ce qui est reçu par la couche précédente auprès des différents équipements à la couche suivante. C'est à dire, cette plateforme convertit et présente les éléments de la couche physique en services qui peuvent être programmés et manipulés par d'autres couches ;

Couche de service : le Framework OSGI est le centre de cette couche. Il permet en premier ordre de recevoir les paquetages des services offerts par la couche précédente puis de les définir pour être utilisés par d'autres couches. Cette couche peut également englober des activités de reconnaissances de voix, de vidéos ou d'ordonnancement ;

Couche de données : elle est composée d'ontologies permettant de gérer les données des services pour fournir un support sémantique de données et de mécanismes de raisonnement qui peut être appliquer sur l'ensemble de contexte ;

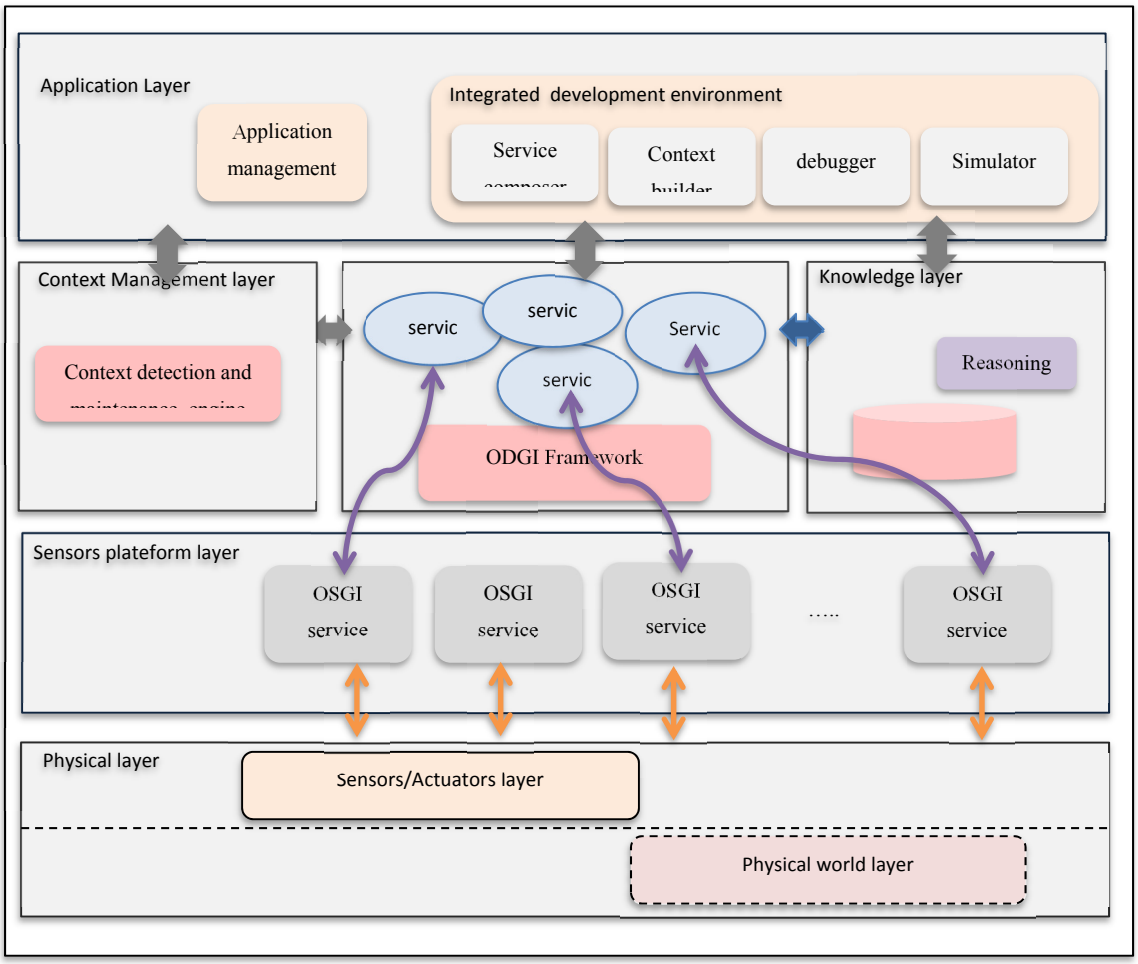


Figure 1.17 Architecture du middleware GTSH
Tirée de Helal (2005)

Couche de gestion de contexte : elle est responsable du contexte de façon générale. Plus particulièrement, elle représente le contexte sous forme de graphe composé d'un ensemble de paquetages révélant l'ensemble des équipements impliqués. Elle permet davantage d'activer des services qui seront utilisés par d'autres applications ;

Couche application : la dernière couche s'occupe à gérer les applications pour être capable d'activer ou annuler l'exécution des services. Elle se dispose également des outils permettant la création et la coopération entre l'ensemble d'entités de cet espace.

1.5.5 SmartLab Project

Le projet SmartLab est une plateforme développée par (Lopez-de Ipina, 2008) ayant pour but d'implémenter un sémantique middleware intégrant la détection, le raisonnement et l'exécution d'information. Il est capable, également, d'assurer la gestion et le contrôle d'un environnement intelligent indépendamment du domaine appliqué. D'abord, il se base sur le standard OSGI qui est une plateforme de services, fondé sur le langage java et formé d'un Framework fournissant, à la fois, une gestion dynamique de l'application, un répertoire de services et un environnement d'exécution pour les modules.

L'architecture de SmartLab (Figure 1.18) est formée de quatre couches facilitant le déploiement et la configuration d'un environnement intelligent :

Première couche : c'est une couche de détection et de déclenchement qui rassemble les différents dispositifs qui peuvent exister au sein de cet espace. Elle est équipée d'un bus d'automatisation, des équipements intelligents (afficheurs, conteneurs et chaises) et peut également employer la VoIP et VideoIP ;

Deuxième couche : c'est une couche d'abstraction de services permettant de transformer les fonctionnalités des dispositifs reçues de la première couche en services. Elle se compose du standard OSGI qui implémente le mécanisme d'empaquetage pour chaque dispositif comme des archives *.jar*. Alors, chaque type de dispositif sera encapsulé dans un paquet ou unité d'exécution pour être utilisé par la couche suivante ;

Troisième couche : nommée par le serveur SmartLab, elle se comporte comme un bus entre les services avancés d'OSGI et les fonctionnalités des équipements déployés dans l'environnement dont le but est la modélisation sémantique du contexte et la gestion des services ;

Quatrième couche : la couche supérieure de l'architecture représente les applications utilisées par les dispositifs de cet espace. Elle est composée essentiellement d'un contrôleur d'environnement muni d'une interface basée sur le concept de gadget web qui est accessible par les clients. De plus, un frontal se forme d'une interface Web permettant la gestion sémantique du contexte et des services empaquetés et la modification des bases de données pour le mécanisme de raisonnement.

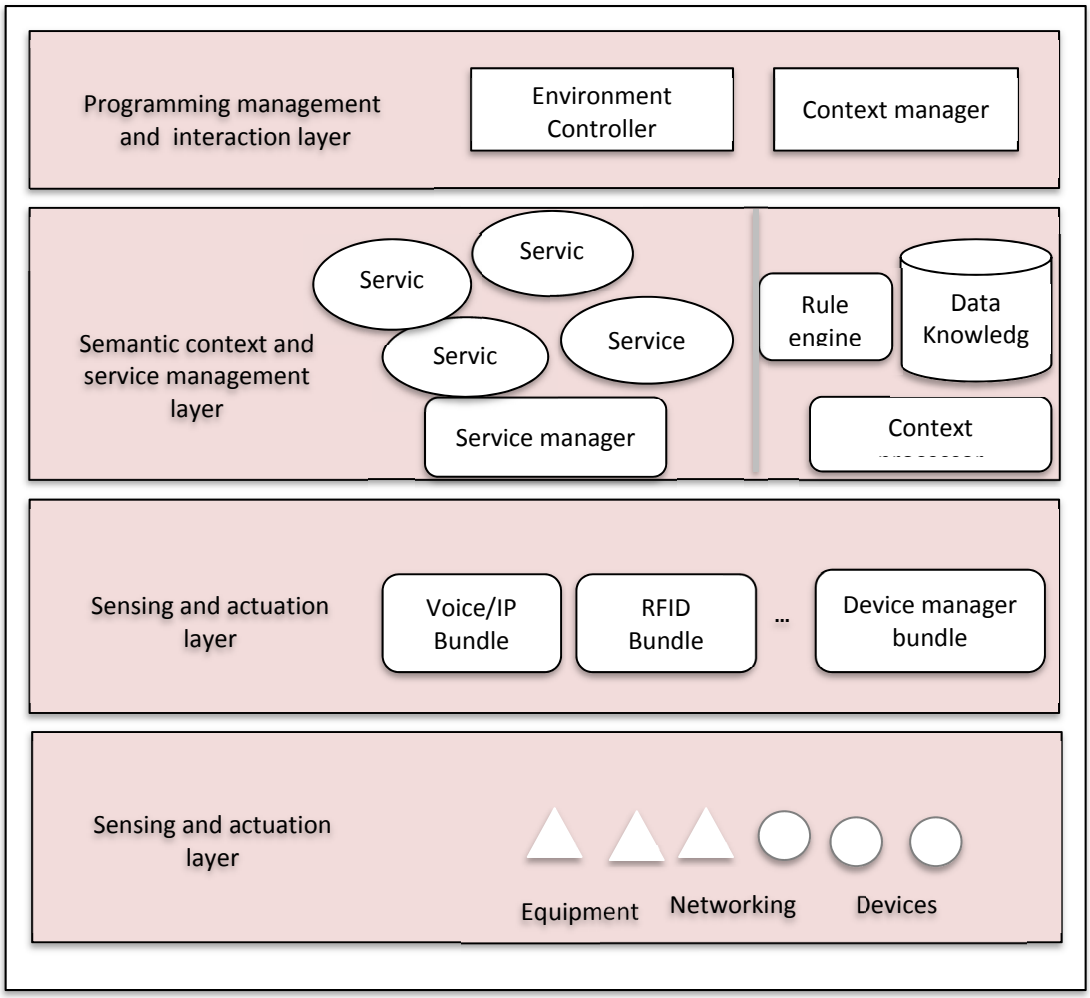


Figure 1.18 Architecture de SmartLab
Tirée de Lopez-de-Ipina (2008)

1.5.6 DSOA Project

DSOA ou (Device Service Oriented Architecture) est une approche développée par (Buzeto, 2010) pour modéliser un espace intelligent caractérisé par différents dispositifs et applications hétérogènes. En réalité, DSOA est une extension du SOA, sur lequel il offre en plus de nouvelles fonctionnalités pour la gestion d'un espace intelligent tel que la distinction entre les interactions synchrone et asynchrone. Cette stratégie définit quatre points essentiels : (1) l'espace intelligent est formé de deux dispositifs au minimum ayant la possibilité de communiquer et d'échanger de l'information avec les autres pour produire des services aux utilisateurs de façon transparente. (2) les ressources se présentent comme un groupe de fonctionnalités logiques (ex : positionnement de l'utilisateur) ou physiques (ex :

clavier) prédéfinis par des interfaces et accessibles par l'environnement. (3) les services sont responsables à exécuter les différentes fonctionnalités dans l'espace intelligent via des interfaces déjà connues par les ressources (Figure 1.19).

DSOA définit trois rôles essentiels accordés aux dispositifs :

Consommateur : c'est l'entité qui a l'accès au service via les ressources ou les applications ;

Fournisseur : c'est l'entité qui est munie d'une interface permettant l'accès au service ;

Enregistreur : c'est l'entité qui assure le suivi des informations des ressources et des services. Afin d'assurer une communication fiable dans cet espace, DSOA, définit trois méthodes d'échanges entre les entités qui sont :

Le transport de données : c'est la stratégie qui favorise l'échange de données entre un fournisseur ou un consommateur de service ;

L'interaction entre dispositifs : elle peut être effectuée de façon synchrone si le consommateur demande un service et le fournisseur lui répond par une réponse à la demande. Sinon, l'interaction sera asynchrone, lorsqu'il n'y aura pas une réponse directe à la demande ;

Le protocole uP : Ubiquitious Protocol ou (uP) consiste en un ensemble de protocoles fournissant des interfaces et favorisant la communication entre les différentes entités dans un tel espace ubiquitaire.

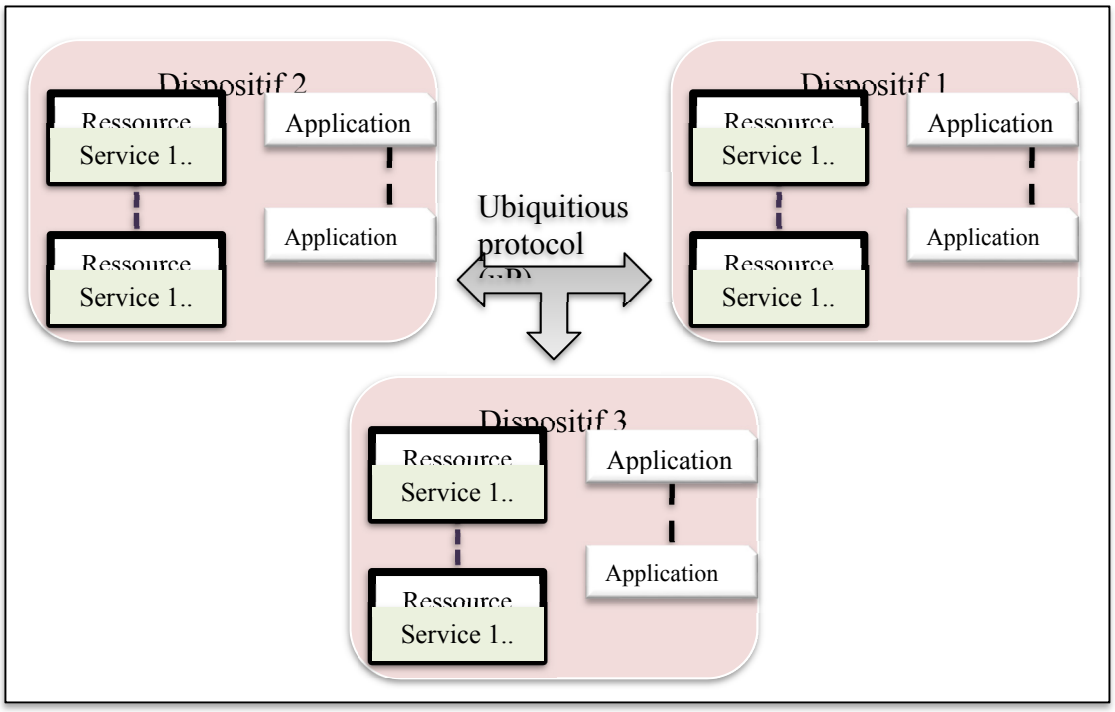


Figure 1.19 Architecture de DSOA
Tirée de Buzeto (2010)

1.5.7 Smart-M3

Smart-M3 est une plateforme NOKIA, multi domaine, multi dispositif, développée par (Honkola, 2010) pour fournir une implémentation particulière supportant l'interopérabilité des dispositifs dans un espace intelligent. Elle se base sur des mécanismes de communications capables de partager l'information entre des entités hétérogènes à travers des protocoles d'accès standardisés dont le but est de produire des services adaptables sensibles au contexte. De manière générale, cette architecture se résume en trois entités (Figure 1.20) :

- L'entité SIB ou (Semantic Information Broker) est l'élément central de l'architecture Smart-M3. Effectivement, il est responsable de toute fonction de stockage, de partage ou de gestion d'informations présente dans l'espace intelligent par le biais du protocole SSAP (Smart Space Acces Protocol). Un tel espace peut contenir un seul SIB ou plusieurs qui peuvent se communiquer entre eux en présentant la même information aux dispositifs ;

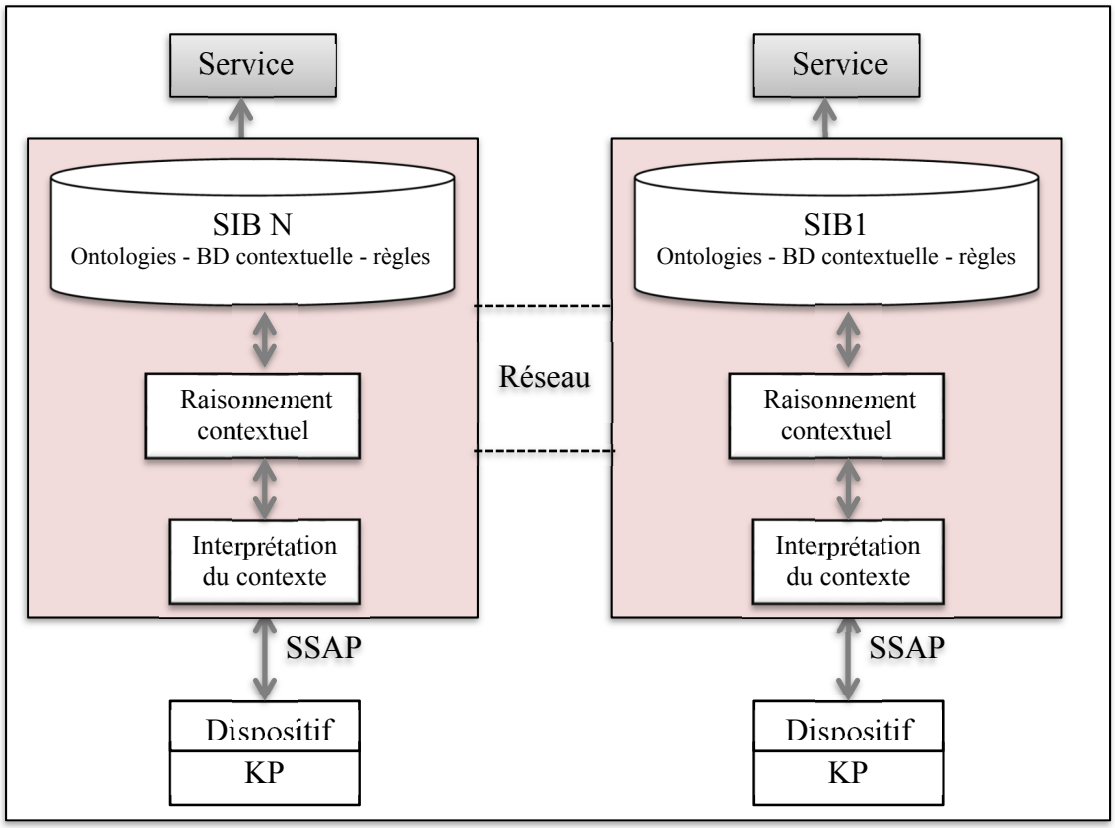


Figure 1.20 Architecture de smart-M3
Tirée de Honkola (2010)

- L'entité KP ou (Knowledge Processors) est une entité produite afin de communiquer avec le SIB, soit, pour lire des informations contenant dans le SIB, soit, pour publier des informations dans le SIB. Notant qu'un dispositif peut contenir une entité KP ou plusieurs de différentes fonctions, de façon que, chacune est responsable d'une seule tâche spécifique. Par conséquent, les entités KP ne peuvent pas communiquer entre eux, c'est à dire que, la transmission de données entre les dispositifs n'est pas supportée. Par contre, ils doivent passer par le noyau SIB qui a la possibilité de communiquer avec les KP ;
- Des ontologies basées sur le langage OWL sont utilisées par les entités SIB pour représenter des informations contextuelles captées après un processus de raisonnement. Les ontologies jouent un rôle important à favoriser aux KPs l'accès et le traitement des informations accordées à leurs types de fonctionnalités dans l'espace intelligent.

1.5.8 CLM middleware

Le middleware CLM (CASAS Lightweight Middleware) est implémenté par (Kusnir, 2010) dans un environnement intelligent nommé par CASAS (Centre for Advantace Stadies in Adaptative Systems). Le but crucial de ce projet et de désigner un middleware léger, rapide, flexible et extensible sur un espace intelligent (Figure 1.21). Cette approche repose sur le mécanisme de publication/souscription de messages. L'ensemble des composons est divisé en modules nommés par 'agents'. Ce genre d'agents peut jouer le rôle d'un publicateur ou un souscripteur ou les deux rôles à la fois.

Un environnement CASAS a pour but d'implémenter un espace intelligent formé d'un ensemble d'agents intelligents qui interagissent comme des fournisseurs de données. C'est le cas des capteurs qui détectent et captent l'information de l'extérieur ou des actionneurs qui signalent l'exécution des décisions prises.

Devant ces besoins, le middleware CLM utilise le langage XML pour l'échange de données entre les différentes entités. Bien évident, XML est un langage de balises utilisé pour définir un format spécifique de données échangées via l'internet. Il se caractérise par sa généralité, sa simplicité et sa convivialité à travers l'internet. Donc, chaque composant de l'architecture utilise le format XML pour exprimer leurs évènements. De plus le middleware CLM définit pour chaque unité participante dans l'architecture un identificateur unique (ID), une variable timestamp pour l'ordonnancement des messages effectués et une variable de localisation indiquant l'emplacement physique de l'entité.

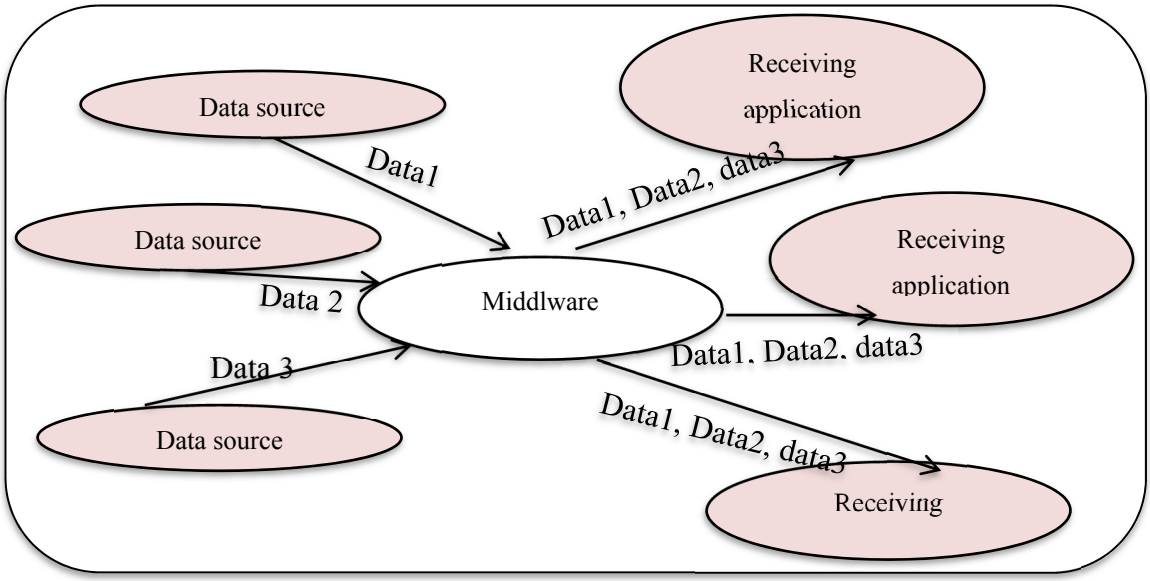


Figure 1.21 Architecture de CLM
Tirée de Kusnir (2010)

En conséquence, l'échange de données est effectué sous formes d'instances de messageries via le protocole XMPP (eXtensible Messaging and Presence Protocol) qui présente des mécanismes de sécurité, des messages échangés à travers la protection mot de passe et le protocole SSL. Cependant, l'implémentation de CLM se focalise principalement sur un gestionnaire de commandes XML et de messages de données. Ce dernier maintient une liste d'entités réceptrices de différents canaux. Alors, il reçoit les messages envoyés par des entités publicatrices sous forme de message XMPP, ensuite, il s'occupe de les renvoyer aux souscripteurs via leurs canaux prédéfinis.

1.5.9 Framework S2CAS

S2CAS ou (Smart Space Context Aware System) est un Framework développé par (chen et al., 2011) et orienté vers la gestion des informations contextuelles et la production des services dans un espace intelligent. La conception de ce système s'articule autour de trois couches majeures (Figure 1.22) :

La couche initiale: représente la première couche qui reçoit les informations contextuelles de l'extérieur. Elle est dotée de mécanismes spéciaux pour la réception et le prétraitement des données captées (logique ou physique) pour les transmettre vers la couche suivante ;

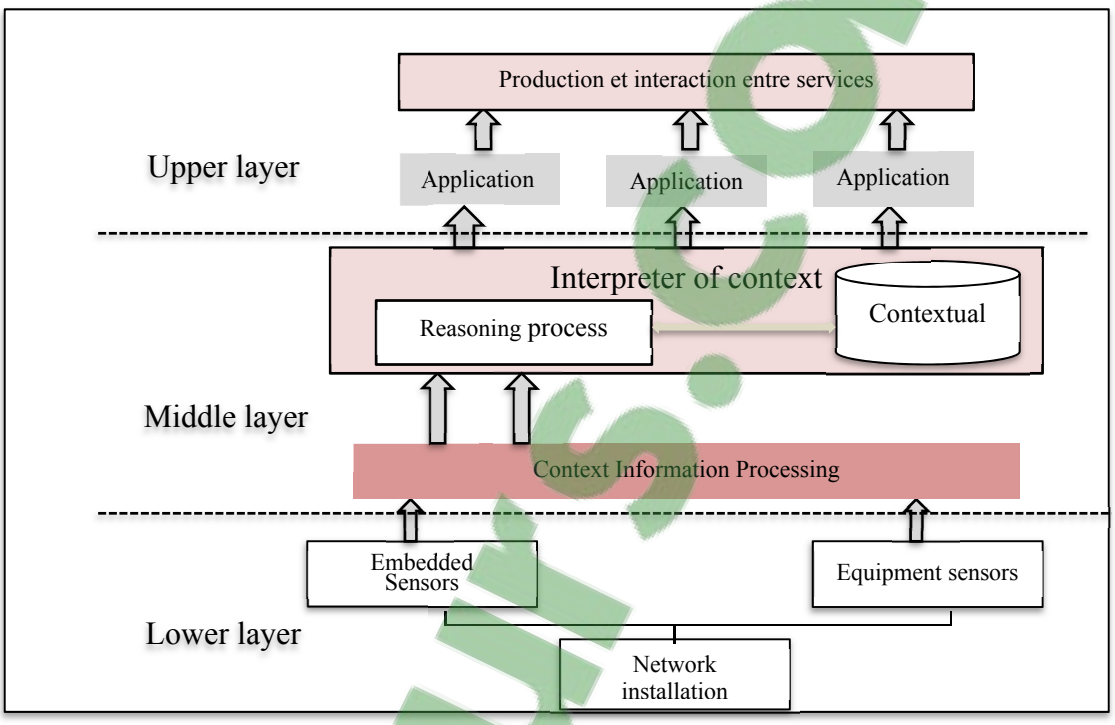


Figure 1.22 Architecture de S2CAS
Tirée de Chen (2011)

La couche intermédiaire : cette couche définit le noyau de ce système du fait qu'elle maintient le processus du traitement, de l'explication et de la gestion des données contextuelles arrivées de la couche inférieure. Le module de traitement du contexte reçoit les données de la couche initiale et les transforme en données plus formelles. Ensuite, le module d'interprétation, composé d'une base de connaissances textuelles et de mécanismes de raisonnement, permet, ainsi, le transfert des données reçues de bas niveau à un format de haut niveau avec un enregistrement historique du contexte ;

La couche supérieure : c'est la dernière couche qui se définit comme interface entre l'utilisateur et sa machine. Cette dernière reçoit les informations du processus d'interprétation de la couche intermédiaire et fournie, par la suite, des services sensibles en se basant sur des règles d'inférences.

1.5.10 GrennerBuildings

GreenerBuildings est un projet basé sur l'approche orientée services (SOA), développé par (Degeler, 2013) et appliqué sur un système de gestion de bâtiments (Building Management System BMS). Le but crucial de ce projet est de construire un bâtiment intelligent (automatisé et adaptable) capable de réaliser des économies énergétiques selon les besoins des utilisateurs. L'implémentation la plus connue de SOA est réalisée avec un bus (middleware) de services échangés entre les dispositifs fournissant l'information (capteurs) et les dispositifs implémentant l'information (actionneurs) (Figure 1.23).

L'architecture d'un tel bâtiment se fonde sur trois couches essentielles :

Couche physique : c'est la première couche de l'architecture. Elle est responsable de la gestion des dispositifs existants dans le système qui sont de différents types: des plugins pour la mesure d'énergie, des contrôleurs KNX, des systèmes de chauffage, d'éclairage ou de température. L'ensemble est interconnecté via un bus qui a pour mission de collecter toute information provenant de l'extérieur et de la transformer en format adapté au traitement de la couche supérieure ;

Couche ubiquitaire : globalement, cette couche assure le bon fonctionnement du système. Elle se compose d'un répertoire de données contenant toutes les informations concernant les dispositifs, entre autres, la configuration et la consommation d'énergie. Le composant contexte est également transformé à un niveau plus élevé de connaissance pour être interprété et mieux géré ;

Couche de composition : c'est la couche supérieure. Elle est responsable dans le contrôle des services et l'exposition des actions à exécutées par l'utilisateur à travers les interfaces manipulées. Cette couche est munie, également, d'un mécanisme de raisonnement et de maintenance capable de prendre des décisions selon les préférences des clients et les accomplir.

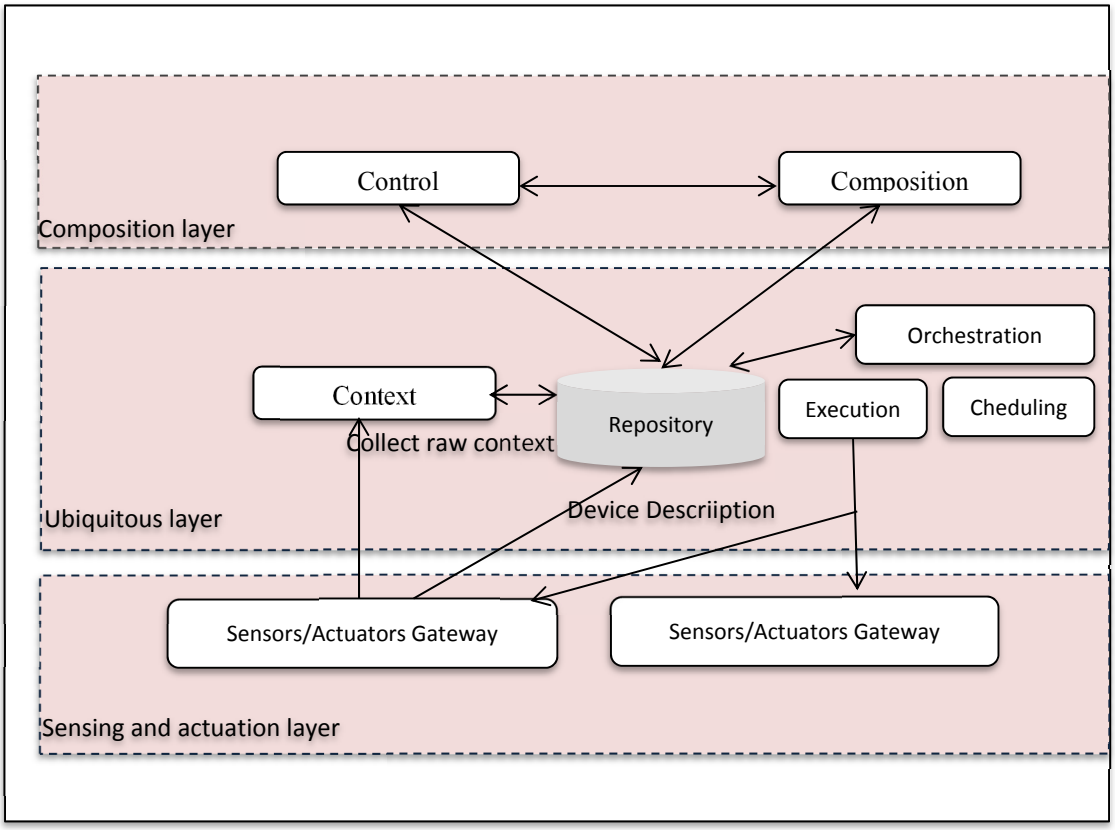


Figure 1.23 Architecture du GreenerBuilding
Tirée de Degeler (2003)

1.5.11 Système auto-adaptable

Cette architecture est développée par (Chun, 2013) dont le principe est de créer des dispositifs adaptables au milieu d'un espace intelligent basé sur des agents, de telle façon que, chaque dispositif comprend un agent adaptable. L'ensemble des dispositifs se coopère pour produire des services de manière transparente à l'utilisateur. L'architecture de ce système est composée d'un logiciel adaptable et des agents d'adaptation. Le logiciel adaptable est formé du thread SAManager et de logiciels de procédures d'apprentissages. Le SAManager est composé d'une liste prédéfinie des contraintes des états normaux du système et une liste de fonctions à exécuter. Les agents d'adaptation sont formés d'une table de statut, une table des défauts et une autre pour les stratégies d'adaptation. Dans la suite, on présente le rôle de chaque composant dans ce système auto-adaptable (Figure 1.24) :

- Le logiciel de procédures d'apprentissage est en relation avec le SAManager. Il se compose d'un ensemble de programmes capables d'exécuter les requêtes demandées par le SAManager comme une invocation, soit pour la création des listes de contraintes ou de conditions, soit, pour l'ajout et la suppression des nouvelles contraintes ;
- Le SAManager est le composant central du système. Du côté des agents d'adaptation, il effectue les demandes de trouver le composant ou la condition reliée à la liste des contraintes et de les sauvegarder après la réception. De l'autre côté, vers le logiciel de procédures, il déclenche le commencement des fonctions à exécuter. Lors des anomalies, il s'occupe de les notifier aux agents d'adaptation ;

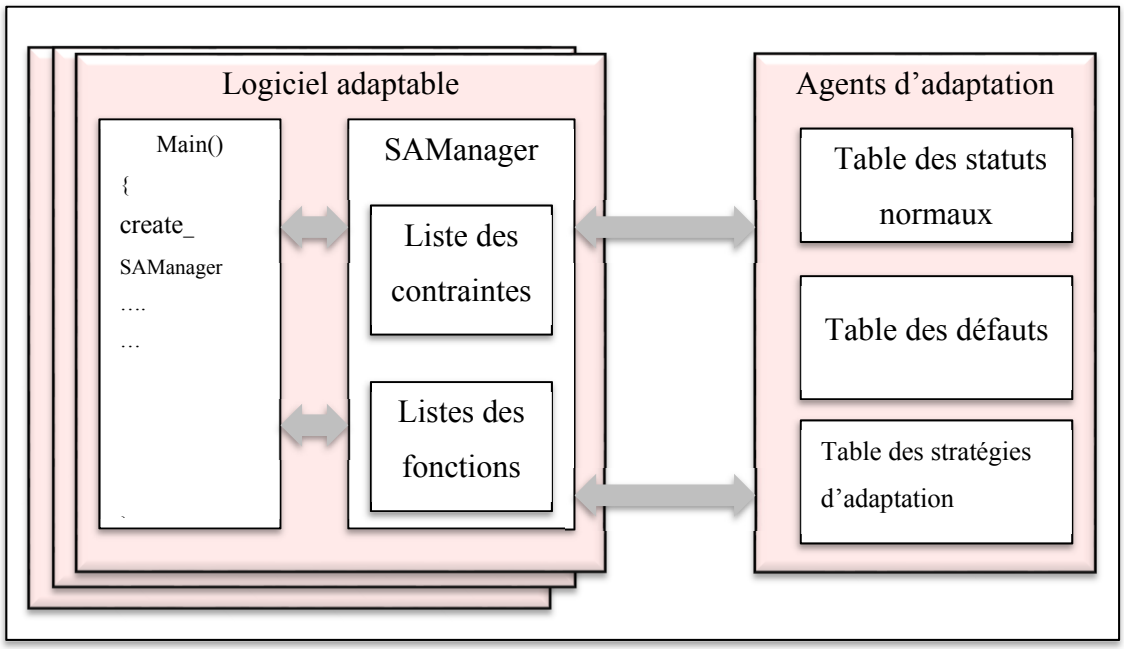


Figure 1.24 Architecture d'un système auto adaptable
Tirée de Chun (2013)

- Les agents d'adaptation sont en relation avec le SAManager dont la fonction est de répondre à ses demandes de recherche, que ce soit des contraintes ou des conditions sinon des stratégies d'adaptation. Les agents d'adaptation reçoivent aussi différents types d'informations de l'extérieur via des capteurs.

Après avoir détaillé l'architecture et le fonctionnement de quelques systèmes pertinents, nous allons analyser en détail les points de similarités entre eux. Nous constatons alors que

l'architecture de chaque modèle diffère de l'autre, et peut comporter différents mécanismes, sauf que, elles fonctionnent pour le même but, celui de répondre aux exigences de l'environnement intelligent dédié. Entre autres, nous avons identifié trois points essentiels à savoir : les modules de l'architecture, contexte et modules de raisonnement. Ces points constituent ainsi la référence pour pouvoir distinguer entre les différents systèmes en termes d'adéquation pour un environnement intelligent.

Modules d'architecture

L'architecture d'un espace intelligent définit la façon dont les modules sont disposés et connectés pour la coordination des tâches réalisées. Notre analyse se base sur l'évaluation de la cohésion et le couplage des modules architecturaux pour chacun des systèmes présentés précédemment.

Cependant, la cohésion est un principe de conception révélant le degré de dépendance entre les différents modules constituant l'ensemble de l'architecture. Entre autre, le couplage permet d'identifier le niveau d'interaction et l'échange entre ces modules. Généralement, ils sont déployés respectivement comme des métriques de mesure, en particulier, pour les systèmes informatiques et peuvent varier entre fortes ou faibles mesures.

Contexte

Le contexte représente la source d'information sur lequel se base le système. Il définit toute donnée (situation) circulant dans l'environnement, reçue par des capteurs et qui peut passer par différents processus dont le but est de déclencher un service convenable à la situation actuelle. Plus précisément, un système intelligent doit prendre en considération la tâche de transformer un contexte d'un bas niveau en un niveau d'abstraction plus élevé. L'abstraction du contexte a un rôle important dans la présentation adéquate du contexte dont le but est de donner une représentation plus compréhensible qui pourra être manipulée par des mécanismes de raisonnements. De ce fait, nous nous sommes intéressés à évaluer le critère

d'abstraction du contexte pour chaque système à l'égard de la sensibilité au contexte qui se manifeste aussi comme une condition très importante pour la conception d'un système intelligent.

Modules de raisonnement

Le raisonnement ou l'inférence du contexte est un facteur important dans un système intelligent. En effet, c'est la source du caractère d'intelligence qui permet de différencier un espace intelligent à un autre espace. Dans le fond, il permet d'identifier la donnée contextuelle et de la transformée du bas niveau, celui reçu par les capteurs à un niveau plus élevé et plus compréhensible par les autres composants du système. L'évaluation de ce critère est analysée par la présence ou l'absence de ce module et par la nature et le type du raisonnement effectué.

Après avoir exposer différents systèmes intelligents, on peut dire que la majorité des systèmes souffrent toujours d'un manque : soit au niveau architectural, soit au niveau de la composition des modules. Effectivement, on remarque que quelques systèmes ne sont pas modulaires ce qui génère un fort couplage entre les modules et/ou une faible cohérence dans les modules. Aussi, il existe ceux qui ne traitent pas l'abstraction du contexte ou la sensibilité du contexte. D'autres, ne possèdent pas des modules de raisonnement

1.6 Conclusion

Dans ce chapitre nous avons introduit les différents axes de recherches liés à notre problématique autour d'un environnement intelligent sensible au contexte. Notre solution englobe les notions de l'intelligence ambiante, le contexte, la sensibilité au contexte et l'auto-adaptation. Nous avons donné un aperçu sur l'évolution de chacun pour bien éclaircir le contexte de notre travail. D'abord, nous avons commencé par la présentation d'environnement intelligent dont l'objectif est de rendre le comportement de l'environnement et de ses utilisateurs intelligents et par conséquent assister l'utilisateur sans que ce dernier

fasse le moindre effort. Étant donné que l'environnement intelligent se base essentiellement sur la notion du contexte, nous avons illustré quelques définitions sur le contexte, la sensibilité au contexte ainsi que les principales catégories et modélisations du contexte. Ensuite, nous avons introduit le concept d'auto-adaptation dans un environnement intelligent par les différentes formes d'adaptation connues dans la littérature. Nous avons aussi mis l'accent sur les façons de raisonnement et d'apprentissage utiles pour réaliser un ajustement de service avant d'être exécutés. Vers la fin du chapitre, nous avons présenté et étudié les architectures les plus pertinentes pour les systèmes intelligents dans le domaine de l'intelligence ambiante. Nous avons décrit l'architecture et le déroulement de ces systèmes pour réaliser des services convenables aux besoins des utilisateurs.

Notre thèse s'inscrit dans ce contexte et vise à répondre aux défis des systèmes SIAC dans le cadre d'une architecture orientée composant. Afin d'appliquer correctement cette architecture dans le cadre d'un système intelligent, nous avons traduit les défis soulevés par ce système à un modèle tiré du pattern MVC. C'est au centre de ce modèle que se situe le module de raisonnement et d'adaptation de services. En effet, ce dernier est responsable de l'ajustement dynamique des services, en tenant compte, des besoins variables exprimés par les utilisateurs. Par conséquent, nous allons présenter plus en détail cette problématique liée à l'architecture d'un espace intelligent efficace à adapter ses services pour bien servir ses utilisateurs.

CHAPITRE 2

TOWARDS AN EFFICIENT SMART SPACE

Somia Belaidouni [a], Miraoui Moeiz [b], Chakib Tadj [a]

[a]Département de génie électrique, École de technologie supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[b]Al-Leith computer college, Umm Al-Qura university, Saudi Arabia
21421 Makkah, Saudi Arabia

Article publié dans le journal «International Journal of Advanced Studies in Computer Science and Engineering, IJASCSE» ,Volume 5, Issue 1, 2016.

Abstract

A smart space offers entirely new opportunities for end users by adapting services accordingly to make life easy. A number of architectural designs have been proposed to design context awareness systems and adaptation behavior. However, the quality of the system depends on the degree of satisfaction of the initials needs. In this paper, we discuss three main indicators of quality design for smart spaces that are strongly related to the context modules and reasoning process: functionality, reusability and changeability. A general layered architecture system is presented to define the principal components that should constitute any context aware adaptive system.

Keywords: smart space; architecture; adaptation; context-awarness; quality.

2.1 Introduction

A smart environment can be defined as a space that is able to acquire and apply knowledge about its environment and inhabitants to provide appropriate services (Handte and al., 2012). Some important features of smart environments are that they possess a degree of autonomy, adapt themselves to changing environments and communicate with humans in a natural way (Das and al., 2005). This ubiquitous environment is composed of different devices such as embedded computers, multimodal sensors and various software programs that aim to facilitate human life by offering abundant information from different devices to accomplish

the required expectations. The capacity to build a system that is able to process the information delivered to meet the user's requirements is dependent on an architecture that can support the dynamism and heterogeneous devices and handle the interactions between the user and system. Therefore, the main objective of any architecture developed for any smart space is the ability to sensor and gather information from the area where it is deployed and to process it by making adequate decisions and executing actions on the physical environment.

The majority of the existing architectures for smart space focus on acquiring data from sensors, interpreting the data and adapting services to adequately assist users to concentrate on their specific tasks. Context awareness and adaptation are tightly related, and the two terms are often used as synonyms (Chaari, 2004). However, adaptation means the ability to change a service and produce another corresponding environment; context awareness is the ability to perceive the different situations of users to adapt actions before execution. The proposed architectures are designed at a limited concentration at the modularity of components. Moreover, they do not take into account the abstraction of context received from the environment. In addition, the awareness of context and the ability of adaptation by using reliable algorithms still remain to be research questions. These different problems involve development complexity in making the system more difficult to maintain and to reuse components.

The main goal of this paper is to define the most important quality characteristic related to a smart system. A set of indicators is introduced to reveal the highest requirements in a height quality system. Subsequently, we present a general architecture with essential components that can be designed to build an adaptable system with high performance. We then present a comparative study between our proposal design and other attractive structures based mainly on the evaluation criteria.

The rest of this paper is structured as follows. In the next section, we review related work. Section 3 identifies three essentials metrics for creating a better architecture for a smart

space. Section 4 provides an overview of the proposed architecture and all of the components necessary for building an adaptable system. In section 5, we analyze and compare our proposed design with others systems, and section 6 concludes the paper.

2.2 Related Work

Researchers studying smart spaces have been actively doing research in recent years, as many related smart systems have been developed and tested in the real world. The main goal for any smart system is to adapt compartments according to the user's preferences and satisfy all demands.

In this section, we showcase some smart environment projects as case studies and discuss them. Our studies are based on projects and not necessarily on awareness context or self-adaptive systems. However, we focus on reliable projects that have proven feasibility and validity. We also aim to study distinct structures with different modules, where some modules can be combined or removed altogether according to their functionalities. We can thus expand our performance analysis and deduct the primary criteria that affect the working of the design. The MavHome project (Cook and al., 2003) aims to create a home that acts like a rational agent. The basic idea is to build a smart home with sensors and actors to produce services adapted to the inhabitants' activities. In the proposed solution, the main idea is to use agents that work together for selecting and initializing the adequate action and hence maximizing the goals for the home. Due to this reason, the design home must be able to receive and process the information and perform reasoning to adapt actions to user preferences. The MavHome architecture is structured in a hierarchical form composed of agents. Each agent is separated into four layers. The physical layer contains all of the hardware that may exist in the home, including sensors, actioners and hardware for communication. The communication layer has the responsibility of transmitting information, queries or requests between agents. The information layer gathers and stores any data in the system and generates knowledge useful for making the appropriate decisions. The decision layer is based on the previous layer information for making decisions and selecting related

actions. The SOCAM project (Gu and al., 2004) is an architecture developed for a service-oriented context-aware middleware to provide context-aware mobile services within a smart space. The SOCAM architecture is composed mainly of the following components: context provider, context interpreter, context knowledge and context reasoner, service locating service, context- aware mobile service and context database. Context providers provide internal and external context at higher orders, which can be used directly by services. The context interpreter is responsible for context processing and defining the reasoning component based on ontologies that allow for robust approach reasoning. The principal task of ontologies may include deriving high-level contexts from low-level contexts, maintaining the consistency of context knowledge and resolving context conflicts (Gu and al., 2004). The context-aware services refer to all elements that use context at different levels, such as agents, applications and services. This component is able to obtain the context of interest by querying the service-locating service to locate all of the context providers, which is the source context. The service locating service deploys multiple services to allow users and applications to discover contexts, locate context providers and adapt to their dynamic changes. The SmartLab project (Lopez-de-Ipina and al., 2012) is an OSGI-based middleware platform that provides a cooperative and programmable intelligent environment. The interest in the project stems from the fact that it is an evolutionary and self-configuration structure that adapts services according to rapid changes in such environments. The SmartLab paradigm has already been used as a base for diverse projects. The sensing and actuation layer is composed of a set of devices that constitute the environment. They include the EIB/KNX automation bus, VoIP and VideoIP, indoor location systems, Smart Display, and Smart container. The next layer is the service abstraction layer that interprets and transforms the functionality of the devices received from the previous layers into software services. The Semantic Context Modeling and Service management, also performed by the server SmartLab, acts like a gateway between the advanced OSGI services and the functionalities of the equipment deployed in this environment. It comprises the Service Manager, which handles the activation and deactivation of devices and, thus, service availability. The Semantic Context Manager stocks knowledge on devices using a common ontology. The final layer is the Programming, Management and Interaction layer, which enables

applications to make use of the SmartLab infrastructure. It also generates two generic applications, Environment controller and Context-Manager Front End, which ensure the control and management of an intelligent environment. CoBra (Chen, 2004) is an architecture and specification based on agents for creating, distributing and managing context-aware applications in an intelligent space. The core of any CoBra-distributed system is the Object Request Broker (ORB). The role of the ORB is to maintain a shared model of context between whole entities in the smart space. In addition, the ORB is able to ensure communication between components and heed to issues related to distribution and heterogeneity of the environment. The concept of the CoBra architecture is based on four main functionalities: context knowledge, context reasoner engine, context acquisition module and privacy management module. Context knowledge uses ontologies for the management and storage of context data. It also provides a set of APIs that allow access to the data context. The context reasoner engine uses a set of ontologies to reason and infer new rules that can be used by the system. It allows also for a more efficient reasoning and sharing of contextual information. The Gator Tech Smart House (GTSH) (Helal and al., 205) is a project developed to construct an experimental laboratory and an actual live-in trial environment to test and prove the performance of technologies and systems developed for smart spaces. The smart house is designed to assist older persons or individuals with special needs to maximize their independence and provide a better life for them. The project architecture is developed in six main layers, and the middleware is the main unit connecting the other entities. The physical layer gathers various devices and appliances in the house. It includes sensors and actuators and can also include equipment such as the TV, a lamp, a doorbell or sensors and actioners. The sensor layer communicates with different devices in the physical layer, receives their information and transforms it into useful information to relay from the physical layer into software services that can be used in other services invested into by the next layer. The service layer comprises the central middleware OSGI, which maintains and controls the service bundle. The middleware store service bundle definitions for each sensor or captor represented in the physical layer. The services are composed of other services that represent the many applications available in the smart space. The knowledge layer represents the ontology mechanism offered for all services and

appliances in the house. This allows for reasoning related to the services used. The context management layer is responsible for defining and eliminating the context of interest to be used by applications. The context is designed as a graphical OSGI bundle linking various sensors and captors. The application layer is the last layer equipped with various tools for managing applications and developing the smart environment, and it is therefore responsible for activating or disabling service execution according to the inhabitant's need. The Smart-M3 NOKIA platform (Honkola and al, 2010) is designed for creating a simple, extensible and interoperable model to be deployed on top of any existing infrastructure in a smart environment. Furthermore, it allows devices to easily share and access local and global semantic information. It mainly consists of two components: Semantic Information Broker (SIB) and knowledge processor (KP). The SIB component maintains and stocks all information by exploiting the Resource Description Framework (RDF). There may be one or many SIBs in the smart space, and they are connected by the protocol SSAP. They work together to manage and provide the same information for any device. KP entities are deployed by developers on different devices participating in the smart space and interact with each other to access SIB for reading or publishing data via the SSAP protocol. Each KP has a specific task, and devices may contain diverse KPs with different functions. However, KPs do not have the ability to communicate with others; instead, they must pass through the SIB unit. The Smart Space Access Protocol (SSAP) is the protocol used by KBs to access a SIB. It performs seven major operations that specify actions the KB or SIB should take. SSAP is session-based, that is, when a KP wants to communicate with the SIB, it must open and active a SSAP session. For this purpose, development of KPs and SIB should support all SSAP operations. This will guarantee interoperability in a heterogeneous environment according to the requirement for smart space. The framework S2CAS (Smart Space Context Aware System) (Chen and al, 2011) is developed to manage context and to provide services that take into account environmental data. The overall project is designed in three major layers. The first layer, which is situated at the bottom of the system, can receive information from different sensors and devices and prepare them for the core layer. The layer is equipped with a mechanism for pre-treatment of the captured data. The context information should be physical or virtual data. The second layer is the core of the system. It includes three essentials

modules that realize the interpretation, treatment and management of context: the context information processing, the context explanation and the context manager. Context information processing involves receiving information from the previous layer and transforming it to more formal data. The context interpreter, which is composed of a context knowledge base and the context inference engine, records the historical context environment and transforms the low-layer context information into high-layer information. The top layer is defined as the interface between the user and system and presents applications and services according to needs. This layer uses the context height level from the interpreter module of the core layer to provide specific context-aware service. The Greenerbuilding project (Degeler et al., 2013) is a service-oriented approach (SOA) for designing and developing a building management system. It is a structure that is dedicated to creating an intelligent office in a green and energy-efficient way while preserving and maintaining comfort for habitation. At the same time, it allows its users to modify and adjust the rules of the system for adapting services according to their needs. A significant component of this project involves using renewable energy, which is an excellent alternative for economic and environmental reasons. In addition, with the introduction of the SOA principles, the building can take into account the heterogeneity of the available devices and capabilities. The physical layer consists of various devices in the building, which can include sensors, actuators, gateways and also low-level protocols. The gateway sensors and actuators collect all information from the devices and transfer it to the height level of the system. The Greenerbuilding is interconnected to smart grid services that make it aware of external and internal energy pricing. This allows Greenerbuilding to adjust demands and reduce the energy costs according to the operations. A ubiquitous layer ensures the functioning of the whole system. It is composed of a repository that contains a database of the devices and their configurations and states. The context component collects information from the sensors and transfers it to the height level. The composition layer is the last layer that supplies the control services and represents the system's interface to its users. It mainly performs reasoning and makes decisions using components based on the Rule Maintenance Engine. It gathers information about user preferences makes decision and decides on actions to be performed according to user requirements. Self-adaptation systems (Chun and al., 2013) are a new project designed to

develop an innovative approach for building, running and managing a self-adaptation system for smart space. The main idea is to make smart and reliable devices handle uncertainty and uncontrolled conditions in smart spaces. It implements agents in such a way that each agent defines a device, and all agents cooperate to provide seamlessly adequate services for users. The system consists of adaptive software and an adaptation agent. The adaptive software consists of a SAManager thread, which is the central component of the system. It contains the predefined normal status constraints (NSCs) and function list to be executed. SAManager performs multiple duties. Initially, it is invoked by the adaptable system to create the first thread. It then monitors and manages the NSC list. Moreover, SAManager is responsible for selecting functions related to adaptation commands from the adaptation agent and notifying that an anomaly occurred.

The adaptation agent includes the adaptation process and logics in contact with the SAManager, and it states its demands and selects the appropriate adaptation strategy using the adaptation strategy table (AST). The adaptation agent receives data from different resources and may collaborate with neighboring adaptation agents to provide services. Thus, the implementation of the adaptation agent can be customized and configured according to the goals of the system.

2.3 Smart Space Requirements

It is widely acknowledged that a smart space is a typical open, distributed and heterogeneous pervasive computing system, which aims to create a ubiquitous, human-centric environment with embedded computers, information appliances and multimodal sensors that facilitate humans to perform tasks efficiently by offering abundant information and assistance from computers (Qin and al., 2006). A prominent and important requirement for a smart space to be able to respond adequately to user's needs is to assure interoperability between components. Interoperability reflects the existence of a co-operative structure of services (van der Meer and al., 2003), which means the capability to mask the heterogeneity of the equipment to facilitate communication within the environment. A smart space is a grouping

of heterogeneous entities (sensors, objects, devices) that interact and service one another to complete different actions. Scalability is another important requirement for designing a smart space. A reliable system designed for a smart space should enable its structures and functions (Wu and al., 2007). A smart system should also adapt the behavior and the interface of an application to the user situation and equipment to provide information about a physical environment, which is shared with inherently dynamic applications (Chaari and al., 2004). In addition, the concept of context is the principal entry on which all of the system processes are based. It denotes various aspects such as where one is, who one is with, what one is doing and what resources are nearby (in other words, location, identity of neighbor entities, activity and the state of the physical environment) (Bucur and Doina, 2008) . The ability to adapt services in such an environment requires some technologies for sensing, acquiring, interpreting and inferring context, which is the principal entry to the smart system. This mechanism is known as awareness of context and is an essential step towards improving the quality of the system.

2.4 Quality Architecture

The quality of an architecture designed for a smart system is measured by the ability to achieve the planned targets (Handte and al., 2012). Thus, height quality must respond to the growing demands of users in a reliable and error-free way. To evaluate the smart space architecture quality, different quality characteristics might be of interest. In the context of this paper, we define three basic characteristics: functionality, reusability and changeability.

- Architecture functionality (Q1) - the architecture of the system satisfies the stated objective of the system.
- Architecture reusability (Q2) - can the architecture (or components) of the system can be reused in other systems?
- Architecture maintainability (Q3) - can the architecture of the system be maintained to extend and adapt to a changed environment?

We choose the architecture functionality characteristic because it is strongly related to the goals of any smart space to provide services for the tasks. Furthermore, we choose reusability

and maintainability, the ISO/IEC9126 sub-characteristics of maintainability because we consider them to be two important concepts that should considered by the architecture.

We further refine architecture functionality into two concrete quality characteristics:
Awareness context: this term signifies that the system adapts its services to a user's preferences and activities in the environment. The awareness context is becoming increasingly more prevalent and can found in many areas of research (Krumm &John, 2009) due to its importance. In a smart environment, a context awareness system must have the ability to detect, sense, interpret and respond to user requests (Jiang et al., 2006). Thus, this property leads to detection of contextual information from different sensors or device sources but also to understanding what type of contextual information should be provided for the appropriate service.

Self-adaptation: the capability of the system to adjust its behavior in response to the environment in the form of self-adaptation has become one of the most promising research directions (Brun and al., 2009). We consider that a self-adaptation architecture is capable of readjusting its behavior to external conditions. It is widely known that a smart space is richly equipped by different sensors/devices and connected to diverse technologies. Within this heterogonous environment, applications must modify and adapt their services to the user's needs according to the current context (Miraoui and al., 2014). Therefore, we can say that an adaptive system is a context-awareness system that applies specific adaptation approaches. Moreover, a smart system is characterized by self-adaptation that does not require any tiers. However, the system triggers its attitude automatically according to the actual context.

We refine architecture reusability and maintainability in the modularity that provides the mechanism for separating the complex system into functional modules. The modularity must be effective: each module should focus exclusively on one aspect of the system, or the modules should be cohesive. Additionally, modules should be independent of each other. A more efficient modularity leads to a more understandable, reusable and modifiable architecture. The maintainability of a software system is the ease with which it can be

modified to changes in the environment, requirements or functional specifications (Bengtsson and al., 2000). The maintainability process involves the capability of making modifications at different levels of conception. Adaptive systems designed for smart spaces are to be modified several times after their initial development, taking into account the dynamism of the environment, to facilitate system performance. A number of studies (Hoefler and al., 2012) have shown that 50% to 70% of the total lifecycle cost of a software system is spent after initial development due to the high cost of software maintenance. This propriety defines an important quality that we should consider for fulfilling system purposes. We now study some quality indicators that can be used to check the achievement of these characteristics.

We define two main indicators that can be measured for the awareness of context:

- Abstraction of context
- Modification of context

The abstraction of context aims to model and present real-world situations. Information from physical sensors is called low-level context and is acquired without any further interpretation (Ye and al., 2007). Moreover, human interactions and behavior are received in unreadable format. This limitation of low-level contextual cues risks reducing the usefulness of context-aware applications (Bettini and al., 2010). One way of overcoming this problem is to derive higher-level context information from raw sensor values, based on context reasoning and interpretation. The main idea of the abstraction process is to create a model that takes raw sensor data as input and generates more significant data relevant to the current situation.

The identification of components and their relationships is the essence of high-level software design (Allen and al., 2001). The modularity of the architecture can be measured by the degree of coherence and coupling (Stevens and al., 1974). These measures capture the degree of interaction and relationships between the elements (Újházi and al., 2010). Software research shows that designs with low coupling and high cohesion enable the production of a more reliable and maintainable system (Holvitie and al, 2014). Principally, cohesion refers to the degree of interconnection within the same module. High cohesion of a module indicates

that the elements are interconnected well for implementing a specific function together, without outside assistance. The coupling defines the degree of interdependence between the modules. High coupling of a module creates a strong dependency with other modules, which means that its functionality is based heavily on the presence of other modules. Figure 2.1 shows the principal characteristics and the relationship between indicators to produce a high-quality architecture.

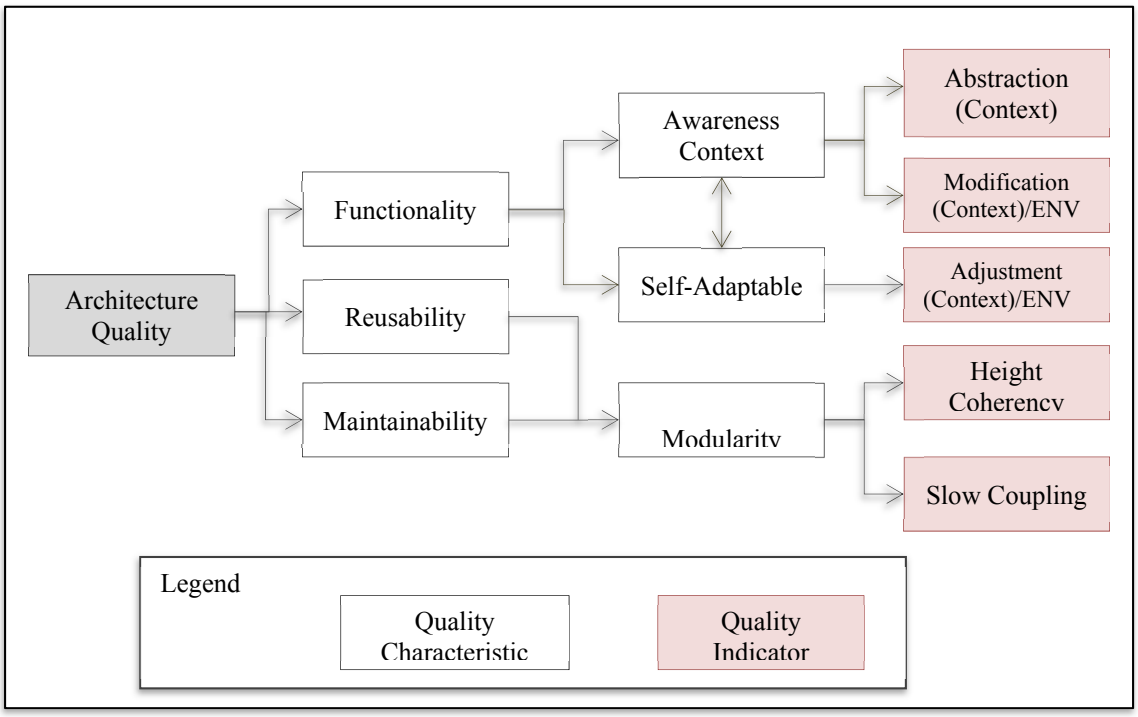


Figure 2.1 Quality architecture for smart space

2.5 Architecture Overview

In this section, we will present the design pattern for adapting services in a smart environment. Using the most successful projects as case studies, we searched for a common pattern and advantageous structure for building a smart system. We wanted to provide a general design for adapting actions according to the different versions of context in a smart space. This pattern may be used as an exemplar for other projects and is intended to reduce efforts and allow developers to concentrate just on specifying requirements tailored to every

project. Figure 2.2 presents the pattern process in an entry system context (such as battery level and network resources) and for users (such as localization, noise, and user needs). The whole pattern may be split into four layers (physical, context, adaptation, decision) in such a way that every layer is a set of different components that interact to achieve specific services that will be presented to the user at the end. It is important to note that all components of a layer must collaborate to generate services to the following layer.

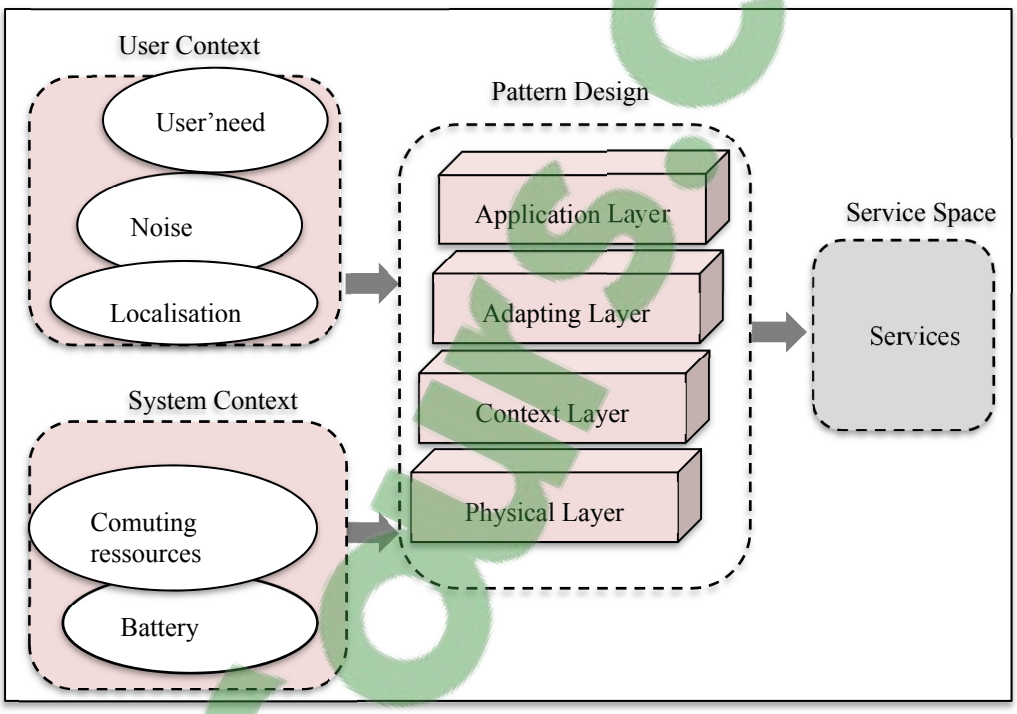


Figure 2.2 Overview of an architecture for a smart space

2.6 General Architecture for Smart Spaces

In this section, we will present a generic design of the smart environment architecture. We intend to provide a quality architecture that satisfies the requirements mentioned earlier. Therefore, our architecture includes four layers with several distinct components in every layer. It is important to note that these components are not exhaustive in terms of availability in the system. However, projects may have a greater or smaller number of components,

depending on the goals and the environment. The general architecture is presented in Figure 2.3.

2.6.1 Physical Layer

Is a wireless network of smart objects. It contains all of the hardware parts of the system, which includes wired and wireless sensors and actuators associated with their infrastructure. The main goal of this layer is to capture and collect external information about the environment and transfer it to the next layer.

2.6.2 Context Layer

Is the backbone of the whole architecture. It is the main support of the system that contains several components with different responsibilities related to the context. We can split this layer into four essential modules that reside in every smart environment in one form or another.

Knowledge Base

The knowledge base is the base of any information related to the system and its components. It can include static information about devices, appliances or users such as configuration information or dynamic information exposed to frequent changes, such as the location of the users or devices in the environment. Moreover, this module is a base of all information required for high-level reasoning. The nature of the data depends on the type of reasoning module. For an ontology-based system, this information would represent ontologies, and for a rule-based system, the reasoning rules would be stored.

Context Module

The main responsibility of this module is to transform low-level data gathered from the previous layer into higher-level information. This module collects sensory data and interprets them to identify the meaning and yield understandable data that are usable by the reasoning

module. The reasoning module includes other components that work together to manage the data.

Aggregators

Aggregators collect different pieces of context information that are logically related in a logical depot. A context is often received from different and distributed sensors. Therefore, aggregators decompose and gather context information in coherent groups for context inference, consistency checking and knowledge sharing.

Interpreter

After combining context information in logical groups, the interpreter will identify the meaning of the context before it can be used. An application may not be interested in the low-level data. However, it requires comprehensible data that are expressed at higher levels of abstraction.

Reasoning Procedure

Context reasoning, also known as context inference, is one of the most important elements in the context aware system and involves different process: checking the consistency of the context and deducing high-level, implicit context from low-level, explicit context (Wang and al., 2004) . For achieving these goals, the reasoning process is based on specific reasoning methods such as rules, fuzzy logic, decision trees and Bayes networks. For our work, we choose the ontology mechanism because it can exploit various existing logic mechanisms to deduce high-level conceptual context from low-level, raw context and check for and solve inconsistent context knowledge due to imperfect sensing (Brun and al., 2009).

Modeling

Context modeling is an important instrument for handling contexts and managing how they are collected, organized and represented, whereas context awareness signifies reasoning about the context (Topcu and Ferit, 2011). In a smart space, context modeling defines the manner for representing context information for reasoning and adaptation processes.

Basically, it aims to provide more simplified descriptions using different mechanisms. There are different models for present context such as the following: graphical models, logical models and ontology-based models. It is important to note that researchers show that an ontology is the most advantageous and beneficial model because it simultaneously allows for recognizing the context being operated in and reasoning about other different contexts. Furthermore, using non-ontology-based models requires much programming effort and tightly couples the context model to the rest of the system (Niemelä and al., 2004).

2.6.3 Adaptation Layer

It comprises the domain-level logic of the system. It aims to adapt services according to the context detected by the preceding layer. It is responsible for analyzing context and deciding upon actions to execute. Such an adaptation is required to serve several goals determined by the nature of the application, the nature of the context and the preferences of the user (Mizouni and al., 2014). According to its responsibilities and definitions, we summarized this layer into six essential processes:

Machine Learning

Machine learning is a type of automatic learning that aims to provide the best possible decision and action to execute. In smart systems, machine learning is applied to support reasoning and inference related to complex information. As defined in (Wang and al., 2011), machine learning in a pervasive system is an innovative approach that integrates wireless, mobile and context-awareness technologies to detect the context of learners in the real world and thus accordingly provide adaptive support, personalized services or guidance.

It integrates wireless, mobile and context-awareness technologies to detect the situation. This process is also linked to database user preferences to consider user preferences during the learning process. Several algorithm are employed depending on the project goals and its requirements, such as genetic algorithms, artificial neural networks, decision trees and reinforced learning.

Figure 2.3 General architecture for a smart space

Activity Recognition

Activity recognition is a process that uses context information about the current state and applies internal knowledge to clarify the data. This activity should be performed at discrete time points in real time in a progressive manner (Chen and al., 2012). This activity is able to

detect a current service or movement based on the information received by sensors to recognize the current state of the ongoing activity and further identify the user's needs.

Adaptation Services

Adaptation tools form the core of a context aware adaptive system. The input of this engine is the learning context, and the output of this component is the adaptation action. The adaptation process is related to learning activities that add adjustments to meet user needs. The process adaptation can be used similar to configuration, personalization or recommendation.

2.7 Performance Analysis

The goal of the performance analysis of any architecture is to verify that the implemented system meets all of the requirements for the current project. In this section, we aim to compare our proposal architecture with seven systems developed for adapting services in a smart space. To conduct our analysis, we focus on three criteria mentioned subsequently: design of the system and the contextual and reasoning modules. These three characteristics are unique for every design and differ according to the environment and the project goals. Table 2.1 shows a comparison of the relevant smart systems. The design of the system reflects the nature of interconnection and relationship between the modules and their elements. The ideal architecture intends to minimize coupling (coup) and maximize cohesion (coh) to reduce the efforts related to understanding and maintaining the systems. By analyzing related work, we note that several systems are highly cohesive and coupled. Such drawbacks make the system complex, inflexible and ineffective. SOCAM and Greenbuilding are two modular architectures that respect a high cohesion and a low coupling. This quality facilitates the changeability and maintainability of systems. The abstraction of the context and context awareness are major criteria for processing and identifying more comprehensible data to provide intelligent and adaptable context-aware services. MavHome, CoBra and SOCAM present the advantages of being aware of the context and providing a level of abstraction during context processing. Reasoning module is very important, especially in a

smart space, where the context is received from sensors at low levels. The reasoning engines have the capability of deriving new concepts to adapt service behavior accordingly. We remark that the CoBra, SOCAM, Smart-M3, Smartlab systems use ontologies to reason and provide a rich formalism for specifying contextual information.

Table 2.1 Comparison of smart systems

Systems	**Component modularity**		**Contextuel module**		**Reasoning module**	
	Coh	**Coup**	**Context abstraction**	**Context awareness**	**Reasoning module**	**Type of reasoning**
MavHome	High	High	+	+	+	Machine learning
CoBra	Low	High	+	+	+	Ontologies
SOCAM	High	Low	+	+	+	Ontologies
Self-adaptive	High	High	+	-	-	-
Smart-M3	High	High	-	+	+	Ontologies
S2CAS	High	High	+	-	+	Rules
Green-building	High	Low	+	-	+	Rules
SmartLab	High	High	+	-	+	Ontologies

By comparing the other systems with our proposed system, we note that our solution responds effectively to all of the requirements (components modularity, contextual/reasoning module), unlike the other solutions we surveyed. A further advantage is that our solution involves a full architecture that contains all of the layers and components for managing the context and responding to user requests. In addition, our solution embeds reasoning and adaptation processes to regulate actions dynamically according to the environment and its users.

2.8 Conclusion

The development of adaptable systems has undergone an important evolution marked by an incremental improvement in techniques for managing and reasoning related to the context.
In this paper, we present the most important characteristics for quality architecture and the requirements necessary for developing a reliable adaptable system: functionality, maintainability and reliability. We subsequently present general-layer architecture with the most necessary components to develop a smart space. From the analysis of some relevant architecture, we show that the modularity of components, awareness of context and process reasoning are three important characteristics that reflect the reliability of systems.

Future development efforts should focus more on the development of modular architectures with high cohesion and low coupling components that are important for facilitating the reusability and extensibility of systems. Context inference is a hard task and requires methods to reason correctly and adapt systems automatically to new situations. We must therefore reinforce the development of more efficient reasoning methods that take into account the dynamism of smart spaces and their entities. Security and privacy are important parameters, which we must consider in the adaptive system to protect sensitive information.

CHAPITRE 3

USING MVCA ARCHITECTURE TO IMPROVE MODULARITY IN SMART SPACE

Somia Belaidouni [a], Miraoui Moeiz [b], Chakib Tadj [a]

[a]Département de génie électrique, École de technologie supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[b]Al-Leith computer college, Umm Al-Qura university, Saudi Arabia
21421 Makkah, Saudi Arabia

Article publié dans le journal «Journal of Computer Science».Volume 13, Issue 10, (2017): 697-707.

Abstract

There has been increasing interest in the use of context awareness, as a technique for designing architectures dedicated to smart spaces in order to adapt and produce suitable services according to user context. In recent years, various architectures have been developed to support context-aware systems. The major challenge with these systems is decomposing the entire architecture into smaller, modular components that facilitate the comprehension and modification of the architecture. In this study, we propose the Model View Controller Adapter (MVCA) architecture, derived from the model-view-controller pattern, which is modular, flexible and capable of adapting services autonomously on behalf of users. The main concept of MVCA architecture is that it decomposes the overall functionalities into modular components with high cohesion and low coupling, which facilitates reusability and maintainability of the system. The MVCA architecture is essentially composed of four components that are responsible for sensing and managing the environmental context in order to adapt and produce services proactively according to user context. To clarify and show the usability of our architecture, we present a scenario-based simulation of MVCA architecture using the Java Agent Development Framework platform.

Keywords: Pervasive Computing, Smart Space, Architecture, Context- Awareness, MVC Pattern, Modularity.

3.1 Introduction

The diverse nature of pervasive computing makes it difficult for software designers to adopt one common model that can meet all requirements (Abdualrazak and al., 2010). Designing context-aware smart spaces is a challenging task for two main reasons: Firstly, supporting different devices and multiple interacting platforms is difficult; and secondly, it is difficult to achieve environmental context awareness and to proactively adapt services to dynamic changes. Addressing these issues means that systems should be capable of providing a uniform and efficient architecture for communicating the entirety of entities within the environment to meet people's needs and adapt to their preferences.

In view of these requirements, our project aims to create a modular and flexible architecture, which takes the context as the first input as the most important element for reasoning, adapting and providing a service in a more suitable form. Modularity is a significant quality attribute in the design of large system architectures (Knoernschild, 2012). Modularity is considered as a key feature to improve several quality attributes such as maintainability, portability, reusability, interoperability and flexibility (Galin, 2004). In the case of our smart environment, modularity is the key to facilitating management of the design complexity and ensuring the development of the overall components. Moreover, the degree of modularity is proportional to the degree of loose coupling and high cohesiveness of a system's software elements (Sharafi et al., 2012).

Therefore, the benefit of modularity is dividing the architecture into loosely coupled and highly cohesive modules, which interact in a collaborative manner to achieve the requirements.

The Model-View-Controller (MVC) is considered as an architectural pattern in software engineering. It adopts the idea of "divide and conquer," (Zhang and Zhu, 2013), which involves dividing the entire architecture into three separate components: Model, view and controller, based on the principle of separating the act controllers, data presentation and

user's actions. This facilitates modification and change in each component without significantly disturbing the others. The benefits of the MVC design have been demonstrated in various interactive applications (Sarker and Apu, 2014).

In this study, we propose a Model View Controller Adapter (MVCA) architecture that is a variation of the established MVC pattern and provides a new perspective on modularization. It divides the overall functionalities of the system into a modularized architecture. MVCA introduces a new component, the adapter, to provide suitable services with a robust and extensible infrastructure in order to provide environmental context awareness and meet the user's needs. Mainly, it is responsible for reasoning regarding the sensed context and adapting services proactively. In general, the adapter is a set of rules which have to be specified for the possible contextual configurations about user interactions and their preferences and each rule triggers a specific set of services according to the current context. In such adapter, we have to imagine and predict all possible context configurations before deploying the system. This set of rules will then be static and can not be modified when the system is operational. Such approach has two main drawbacks: (a) it is impossible to predict all possible context configurations and (b) the adaptation system is not dynamic and cannot evolve as the system operates (it is limited to the static set of rules). Instead of rule base approach, the component contains also learning engine in order to adapt services to all possible and meaningful context configurations. This machine learning approach can select a previous choice about the service and can adapt itself by another new choice about service from user feedback, which make it adaptative.

The remainder of this paper is organized as follows. In the next section, we provide an evaluation of some existing software architectures. In section 3, we describe our proposed architectural pattern to overcome the identified problems. In section 4, we present a scenario implementation and report the main results from the simulation studies. Finally, our conclusion and future work are presented in section 5.

3.2 Related Work

A wide variety of context-aware schemes have been proposed over the past few years. However, most of these provide only partial context-awareness functionality (Zhang et al., 2013) and aim to satisfy certain quality attributes, such as interoperability, flexibility and maintainability. The CASS tool (Clarke and Fahy, 2004) is middleware for supporting the development of context-aware applications. It divides the entire architecture into independent components and achieves effective abstraction of contextual information. This architecture provides efficient modularity that facilitates the modification of server components; in particular, the inference engine. Octopus (Firner and al., 2012) is dynamically extensible middleware for facilitating smart home/office domain-specific applications. It is constructed on a simple layered architecture to tackle the problems of data management and fusion; however, it appears to have a lower tolerance for problems. FlexRFID (El Khaddar and al., 2015) aims to provide a policy-based middleware solution for facilitating the development of context-aware applications and integrating heterogeneous devices. Authors have furthermore proposed a multi-agent (Miraoui et al., 2016) for a smart living room that focuses on context-awareness, which is a key enabler for service adaptation in smart spaces. The project aims to provide architecture with a high degree of modularity; that will have a positive impact on several software qualities, such as modifiability, reusability, maintainability and extensibility.

The architecture proposed in (Aloulou et al., 2012) provides an adaptable and dynamic platform that integrates new assistive services at runtime and enables their binding to specific patients. This system adopts modularity by using the Service-Oriented Approach (SOA), which improves its flexibility, scalability and configurability. The FIWARE (FIWARE Project, 2016) aims to provide cutting-edge infrastructure in which the creation and delivery of services, high quality of service and effective security are achieved. It is a generic platform that could adaptively be applied to various usage areas. Preuveneers and Berbers (2007), authors have proposed a resource-aware and context-driven application middleware for mobile devices that has a layered design and provides introspection and intercession capabilities within each layer to support a more flexible self-adaptation strategy at runtime.

The MUSIC project (Rouvoy et al., 2009) is a middleware Support for Self-Adaptation in ubiquitous and service-oriented environments, which addresses aspect-oriented adaptations. To accomplish this, authors uses computations and evaluations of alternative application configurations in response to context changes and the selection of a feasible configuration. The SECAS (Chaari et al., 2008) is a project that ensures the deployment of adaptive context-aware applications. The authors aim to develop a platform, which makes the services, data and the user interface of applications adaptable to different context situations. Focuses on this point by providing the necessary tools to adapt existing applications to new contextual situations that were not taken in consideration at their design stage. The Chisel system (Keeney and Cahill, 2003) is an open framework for dynamic adaptation of services using reflection in a policy-driven, context-aware manner. The system is based on decomposing the particular aspects of a service object that do not provide its core functionality into multiple possible behaviours. The principal aim of this project is to build a framework supporting unanticipated dynamic adaptation that will take account of contextual information from as many sources as possible. The PACE middleware (Henricksen et al., 2005) is used to support the development of a context-aware vertical handover application which investigates a variety of issues related to pervasive computing, including the design of context-aware applications and solutions for modeling and managing context information. Il offers a toolkit that facilitates interaction between application components and the context and preference management systems.

3.3 Overview of MVCA

The MVCA architecture is based on the MVC pattern, with an additional component, namely the adapter, which is used to adapt services in unforeseen situations. This architecture is modular and based on the separation of the four components that represent the functionalities of a smart space. These modules interact and collaborate to handle contextual data in order to produce the most suitable service.

The context is the key element to take into account because it represents the main input data on which our architecture functionalities are based. As illustrated in Fig 3.1, we have decomposed the smart space into three main parts: The context is represented by the contextual model, the adaptation part is realized by the adapter and controller and the service part is represented by the view component.

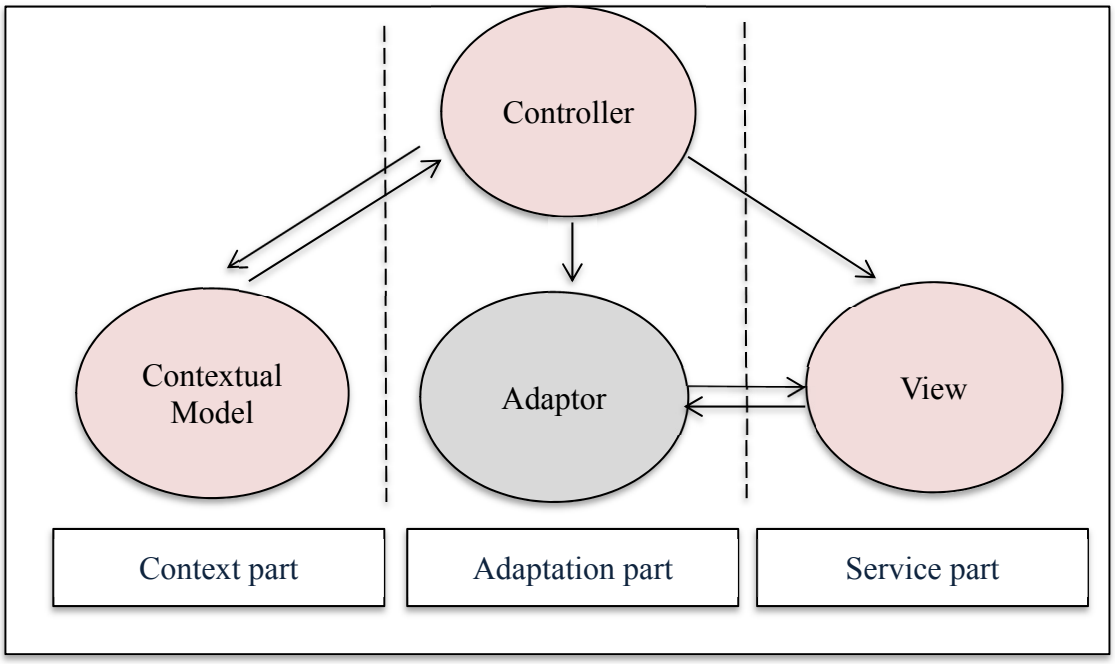


Figure 3.1 MVCA architecture

3.4 Detailed MVCA Architecture

In terms of software, the system consists of four main components: The controller, contextual model, adapter and view, as shown in Fig. 3.2 (1) The controller receives a variety of contexts sensed from the environment.

The context is then interpreted in a suitable form to be comprehensible and used by other components. (2) The contextual model represents the storage of all context information. This component contains sensed data associated with the appropriate services, as well as a pre-installed list of user preferences. Moreover, it is responsible for responding to controller requests by sending the appropriate services, if they are found. It can also receive new

services that have been adapted by the adapter component due to unforeseen circumstances. (3) The adapter module is the core component of the architecture, as it maintains the overall mechanisms and rules to adapt services to new situations. Firstly, the adapter receives contexts from the controller and attempts to reason and adapt suitable services. Then, it sends the adapted services, accompanied by their context information, to the contextual model, to be saved for future utilization. (4) Finally, the view component is responsible for determining the system output. It receives queries from the adapter to execute services such as displaying data or requests sent to actioners.

3.4.1 Controller

The controller is the first component to interact with the environment. Thus, it is related to sensors and devices implanted in the environment in order to intercept and collect any context information or situation changes. This component is the backbone of the architecture because it provides the main support for context to be used by others components. Therefore, it includes procedures for interpreting and reasoning, in order to transform gathered low-level raw data into higher- level information. The controller component performs three major interactions:

- Transfers the appropriate service from the contextual model component to the actual context.
- If the service is found, sends the service in question to the view component for execution.
- Otherwise, sends the context information to the adapter component.

3.4.2 Contextual Model

The contextual model acts as data storage for all contexts collected by the controller component. This component incorporates a context history accompanied by particular service criteria that may be used to establish trends and predict future services. It also saves also users' information and preferences in order to satisfy their needs.

The contextual model is responsible for the following interactions:

- Checks in the base if there is a match between the context received from the controller and the recorded context,
- Sends a message response to the controller component containing the appropriate service to execute (if found), else sends a negative response,
- Saves the adapted services with their context information received from the adapter component.

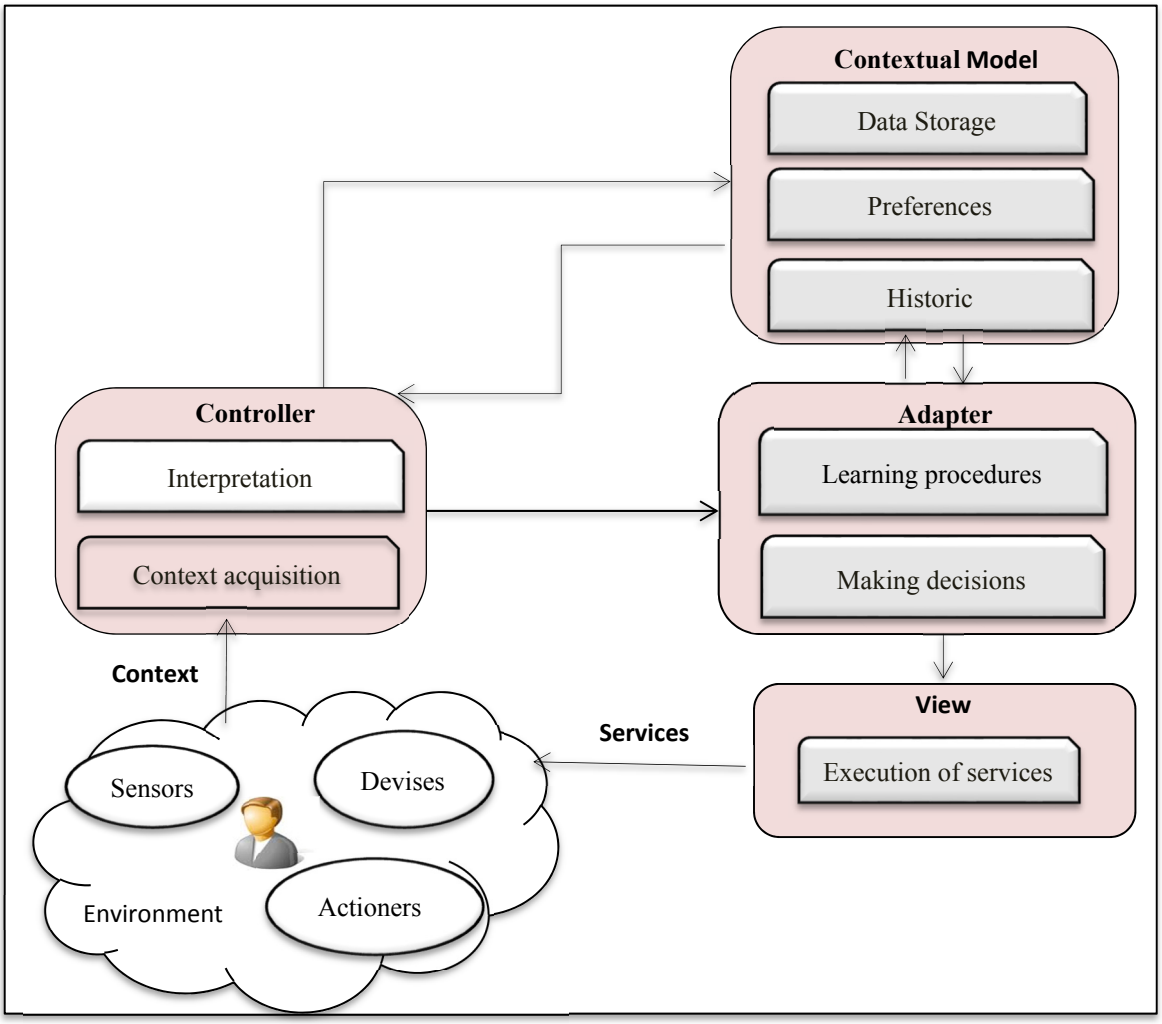


Figure 3.2 MVCA components

3.4.3 Adapter

The adapter component contains the domain-level logic of the system. It holds a database of adaptation rules. This rules define the adaptation operations that the adapter has to compute on the service it adapts and the context parameters related to it.

Besides, it is based on user's behavior learning using different types of procedures for automatic learning of the best possible services to execute, as well as a decision-making process, which is responsible for determining which service should be performed in a given situation in order to provide personalized and relevant information to end-user. Then, mainly this process involves two general requirements that form cycle. First gathering preferences of the user (with feedback or history of interaction for instance) and then adapting the engine to produce a suitable service (for instance by adjusting parameters related to preferences). For example, an inhabitant that gets up generally at 7 in the morning. One day, since he has a meeting in another city, his alarm rings at 6 'o'clock because the inhabitant set the adaptation system on the adaptation server before he went to sleep. Based on this example, we deduce that. That is why the system must be aware to the context and react to changes of this context in a continuous way. In order to get the right context of the user and his environment to fulfill adaptation process, the adapter component continuously interacts with the other components to perform the following operations:

- Receives the context information from the controller component;
- Adapts suitable services according to situation changes ;
- Retrieves useful information during the adaptation process as required;
- Sends adapted services that are found, along with their context information, to the contextual model to be saved for future use ;
- Sends adapted services to the view component for execution.

When the adaptation process was performed and the service is executed, the user must have the option to accept or reject the service providing a feedback that will be used to adjust the parameters of the techniques implemented.

3.4.4 View

The view component aims to generate and provide services to the user. It mainly contains mechanisms for interacting with actuators, which perform services at the final stage. These services are in the form of notifications or recommendations forwarded to user devices and can also be queries performed by actuators to execute services. The view component is responsible for the following operations:

- Receives services from the adapter component ;
- Sends notifications or recommendation to user devices ;
- Sends queries to actuators to perform received services.

In this study, the MVCA architecture is implemented using the Java Agent Development Framework (JADE) platform (Bellifemine et al., 2005). Figure 3.3 shows the four different types of agents representing the MVCA architecture components. It should be noted that we have added a new agent, “context,” which represents the context information sensed from the environment.

The interaction between different agents follows a process, consisting of the following phases:

- When a contextual agent obtains external information, it informs the controller agent of the available context in order to find the appropriate service ;
- The controller agent reads and processes the received information and requests the appropriate service from the contextual model ;
- When the contextual model receives context information, it checks its base to determine whether there is a match between the received and saved contexts. A negative response is sent when no match is found and a positive response when a match between service and context elements is identified ;
- If the service is identified by the contextual model agent, the controller agent sends a query to the view agent with the service’s criteria in order to be executed, as an appropriate service for user needs ;

- If the service is not identified, the controller agent informs the adapter agent about the identified context requiring an adaptation service ;
- The adapter agent uses mechanisms to adapt and determine suitable services, which are then transferred to the view agent. Moreover, it communicates the new adapted service to the contextual model agent according to its context information, to be added to the database for future work;
- The adapter agent retrieves certain information from the model contextual agent that will be used in learning processes, such as historical or preference data, in order to determine the appropriate service;
- Once the view agent receives the specific service, it responds to orders received from the adapter agent and sends notifications, recommendations, or queries to actioners installed in the environment.

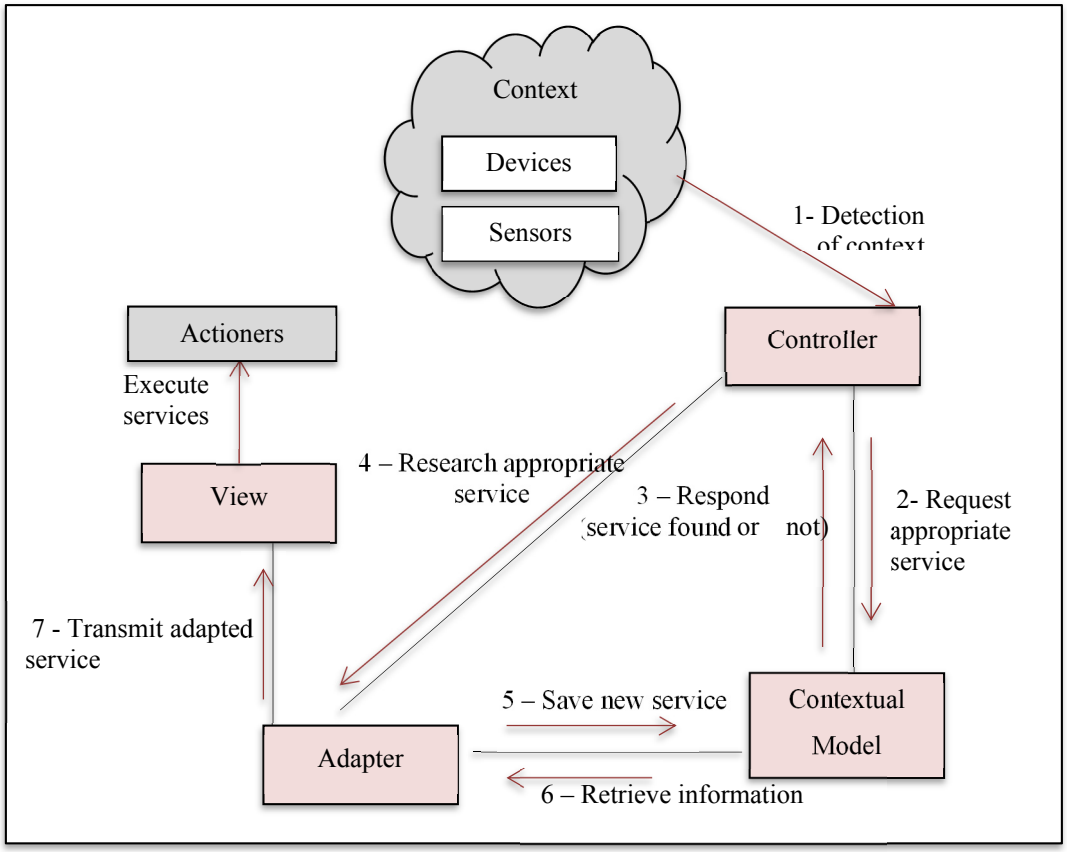


Figure 3.3 Agents in MVCA architecture

In order to verify the performance of the MVCA architecture, we implement a simple scenario using JADE agents. The scenario describes the daily routine of an employee in his office from 8:00 a.m. until 4:00 p.m.

The entire scenario can be divided into five scenes. The basic concept of each scene is that the context agent is the first to sense the context that will be interpreted and processed by the controller agent in order to infer the appropriate action to be taken, either directly by the contextual model agent or by the adapter agent. Then, the view agent receives the services to be executed. Parameters reflecting all contextual variation in the environment are taken into account; for example, working day, time, place, temperature, light and device interaction.

JADE Agents

The scenario describes the daily routine of an employee in his office from 8:00 a.m. until 4:00 p.m. The entire scenario can be divided into five scenes. The basic concept of each scene is that the context agent is the first to sense the context that will be interpreted and processed by the controller agent in order to infer the appropriate action to be taken, either directly by the contextual model agent or by the adapter agent. Then, the view agent receives the services to be executed. Parameters reflecting all contextual variation in the environment are taken into account; for example, working day, time, place, temperature, light and device interaction.

Table 3.1 illustrates the different scenes, each of which describes a particular service. We choose to identify scenes according to the three most important pieces of information: (1) contextual data, which can either be planned situations in the system or unanticipated situations requiring an adaptation process; (2) the equipment performing the action; and (3) the performed service, identified by its nature, which can be either a default service (predefined service according to a pre-established context in the system) or adapted service (new service resulting from an adaptation process in an unanticipated situation).

Table 3.1 Description of scenario scenes

	Contextual data		Equipment	Executed service	
	Planned situation	**Unanticipated situations**		**Description**	**Description**
Scene 1	Working day Time = 8:00 a.m. Opening door		Light system	Request sent to lamps. (Default service)	Lamps are turned on
Scene 2	Conference room: A50	Alternative room: B30	Cellular	Notification (Adapted service)	Message indicating the new conference room.
Scene 3	Ali lefts his office		Portable PC Light system	Requests sent to portable PC and light system (Default service)	PC and light system goes into power save mode.
Scene 4	Ali return to the office	The temperature is higher than 25 degrees.	Air-conditioning system	Request sent to air-conditioning system. (Adapted service)	Air conditioning turned on to reach the desired temperature.
Scene 5	Time = 16 a.m. Door closing. Ali lefts his office		Portable PC Light system Air-conditioning system	Requests sent to equipment. (Default service)	Equipment are switched off.

We take the third and fourth scenes, representing two different situations, as an example. The first is a planned situation; that is, a situation with a known service. The sensed context relates to the day (working day), time (8:00 a.m.) and interaction of equipment (door). This information is identified as the main input of the MVCA architecture, to be transmitted to the controller component, which in turn attempts to retrieve the appropriate service from the

contextual model module. In response, the contextual model sends an adequate service description located in its local database. This is based on the fact that, in the case of planned situations, the contextual model component has already saved different context information, such as preferences or known conditions associated with suitable services, to be executed in the future. Finally, the service will be transmitted directly to the view component, which is responsible for sending orders to turn the light on.

The second scene involves unanticipated situations that have not been previously saved in the system. The context agent senses that the conference room is occupied, a situation that is not predefined in the system. Therefore, the controller agent receives a negative response (service not found) from the contextual model. Table 3.2 illustrates different interactions between agents in the two scenes.

Table 3.2. Interactions between agents in two different scenes

Scenes	Context	A.Ctxt	A.Contr	A.MDL	A.Adap	A.View	Query
Scene 1	Working day Time = 8:00 a.m. -Opening door	E	R				Inform
			E	R			Request
			R	E			Respond (Service found)
			E			R	Execute service
Scene 2	Conference room occupied	E	R				Inform
			E	R			Request (Service not found)
			E	R			Respond
			E		R		Request
				R	E		Responds (adaptable service)
					E	R	Execute service

The controller then communicates this context information to the adapter, which can reason and adapt a suitable service; in this case, a notification about changing the conference room. Accordingly, the view sends notifications to the user's cell phone to inform them about the alternative room where the conference can be held. We now discuss the implementation results. We developed a simple application to test our architecture. Fig 3.4 displays a snapshot of the application interface, which is designed to load a scenario in a text format and launch the creation of agents for the simulation. Furthermore, the interface shows different scenes extracted from the associated scenario, with the elapsed time between the start and completion of each scene. The interface also presents an automat diagram that shows the various interactions between agents in each scenario. An agent that is known to simply receive a message is colored red. This explains the communication flow among agents in the MVCA architecture.

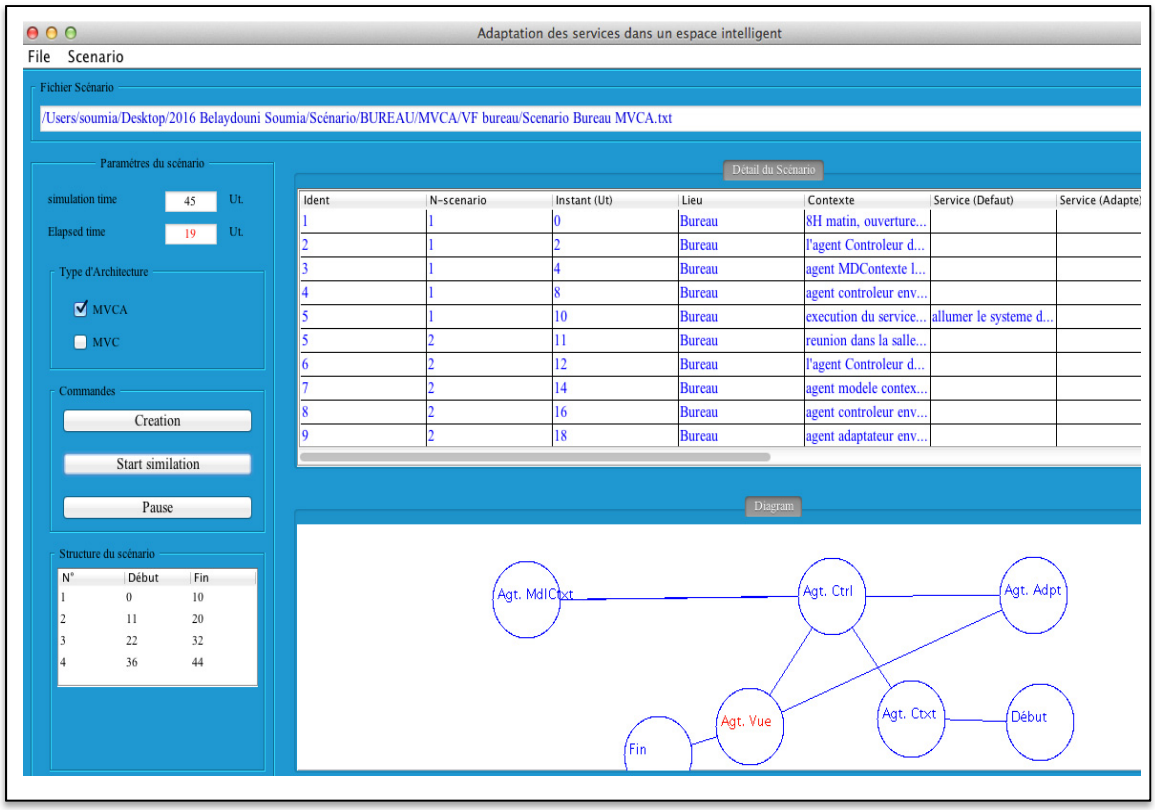


Figure 3.4 Interface of MVCA implementation

The sequence diagram shown in Fig 3.5 depicts the scenario execution, which is carried out by the exchange of messages between different agents composing the MVCA architecture. The diagram is also used to show how the different components of our architecture cooperate to accomplish the proposed scenes. The diagram illustrates the occurrence of events for invoking specific operations using different behavior messages like “inform, request.” Execution times, sender, location and context are annotated in the sequence diagram, as shown in Fig3.7.

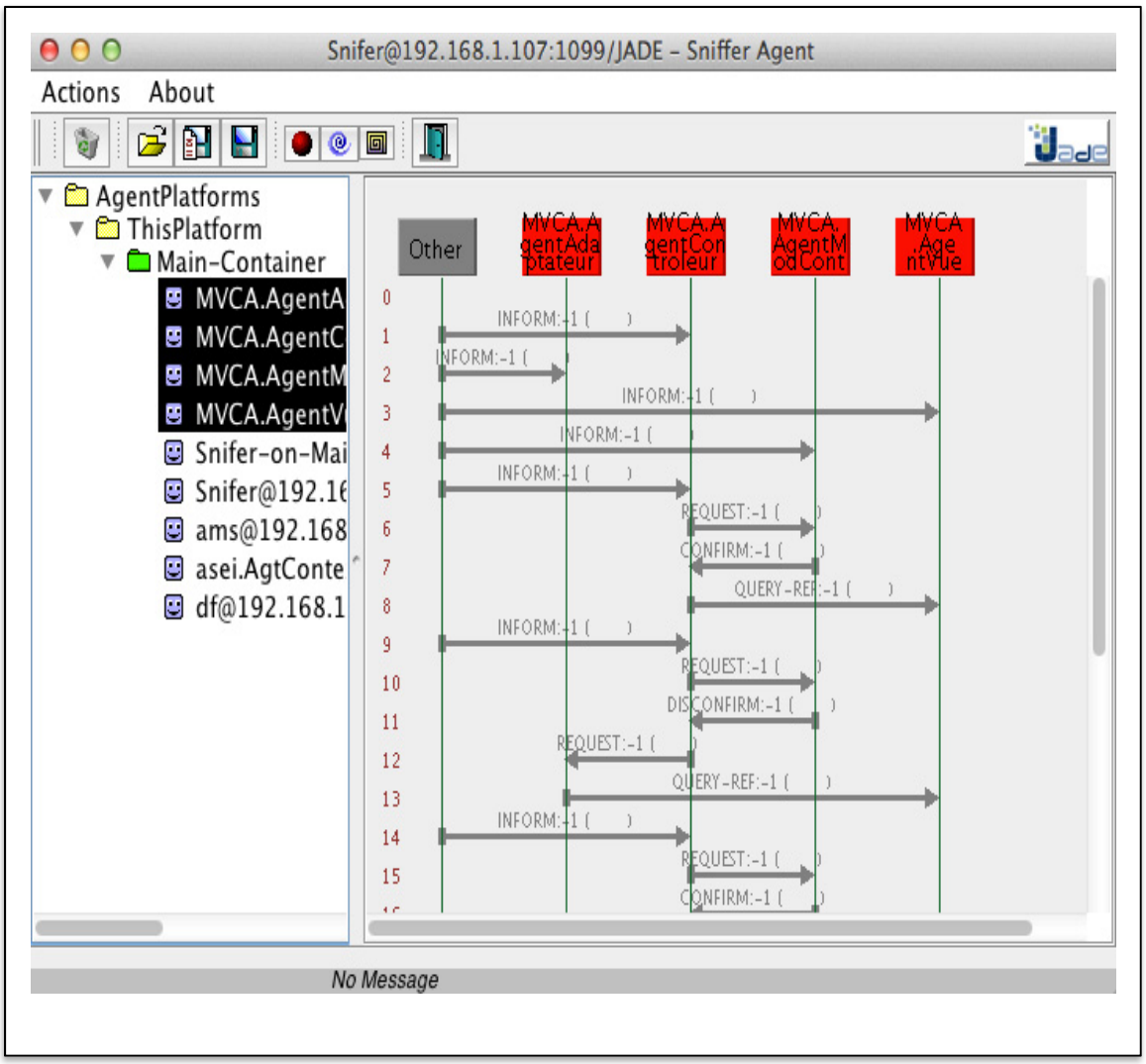


Figure 3.5 Interface of MVCA implementation

In this system, modularity is realized by separating the MVCA functionality into independent parts (Figure 3.6), so that each module focuses only on one or a small number of interactions. This method contributes to maximizing the interaction within each component (cohesion) and minimizing dependencies (coupling) between components. It also facilities the modification of components to satisfy local requirements. Moreover, it contributes to reducing system maintenance efforts and maximizing the system's reusability.

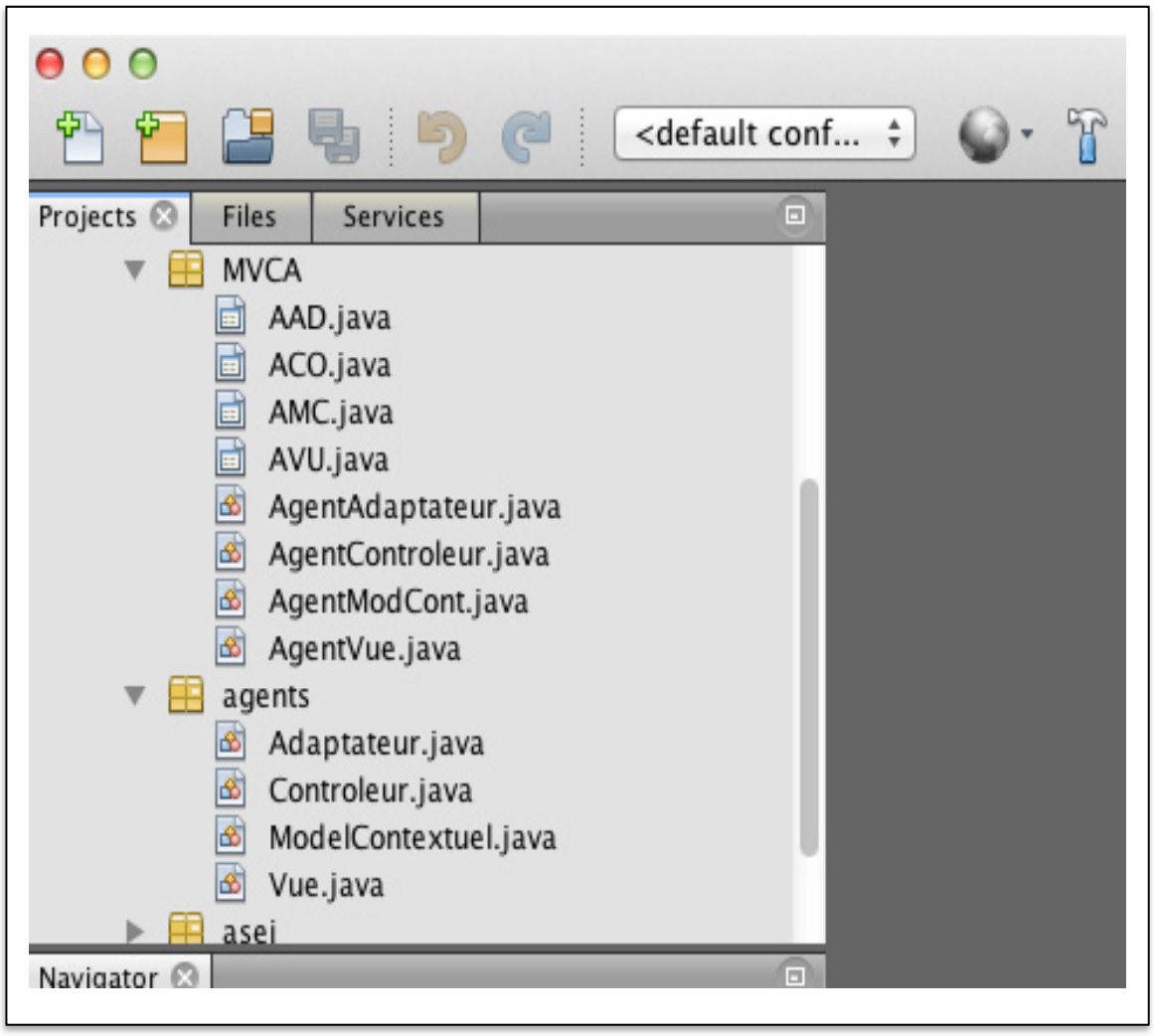


Figure 3.6 Modular decomposition of MVCA architecture

Figure 3.7 shows four tracking tables for the components, and shows interactions during the pre-established scenes at each instant. For example, in the controller table report, the first interaction shows that at 5:40:12, the controller component receives a message from the context agent with sensed information (working day, time: 8:00 a.m., door opens), and the user location is “office.” This context is transferred to the contextual model agent at 5:40:14 to determine the appropriate service (turn on lamps), which is executed by the view agent at 5:40:20. The execution times of the different component interactions components highlight the progress of the MVCA architecture. Moreover, we can retrace errors that are a result of poor component operation.

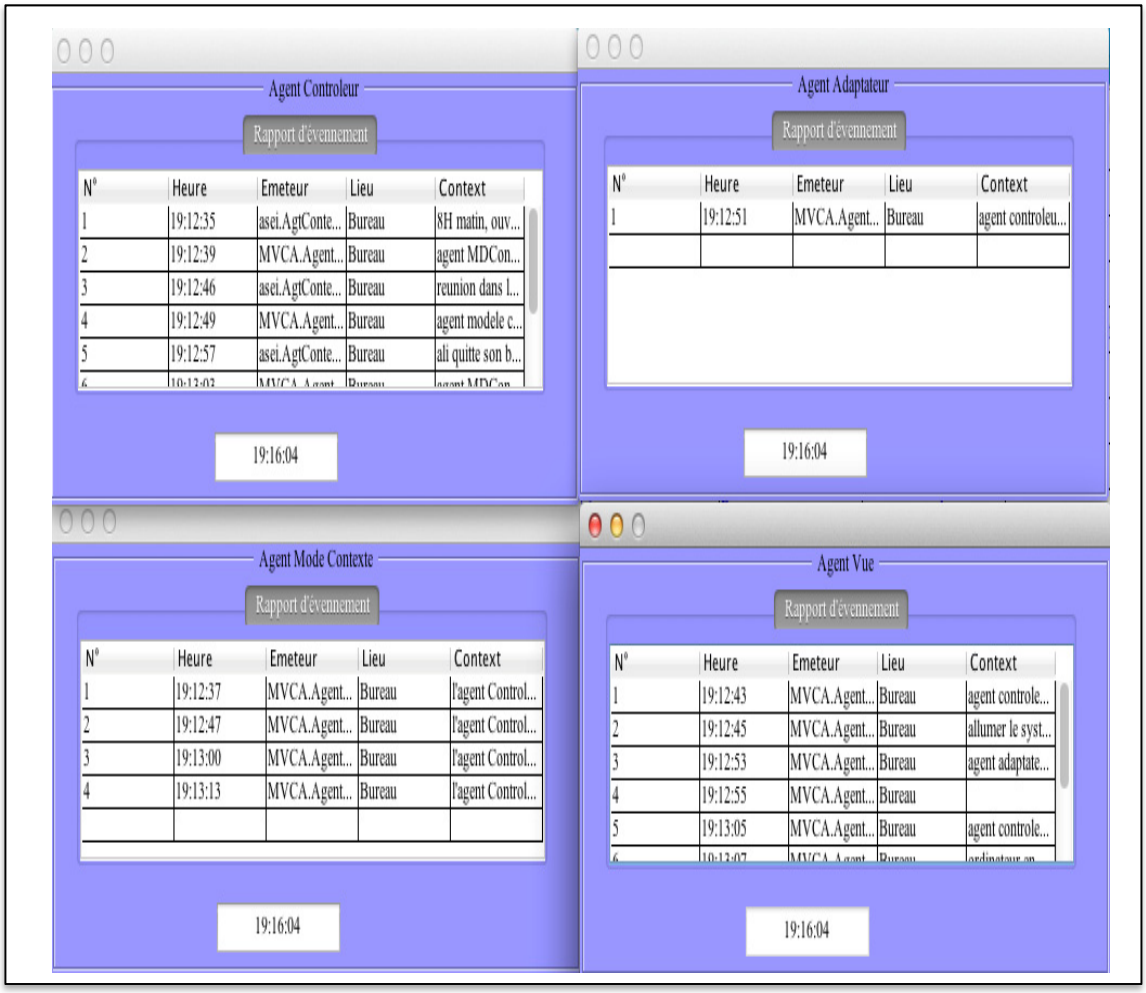


Figure 3.7 Component event report

3.5 Conclusion

Smart environments are characterized by complex systems that require a balance between transparency and context awareness. Architectures for such systems must respond to requirement space and should integrate a modular and flexible design that can produce appropriate services at the right time. In this article, we propose MVCA architecture, derived from the MVC pattern. The main idea behind this system is to decompose the overall functionalities of into four separate components, which are responsible for sensing, managing context information and reasoning to execute the most suitable service for users. This approach allows for high cohesion and low coupling, which are important measures of architecture quality. Therefore, this method facilitates system reusability and maintainability. In future works, we plan to extend the MVCA architecture to handle and process various complex scenarios and to evaluate the performance of the architecture and its implementation in depth.

CHAPITRE 4

QL-CBR HYBRID APPROACH FOR ADAPTING CONTEXT–AWARE SERVICES

Somia Belaidouni [a], Miraoui Moeiz [b], Chakib Tadj [a]

[a]Département de génie électrique, École de technologie supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[b]Al-Leith computer college, Umm Al-Qura university, Saudi Arabia
21421 Makkah, Saudi Arabia

Cet article a été soumis dans «Journal of Intelligent Systems » (jisys) le 10 avril 2018

Abstract

A context-aware service in a smart environment aims to supply services according to user situational information, which changes dynamically. Most existing context-aware systems provide context-aware services based on supervised algorithms. Reinforcement algorithms are another type of machine-learning algorithm that have been shown to be useful in dynamic environments through trial-and-error interactions. They also have the ability to build excellent self-adaptive systems. In this study, we aim to incorporate reinforcement algorithms (Q-learning) into a context-aware system to provide relevant services based on a user's dynamic context. To accelerate the convergence of reinforcement learning (RL) algorithms and provide the correct services in real situations, we propose a combination of the Q-learning and case-based reasoning (CBR) algorithms. We then analyze how the incorporation of CBR enables Q-learning to become more efficient and adapt to changing environments by continuously producing suitable services. Simulation results demonstrate the effectiveness of the proposed approach compared to the traditional CBR approach.

Keywords: context-aware service; smart space; auto-adaptation, reinforcement learning; Q-learning; supervised learning; CBR.

4.1 Introduction

There is an increasing demand for adaptive systems that dynamically change their behavior in real-time in response to changes in user preferences and contexts. Such systems come in many different forms, but they generally follow the same process to respond to user needs.

This process is called an adaptation loop (Betty and al., 2009). This loop begins by capturing and analyzing changes in context, then using an adaptation mechanism to determine the appropriate service. Finally, the system executes the correct service to meet user expectations. Adaptive systems are context-aware systems that sense changes in their environments and respond by changing their behavior and/or structure appropriately (Colman and al., 2014). Such systems are able to adapt their services in response to feedback from the surrounding environment. Context-aware systems are examples of adaptive systems, which must (a) model, process, and manage their environmental information, and (b) adapt according to changes in their environments (Hussein and al., 2010). The origin of the term "context awareness" is attributed to Schilit and Theimer (Schilit and al., 1994), who asserted that context sensitivity is the ability to discover and react to changing situations. The main goal of context-aware systems is to provide users with relevant information and/or services based on their current context. Therefore, the notion of context is critical and several authors have proposed different definitions (Schilit and al., 1994; Brézillon and al., 1999; Meissen and al., 2004; Kayes and al., 2014; Miraoui and Tadj, 2007; Abowd and al., 1994). The most-cited definition describes context as any information that can be used to characterize the situation of an entity (person, object, or physical computer). In general, we can consider context as any information that is relevant to the user and their surroundings. The main difficulty when dealing with context is how to properly adapt services in view of the high dynamicity of contextual values. A change in one contextual feature may generate changes in many other contextual ones. That is why context-aware systems must be adaptable to highly dynamic user situations and context models must be able to capture the dynamic interactions between contextual features.

Several studies on context-aware adaptive systems have addressed the problem of service adaptation in context-aware systems. The majority of previous methods were inspired by supervised learning models, such as Bayesian statistics, neural networks, and naive Bayes' classifiers. Reinforcement algorithms are another type of machine-learning algorithm that have been shown to be useful in dynamic environments through trial-and-error interactions. They also have the ability to build excellent self-adaptive systems (Sutton and Barto , 1998).

However, this approach has not been widely used in context-aware systems. Our goal in this study is to utilize reinforcement algorithms in a context-aware system to respond to user needs.

Reinforcement learning (RL) represents a class of problems in which an autonomous agent acting in a given environment improves its behavior by progressively maximizing a calculated function based on a succession of scalar responses (rewards), which are received from the environment (Dorigo and Bersini, 1994). In other words, instead of learning from examples provided by an external supervisor, RL is accomplished by directly interacting with an environment. Therefore, incorporating RL into context-aware systems to select and adapt services according to user situations should enable the system to dynamically execute the best services. However, one of the shortcomings of RL is that the convergence of the value function and learning time can be computationally prohibitive for very complex and dynamic systems, such as smart spaces. Applied reinforcement algorithms can classify an entire group of services and specify different situations that can occur for each service, but they are not capable of adjusting or adapting services based on the current context. One method to speed up the convergence of RL algorithms is to utilize previous domain knowledge stored as a case base. This method incorporates case-based reasoning (CBR) techniques into existing reinforcement algorithms to facilitate the modification and selection of optimal services. This study investigates the combination of RL techniques and CBR algorithms to select the best services and adapt them according to contextual changes.

The remainder of this paper is organized as follows. Related work on context-aware reasoning systems is summarized in Section 2. In Section 3, an overview of the background knowledge required for the reinforcement approach is introduced. This is followed by an overview of CBR algorithms in Section 4. In Section 5, our hybrid approach is described in detail. In Section 6, we present a simulation scenario to objectively demonstrate how our approach performs and discuss the obtained results. Finally, conclusions are presented in Section 7.

4.2 Related work

Several studies have been performed on the adaptation of services in smart spaces and considerable effort has been invested to create and test different reasoning algorithms that are able to adapt and produce suitable services according to changes in an environment. Yuan and Herbert introduced a personalized, flexible, and extensible hybrid-reasoning framework for Context Aware Real-time Assistant (CARA) systems in smart-home environments to provide context-aware sensor data fusion and anomaly detection mechanisms that support the activities of daily living analysis and alert generation (Yuan, and Herbert, 2014). Adaptive control of home environments (ACHE) (Mozer, 2004) is an adaptive-house model that controls the comfort systems in a home, such as lightning, ventilation and air, and water heating (Colman and al., 2014; Hussein and al., 2010). The objective of ACHE is to use reinforcement algorithms, specifically Q-learning, to predict inhabitant actions and adjust systems to decrease energy consumption. Additionally, authors incorporated neural networks to predict which zone(s) would become occupied in the next timeframe. Wang et al. (2014) proposed a novel learning-classifier system based on the co-evolution eXtended classifier system (XCS) to perform context-aware mobile service adaptation.

The main concept in their system is to map user contexts onto classifier conditions and mobile services onto classifier actions. The generation of adaptation rules is then transformed to provide mobile service adaptation through the matching and competition of classifiers. Mandato and al. (2000) proposed an approach to modeling and implementing context-aware adaptive software systems. They mainly considered explicit user preferences and implicit user circumstances. This context enables users to gain more control over the services they access. They proposed a component model (CAMP) that incorporates explicit support for the definition of system functionality, context, and management. Ni and al. (2009) proposed a context-dependent task approach to manage pervasive services. They adopted an implementation of the case-based reasoning (CBR) method to recognize tasks, which facilitated task-oriented system design in smart home environments. Lum and Lau (2002) proposed a content adaptation system that can determine the optimal content version for

presentation and the best strategy for deriving and generating that version. Their system's most crucial component was the decision engine, which utilized decision trees to determine the optimal content for presentation by focusing primarily on user preferences, intended target device capabilities, and network conditions. The CARA system (Yuan and Herbert, 2014) adopted a context-aware hybrid-reasoning framework by means of case-based reasoning and fuzzy rule-based analogy interpretation of sensor data within a wider context to perform reasoning with all available knowledge for situation assessment and perform actions based on the results of the reasoning process. Miraoui and al. (2017) presented a hybrid approach for context-aware service adaptation in a smart living room based on naïve Bayes', fuzzy logic, and CBR. Kabir and al. (2015) focused on the use of two effective learning algorithms: the back propagation neural network and temporal differential (TD) class of RL to predict the demand of home users and proactively provide the proper services. Ali and al. (2008) analyzed the prerequisites for user-centered prediction of future activities and presented an algorithm for autonomous context aware user activity prediction. They proposed combining the fuzzy-state and Q-learning algorithms to predict or anticipate a future situation in an assistive environment. Hongand al. (2009) proposed an agent-based framework for providing personalized services using context history via context-aware computing. They mainly focused on context history to derive preference rules and recommend personalized services.

4.3 Reinforcement Learning (RL) and the Q-learning algorithm

RL algorithms have been applied successfully to the online learning of optimal control policies in Markov decision processes (MDPs) (Xu and al., 2002). RL is an approach where an agent acts in an environment and learns from its previous experiences to maximize the sum of rewards received based on its action selection (Sutton and Barto, 1998). The reward is classically a continuous function between −1 and 1, where 1 corresponds to satisfaction, −1 to disapproval, and 0 to no opinion. Q-learning falls under the class of RL algorithms (Rakshit and al., 2013). It uses the temporal difference method to solve problems by estimating a value function called the Q-value for each state-action pair. For separate cases,

the Q-learning algorithm assumes that the state set S and action set A can be divided into discrete values. At a given step t, the agent observes the state (context) c ∈ C and then chooses a service (action) s ∈ S. After executing the action, the agent receives a reward r that reflects how desirable that action was (in a short-term sense). The state will change to the next state c_{t+1} based on the action s_t. The agent will choose the next action s_{t+1} according to its prior knowledge. This process is illustrated in Figure 4.1. The Q-learning principle largely follows a Markov decision process that is defined by C, S, R, and P as:

• C is the set of all possible states or contexts in the environment ;

• S is the list of possible actions or services to execute ;

• R is the reward function that indicates the opportunity cost to choose action a in situation s ;

• P is the transition function modeling the environment. P(c, s, c′) is the probability of being in a situation c′ when applying a service s in a situation c (Bianchi and al., 2009).

This process forms the Q-matrix. The learning algorithm executes best the possible *action* in a particular *state* to reach the goal state, which is assigned by the agent(*s*). Finally, a Q-matrix is obtained through a finite number of iterations using learning parameters. The maximum value of *Q* is calculated by considering all possible actions at a particular state.

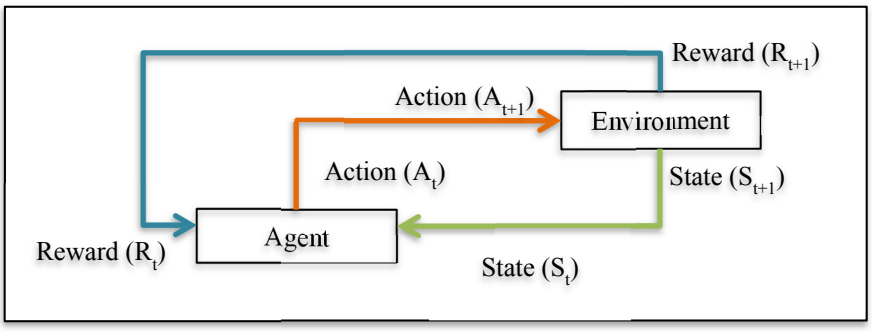


Figure 4.1 Reinforcement approach

4.4 Case Based Reasoning (CBR)

CBR has become one of the most successful applied methods of artificial intelligence (machine learning). The CBR mechanism uses knowledge regarding previous situations

(cases) to solve new problems by finding a similar past case and reusing it in a new problem situation (Aamodt and Plaza, 1994). The fundamental concept is the assumption that similar problems have similar solutions, meaning the CBR algorithm retrieves previously solved problems similar to the current problem and attempts to modify the previous solution to fit the current problem. The CBR process relies on three main operations: retrieval, adaptation, and case memorization (Kolodner, 1997; Riesbeck and Schank, 2013). Adaptation is the heart of the CBR process and is performed by the inference engine. The case-based inference engine of a CBR system solves new problems by retrieving and adapting previous problem-solving experiences (Ahmed, 2014), as shown in Figure 4.2. Adapting services in CBR involves obtaining a context description, measuring the similarity of the current context to previous contexts stored in the case base along with their known solutions, retrieving one or more similar cases, and attempting to reuse the solution(s) of the retrieved case(s), possibly after adapting them, as depicted in Figure 4.2. Other steps that are typically found in CBR systems are the evaluation of the proposed solution, revision of the solution (if required in light of its evaluation), and retention (learning) of new cases (if the system has learned to solve a new problem). In our study, we used the case definition proposed by Ros (2009; A2007; J2007), which is composed of three parts: the context description (C), service description (S), and case scope (P), which are formally described using three-tuples: case(C, S, P). The context description C corresponds to a sensed user situation in which the case can be used. In general, it is a combination of various captured contexts surrounding the user. For example, in a smart environment; a context description may include the user's localization, activity, or time or battery charge level of the user's mobile phone. The service S describes the solution or action that must be executed to respond to user needs. Case retrieval is the most important process in a CBR system and is considered to be the most basic component (Richter and Weber, 2013). It is typically driven by a similarity measure between new context and solved problems in the case base. When a new situation occurs, case retrieval indicates how similar a context (problem) and case are.

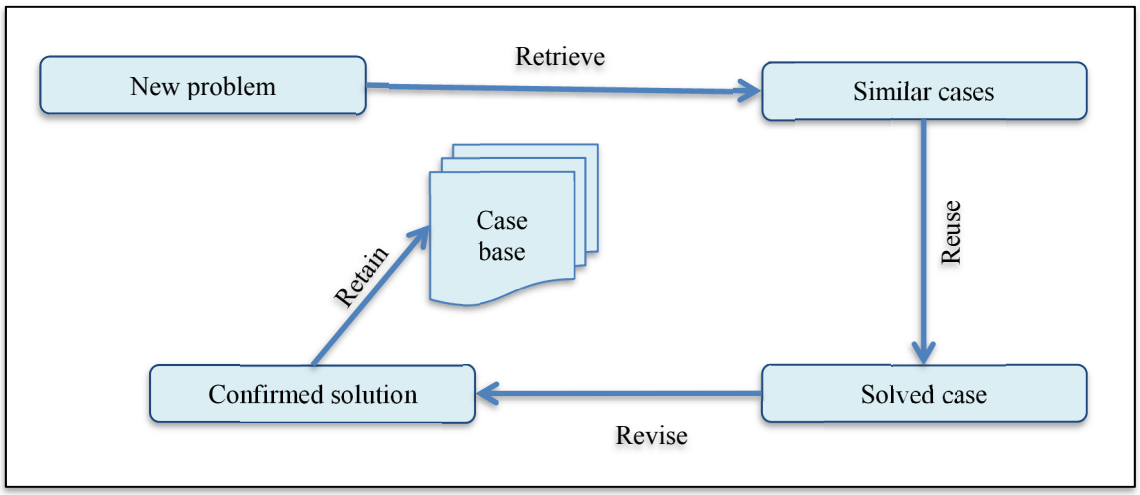


Figure 4.2 CBR Cycle

In CBR, the global approach to measuring the similarity between cases is primarily based on calculating local similarities between attributes, which can be customized for each case base (Richter and Weber, 2016). In most cases, the similarity function is defined by the distance between the attributes of the new context and those of the contexts in the case base. This function is expressed by equation (1):

$$\text{Similarity}\ (\text{context}_{new}, \text{context}_{old}) = \sum_{i=1}^{n} w_i \times sim\ (a_i^{\text{context}_{new}}, a_i^{\text{context}_{old}}), \qquad (4.1)$$

where context_{new} is the new captured context, context_{old} is the context stored in case base, a_i is the attribute of either the new context or old context, and w_i is the weight of attribute a_i. The sum of the similarities of all attributes is calculated to provide a measure of the overall similarity between the old context in the case base and the new context.

4.5 The proposed approach (QL-CBR)

A smart environment is composed of different devices communicating in order to provide proactively adapted services to users (resp. inhabitant). Each device can provide different services and each service has different forms according to changes in the environment. The extent of service changes and their forms depends on many factors, including the user

environment and their preferences. The adaptation process consists of adjusting services or their forms depending on the sensed context. In order to provide Q-learning with the capability of adapting services to changes in context, we propose a novel approach called Q-learning case based reasoning (QL-CBR), which enhances the Q-learning algorithm with the abilities to classify contexts and their appropriate services, and adapt services to the current situation. The process of Q-learning adaptation using CBR is illustrated in Figure 4.3. This process consists of a Q-learning phase, retrieval phase, and adaptation phase.

4.5.1 Q-Learning phase

First, we consider an autonomous agent interacting with a smart environment via perception and action. For each interaction, the agent senses the current context and chooses an action (service) to perform. A reward (r) is then given to the agent to indicate the desirability of the resulting situation. The best services can then be identified through a trial-and-error process. The goal of this state is to find the most suitable service according to the sensed context. Assume that there are K contexts $\{C_1, C_2, \ldots, C_K\}$. Each context C is composed of A_m attributes $\{A_1, A_2, \ldots, A_M\}$ and can perform S_N services $\{S_1, S_2, \ldots, S_N\}$. The rows of the Q-matrix are composed of different attributes that represent various states, as shown in Table 4.1. In other words, each line is a combination of A_k attributes that represent one context that requests the execution of an adequate service. To set this service as a goal, we associate a reward value with each attribute (i.e., link between nodes). The different attributes that lead to the goal have an instant reward of 100. The others have zero reward. At a particular step t, the agent observes the context C_t and then chooses a service S_t. After executing the service, the agent receives a reward r_{t+1}, which reflects how desirable that service is. The context will then change into the next context C_{t+1}. The agent will then choose the next service S_{t+1} according to the best acquired knowledge. The goal of Q-learning is to learn a policy π by learning the service values. The policy π is a rule that the agent follows in selecting actions with the highest value in each state [27].

The Q-learning update rule is:

$$Q_{t+1}(c_t, s_t) \leftarrow Q_t(c_t, s_t) + \alpha[r_{t+1} + \gamma \max Q\,(c_{t+1}, s_t) - Q_t(c_t, s_t)], \qquad (4.2)$$

where γ is the discount factor ($0 \leq \gamma < 1$) and α is the learning rate.

When the Q-matrix approaches a state of convergence, we conclude that our agent has learned the most optimal paths to the goal service. Tracing the best sequences of attributes is as simple as following the links with the highest values at each state. At the end of the learning process, the Result-Q contains all possible contexts with theirs appropriate services to form the principal rules that are used in the next phase.

Table 4.1 Composition of the Q-matrix

Contexts	C1	C2	…	C_K		S_1	…	S_N
C1	C_{11}	C_{12}		C_{1M}		..	..	..
C2	C_{21}	C_{22}		C_{2M}		..	..	..
⋮	⋮		⋮	⋮		⋮		⋮
C_K	C_{K1}	C_{K2}	…	C_{KM}		..	…	..

4.5.2 Retrieval phase

In this stage of the procedure, we use the retrieval phase of the CBR algorithm to retrieve similar cases to a captured new context ($context_{new}$). For this purpose, we use WordNet (Fellbaum, 1998), which is a lexical database based on semantic similarity measures. WordNet organizes an entire word set (synset) into a hypernym tree that can be used for reasoning based on the similarity between two words. The Result-Q must be visited to retrieve all rules with a similar profile to $context_{new.}$ This similarity is determined by calculating the weights between attribute rules and those of $context_{new}$ as follows: Suppose that our captured context is $context_{new}$ = ((A_1, A_2, …, A_M) and the first rule in Result-Q is defined by Rule1 = (A r1, A r2, …, A rM). Then, the similarity between them is represented

by W (newcontext1, Rule1)=(w1, w2, …, wz), where w_i indicates the weight value of the attributes Ai and A rM. This value corresponds to the degree of similarity between two contexts.

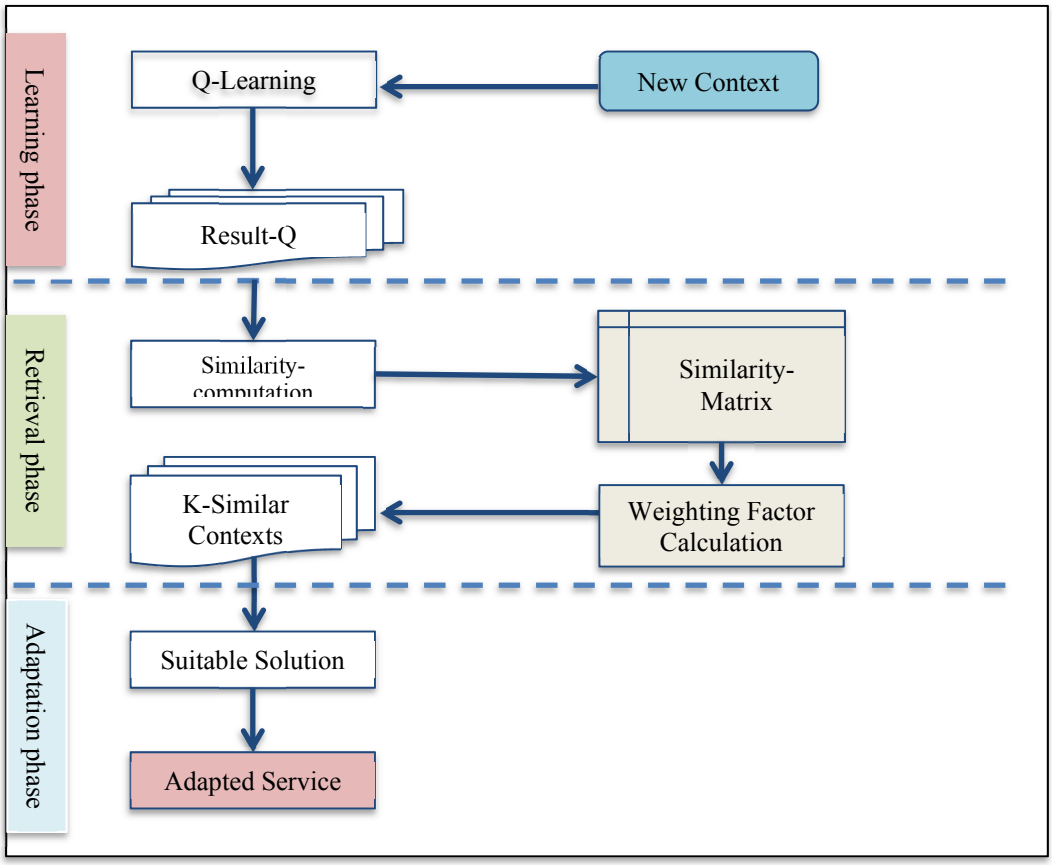


Figure 4.3 Process of QL-CBR

Next, searching for the most suitable result is performed by calculating Euclidian distances, which is repeated for each rule in the Q-Result, to find the K-nearest cases using a distance measure and selecting the class of the majority of these K cases as the appropriate solution.

4.5.3 Adaptation phase

The adaptation phase represents the process of transforming the service of the most similar retrieved case from Result-Q into an appropriate solution for the new context. At this stage of

the process, we use the minimum distance between the retrieved contexts to resolve and determine the appropriate service. Once an adequate case is selected, if the result coincides completely with the old context, the appropriate service is the same as the result and will be considered as a default service. However, if some attributes only coincide partially, the appropriate service is only the most-similar rule and will be considered as adaptable.

4.6 Application scenario and simulation

For an experiment to analyze the proposed approach, suppose that there is a context-aware scenario for a user named Jack, who is an employee of the Gaz Company and is studying for his Master's degree in Biochemistry. He spends his days at work and attends the university in the evening to attend or revise his courses. At night, he returns home. Jack often uses his cellular phone for his work or studies to meet his needs. In this scenario, we consider the cellular phone as the service-providing device. In our work, we focus on incoming call notifications as services that can be changed according to the current context. The adaptation process in the first step consists of collecting the set of data that change the form of the service. This data includes fives attributes that represent the context of the scenario namely: day, time, localization, battery, and activity.

The second step consists of specifying the set of possible values for each context element as follows:

- Day type (weekday, weekend) ;
- Time (morning, afternoon, evening, night) ;
- Localization (home, company, restaurant, university class, university library) ;
- Battery Charge Level (high, low) ;
- Activity (working, eating, studying, resting, sleeping).

In the default state, the cellular phone notifies Jack of incoming calls by using ringtones. However, in some situations, it can notify Jack of incoming calls using other methods,

including silent notification, audio calls, vibration, and video calls. Table 4.2 lists the scenario contexts and services according to context changes.

Silent: when the user is sleeping. In our case, we assume that the user is usually sleeping at night.

Audio call: we assume that the user receives audio call during his free time or when he is at home or at university.

Vibrator: we assume that the user can receives calls using vibrator form of his cellular when he is studying either in the university library or at home.

Video-call: we assume that the user can receive incoming calls notification using video call form when he is eating or resting anywhere (home, restaurant). We need also to specify the contradictory rules to eliminate meaningless context configurations that are:

- At week-end, jack has not studies or work ;
- In case Jack drives his car, he cannot eat, work or sleep ;
- In the morning of the week-day, the localization cannot be university–class or university-library ;
- In the night, the localization cannot be the company ;
- At university, Jack cannot sleep or work.

The final step consists of using the trained model to choose the most appropriate services according to the current context. For the implementation of the application scenario, we used the Java 1.8 platform. Our goals were to find a good tradeoff between all parameters, obtain the best performance for all schemes, and evaluate the schemes objectively. The experimental settings for these algorithms were set as follows: The reward was r = 100 for each attribute that could reach the target. Otherwise the reward was zero. The discount factor was $\gamma = 0.8$ and the learning rate was $\alpha = 0.01$.

Table 4.2 scenario's contextual information and services according to changes

Device	Form of Context		Current Service	Modified Form of Context	Adapted-Service
Cellular phone	Context-type	Attribute-id	"Audio call" Or "Vibrator" Or "Video call"	Activity= "meeting"	Silent
	Day type	weekday Weekend		Activity= "sleeping"	
				Localization= "universityclass"	
				Charge Level= "low"	
	Time	Morning Afternoon Evening Night	"Audio call" Or "Video call" Or "Silent"	Localization= "car"	Vibrator
				Localization= "home" Activity= "studying"	
				Localization= "university library"	
	Location	Home Company Restaurant University class University library	"Audio call" Or "Silent" Or "Vibrator"	Activity= " eating"	Video call
				Activity= "resting"	
				Day type="weekend"	
	Battery	High Low	"Silent" Or "Vibrator" Or "Video call"	Day type="weekday"	Audio call
				Localization=" company"	
	Activity	Working Eating Studying Resting Sleeping		Activity="resting"	

Figure 4.4 illustrates the tree of relationships between attributes and services. We note that user profiles and services under the root begin with the day type attribute and end with

services. It should also be noted that the different combinations of attributes form a context that reaches an appropriate service and that a single service can be reached by different contexts.

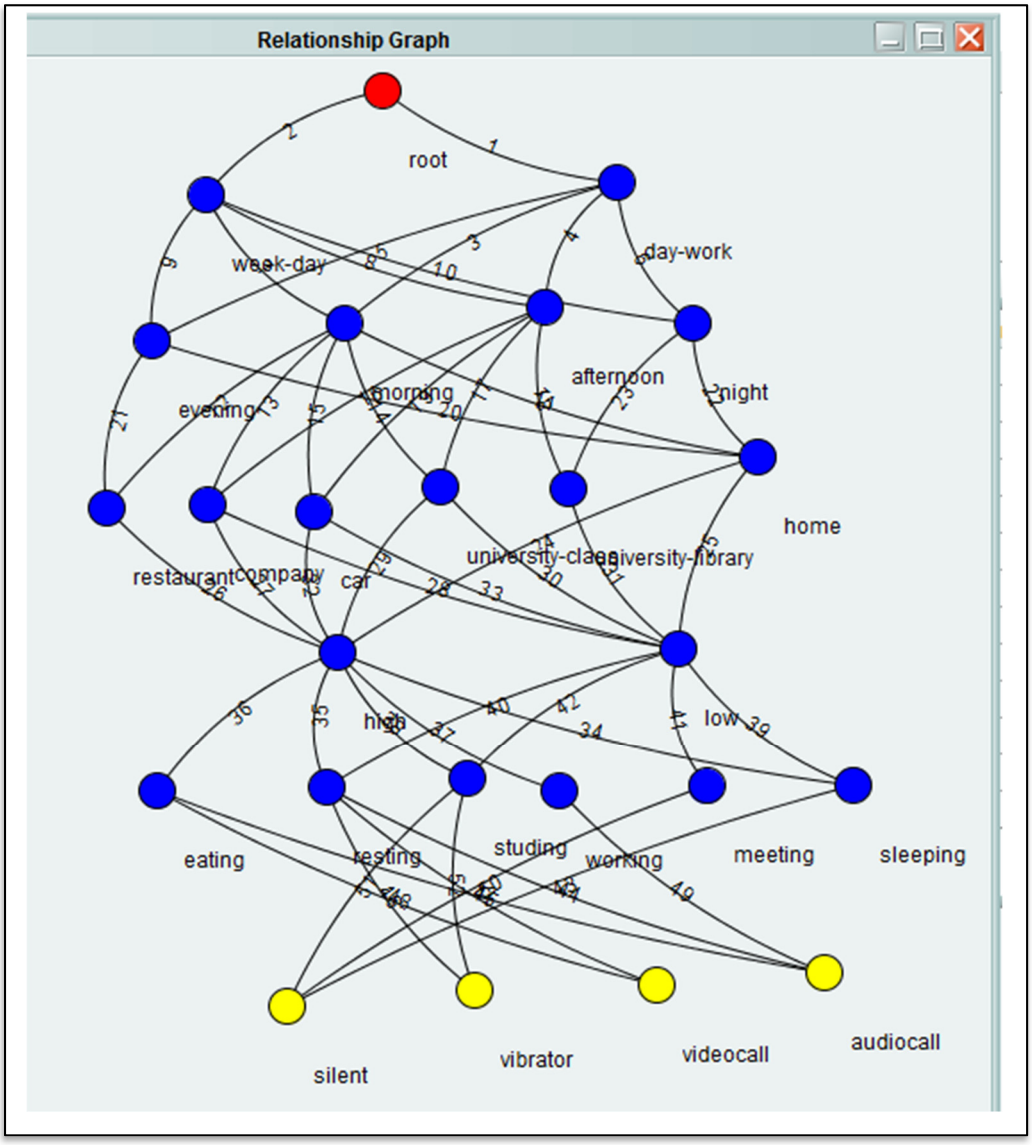


Figure 4.4 Relationship Graph

Figure 4.5 presents the final matrix of the Q-learning algorithm. At this level, the agent can converge to the goal (adequate service) in an optimal manner. It can trace the sequence of appropriate attributes by finding the service that maximizes Q for this state.

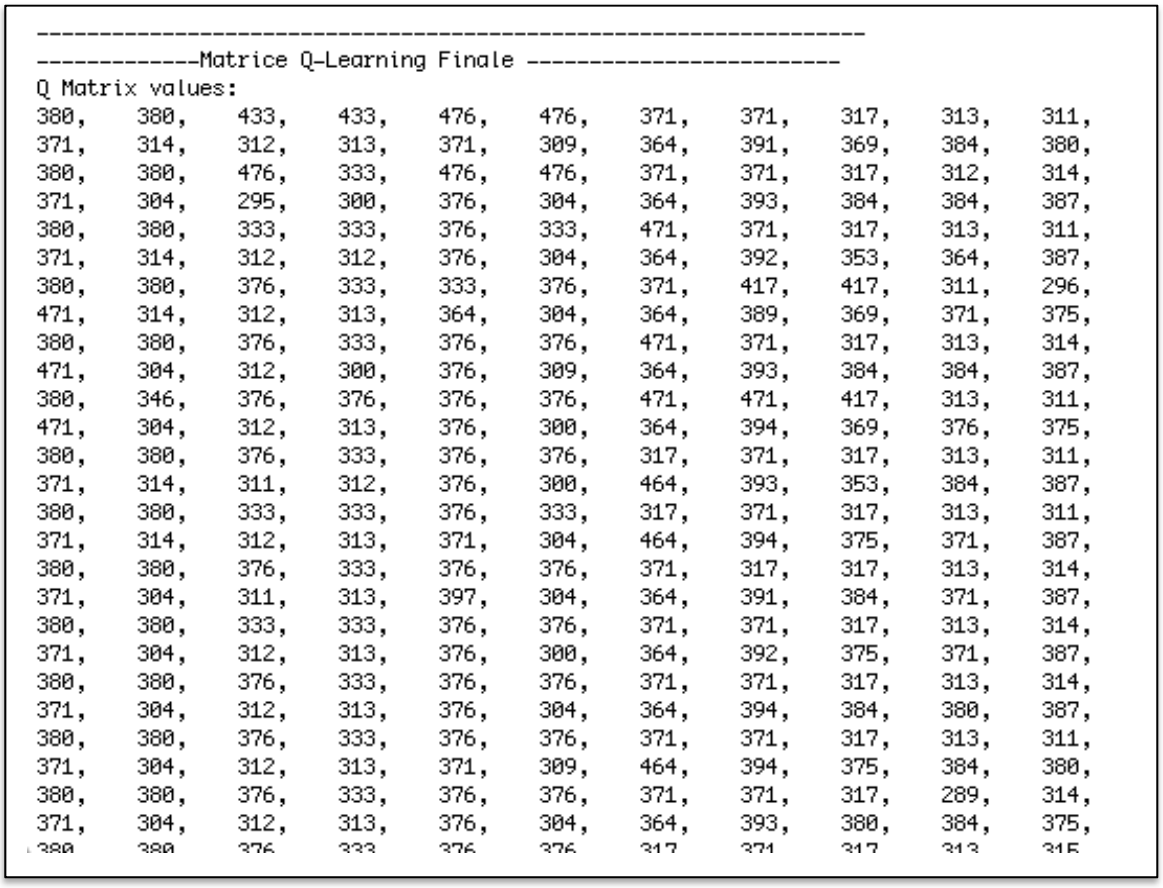


Figure 4.5 Final matrix of Q-learning

Figure 4.6 presents the Q-learning results. We can deduce that the Q-learning classified all situations with the appropriate services. For example, the silent service was identified for five different contexts.

Following the Q-learning process, the CBR algorithm begins by calculating the similarity between the attributes of the new context and the case base using WordNet measures (Figure 4.7). For example, let A_1 = « night » and A_2 = « night ». We then obtain sim (A_1, A_2) = 1. If A_1 = « week-day » and A_2 = « week-end », then sim (A_1, A_2)= 0.125. If A_1 = 'company' and A_2 = « morning», then sim (A_1, A_2)= 0. At the end of this stage, we obtain the most-similar situations.

This study aims to verify the performance of combining Q-learning and the CBR algorithm. The fundamental advantage of combining Q-learning and the CBR algorithm is the ability to

adapt services to users according to their context and provide the most suitable services. In order to verify the reliability of our proposed method, we examined the results of services requiring reformulation. If an adapted service had the same form when using the CBR algorithm, then the service was considered to be well adapted. Otherwise, it was considered to be poorly adapted. The experimental results of service adaptation via QL-CBR and CBR are presented in Figures 4.8 and 4.9, respectively.

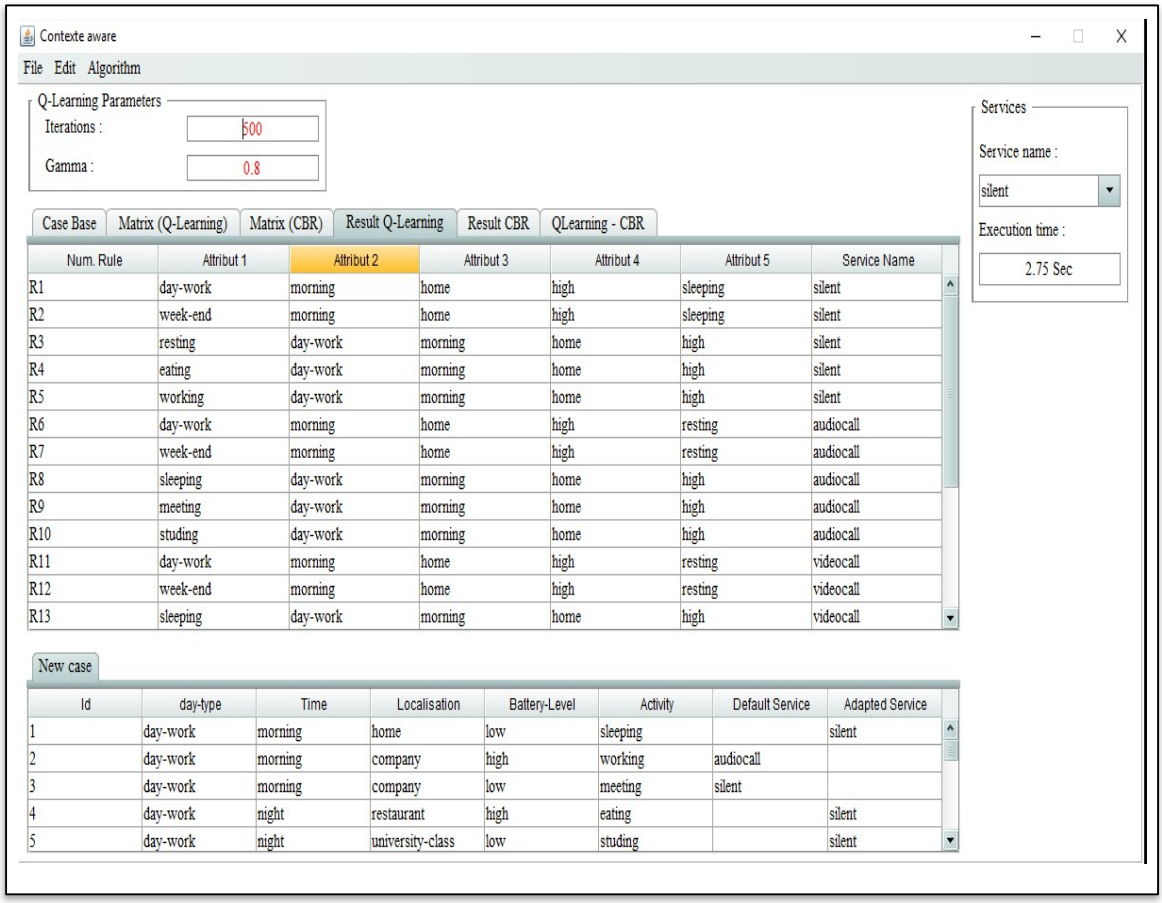
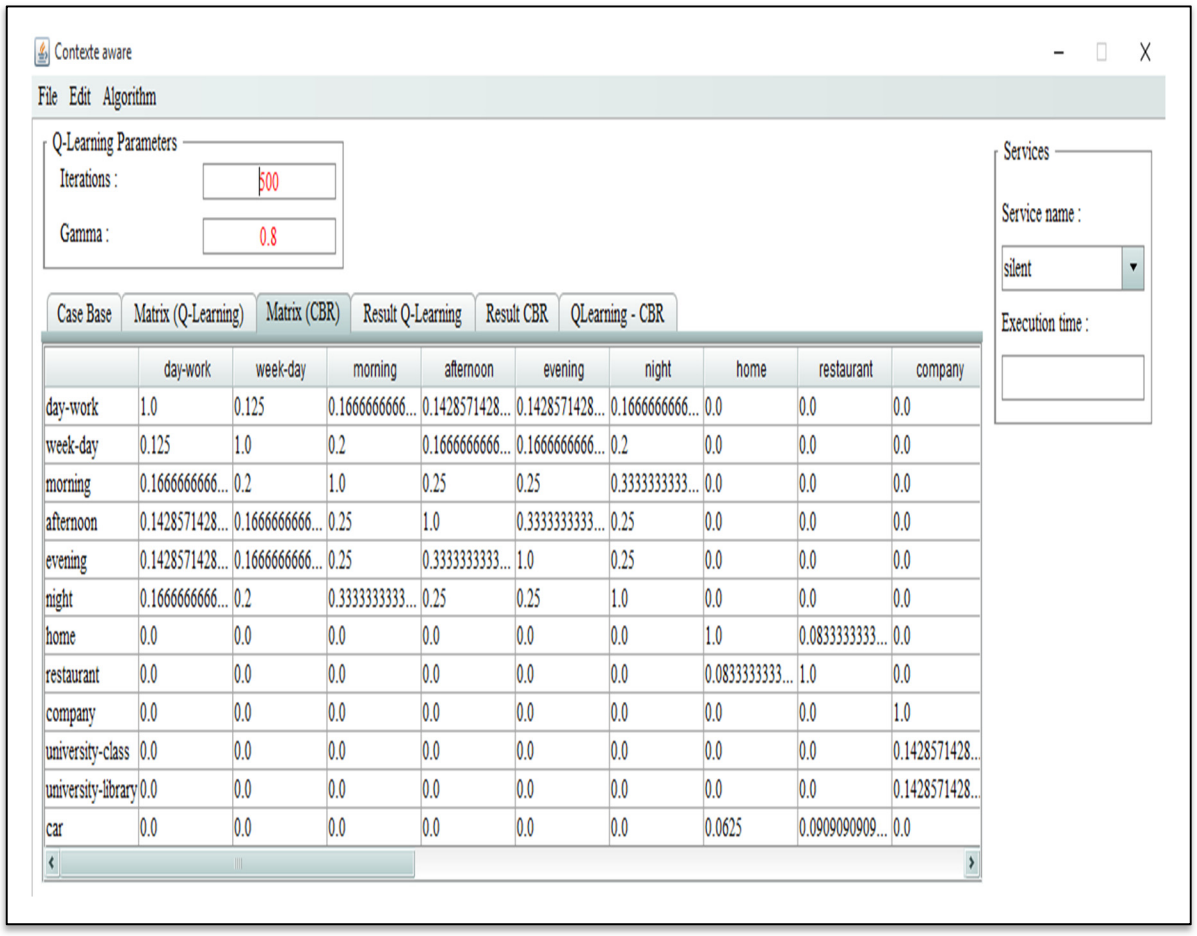


Figure 4.6 Results of Q-learning

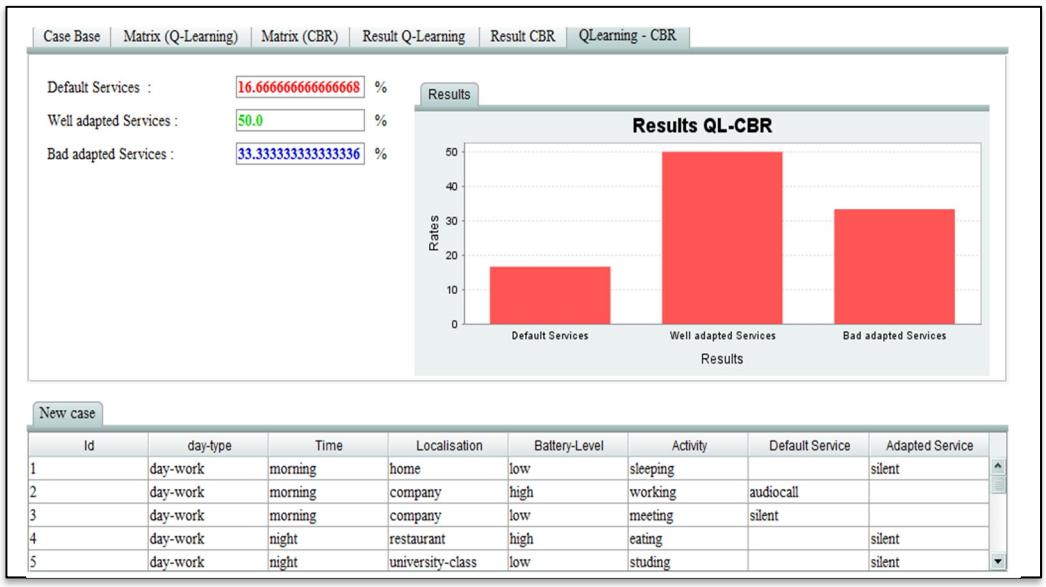
It can be seen clearly that success rate is 50% compared to CBR which has 38.09% service adapted. Moreover, the well-adapted service is 50% against 33,33% of bad adapted one. That means the percentage of well-adapted service has largely exceeded the bad-adapted one in QL-CBR, which proves the effectiveness of this method.



	day-work	week-day	morning	afternoon	evening	night	home	restaurant	company
day-work	1.0	0.125	0.1666666666...	0.1428571428...	0.1428571428...	0.1666666666...	0.0	0.0	0.0
week-day	0.125	1.0	0.2	0.1666666666...	0.1666666666...	0.2	0.0	0.0	0.0
morning	0.1666666666...	0.2	1.0	0.25	0.25	0.3333333333...	0.0	0.0	0.0
afternoon	0.1428571428...	0.1666666666...	0.25	1.0	0.3333333333...	0.25	0.0	0.0	0.0
evening	0.1428571428...	0.1666666666...	0.25	0.3333333333...	1.0	0.25	0.0	0.0	0.0
night	0.1666666666...	0.2	0.3333333333...	0.25	0.25	1.0	0.0	0.0	0.0
home	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0833333333...	0.0
restaurant	0.0	0.0	0.0	0.0	0.0	0.0	0.0833333333...	1.0	0.0
company	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
university-class	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1428571428...
university-library	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1428571428...
car	0.0	0.0	0.0	0.0	0.0	0.0	0.0625	0.0909090909...	0.0

Figure 4.7 Results of similarity calculations

Figure 4.8 presents the results of QL-CBR after 1,000 iterations. Overall, combining Q-learning and CBR seems to yield superior performance in terms of service adaptation. We noticed that the rate of poorly classified services decreased and that of well-adapted services increased with the number of iterations. We also noticed that the overall rate of adapted service increased with the number of iterations. For example, at 100 iterations, the ratio of well-adapted services was approximately 40% (Figure 4.9). However, at 1,000 iterations, this ratio reached 90%.



Id	day-type	Time	Localisation	Battery-Level	Activity	Default Service	Adapted Service
1	day-work	morning	home	low	sleeping		silent
2	day-work	morning	company	high	working	audiocall	
3	day-work	morning	company	low	meeting	silent	
4	day-work	night	restaurant	high	eating		silent
5	day-work	night	university-class	low	studing		silent

Figure 4.8 Results of QL-CBR

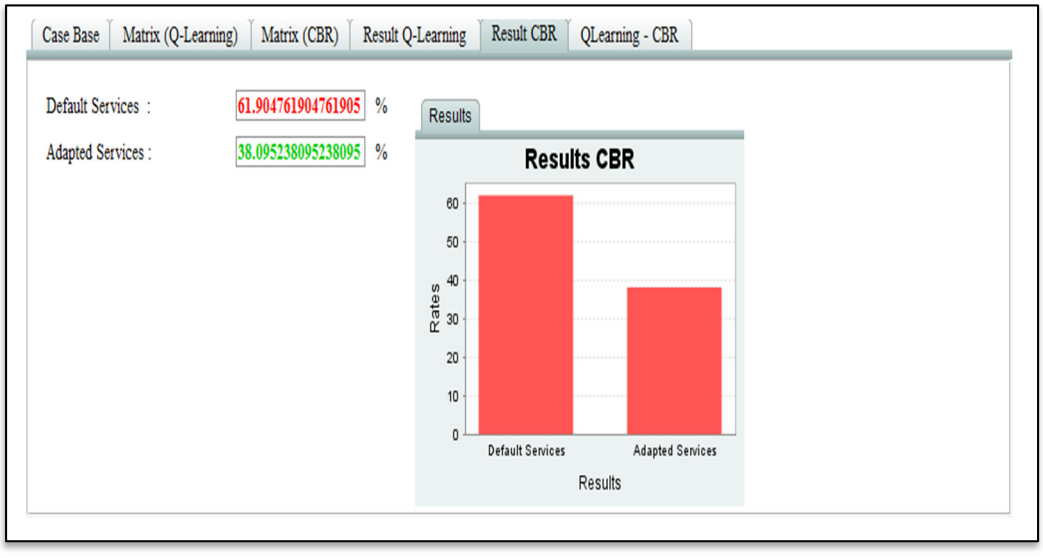


Figure 4.9 Result of CBR

4.7 Conclusion

In this paper, we introduced an RL algorithm for context-aware systems for adapting services. Our proposed approach consists of combining Q-learning and CBR, which facilitates the consideration of the current context for service adaptation. The entire process

can be divided into three phases of operation. In the first phase, we introduce the Q-learning algorithm, which has the capability to classify all situations with appropriate services. It should be noted that this result is used as the base for the next phase, which incorporates the CBR algorithm to retrieve the most-similar contexts to a new context. The final phase involves adapting one or more solutions, if necessary, to fit the new situation. The obtained results are very encouraging and indicate that our approach is able to provide reasonable services. One major remaining problem is long execution time. More generally, when using reinforcement algorithms, particularly Q-learning, in context-aware systems to respond to user needs, services must be effectively adapted prior to execution. However, the deployment of such a system in a real-world operational environment is a challenging task that still requires further investigation. Our future work will consist of testing additional reinforcement algorithms with time constraints in the learning phase and performing adaptation tasks for more than one.

CONCLUSION

Au cours des deux dernières décennies, on assiste au développement de services et d'application dédiées à l'amélioration de l'état et du bien être de ces personnes dans leurs espaces de vie. L'espace intelligent définit tout espace autour des utilisateurs, capable de se réagir intelligemment pour produire des services assurant la satisfaction des utilisateurs. Un tel espace se voit comme une panoplie d'équipements et de dispositifs hétérogènes, de différentes technologies qui s'interrogent par différents moyens de communication.

Les travaux de recherche présentés dans cette thèse ont fait l'objet de cinq contributions majeures. Ces différentes contributions pavent des chemins orientés vers la réalisation d'un système intelligent capable de réaliser une adaptation dynamique de services dans un environnement ambiant.

La première contribution a porté sur la proposition d'un modèle de qualité assez globale qui intègre toutes les exigences nécessaires pour un système SIAC, plus performant dans un environnement ambiant. Elle s'est amorcée par une étude sur les identificateurs de qualité qui doivent exister dans une architecture d'espace intelligent. Essentiellement, la qualité d'un SIAC, considère trois métriques majeures : la fonctionnalité, la réutilisabilité et la modificabilité. Ces métriques nous permettent d'analyser si l'architecture que nous proposons, atteint réellement l'objectif de fournir des services mieux adaptés à l'utilisateur. Une architecture est dite fonctionnelle si elle satisfait deux exigences essentielles : la sensibilité au contexte et l'auto-adaptation. Évidemment, un SIAC doit être sensible au contexte, capable de détecter les changements des situations pour être en mesure d'anticiper les besoins des utilisateurs. Toutefois, l'auto-adaptation dépend fortement de la sensibilité au contexte. En effet, les systèmes sensibles au contexte se caractérisent par leur capacité à adapter les services, par la prise en compte du contexte environnant, afin d'augmenter leur efficacité

La deuxième contribution a porté sur la proposition d'une architecture générale qui incarne notre vision d'un système SIAC. Cette architecture est construite suivant un modèle multicouche afin de supporter un système adaptatif et réactif aux évènements. Chaque couche repose sur plusieurs modules interconnectés et engagés dans un processus de collaboration afin de réaliser les objectifs spécifiés lors de la détection des évènements dans un environnement ambiant.

La troisième contribution a porté sur la proposition d'une architecture modulaire de forte cohésion et de faible couplage, adaptée à la conception d'applications pour des systèmes intelligents. Relativement aux besoins des SIAC, nous avons opté pour quatre modules séparés pouvant coopérer pour produire des services : Contrôleur, Modèle contextuel, Adaptateur et Vue. Le Contrôleur est lié directement avec le monde physique. Il reçoit toute donnée contextuelle détectée, provenant soit de l'utilisateur, soit de l'environnement. Il est muni de mécanismes capables d'interpréter le contexte et de l'envoyer au Modèle contextuel. Celui-ci, joue le rôle d'une base de donnée permettant le stockage des données contextuelles. Il possède des mécanismes pour trouver l'intégralité des données recherchées. Il dispose également d'un registre spécialisé dans la sauvegarde des préférences des utilisateurs. Le Modèle contextuel est lié directement avec l'adaptateur pour lui transmettre certain contexte ci-nécessaire. Le rôle principal du module Adaptateur est d'assurer l'adaptation des services avant d'être envoyés au module Vue pour être exécutés. Le module Adaptateur est doté de mécanismes de raisonnement donnant la capacité d'effectuer des ajustements dépendamment du contexte présent. La Vue est le dernier module de l'architecture. Elle s'intéresse beaucoup plus à fournir et à exécuter les services à l'utilisateur.

La quatrième contribution a porté sur la proposition d'un moteur d'adaptation structurant ainsi que des opérations de raisonnement et d'apprentissage nécessaires à la supervision du processus d'adaptation dynamique. La solution proposée suit une hybridation entre un algorithme par renforcement (Q-learning) et un autre supervisé (CBR). Le processus Q-learning se manifeste le premier à travers une sensibilité au contexte environnemental qui détecte un nouveau contexte nécessitant une adaptation. L'algorithme Q-learning part d'une

observation sur le contexte pour identifier le problème de conception instantané puis le résout. Autrement dit, il va découvrir par une suite de tentatives et d'erreurs ce qu'il convient de faire comme service en différentes situations jusqu'à aboutir au service désiré. Tandis que le CBR consiste à adapter un service au nouveau contexte en se basant sur un ensemble de contextes déjà résolus. Le processus de décision est guidé par un ensemble d'objectifs et de contraintes traduisant la notion d'adaptation pour le service identifié.

La démarche suivie et les contributions apportées présentent plusieurs avantages.
Premièrement, le modèle de qualité constitue un support solide pour un système SIAC plus performent et plus efficaces à servir ses utilisateurs. Cet aspect essentiel procure au système la capacité d'être sensible au contexte pour l'adaptation et la prise de décision, afin d'assurer l'efficacité et l'évolutivité du système par rapport à l'évolution de l'environnement ambiant et des utilisateurs, en lui permettant une plus grande ouverture vers les différentes entités de l'environnement ambiant notamment : les services, les évènements, le contexte et les utilisateurs.

Deuxièmement, la structuration modulaire permet de répondre aux besoins de notre système en termes d'évolutivité, de réutilisation et d'adaptabilité. En effet, un module peut être facilement remplacé par un nouveau module sans impacts majeurs sur le fonctionnement du système. Il peut même être réutilisé pour d'autres besoins applicatifs.

Troisièmement, l'adaptation de services permet de gérer les changements de contexte pour être en mesure de produire des services convenables. Notre approche d'adaptation a montré l'efficacité, à suivre l'utilisateur dans son contexte pour lui proposer des services convenables.

Enfin, nous avons montré, grâce à plusieurs scénarios d'adaptation dans un espace intelligent, l'intérêt et la faisabilité de notre approche dans un environnement ambiant. À travers ces scénarios, nous avons montré qu'il est tout à fait possible de présenter à l'utilisateur des services convenables à sa situation dans différents contextes et avec divers

dispositifs. En effet, un service peut être ajusté selon plusieurs paramètres entourant l'utilisateur. Grâce à la flexibilité de notre architecture, l'adaptation de services est gérée dynamiquement en fonction du contexte environnemental, et ce, contrairement à des systèmes statiques où des adaptations sont faites via l'intervention de l'utilisateur.

Par ailleurs, la mise en oeuvre et l'évaluation de notre algorithme montre clairement leurs apports en termes de taux d'adaptabilité et de réactivité face à la nature dynamique de l'environnement intelligent. Les tests d'évaluation des performances ont montré que le taux des services bien-adaptés reste élevé par rapport au taux des services mal-adaptés. Ce taux reste également meilleur, par rapport, à plusieurs approches proposées dans le domaine de l'adaptation de services.

Perspectives de ces travaux

Sur la base du travail réalisé, nous pouvons dresser plusieurs perspectives de recherche à court, à moyen et à long terme.

Sécurité et respect de la vie privée : l'ouverture de l'environnement intelligent à des entités inconnues a priori et à des composants qui peuvent être développés par des parties tierces posent des problèmes de sécurité. Il est alors nécessaire de procéder à une authentification des entités de l'environnement. Il est également nécessaire de gérer les autorisations d'accès de ces entités à des données personnelles des utilisateurs.

Applications à des scénarios plus complexes : nous envisageons, comme perspectives à court terme, la mise en œuvre d'un scénario plus complexe avec plusieurs utilisateurs qui peuvent être servis par différents services.

Amélioration du processus d'adaptation : notre modèle d'adaptation se base sur un algorithme par renforcement et un autre supervisé. Il est nécessaire de modifier les

algorithmes utilisés en tenant compte de l'aspect dynamique de l'environnement et de ses utilisateurs.

La définition et la catégorisation des services : dans le domaine de l'informatique ambiante, il est préférable de proposer des modèles de services permettant d'améliorer le processus de l'adaptation.

Extension des tests d'évaluation des performances : il s'agit d'effectuer d'autres tests de performances notamment sur l'approche d'adaptation de services. C'est à dire, d'envisager d'autres paramètres de comparaisons en introduisant d'autres variantes d'apprentissage.

ANNEXE I

MACHINE LEARNING TECHNOLOGIES IN SMART SPACE

Abstract— Context-awareness is the key element for building a smart environment that responds to users needs. The goal of such environment is to provide proactively services according the demand of users by considering the user's context information. Machine learning techniques can provide several benefits. They can apply in many context aware systems to help to provide services. They have the possibility to make better prediction and adaptation than other. In this paper, we present the main goals of machine learning goals and some learning algorithms applied in smart space.

Keywords- context-awareness; smart environment; machine learning; prediction; adaptation; performance.

1. Introduction

A smart environment can be defined as an environment that is able to acquire and apply knowledge about the environment and its inhabitants in order to improve their experience in that environment (DAS and al., 2006). In effect, such environment can perceives the state of the space using sensors, analyzes the state using learning and reasoning techniques and adapt behaviours according to users in order to provide ease daily life to increase their comfort and usefulness. Dynamism, complexity are very important characteristic in smart space (STENUDD and al., 2010). Indeed, there are different devices networked via an infrastructure of heterogeneous access technologies. Also, the user behaviour or Preferences may change at any time.

In addition, the awareness of context is a key feature to develop an adaptable a smart system that have the ability to sense and to react accordingly to modifications. Also, Smart space must be able to control and to adapt services automatically. Machine learning techniques have been widely used for this objective. Machine learning as a domain capable of supporting the solution of complex problems is able to provide significant help (SMOLA and al., 2008). These algorithms can be applied in a predictive sense or to investigate internal relationships of a dataset (STIMPSON and al., 2015). It can be divided into four main

groups: supervised learning, unsupervised learning, semi-supervised learning and reinforcing. Each of these groups has their advantages and their drawbacks and utilizes different approaches to target different goals.

The Article is organized as follows. Section 2 describes important goals of machine learning in smart environments. Section 3 shows the principal phases that are necessary in learning process. Section 4 discuss about the different types of machine learning and some techniques applied in smart space. Section 5 concludes the article.

2. Machine learning goals

Important features of smart environments are that they possess a degree of autonomy, adapt themselves to changing environments, and communicate with humans in a natural way. Application of machine learning in context aware system can be employed to achieve specific goals. Generally, these goals belong to four mains classes:

A. Recognition

Several approaches already exist devoted to recognition problem to identify events or activities of users in smart environments. The activity recognition is usually done through two steps: activity pattern clustering and activity type decision (BOUROBOU and al., 2015). In most cases, recognition problems are processed by supervised learning algorithms witch assumes that a training set is consisting of a set of instances that have been properly labeled by hand with the correct output.

B. Prediction

Prediction aims to predict what is going happened in the future. In a smart environment, prediction allows providing information useful for future locations and activities predict the most probable event or subsequent activity. This type of problem can be solved by online

training approach that can learn from input data over time, to predict well the out put data (STIMPSON and al., 2015).

C. Adaptation

Adapting user services according to the context aims to provide the proper services by considering the user and the environmental information. Machine learning algorithms provide several benefic for context aware system. Indeed, it can be applied to support reasoning, inferences and also to deal with complex or fuzzy information (MOTTI and al., 2012).

D. Optimization

Optimization is very important feature in smart environment. It aims to increase their performance and effectiveness. It can resolving by using Reinforcement algorithms that can explore idealized learning situations and evaluate the effectiveness of various learning methods (SUTTON and al., 1998).

3. Learning process in smart environment

Machine learning used in smart environment offer major opportunities to provide context aware services. Context awareness is about capturing a broad range of contextual attributes (such as the user's current positions, activities, and their surrounding environments) to better understand what the user is trying to accomplish, and what services the user might be interested (LEE and Wei-Po, 2007). In these consequences, learning became an important and indispensable to reason and to make the best decision.

Indeed, it is considered necessary for knowledge creation (BHATT and al., 2002). In addition, artificial intelligence includes several sub-categories such as detection, knowledge representation and machine learning, machine perception, among others (BKASSINY and al., 2013). In our article, we focus mainly on four process detection, interpretation, learning,

reasoning as illustrated in Fig 1. First, different sensors installed in the environment in order to capture context in the environment accomplish detection phase. After that, interpretation is the ability to interpret the context information received to have a use full data. After acquiring the sensing observations, learning mechanisms take the use full context and try to classify and to organize the observations according to the algorithm. Finally, the reasoning phase allows using the context information knowledge acquired through learning to achieve its objectives.

4. Machine learning application in smart environment

Diverse machine-learning algorithms have been developed to cover variety of data and problem types in smart environment. Machine learning has branched into several subfields dealing with different types of learning tasks. We give a rough taxonomy of learning paradigm. Mainly, there are four categories: Supervised learning, unsupervised learning, semi-supervised learning and reinforcing algorithms. The next section presents some learning strategies applied for different raisons in smart environment.

A. Supervised learning

Supervised machine learning is the search for algorithms that reason from externally supplied instances to produce general hypotheses, which then make predictions about future instances (KOTSIANTIS and al., 2007). Mainly, supervision is provided in the form of a set of labeled training data, each data point having a class label selected from a fixed set of classes (MITCHELL and Tom, 1999). The use of the supervised activity classification approaches has shown promising results (ALTUN and al., 2010).

Supervised learning methods are widely used in smart environment to solve several problems. The ACHE system used neural networks and reinforcement learning to control devices (MOZER and Michael, 1998). Authors in (BOUROBOU and al., 2015) propose a hybrid approach consisting of the neural network algorithm based on temporal relations and

K-pattern clustering to recognize and predict user activities in IoT based smart environments. In (HSU and al., 2007) neural network are used for inferring the user's viewing preferences to develop a personalized contextual TV recommendation system. Authors in (FLEURY and al., 2010) use SVM algorithm to classify the classification of the Activities of Daily Living in a Health Smart Home. In (), authors used Decision tree based on context history to infer the preferences of the user for providing the personalized services based on context history on context-aware computing. A naïve Bayesian Classifier is used to learn user activity and availability directly from sensor data according to given user feedback (MÜHLENBROCK and al., 2004).

B. Unsupervised learning

In unsupervised learning no information about the input is given and thus the system cannot know anything about the correctness of the outcome (STENUDD and al., 2010). It try to directly construct models from unlabeled data either by estimating the properties of their underlying probability density (called density estimation) or by discovering groups of similar examples (called clustering) (GU and al., 2010). The use of an unsupervised approach is applied for different activities recognition in smart space when it is difficult to have labels for the data. Authors propose an unsupervised approach based on object-use fingerprints to recognize human activities without manual labeling, making our effort towards scalability, applicability and adaptability for real-life deployment (TRABELSI and al., 2013). Hidden Markov models (HMM) are used in (LV and al., 2006) for both segmentation and recognition of to enable real-time assessment and feedback for physical rehabilitation.

C. Reinforcement learning

Reinforcement learning is a learning paradigm concerned with learning to control a system so as to maximize a numerical performance measure that expresses a long-term objective (SZEPESVARI and al., 2010). Q-learning (WATKINS and al., 1992) is a model-free reinforcement learning method based on learning the expected utility given a state decision.

Authors in (LI and al., 2014) propose Q-learning algorithm to provide a more efficient way for on-line decision making, with more flexibility and adaptiveness with relatively good performance. This algorithm is implemented also in simulation to demonstrate how the performance of the new Markov Decision Process (MDP) representation is comparable to that of a Linear Time-Invariant (LTI) one on a reference-tracking scenario (KARA and al, 2012). Reinforcement algorithms are used in Mavhome project (COOK and al., 2003) to acquire an optimal decision policy to automate basic functions in order to maximize the inhabitants' comfort and minimize the operating cost of the home.

D. Semi-supervised learning

Semi-supervised learning is a learning paradigm concerned with the study of how computers and natural systems such as humans learn in the presence of both labeled and unlabeled data (ZHU and al., 2009). The goal of semi-supervised learning is to combine a large amount of unlabeled data, together with the labeled data, to build better classifiers (ZHU, 2005). This category requires less human effort and building costs.

There are some popular semi-supervised learning models, including self-training, mixture models, co-training and multi-view learning, graph-based methods, and semi-supervised support vector machines. Authors in (COOK and al., 2010) combine between supervised and semi-supervised learning to recognizing ADL activities and to provide context-aware services, such as health monitoring and intervention in different smart space.

5. Performance comparison of machine learning algorithms

Machine learning algorithms that have been used for solving different problems in context aware spaces generally fall into the categories of being supervised, unsupervised, semi-supervised or with reinforcement. Nevertheless, the advantages or disadvantages of each one depend on what learning algorithm want to solve.

Neural networks are the most widely used like a supervised learning. Indeed, they offer a number of advantages including requiring less formal statistical training, ability to implicitly detect complex nonlinear relationships between dependent and independent variables, ability to detect all possible interactions between predictor variables, and the availability of multiple training algorithms (TU and al., 1996). On the other hand, disadvantages include its "black box" nature, greater computational burden, and proneness to over fitting. Decision trees algorithm are non-parametric algorithm and easy to interpret and explain. Their main disadvantage is that they easily over fit. SVMs can work well with an appropriate kernel even when data isn't linearly separable, they have a High accuracy and nice theoretical guarantees regarding over fitting. Whoever, it is hard to interpret.

In Unsupervised Learning, there is no outcome measure; we observe only the features and the goal is to describe the associations and patterns among a set of input measures (RAMÓN and al., 2012). Its major disadvantage is the lack of direction for the learning algorithm and that the absence of any interesting knowledge discovered in the set of features selected for the training. Clustering is a form of unsupervised learning that finds patterns in the data by putting each data element into one of K clusters, where each cluster contains data elements most similar to each other (WOOLEY and Bruce, 1999).

Semi-supervised learning is an interesting field. It's a hybrid between clustering and supervised learning, potentially useful on scenarios where labeling effort is not ready available or expensive. These algorithms try to solve a supervised learning approach using labeled data, augmented by unlabeled data. So, adding cheap and abundant unlabeled data, you are hoping to build a better model than using supervised learning alone.

Reinforcing algorithms learn more control policies, especially in the absence of a priori knowledge and a sufficiently large amount of training data. However, it suffers from a major drawback: height calculation cost because an optimal solution requires that all states be visited to choose the optimal one.

6. CONCLUSION

In this paper, we have discussed about the importance of the use of Machine Learning in context aware system. We have presented the major goals of learning methods. We also show the main steps to achieve the learning process in context aware space. This process must acquire the context of the environment to be able to adapt services to users. A discussion of the most important and commonly used, learning algorithms was provided to solve different problems in smart environment.

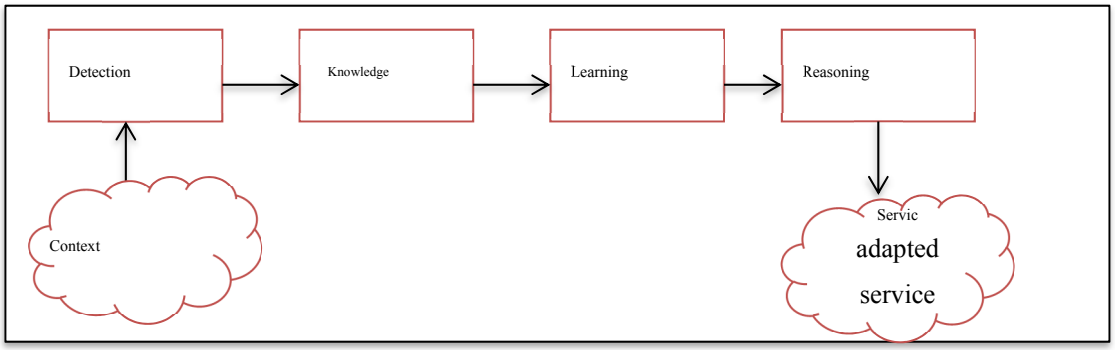


Figure 1. The main steps for learning process in context aware system

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Aamodt, A., & Plaza, E. 1994. « Case-based reasoning: Foundational issues, methodological variations, and system approaches ». *AI communications*, p. 39-59.

Abdualrazak, B., Y. Malik and H.I. Yang, 2010. « A taxonomy driven approach towards evaluating pervasive computing system ». Proceedings of the International Conference on Smart Homes and Health Telematics, Springer , p.32-42.

Abowd, D., Dey, A. K., Orr, R., & Brotherton, J. 1998. « Context-awareness in wearable and ubiquitous computing ». Virtual Reality, 3(3), p. 200-211.

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. 1999, September.« Towards a better understanding of context and context-awareness ». In International Symposium on Handheld and Ubiquitous Computin, Springer, p. 304-307.

Acampora, G., Loia, V., Nappi, M., & Ricciardi, S, 2005,July. « Ambient intelligence framework for context aware adaptive applications ». In Computer Architecture for Machine Perception, Seventh International Workshop, IEEE, p. 327-332.

Ahmad, H. 2014, April, « Computer fault daignosis system using case-based reasoning ». International Journal of Latest Research in Science and Technology, p. 2278-5299

Ali, F. M., Lee, S. W., Bien, Z., & Mokhtari, M. 2008, August. « Combined fuzzy state q-learning algorithm to predict context aware user activity under uncertainty in assistive environment ». In *Software Engineering, Artificial Intelligence, IEEE,* p. 57-62.

Allen, E. B., Khoshgoftaar, T. M., & Chen, Y. 2001.« Measuring coupling and cohesion of software modules: an information-theory approach ». In Software Metrics Symposium, IEEE, p. 124-134.

Aloulou, H., M. Mokhtari, T. Tiberghien, J. Biswas and P. Yap, 2014. « An adaptable and flexible framework for assistive living of cognitively impaired people ». IEEE journal of biomedical and health informatics, p. 353-360.

Altun, K., Barshan, B., & Tunçel, O. 2010 . « Comparative study on classifying human activities with miniature inertial and magnetic sensors ». Pattern Recognition, p. 3605-3620.

Amoui, M., Salehie, M., & Tahvildari, L. 2009. « Temporal software change prediction using neural networks ». International Journal of Software Engineering and Knowledge Engineering, p. 995-1014.

Belaidouni, S., Miraoui, M., & Tadj, C. 2016.« Towards an efficient smart space darchitecture ». arXiv preprint arXiv:1602.05109.

Bellifemine, F., F. Bergenti, G. Caire and A. Poggi, 2005. « Jade - A Java Agent Development Framework. In : Multi-Agent Programming ». Springer, p. 125-147.

Benazzouz, Y. 2011. « Context discovery for autonomic service adaptation in intelligent ». space. Adv. Next Gener. Serv. Serv. Archit, 14, 281.

Bengtsson, P., Lassing, N., Bosch, J., & Vliet, H. V. 2000. « Analyzing software architectures for modifiability ».

Betty, H. C., Rogério, D. E., Holger, G., Inverardi, P., & Magee, J. 2009. « Software engineering for self-adaptive systems ». Lecture Notes in Computer Science, 5525.

Bianchi, R. A., Ros, R., & De Mantaras, R. L. 2009, July. « Improving reinforcement learning by using case based heuristics ». In International Conference on Case-Based Reasoning, Springer, p. 75-89.

Bkassiny, M., Li, Y., & Jayaweera, S. K. 2013. « A survey on machine-learning techniques in cognitive radios ». IEEE Communications Surveys & Tutorials, p. 1136-1159.

Bourobou, S. T. M., & Yoo, Y. 2015. « User activity recognition in smart homes using pattern clustering applied to temporal ANN algorithm ». Sensors, p. 11953-11971.

Boytsov, A., & Zaslavsky, A. 2010. « Extending context spaces theory by proactive adaptation ». In Smart Spaces and Next Generation Wired/Wireless Networking, Springer, p. 1-12.

Brézillon, P. 2002). « Making context explicit in communicating objects ». In: C. Kintzig, G. Poulain, G. Privat, P.-N. Favennec (Eds.): Communicating Objects. Hermes Science Editions, Lavoisier, p. 295-303.

Brézillon, P., & Pomerol, J. C. 1999, July. « Contextual knowledge and proceduralized context ». In Proceedings of the AAAI-99 Workshop on Modeling Context in AI Applications, AAAI Technical Report.

Brézillon, P., Borges, M., Pino, J. A., & Pomerol, J. C. 2004, July. « Context-awareness in group work: Three case studies ». In Proc. of.

Brown, P. J. 1995. « The stick-e document: a framework for creating context-aware applications ». Electronic Publishing-Chichester-.p, 259-272.

Brumitt, B., Meyers, B., Krumm, J., Kern, A., & Shafer, S. 2000, September.« Easyliving: Technologies for intelligent environments ». In International Symposium on Handheld and Ubiquitous Computing, Springer, p. 12-29.

Brun, Y., Serugendo, G. D. M., Gacek, C., Giese, H., Kienle, H., Litoiu, M., ... & Shaw, M. 2009.« Engineering self-adaptive systems through feedback loops ». In Software engineering for self-adaptive systems, Springer,p. 48-70.

Brusilovsky, P. 1996. « Methods and techniques of adaptive hypermedia ». In Adaptive hypertext and hypermedia, Springer, p. 87-129.

Buthpitiya, S., Luqman, F., Griss, M., Xing, B., & Dey, A. K. 2012, March.« Hermes--A context-aware application development framework and toolkit for the mobile environment ». In Advanced Information Networking and Applications Workshops (WAINA), 26th International Conference on IEEE, p. 663-670.

Buzeto, F. N., de Paula Filho, C. B., Castanho, C. D., & Jacobi, R. P. 2010, May.« Dsoa: A service oriented architecture for ubiquitous applications ». In International Conference on Grid and Pervasive Computing, Springer, p. 183-192.

Chaari, T., F. Laforest and A. Celentano, 2008. « Adaptation in context-aware pervasive information systems: The SECAS project ». Int. J. Pervasive Comput. Communications, p. 400-425.

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

Chen, H. 2004.« An intelligent broker architecture for pervasive context-aware systems» (Doctoral dissertation, University of Maryland, Baltimore County).

Chen, H., Perich, F., Finin, T., & Joshi, A. 2004, August. « Soupa: Standard ontology for ubiquitous and pervasive applications ». In Mobile and Ubiquitous Systems: Networking and Services, MOBIQUITOUS 2004. The First Annual International Conference on, IEEE,p. 258-267.

Chen, Y., Tang, Y., Xu, J., & Fang, Z. 2011, April. « Research on smart space oriented context-aware system ». In Electric Information and Control Engineering (ICEICE), IEEE,p. 698-700.

Cheverst, K., K. Mitchell, et N. Davies. 2002. « The role of adaptive hypermedia in a contextawaretourist guide ». Commun. ACM 45 (5),p. 47–51.

Christos Mettouris and George A. Papadopoulos. March 2014. « Ubiquitous recommender systems », Computing, Springer, p. 223-257.

Chun, I., Park, J., Lee, H., Kim, W., Park, S., & Lee, E. 2013. « An agent-based self-adaptation architecture for implementing smart devices in Smart Space ». Telecommunication Systems, p. 2335-2346.

Clarke, P. and S. Fahy, 2004.« CASS-middleware for mobile context-aware applications. Proceedings of the Workshop on Context Awareness », In Workshop on context awareness, MobiSys. CiteSeerX, p. 1-6.

Colman, A., Hussein, M., Han, J., & Kapuruge, M. 2014. « Context Aware and Adaptive Systems ». In Context in Computing, Springer, p. 63-82.

Cook, D. J. 2012 . « Learning setting-generalized activity models for smart spaces ». IEEE intelligent systems, p. 32-38.

Cook, D. J., Youngblood, M., Heierman, E. O., Gopalratnam, K., Rao, S., Litvin, A., & Khawaja, F. 2003, March.« MavHome: An agent-based smart home ». In Pervasive Computing and Communications, IEEE,p. 521-524.

Cook, D. J., Youngblood, M., Heierman, E. O., Gopalratnam, K., Rao, S., Litvin, A., & Khawaja, F. 2003, March . « MavHome: An agent-based smart home ». In Pervasive Computing and Communications, Proceedings of the First IEEE International Conference, p. 521-524.

Cremene, M., Riveill, M., Martel, C., Loghin, C., & Miron, C. 2004. « Adaptation dynamique de services ». arXiv preprint cs/0411056.

Das, S. K., & Cook, D. J. 2006, June. « Designing and modeling smart environments ». In Proceedings of the 2006 International Symposium on on World of Wireless, IEEE Computer Society, p. 490-494.

De Groot, B., & van Welie, M. 2002, September. « Leveraging the context of use in mobile service design ». In International Conference on Mobile Human-Computer Interaction . Springer, p. 334-338.

De Lemos, R., Giese, H., Müller, H. A., Shaw, M., Andersson, J., Litoiu, M., ... & Weyns, D. 2013.« Software engineering for self-adaptive systems: A second research roadmap ». In Software Engineering for Self-Adaptive Systems II, Springer, p. 1-32.

Degeler, V., Gonzalez, L. I. L., Leva, M., Shrubsole, P., Bonomi, S., Amft, O., & Lazovik, A. 2013, December.« Service-oriented architecture for smart environments (short paper) ». In Service-Oriented Computing and Applications (SOCA), IEEE, p. 99-104.

Degeler, V., Gonzalez, L. I. L., Leva, M., Shrubsole, P., Bonomi, S., Amft, O., & Lazovik, A. 2013, December.« Service-oriented architecture for smart environments (short paper) ». In Service-Oriented Computing and Applications (SOCA), IEEE, p. 99-104.

Derntl, M., & Hummel, K. A. 2005, March. « Modeling context-aware e-learning scenarios ». In Pervasive Computing and Communications Workshops, Third IEEE International Conference on, IEEE, p. 337-342.

Dey, A. K. 2001. « Understanding and using context ». Personal and ubiquitous computing, 5(1), 4-7.

Dey, A. K., Abowd, G. D., & Salber, D. 2001. « A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications ». Human-computer interaction, 16(2).p, 97-166.

Dobson, S. 2005. « Leveraging the subtleties of location ». In Proceedings of the 2005 joint conference on Smart objects and ambient intelligence : innovative context-aware services :usages and technologies , New York, NY, USA, ACM, p. 189–193.

Dorigo, M., & Bersini, H. 1994.« A comparison of Q-learning and classifier systems ». *From animals to animats*, p. 248-255.

Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J. & Burgelman, J.-C. 2001.«Scenarios for ambient intelligence in 2010». In Technical report, IST Advisory Group (ISTAG), IPTS-Seville.

El Khaddar, M.A., M. Chraibi, H. Harroud, M. Boulmalf and M. Elkoutbi et al., 2015.« A policy-based middleware for context-aware pervasive computing ». Int. J. Pervasive Comput. Communi., p 43-68.

Fellbaum, C. 1998. *WordNet*. John Wiley & Sons, Inc..

Firner B., R.S. Moore, R. Howard, R.P. Martin and Y. Zhang, 2011. « Smart buildings, sensor networks and the internet of things ». Proceedings of the 9th Conference on Embedded Networked Sensor Systems, p: 337-338.

FIWARE Project, 2016. https://www.fiware.org/

Fleury, A., Vacher, M., & Noury, N. 2010 . « SVM-based multimodal classification of activities of daily living in health smart homes: sensors, algorithms, and first experimental results ». IEEE transactions on information technology in biomedicine, p. 274-283.

Fleury, Anthony, VACHER, Michel, et NOURY, Norbert. SVM-based multimodal classification of activities of daily living in health smart homes: sensors, algorithms, and first experimental results. IEEE transactions on information technology in biomedicine, 2010, vol. 14, no 2, p. 274-283.

Frey, J., Bird, C., & Willoughby, C. 2013.« Smart meeting spaces for knowledge transfer ». In Innovation through Knowledge Transfer, Springer, p. 31-38.

Galin, D., 2004.« Software Quality Assurance: From Theory to Implementation ». 1st Edn., Pearson Education Limited, p: 590.

Gu, T., Chen, S., Tao, X., & Lu, J. 2010 . « An unsupervised approach to activity recognition and segmentation based on object-use fingerprints ». *Data & Knowledge Engineering*, p. 533-544.

Gu, T., Pung, H. K., & Zhang, D. Q. 2004, May.« A middleware for building context-aware mobile services ». In Vehicular Technology Conference, IEEE,p. 2656-2660.

Gu, T., Pung, H. K., & Zhang, D. Q. 2005. « A service-oriented middleware for building context-aware services ». Journal of Network and computer applications, p.1-18.

Handte, M., Schiele, G., Matjuntke, V., Becker, C., & Marrón, P. J. 2012. « 3PC: System support for adaptive peer-to-peer pervasive computing ». ACM Transactions on Autonomous and Adaptive Systems.

Helal, S., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y., & Jansen, E. 2005.« The gator tech smart house: A programmable pervasive space ». Computer, p. 50-60.

Henricksen, K., & Indulska, J. 2006.« Developing context-aware pervasive computing applications: Models and approach ». Pervasive and mobile computing, p. 37-64.

Henricksen, K., J. Indulska, T. McFadden and S. Balasubramaniam, 2005. « Middleware for distributed context-aware systems ». Proceedings of the OTM Confederated International Conferences on the Move to Meaningful Internet Systems, Springer, p. 846-863.

Hess, C. K., & Campbell, R. H. 2003, May. « A context-aware data management system for ubiquitous computing applications ». In Distributed Computing Systems, Proceedings. 23rd International Conference on, p. 294-301

Holvitie, J., & Leppänen, V. 2014, September.« Illustrating Software Modifiability--Capturing Cohesion and Coupling in a Force-Optimized Graph ». In Computer and Information Technology (CIT), IEEE, p. 226-233.

Hong, J., Suh, E. H., Kim, J., & Kim, S. 2009. « Context-aware system for proactive personalized service based on context history ». *Expert Systems with Applications*, p. 7448-7457.

Honkola, J., Laine, H., Brown, R., & Tyrkkö, O. 2010, June.« Smart-M3 information sharing platform ». In Computers and Communications (ISCC), IEEE, p. 1041-1046.

Hsu, S. H., Wen, M. H., Lin, H. C., Lee, C. C., & Lee, C. H. 2007, May. « AIMED-A personalized TV recommendation system » . In European Conference on Interactive Television, Springer, p. 166-174.

Hussein, M., Han, J., & Colman, A. 2010 .« Context-aware adaptive software systems: A system-context relationships oriented survey ». Technical Report# C3–516_01, Swinburne University of Technology.

Kabir, M. H., Hoque, M. R., Seo, H., & Yang, S. H. 2015. « Machine learning based adaptive context-aware system for smart home environment ». *International Journal of Smart Home*, p. 55-62.

Kara, E. C., Berges, M., Krogh, B., & Kar, S. 2012, November . « Using smart devices for system-level management and control in the smart grid: A reinforcement learning framework ». In Smart Grid Communications (SmartGridComm), IEEE, p. 85-90.

Kayes, A. S. M., Han, J., & Colman, A. 2014, June.« PO-SAAC: a purpose-oriented situation-aware access control framework for software services ». In International Conference on Advanced Information Systems Engineerin, Springer, p. 58-74.

Keeney, J. and V. Cahill, 2003. « Chisel: A policy-driven, context-aware, dynamic adaptation framework ». Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks, IEEE, p: 3-14.

Kim, E., & Choi, J. 2008, December. « A context management system for supporting context-aware applications ». In Embedded and Ubiquitous Computing, International Conference on IEEE, p. 577-582.

Knoernschild, K., 2012.« Java Application Architecture: Modularity Patterns with Examples Using OSGi ». Prentice Hall Press,p. 384.

Kolodner, J. L. 1997.« Educational implications of analogy: A view from case-based reasoning ». *American psychologist*, p. *52*-57.

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. 2007. « Supervised machine learning: A review of classification techniques ». Emerging artificial intelligence applications in computer engineering, p. 3-24.

Laddaga, R., Robertson, P., & Shrobe, H. 2001, May.« Results of the second international workshop on self-adaptive software ». In International Workshop on Self-Adaptive Software, Springer,p. 281-290.

Lee, W. P. 2007 . « Deploying personalized mobile services in an agent-based environment ». Expert Systems with Applications, p. 1194-1207.

Li, D., & Jayaweera, S. K. 2014, November . « Reinforcement learning aided smart-home decision-making in an interactive smart grid ». In Green Energy and Systems Conference (IGESC), IEEE, p. 1-6.

Lopez-de-Ipina, D., Almeida, A., Aguilera, U., Larizgoitia, I., Laiseca, X., Orduna, P., ... & Vazquez, J. I. 2008.« Dynamic discovery and semantic reasoning for next generation intelligent environments ».

Lum, W. Y., & Lau, F. C. 2002. « A context-aware decision engine for content adaptation ». *IEEE*, p. 41-49.

Lv, F., & Nevatia, R. 2006, May . « Recognition and segmentation of 3-d human action using hmm and multi-class adaboost ». In European conference on computer vision, Springer, p. 359-372.

Macías-Escrivá, F. D., Haber, R., Del Toro, R., & Hernandez, V. 2013.« Self-adaptive systems: A survey of current approaches, research challenges and applications ». Expert Systems with Applications, p. 7267-7279.

Mandato, D., Kovacs, E., Hohl, F., & Amir-Alikhani, H. 2000.« CAMP: A context-aware mobile portal ». In *Service Portability and Virtual Customer Environments, IEEE* ,p. 52-61.

McBurney, S., Papadopoulou, E., Taylor, N., & Williams, H. 2008, December.« Adapting pervasive environments through machine learning and dynamic personalization ». In Parallel and Distributed Processing with Applications, International Symposium on IEEE, p. 395-402.

McGuinness, D. L., Fikes, R., Hendler, J., & Stein, L. A. 2002. « DAML+ OIL: an ontology language for the Semantic Web ». IEEE Intelligent Systems, 17(5), p.72-80.

McKinley, P. K., Sadjadi, S. M., Kasten, E. P., & Cheng, B. H. 2004. « Composing adaptive software ». Computer, p. 56-64.

Meissen, U., Pfennigschmidt, S., Voisard, A., & Wahnfried, T. 2004, March. « Context-and situation- awareness in information logistics ». In *International Conference on Extending Database Technology, Springer,* p. 335-344.

Miraoui, M., & Tadj, C. 2007, November. « A service oriented definition of context for pervasive computing ». In Proceedings of the 16th International Conference on Computing, IEEE computer society press.

Miraoui, M., & Tadj, C. 2007. « A service oriented architecture for context-aware systems ». Journal of Research in Computing Science, 29, p. 236-244.

Miraoui, M., Cherif, R., Rtimi, N., & Tadj, C. 2014, January. « Context-aware services adaptation for a smart living room ». In Computer Applications & Research (WSCAR), IEEE, p. 1-5.

Miraoui, M., El-Etriby, S., Tadj, C., & Abid, A. Z. 2017.« A Hybrid Modular Context-Aware Services Adaptation for a Smart Living Room ». *Intelligent Automation & Soft Computing*, p.1-9.

Miraoui, M., S. El-etriby, A.Z. Abid and C. Tadj, 2016. « Agent-based context-aware architecture for a smart living room ». Int. J. Smart Home.

Miraoui, M., Tadj, C., & Amar, C. B. 2009, October.« Dynamic context-aware service adaptation in a pervasive computing system ». In Mobile Ubiquitous Computing, Systems, Services and Technologies, Third International Conference on IEEE,p. 77-82.

Mitchell, T. M. 2004. « The role of unlabeled data in supervised learning ». In Language, Knowledge, and Representation, Springer, p. 103-111.

Motti, V. G., Mezhoudi, N., & Vanderdonckt, J. 2012. « Machine Learning in the Support of Context-Aware Adaptation ». In CASFE.

Mozer, M. C. 1998, March . « The neural network house: An environment hat adapts to its inhabitants ». In Proc. AAAI Spring Symp. Intelligent Environments.

Mozer, M. C. 2004. « Lessons from an adaptive house. *Smart environments: Technologies, protocols, and applications ». J. Wiley & Sons*.

Muhlenbrock, M., Brdiczka, O., Snowdon, D., & Meunier, J. L. 2004, March . « Learning to detect user activity and availability from a variety of sensor data ». In Pervasive Computing and Communications, Proceedings of the Second IEEE Annual Conference , p. 13-22.

Ni, H., Zhou, X., Zhang, D., Miao, K., & Fu, Y. 2009, July.« Towards a task supporting system with CBR approach in smart home ». In *International Conference on Smart Homes and Health Telematic, Springer,* p. 141-149.

Papadopoulou Elizabeth, McBurney Sarah, Taylor Nick, and Williams M. Howard. 2008. « Adynamic approach to dealing with user preferences in a pervasive system ». In Proceedings of the 2008 IEEE International Symposium on Parallel and Distributed Processing with Applications, p. 409–416.

Petrelli, D., E. Not, C. Strapparava, O. Stock, et M. Zancanaro. 2000. « Modeling context i s context i s like taking pictures. In Conference on Human Factors in Computers, Workshop "The What, Who, Where, When, Why and How of Context-Awareness" .

Pignotti, E., Edwards, P., & Grimnes, G. A. 2004, August. « Context-aware personalised service delivery ». In Proceedings of the 16th European Conference on Artificial Intelligence,p. 1077-1078.

Preuveneers, D. and Y. Berbers, 2007. « Towards context-aware and resource-driven self-adaptation for mobile handheld applications ». Proceedings of the Symposium on Applied Computing, ACM,p: 1165-1170.

Rakshit, P., Konar, A., Bhowmik, P., Goswami, I., Das, S., Jain, L. C., & Nagar, A. K. 2013. « Realization of an adaptive memetic algorithm using differential evolution and q-learning: a case study in multirobot path planning. *IEEE Transactions on Systems,p.* 814-831.

Ramón, manuel, Martínez-Pastor, Felipe, García-Álvarez, Olga., Maroto-Morales, Alejandro., Soler, A. J., Jiménez-Rabadán, P., ... & Garde, J. J. 2012 . « Taking advantage of the use of supervised learning methods for characterization of sperm population structure related with freezability in the Iberian red deer ». Theriogenology, p. 1661-1672.

Reignier, P., Brdiczka, O., Vaufreydaz, D., Crowley, J. & Maisonnasse, J. 2007. « Contextaware environments : from specification to implementation » in Expert Systems : The Journal of Knowledge Engineering 24(5), p.305–320.

Réty, J. H., Martin, J. C., Pelachaud, C., & Bensimon, N. 1998. Coopération entre un hypermédia adaptatif éducatif et un agent pédagogique. Actes de H2PTM, 3.

Richter, M. M., & Weber, R. O. 2013. « Basic CBR elements. In *Case-Based Reasoning». Springer,*p. 17-40.

Richter, M. M., & Weber, R. O. 2016. « *Case-based reasonin» g*. Springer-Verlag Berlin An.

Riesbeck, C. K., & Schank, R. C. 2013. « *Inside case-based reasoning »*. Psychology Press.

Ros, R., & Arcos, J. L. 2007, January. « Acquiring a Robust Case Base for the Robot Soccer Domain ». In *IJCAI,* p. 1029-1034.

Ros, R., Arcos, J. L., De Mantaras, R. L., & Veloso, M. 2009. « A case-based approach for coordinated action selection in robot soccer ». *Artificial Intelligence*, p.1014-1039.

Ros, R., De Màntaras, R. L., Arcos, J. L., & Veloso, M. 2007, August. « Team playing behavior in robot soccer: A case-based reasoning approach ». In International Conference on Case-Based Reasoning, Springer, p. 46-60.

Rouvoy, R., P. Barone, Y. Ding, F. Eliassen and S. Hallsteinsen et al., 2009. « Music: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments ». In: Software Engineering for Self-Adaptive Systems, Springer, p : 164-182.

Sabri, L., 2013. « Modèles sémantiques et raisonnements réactif et narratif, pour la gestion du contexte en intelligence ambiante et en robotique ubiquitaire » (Doctoral dissertation, Université Paris-Est).

Salber, D., Dey, A. K., & Abowd, G. D. 1998. « Ubiquitous computing: Defining an hci research agenda for an emerging interaction paradigm ». Georgia Institute of Technology.

Salehie, M., & Tahvildari, L. 2012.« Towards a goal-driven approach to action selection in self-adaptive software ». Software: Practice and Experience, p. 211-233.

Sanchez, A., & Tercero, R. 2010, November. « Smart home technologies: Uses and abuses ». In Artificial Intelligence (MICAI), IEEE,p. 97-102.

Sarker, I.H. and K. Apu, 2014. « MVC architecture driven design and implementation of java framework for developing desktop application ». Int. J. Hybrid Inform. Technol., p. 317-322.

Schilit, B. N., & Theimer, M. M. 1994. « Disseminating active map information to mobile hosts ». IEEE network, p. 22-32.

Schilit, B., Adams, N., & Want, R. 1994, December. « Context-aware computing applications ». In Mobile Computing Systems and Applications, First Workshop on IEEE,p. 85-90.

Sharafi, S.M., M. Ghasemi and N. Nematbakhsh, 2012. « Using architectural patterns to improve modularity in software architectural design ». Proceedings of the International Conference on Software and Computer Applications, ACSIT Press, p. 65-70.

SMOLA, Alex et VISHWANATHAN, S. V. N. « Introduction to machine learning ».Cambridge University, p.34, 2008.

Soylu, A., De Causmaecker, P., & Desmet, P. 2009. « Context and adaptivity in pervasive computing environments: Links with software engineering and ontological engineering ». Journal of Software, p. 992-1013.

Stenudd, S. 2010. « Using machine learning in the adaptive control of a smart environment ». Utigivare, Vuorimiehentie.

Stimpson, A. J. 2015 . « A machine learning approach to modeling and predicting training effectiveness (Doctoral dissertation, Massachusetts Institute of Technology) ».

Strang, T., & Linnhoff-Popien, C. 2004, September.« A context modeling survey ». In Workshop Proceedings.

Sutton, R. S., & Barto, A. G. 1998 . « Reinforcement learning: An introduction ». Cambridge: MIT press.

Szepesvári, C. 2010 . « Algorithms for reinforcement learning. Synthesis lectures on artificial intelligence and machine learning », p. 1-103.

Trabelsi, D., Mohammed, S., Chamroukhi, F., Oukhellou, L., & Amirat, Y. 2013 . « An unsupervised approach for automatic activity recognition based on hidden Markov model regression ». IEEE Transactions on automation science and engineering, p. 829-835.

Tu, J. V. 1996 . « Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes ». Journal of clinical epidemiology, p.1225-1231.

Uschold, M., & Gruninger, M. 1996. « Ontologies: Principles, methods and applications ». The knowledge engineering review, p. 93-136.

Vainio, A. M., Valtonen, M., & Vanhala, J. 2008. « Proactive fuzzy control and adaptation methods for smart homes ». IEEE .

Wang, S., Zheng, Z., Wu, Z., Sun, Q., Zou, H., & Yang, F. 2014. « Context-aware mobile service adaptation via a Co-evolution eXtended Classifier System in mobile network environments ». *Mobile Information Systems*, p. 197-215.

Wang, X. H., Zhang, D. Q., Gu, T., & Pung, H. K. 2004, March. « Ontology based context modeling and reasoning using OWL ». In Pervasive Computing and Communications Workshops, IEEE,p. 18-22.

WATKINS, Christopher JCH et DAYAN, Peter.1992. « Q-learning. Machine learning » .p. 279-292.

Weyns, D., Iftikhar, M. U., Malek, S., & Andersson, J. 2012, June.« Claims and supporting evidence for self-adaptive systems: A literature study ». In Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, IEEE,p. 89-98.

Wooley, B. A. 1999 . « Scaling Clustering for the Data Mining Step in Knowledge Discovery ». Department of Computer Science, p. 325-8073.

Xu, X., He, H. G., & Hu, D. 2002. « Efficient reinforcement learning using recursive least-squares methods ». *Journal of Artificial Intelligence Research*, p. 259-292.

Yuan, B., & Herbert, J. 2014.« Context-aware hybrid reasoning framework for pervasive healthcare ». *Personal and ubiquitous computing*, p. 865-881.

Zuo, Z. and Zhou, M. 2003. « Web Ontology Language OWL and its description logic foundation » in Parallel and Distributed Computing, Applications and Technologies, Proceedings of the Fourth International Conference on,p. 157-160.

Zhang, D., H. Huang, C.F. Lai, X. Liang and Q. Zou et al., 2013. Survey on context-awareness in ubiquitous media. Multimedia Tools Applic., p. 179-211.

Zhang, H. and S. Zhu, 2013.« B/S implementation of internet-based electrical engineering lab with MVC architecture ». Proceedings of the 10th IEEE International Conference on Control and Automation, IEEE Xplore Press, p. 551-555.

Zhu, X. 2005. « Semi-supervised learning literature survey ».

Zhu, X., & Goldberg, A. B. 2009 . « Introduction to semi-supervised learning ». Synthesis lectures on artificial intelligence and machine learning, p. 1-130.

Zimmermann, A. 2003, June. « Context-awareness in user modelling: Requirements analysis for a case-based reasoning application ». In International Conference on Case-Based Reasoning, Springer, p. 718-732.