République Algérienne démocratique et populaire
Université Badji Mokhtar Annaba
Faculté des Sciences de l'Ingéniorat
Département d'informatique
Laboratoire d'Ingénierie des Systèmes Complexes

T H È S E

Présentée en vu de l'obtention du diplôme de

Doctorat

Reconfiguration dynamique des architectures à base d'objets connectés

Domaine: Mathématiques et informatique
Filière: Informatique
Spécialité: Ingénierie des systèmes complexes

Préparée et soutenue publiquement par:

Mme Amira Hakim

Le: 30-06-2019

Devant le jury:

Président	Mme	Souici-Meslati	Labiba	**Pr**	Université Badji Mokhtar- Annaba
Directeur	Mr	Amirat	Abdelkrim	**Pr**	Université Med Cherif Messadia-Souk Ahras
Co-directeur	Mr	Oussalah	Mourad	**Pr**	Université de Nantes- France
Examinateur	Mme	Atil	Fadila	**Pr**	Université Badji Mokhtar- Annaba
Examinateur	Mme	Bounour	Nora	**MCA**	Université Badji Mokhtar- Annaba

Année universitaire: 2018-2019

Remerciements

A l'issue de la rédaction de cette thèse, je suis convaincue que la recherche est loin d'être un travail solitaire. En effet, je n'aurais jamais pu réaliser ce travail doctoral sans le soutien d'un grand nombre de personnes dont la générosité, la bonne humeur et l'intérêt manifestés à l'égard de ma recherche m'ont permis de progresser dans cette phase délicate.

En premier lieu, je tiens à remercier mon directeur de thèse Professeur *Abdelkrim Amirat*, pour la confiance qu'il m'a accordée en acceptant d'encadrer ce travail doctoral, pour ses multiples conseils et pour tout le temps qu'il a consacré à diriger cette recherche. J'aimerais également lui dire à quel point j'ai apprécié sa grande disponibilité et son respect sans faille des délais serrés de lecture de mes documents. Enfin, j'ai été extrêmement sensible à ses qualités humaines d'écoute et de compréhension tout au long de ce travail.

J'exprime mes remerciements les plus sincères au Professeur *Mourad Oussalah* mon co-directeur de thèse. Il m'a apporté tous les conseils et les encouragements dont a besoin un doctorant pour achever sa formation à la recherche dans les meilleures conditions qui soient. Ses conseils et sa vigilance m'ont accompagné tout au long de cette aventure. Je lui remercie également pour son accueil chaleureux au laboratoire LS2N à l'Université de Nantes où j'ai effectué un stage qui m'a été très bénéfique.

Je voudrais remercier Professeure *Labiba Souici-Meslati* de m'avoir fait l'honneur de présider mon jury. Je veux également exprimer toute ma gratitude au Professeure *Fadila Atil* et Dr *Nora Bounour* pour avoir accepté de rapporter mon travail.

Je tiens à remercier tous les membres du laboratoire LISCO et tout particulièrement Professeur *Djamel Meslati*, Professeure *Labiba Souici-Meslati*, Professeure *Atil Fadila*, Dr *Nora Taleb* et Dr *Nora Bounour* pour leurs qualités humaines remarquables, leurs encouragements et leurs conseils tout au long de ces quatre années de thèse.

Sur le plan personnel, j'adresse le plus grand merci à mes parents et à ma sœur pour leurs soutient sans failles et leurs encouragements et prières. Un remerciement particulier à mon mari qui m'a toujours soutenu et qui m'a permis de tenir le cap jusqu'au bout, même dans les moments difficiles. J'adresse également un remerciement à ma belle famille pour leurs prières et leurs soutient.

Dédicaces

A MES PARENTS

Résumé

L'Internet des objets (IoT) représente l'évolution actuelle d'Internet, qui ouvre d'énormes possibilités pour un grand nombre d'applications novatrices qui promettent de révolutionner et d'améliorer la qualité de la vie de l'homme.

Pour cette raison, une grande attention a été portée à ce thème sous différents perspectives. Avec l'émergence rapide de l'Internet des objets dans la plupart des aspects de la vie quotidienne, la complexité des systèmes augmente relativement. Le problème traité par cette thèse est la nécessité de disposer d'un mécanisme permettant aux systèmes IoT de fonctionner sans arrêts ni ruptures, quels que soient les changements qui affectent leurs contexte.

La solution que nous proposons à ce problème comprend deux étapes principales: premièrement, donner une modélisation adéquate aux architectures IoT, c'est-à-dire proposer un modèle générique pouvant être utilisé quel que soit le domaine d'application du système. Deuxièmement, définir la mécanique opératoire à mettre en œuvre par l'architecture proposée, c'est-à-dire le processus de reconfiguration du système de manière dynamique.

Nous proposons un processus de reconfiguration dynamique contextuelle à appliquer au niveau architectural des systèmes IoT; le processus repose sur la boucle de l'informatique autonome MAPE-K. Il comprend quatre étapes principales: la surveillance du contexte, l'analyse des données acquises, la planification de la reconfiguration et enfin son exécution.

L'originalité de notre travail réside dans l'utilisation de styles architecturaux pour rendre réutilisables toutes les évolutions architecturales appliquées au système.

Mots clés: Internet des objets; Objets connectés; Sensibilité au contexte; Reconfiguration de l'IoT; Architecture logicielle; Intelligence ambiante.

Abstract

Internet of Things (IoT) is the current evolution of the Internet, which is opening tremendous opportunities for a large number of novel applications that promise to revolutionize and improve the quality of human life.

For this reason, much attention has been oriented towards this theme from different perspectives. With the rapidly expanding reach of the IoT into most aspects of everyday life, systems complexity is relatively growing. The problem treated by this thesis is the necessity of having a mechanism that enables IoT systems to perform with transparency without stops or breaks regardless of the changes that affect the surrounding context.

The solution that we propose to this problem consists of two main steps: First, giving an adequate modeling to IoT architectures, i.e. proposing a generic model that can be used regardless of the application domain of the system. The second step consists of defining the operating mechanic to be implemented by the proposed architecture, i.e., the process of reconfiguring the system in a dynamic way.

We propose a contextual dynamic reconfiguration process to be applied on the architectural level of IoT systems; the process relies on autonomic computing MAPE-K loop. It consists of four main steps including context monitoring, analysis of the acquired data, planning the reconfiguration and finally executing it.

The originality of our work is the use of architectural styles to make reusable the architectural evolutions applied on the system.

Key words: Internet of things; Connected objects; Context awareness; IoT reconfiguration; Software architecture; Ambient intelligence.

ملخص

إن إنترنت الأشياء هو التطور الحالي للإنترنت ، والذي يفتح فرصاً هائلة لعدد كبير من التطبيقات الجديدة التي تعد إحداث ثورة في نوعية الحياة البشرية وتحسينها لهذا السبب تم توجيه الكثير من الاهتمام لهذا الموضوع من جهات نظر مختلفة.

مع الانتشار المتزايد لإنترنت الأشياء في معظم جوانب الحياة اليومية ، فإن تعقيد الأنظمة ينمو نسبيا و تتمثل المشكلة التي عولجت في هذه الرسالة في ضرورة وجود آلية تمكن أنظمة إنترنت الأشياء من الأداء بشفافية دون توقف أو انقطاع بغض النظر عن التغييرات التي تؤثر على السياق المحيط.

يتألف الحل الذي نقترحه لهذه المشكلة من خطوتين رئيسيتين : أولاً ، إعطاء نموذج ملائم لهندسة إنترنت الأشياء أي اقتراح نموذج عام يمكن استخدامه بغض النظر عن مجال التطبيق الخاص بالنظام وثانيا ، تحديد ميكانيكيـــــة التشغيل الذي ستنفذها البنية المقترحة ، أي عملية إعادة تشكيل النظام بطريقة ديناميكية.

نقترح إجراء عملية إعادة تشكيل ديناميكية سياقية على المستوى الهندسي لأنظمة إنترنت الأشياء ؛ تعتمد العملية على حلقة الحوسبة التلقائية MAPE/K وتتكون من أربع خطوات رئيسية تشمل رصد السياق ، وتحليل البيانات المكتسبة ، والتخطيط لإعادة التشكيل ، وتنفيذها في النهاية.

اصالة عملنا هو استخدام الاساليب المعمارية للتمكن من اعادة استخدام التطورات المعمارية المطبقة على النظام.

الكلمات المفتاحية: إنترنت الأشياء ؛ كائنات متصلة ؛ الوعي بالسياق ؛ إعادة تشكيل إنترنت الأشياء ؛ بنية البرمجيات ؛ الذكاء المحيط.

Table des matières

Liste des figures

Liste des tableaux

INTRODUCTION

1. Cadre de la thèse

Cette thèse s'inscrit dans l'intersection de trois domaines de recherche à savoir: l'architecture logicielle, les systèmes à objets connectés et la sensibilité au contexte. Cette intersection s'explique par le fait qu'un système à objets connectés est un système hautement dynamique et sensible au contexte. L'architecture de ces systèmes évolue fréquemment en fonction des changements de contexte. Pour une utilisation meilleure des systèmes à objets connectés, les évolutions qu'ils subissent ne devraient pas affecter la disponibilité des systèmes. Nous nous intéressons plus précisément à la reconfiguration dynamique des architectures logicielles des systèmes à objets connectés.

Les systèmes à objets connectés, les systèmes cyber-physiques, l'Internet of Things sont des termes qui désignent la révolution technologique qui s'impose. Ce terme apparu premièrement en 1999 par Kevin Ashton pour définir l'interconnexion entre l'Internet et les objets du quotidien. La nouvelle Internet émerge et est censée changer le monde: il ne s'agit pas seulement de relier les gens, mais aussi de connecter diverses choses ou objets. Ce peut être une personne avec un implant cardiaque qui transmet des données, un animal qui porte une puce intelligente, une voiture dotée de capteurs indiquant la pression des pneus ou tout autre objet, créé ou non par l'homme, auquel une adresse IP est assignée et qui peut transmettre des informations.

Les statistiques actuelles prévoient que 212 milliards d'appareils seront connectés d'ici 2020 (Al-Fuqaha, Guizani et al. 2015). Cette croissance incessante est due aux atouts qu'offre l'IoT au monde et à l'homme. Cependant, il existe une multitude de domaines d'application qui vont être améliorés par l'émergence de l'IoT comme l'infrastructure intelligente, la santé, les chaînes d'approvisionnement et les applications sociales …etc.

L'augmentation du nombre d'objets connectés entrainera un accroissement de la complexité. Cet accroissement s'est vu tout naturellement accompagné de questions cruciales sur l'évolution des architectures. Quels mécanismes utiliser pour évoluer

l'architecture d'un système en cours d'exécution, et comment faire pour ne pas réinventer la roue a chaque évolution ?

Les objets connectés sont souvent influencés par l'environnement qui les entourent, ils sont amenés à évoluer en fonction des changements de contexte où des objets viennent d'apparaitre et d'autres disparaissent à la volée. Ces changements ne peuvent pas être prévus au temps de conception, c'est ce qui définit le caractère dynamique de ces objets.

Tenir le système en marche en dépit de ces changements est d'une importance primordiale. Nous somme face à des systèmes omniprésents qui peuvent parfois être critiques. Une reconfiguration dynamique contextuelle sera exigeante en termes de minimisation d'interventions et d'administrations externes. Ainsi, les coûts liés à l'indisponibilité des applications seront réduits.

L'objectif principal à travers la réalisation de cette thèse est de remédier aux problématiques posées pour permettre une reconfiguration dynamique des systèmes à base d'objets connectés.

2. Problématique de la thèse

Avec la portée croissante de l'Internet dans la plupart des aspects de la vie quotidienne, la complexité des systèmes augmente relativement. Les objets connectés sont très dynamiques, c'est-à-dire qu'ils sont entourés d'un environnement qui subit des changements de contexte fréquemment. L'objectif principal de cette étude est de pouvoir prendre en charge ces modifications sans nécessiter des pauses ou d'arrêt du système d'hébergement. Les systèmes IoT doivent être reconfigurés de manière dynamique, ce qui est loin d'être trivial, car les changements ne sont pas et ne peuvent pas être prévus au préalable (Whitmore et al. 2015). Nous ne voulons pas que la reconfiguration soit centralisée ou qu'elle affecte les performances du système ou la qualité des services qu'il offre. Les systèmes IoT devraient pouvoir s'adapter aux changements internes et externes de l'environnement (Darwish et Hassanien 2018).

L'émergence et la propagation des objets connectés est sans cesse croissante. Néanmoins, quelques questions restent à régler pour établir l'Internet des Objets comme une bonne pratique. (Pirotte juin 2014) et (Jain 2015) ont dénombré quelques défis ayant un immense impact sur la bonne utilisation de l'IoT comme l'adressage de dispositifs, la gestion efficace des ressources énergétiques, le partage et allocation d'espaces pour stocker

les objets dans le Cloud, l'analyse d'une très grande masse d'informations issues d'un immense nombre de capteurs, l'hétérogénéité et invariance à l'échelle, la sécurité et confidentialité des données et la dynamique et la haute disponibilité.

Nous adressons dans ce travail le défi qui concerne l'aspect dynamique et haute disponibilité des objets connectés. Derrière ce défi sont engendrées les occupations suscitées.

Face à un environnement qui est soumis à des changements constants en termes de nouvelles exigences et à la mobilité des utilisateurs, la gestion de l'aspect reconfiguration dynamique des systèmes basés sur l'IoT est très cruciale. Par conséquent, la reconfiguration contextuelle de ces systèmes est exigeante en termes de réduction des coûts liés à l'indisponibilité des systèmes durant les opérations de maintenance. En permettant d'éviter autant que possible les interventions humaines et les actions de maintenance. La reconfiguration reflète le changement dans la structure du système et son comportement par l'ajout, la suppression ou la modification de ses composants ou les liens entre eux.

La problématique essentielle traitée dans cette thèse étant la reconfiguration dynamique des systèmes à objets connectés au niveau architectural. La proposition d'une solution à ce sujet a donné naissance à plusieurs sous problématiques:

1. Il n'existe pas encore une architecture standardisée qui prend en charge tous les aspects de l'Internet des Objets;
2. Une très grande masse de données sera acquise, quels mécanismes pour filtrer ces données?
3. Comment modéliser les informations contextuelles acquises et comment faire pour les conserver de sorte qu'ils soient facile a interroger lors de la prise de décision;
4. Comment décider si les changements de contexte nécessitent une reconfiguration;
5. Quels mécanismes utiliser pour reconfigurer le système, et puis comment faire pour réutiliser les connaissances de reconfiguration dans une situation récurrente.
6. Comment s'assurer de la cohérence du système après son évolution.

3. Contributions

Le développement de systèmes informatiques mobiles nécessite des activités de modélisation qui précèdent la conception technique d'un système logiciel. Le contexte d'un système mobile comprend un large éventail d'aspects techniques, physiques, sociaux et

organisationnels. Certains de ces aspects doivent être intégrés dans le système. La sélection des aspects qui sont nécessaires devient de plus en plus une tâche complexe dans les systèmes mobiles comparés à ce qu'avons vu précédemment dans les systèmes d'information traditionnels. C'est dans ce contexte que se situe ce travail de thèse. Il est question de la prise en charge, sur le plan de démarche, modélisation et sur le plan implémentation, de la reconfiguration dynamique des architectures à base d'objets connectés.

L'intérêt derrière cette proposition revient aux gains obtenus à travers la diminution des couts issus de l'adaptation statique des systèmes.

Face aux problématiques citées précédemment, nous proposons dans cette thèse une architecture réflexive qui implémente un processus de reconfiguration dynamique des systèmes à objets connectés. Le processus s'inspire des étapes de l'informatique autonome, il repose sur l'acquisition d'informations contextuelles, l'analyse de ces informations, la planification de la reconfiguration et enfin son exécution. A la fin de ces étapes, un mécanisme d'autogestion du système sera déclenché pour s'assurer de la conservation des paramètres non fonctionnels du système après avoir l'évolution.

Nous avons travaillé sur le sujet en essayant de répondre aux quatre dimensions qui définissent l'évolution dans les systèmes à objets connectés et qui vont aider à construire le processus correspondant. La première dimension correspond à qui intervient dans ces systèmes, c'est-à-dire qui est le responsable derrière le déclenchement du processus de reconfiguration. Nous parlerons du « contexte » et de la « sensibilité au contexte » et des mécanismes appropriés pour son exploitation. La deuxième dimension correspond à quel moment doit-on intervenir, les changements dans les systèmes à objets connectés ne peuvent pas être prévus au temps de conception, le système évolue en cours de son exécution. La troisième dimension correspond à quoi on s'intéresse: les objets connectés, l'Internet des Objets, ce type de systèmes où tous les objets du monde réel sont interconnectés, communiquent et interagissent. La dernière dimension reflète le comment: La technique de réflexion est utilisée. Le processus de reconfiguration contextuelle dynamique consiste en trois étapes en s'inspirant de la boucle de l'informatique autonome y compris l'acquisition des données, la prise de décision, l'exécution de la reconfiguration.

Après avoir proposé un processus de reconfiguration, il nous reste à évaluer les performences de celui-ci. Pour ce faire, une étude de cas dans le domaine de santé

électronique à été introduite où une personne âgée atteinte du diabète vie seule à la maison. Dans le but de maintenir sa vie à bien, nous visons à rendre sa maison intelligente en connectant les objets qu'il utilise. De ce fait, nous avons minimisé les interventions de la personne dans les changements autour de lui.

Les contributions essentielles de cette thèse sont:

1. La proposition d'un modèle d'architecture générique pour l'Internet des objets
2. Le développement d'un processus de reconfiguration contextuelle dynamique implémenté par l'architecture proposée.
3. Utilisation du Cloud pour stocker et traiter les informations contextuelles
4. Utilisation de styles d'évolution pour rendre les évolutions architecturales réutilisables afin d'exploiter les compétences déjà acquises.
5. Simulation d'une maison intelligente qui supporte les évolutions dynamiques.

4. Plan du manuscrit

En plus de l'introduction et la conclusion générale, ce manuscrit est organisé en cinq chapitres dont la structure est la suivante:

Chapitre 1: Le but de ce chapitre est de poser clairement le domaine des objets connectés. Nous mettons le point sur l'Internet des Objets, l'étude du domaine, ses stations historique et les concepts de base. Ainsi que leurs caractéristiques et les défis autours d'eux et les architectures de ces systèmes.

Chapitre 2: Le deuxième chapitre introduit la notion de contexte et la sensibilité au contexte, les travaux existants et les étapes de prise en compte du contexte. A la fin du chapitre, les travaux sur la conscience au contexte des systèmes à objets connectés ont été discutés.

Chapitre 3: L'évolution des architectures logicielles est étudiée dans ce chapitre, notamment ses définitions, ses types, les dimensions qui forment une évolution. Une étude sur les modèles d'évolution existants est donnée selon différents critères. La notion de style d'évolution est ainsi introduite.

Chapitre 4: Dans ce chapitre, nous nous intéressons à la reconfiguration dynamique des systèmes à objets connectés sur le niveau architectural. Nous détaillons le fonctionnement du processus correspondant y compris l'architecture proposée, ses composants et les étapes du processus qu'elle implémente. Le gestionnaire d'évolution est ainsi présenté.

Chapitre 5: Après avoir proposé un processus de reconfiguration, il est nécessaire de l'évaluer face à un exemple concret. La réalisation du processus est faite à l'aide de l'outil Emfatic d'Eugenia sous Eclipse. Un plugin décrivant le cas d'étude est résultant.

L'étude de cas consiste en une maison intelligente dotée d'objets connectés pour donner plus de confort à son propriétaire (patient diabétique âgé). Une simulation de la maison est faite en utilisant le simulateur Cisco Packet Tracer.

La structure de la thèse est illustrée dans la Figure 1.

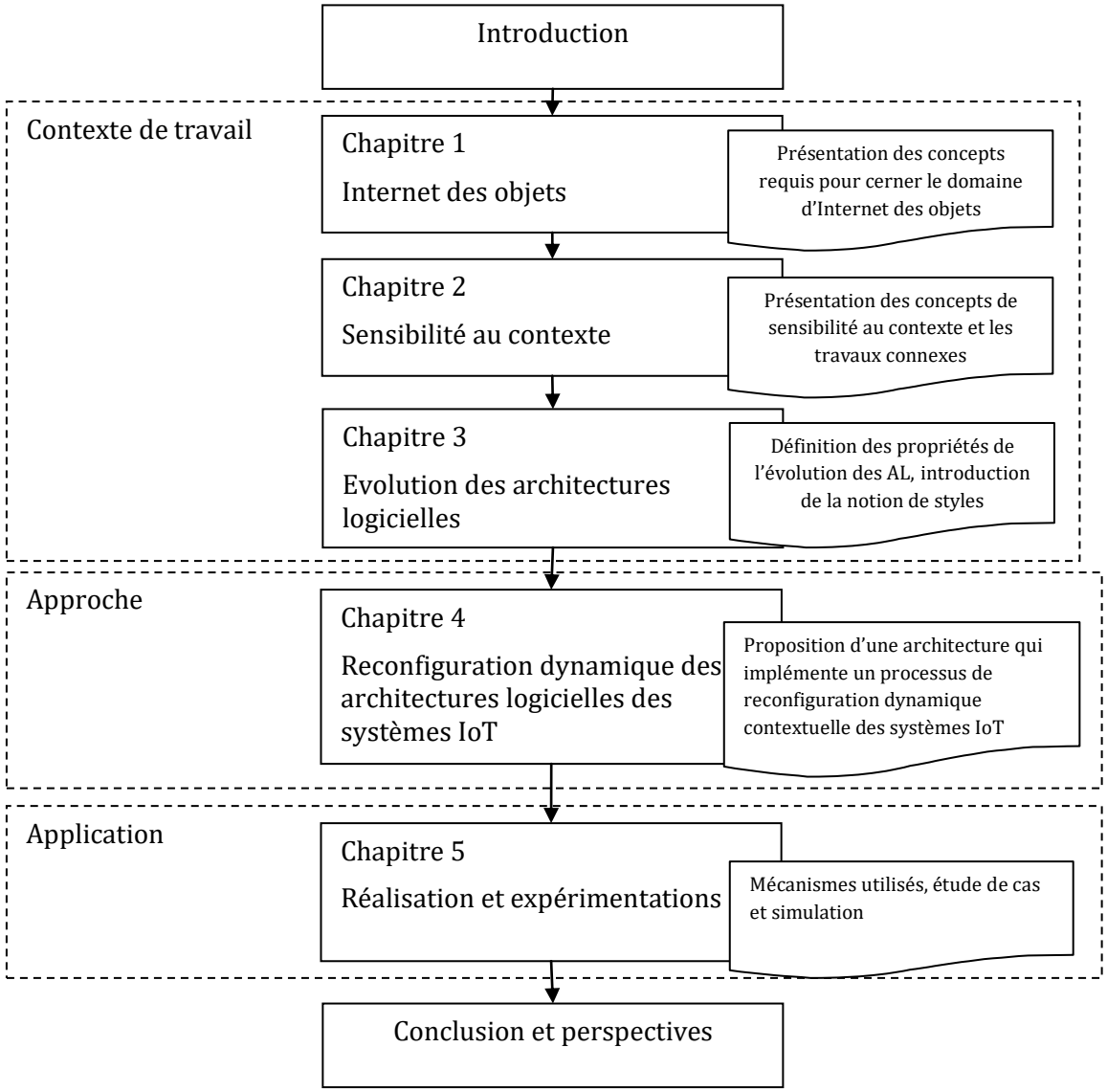

Figure 1- Structure de la thèse

CHAPITRE 1

Internet des Objets: Etat du domaine

1.1. Introduction

Le travail présenté dans cette thèse relève de l'évolution dans les architectures logicielles des systèmes à objets connectés. Auparavant, il est nécessaire de poser clairement le domaine et lorsque nécessaire, d'éclairer certains points et caractéristiques que nous jugeons importants pour notre travail. L'objectif du premier chapitre est de clarifier la notion d'objets connectés. Dans ce but, nous proposons un état du domaine d'une révolution désormais majeure qui a mûrie de manière significative ces vingt dernières années.

Un nombre important de recherches ont été conduites sur la nouvelle forme de l'Internet: L'internet des objets de l'anglais Internet of Things (IoT). Kevin Ashton a introduit ce terme pour la première fois dans une présentation pour l'entreprise Procter and Gamble (P&G) en 1998 (Ashton 2009). Lu Tan et al (Tan and Wang 2010) définissent l'IoT comme: *"things have identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environment and user context"*. L'IoT vient de déclencher une révolution dans l'ère de l'Internet et de la communication en concrétisant la vision de l'informatique ubiquitaire telle qu'imaginée en 1991 par Mark Weiser (Weiser 1999), où la technologie s'efface peu à peu dans l'environnement des utilisateurs, intégrée naturellement à l'intérieur des objets du quotidien.

Ce chapitre met le point sur l'Internet des Objets, l'étude du domaine, ses stations historique et les concepts de base. Ainsi que leurs caractéristiques et les défis autours d'eux et les architectures de ces systèmes.

1.2. Concepts de base

Les concepts de base relatifs à l'internet des objets sont énumérés dans cette section. Ces concepts nous permettrons d'identifier les éléments clés qui composent ces systèmes, les composants structurels et comportementales de leurs architectures.

1.2.1. Objet

Un objet est défini par le dictionnaire Larousse (Larousse 1865) comme *« Toute chose concrète, perceptible par la vue, le toucher »* par exemple, un livre, une voiture, une machine à café. Cependant, nous parlons d'objets connectés pour définir des types d'objets dont la vocation première n'est pas d'être des périphériques informatiques ni des interfaces d'accès au web, mais auxquels l'ajout d'une connexion Internet a permis d'apporter une valeur supplémentaire en termes de fonctionnalités, d'information, d'interactions avec l'environnement.

Dans le contexte précis de l'Internet des objets, et ce quelle que soit la vision, cet objet possède au minimum un identifiant unique attaché à une identité (Atzori, Iera et al. 2010) exprimant d'une part ses propriétés immuables (type, couleur, poids, etc.) et son état c'est-à-dire l'ensemble de ses caractéristiques pouvant évoluer au cours du temps (position, niveau de batterie, etc.) (Billet 2015).

Nous distinguons deux types d'objets connectés selon (Guillaume Plouin 2011), les objets passifs et les objets actifs.

- Les objets passifs: Ils utilisent généralement un tag (puce RFID, code barre 2D). Ils ont une capacité de stockage faible (de l'ordre du kilo octet) et permettent de jouer le rôle d'identification. Ils peuvent parfois, dans le cas d'une puce RFID, embarquer un capteur (température, humidité) et être réinscriptibles.
- Les objets actifs: ils peuvent être équipés de plusieurs capteurs, avec une grande capacité de stockage, capables d'accomplir des calculs et être en mesure de communiquer sur un réseau.

Les objets de l'environnement de l'IoT permettent de capter c'est-à-dire transformer une grandeur physique analogique en un signal numérique, stocker: rassembler des données brutes, produites en temps réel, arrivant de façon non prévue et, transmettre des données issues du monde physique.

1.2.2. Internet des objets

De l'anglais « Internet of Things », appelé aussi systèmes à objets connectés, systèmes cyber-physiques ; ce terme a été introduit pour la première fois en 1999 par Kevin Ashton dans une présentation pour l'entreprise Procter and Gamble (Ashton 2009). Il posait son idée comme suit: « *Si nous avions des ordinateurs qui savaient tout ce qu'il y avait à savoir, en utilisant les données qu'ils ont rassemblées sans aucune aide de notre part, nous serions en mesure de tout suivre et de tout compter et de réduire considérablement le gaspillage, les pertes et les coûts. Nous saurions quand les choses doivent être remplacées, réparées ou rappelées, et si elles sont fraîches ou dépassées.* »

Au cours de la dernière décennie, l'IoT a attiré une attention considérable dans les milieux académiques et industriels. Les principales raisons derrières cet intérêt sont les fonctionnalités offertes par l'IoT. Il promet de créer un monde où tous les objets qui nous entourent sont connectés à Internet et communiquent les uns avec les autres avec un minimum d'interventions humaine. Le but ultime est de créer un monde meilleur pour les êtres humains, où les objets qui nous entourent savent ce que nous aimons, ce que nous voulons et ce dont nous avons besoin et agissent ainsi sans instructions explicites.

Nous citons dans ce qui suit les définitions les plus pertinentes qui ont été données à l'IoT à travers la littérature:

– Définition de (Tan and Wang 2010): *"Things have identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environment, and user contexts."*

– Définition de (Commission 2008): *"The semantic origin of the expression is composed by two words and concepts: Internet and Thing, where Internet can be defined as the world-wide network of interconnected computer networks, based on a standard communication protocol, the Internet suite (TCP/IP), while Thing is an object not precisely identifiable Therefore, semantically, Internet of Things means a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols."*

– Définition de (Vermesan, Friess et al. 2011): *"The Internet of Things allows people and things to be connected Anytime, Anyplace, with Anything and Anyone, ideally using Any path/network and Any service."*

– Définition de le CERP-IdO « Cluster des projets européens de recherches sur l'Internet des objets » (Sundmaeker, Guillemin et al. 2010) «Une infrastructure dynamique d'un réseau global. Ce réseau global à des capacités d'auto configuration basées sur des standards et protocoles de communication interopérables. Dans ce réseau, les objets physiques et virtuels ont des identités, des attributs physiques, des personnalités virtuelles et des interfaces intelligentes, et ils sont intégrés au réseau d'une façon transparente». Cette définition montre les deux aspects de l'IoT: temporel et spatial qui permettent aux personnes de se connecter de n'importe où et à n'importe quel moment à travers des objets connectés.

L'IoT est selon (Atzori, Iera et al. 2010), un paradigme qui peut être vu comme une convergence de trois visions différentes à savoir une vision orientée objets, une vision orientée Internet et une vision orientée sémantique. Dans la vision orientée objets s'effectue le suivi et la surveillance de l'objet à l'aide de capteurs et des technologies omniprésentes. Dans la vision orientée Internet, l'objet physique est converti en objet intelligent. Le protocole Internet est attribué à un objet utilisé pour accéder à l'objet de manière unique. L'objet activé par le capteur est converti en puce, qui est identifiée de manière unique et surveillée en permanence. Dans la vision sémantique, une quantité énorme de périphériques physiques activés par capteur sont connectés sur Internet. Les capteurs surveillent et génèrent en continu des données massives. Les données générées sont éventuellement redondantes et se présentent sous forme homogène et hétérogène. La vision sémantique aide à traiter les données de manière significative et à prendre les mesures et décisions nécessaires de manière appropriée.

L'IoT mets en œuvre des moyens permettant à une grandeur physique de renseigner un système informatique et, inversement, des moyens permettant à un système informatique d'agir sur le monde physique. L'IoT utilise deux types d'éléments pour interagir avec le monde physique : des capteurs et des actionneurs.

1.2.3. Capteurs

Les capteurs permettent de recueillir des informations depuis le monde physique et de les transmettre vers le système informatique. Ils traduisent une grandeur physique en un signal électrique. Ce dernier est ensuite numérisé pour être transmis au système informatique. Par exemple, un capteur de température permet de traduire l'amplitude de la

température en une tension électrique. Cette dernière est numérisée puis transmise. Exemples de capteurs: lumière, proximité, position, déplacement, accélération, rotation, température…etc.

Les réseaux de capteurs sans fil (les Wireless Sensor Network (WSN)) et l'identification par radiofréquence (RFID) sont considérés comme les deux principaux composants des technologies de détection et de communication pour l'IoT (Miorandi, Sicari et al. 2012).

La technologie RFID est considérée comme un développement important dans le domaine des appareils embarqués. En fait, la RFID permet la conception de micro-puces minuscules (appelées tags), qui peuvent être ajoutées à un objet de notre vie quotidienne. En conséquence, les données stockées dans ces balises peuvent automatiquement être utilisées pour identifier et extraire des informations utiles de l'objet. Ainsi, le tag agit comme un code à barres électronique.

Les WSNs joueront également un rôle crucial dans le futur déploiement de l'IoT. Ils peuvent coopérer avec les systèmes RFID pour mieux suivre l'état des choses. WSN est en mesure d'accroître leur prise de conscience de l'environnement. Par conséquent, ils agissent comme un pont supplémentaire entre le monde physique et le monde numérique (Abdmeziem, Tandjaoui et al. 2016).

Les réseaux de capteurs consistent en un nombre très élevé de nœuds de détection communiquant de manière multi-sauts sans fil. En général, les nœuds communiquent leurs résultats de détection à un petit nombre de nœuds spéciaux appelés récepteurs.

1.2.4. Actionneurs

Les actionneurs sont des dispositifs qui transforment une donnée digitale en phénomène physique pour créer une action. Ils permettent au système informatique d'agir sur le monde physique en modifiant son état. Par exemple, un actionneur peut allumer un appareil à distance. Exemple d'actionneurs: afficheurs, alarmes, caméras, haut-parleurs, interrupteurs, lampes, moteurs, pompes, serrures, vannes, ventilateur…etc.

Concrétiser la vision future de nos sociétés dans le cadre de l'IoT ne peut être réalisé en limitant la portée de l'amélioration technologique au cyberespace. En fait, un soutien physique (c'est-à-dire l'actionnement) dans le monde réel est absolument nécessaire(Hu, Guoqiang, Tay et al. 2012).

Revenant aux éléments de l'IoT, selon (Atzori, Iera et al. 2010), les techniques de communication essentiels, câblés ou sans fils et les éléments de l'IoT sont:

1. Technologies d'identification, de détection et de communication: Les systèmes RFID, les WSN.
2. Le middleware: Il repose sur les couches suivantes:

   2.1. Application: Exportation des fonctionnalités du système vers l'utilisateur final.

   2.2. Composition de service: Il fournit les fonctionnalités pour la composition de services uniques offerts par des objets en réseau pour construire des applications spécifiques.

   2.3. Gestion des services: Gestion des fonctionnalités offertes par chaque objet, cette couche peut permettre le déploiement à distance de nouveaux services à l'exécution afin de satisfaire les besoins de l'application

   2.4. Abstraction d'objets: L'IoT repose sur un ensemble vaste et hétérogène d'objets, chacun fournissant des fonctions spécifiques accessibles à travers son propre dialecte.

   2.5 Confiance, sécurité et confidentialité.

Pour assurer leurs fonctionnements, les systèmes IoT nécessitent avoir des mécanismes qui leurs permettent de capturer, identifier, agir, communiquer et gérer les informations contextuelle. Pour cela, (Ray 2018) dénombre les blocs fonctionnels suivants de l'IoT:

- **Dispositif**: Un système IoT est basé sur des dispositifs fournissant des activités de détection, d'actionnement, de contrôle et de surveillance. Les appareils IoT peuvent être de types variés, tels que des capteurs portables, des montres intelligentes, des lampes à LED, des automobiles et des machines industrielles.
- **Communication**: Le bloc de communication établit la communication entre les périphériques et les serveurs distants. Les protocoles de communication IoT fonctionnent généralement dans la couche liaison de données, la couche réseau, la couche de transport et la couche application.
- **Services**: Un système IoT gère divers types de fonctions, telles que des services de modélisation de périphérique, de contrôle de périphérique, de publication de données, d'analyse de données et de découverte de périphérique.
- **Gestion**: Le bloc de gestion fournit différentes fonctions permettant de gérer un système IoT afin de rechercher la gouvernance sous-jacente du système IoT.

- **Sécurité**: Sécuriser le système IoT en fournissant des fonctions telles que l'authentification, l'autorisation, la confidentialité, l'intégrité et la sécurité des données.
- **Application**: La couche application agit en tant qu'interface fournissant les modules nécessaires pour contrôler et surveiller divers aspects du système IoT. Les applications permettent aux utilisateurs de visualiser et d'analyser l'état du système au stade actuel de l'action, prédisant parfois des perspectives futuristes.

1.2.5. Cloud computing

La notion de Cloud computing s'est vue attachée et accompagnée au sujet de l'Internet des objets et plus spécifiquement à l'Internet mobile. Il s'agit d'une infrastructure où le stockage et le traitement des informations surviennet en dehors de l'appareil mobile. De ce fait, les appareils mobiles n'ont pas besoin d'une configuration puissante puisque tous les modules complexes en termes de gestion peuvent etre traités dans le cloud.

L'un des résultats les plus importants de l'IoT est l'énorme quantité de données générées à partir d'appareils connectés à Internet (Gubbi, Buyya et al. 2013). De nombreuses applications IoT requièrent un stockage de données considérable, une vitesse de traitement considérable pour permettre la prise de décision en temps réel et des réseaux haut débit pour la transmission de données, audio ou vidéo. Le Cloud computing fournit une solution idéale pour gérer des flux de données énormes et les traiter en temps réel pour un nombre important d'appareils IoT et d'êtres humains (Lee and Lee 2015).

1.3. Historique

L'Internet des objets représente l'évolution naturelle, et direction, de toutes les avancées technologiques de la dernière décennie. Avant d'examiner en profondeur l'Internet des objets, il convient d'examiner en premier l'évolution de l'Internet.

Figure 2 illustre les cinq phases de l'évolution d'Internet à savoir la création de réseaux informatique, l'internet, l'internet mobile, le web social, et enfin l'internet des objets ou le machine to machine (M2M).

- Fin des années 60, la communication entre deux ordinateurs est devenue possible par l'utilisation d'un réseau informatique.

- Au début des années 1980, le TCP/IP a été introduit et la commercialisation de l'Internet a commencé.
- Au début des années 90, la première vague d'Internet a commencé avec la création et la fondation d'infrastructures du web en ligne. Beaucoup d'entreprises à travers le monde ont créé des machines, des logiciels *et* des réseaux pour rendre Internet disponible aux utilisateurs dans un usage privé et commercial, il s'agit du web 1.0.
- Au début des années 2000, la seconde vague déferle. Il s'agit du web 2.0 ou web social. Il privilégie la dimension de partage et d'échange d'informations et de contenus (textes, vidéos, images ou autres). Il se manifeste par l'émergence des réseaux sociaux et des blogs. Aussi, des moteurs de recherche comme Google qui aident les internautes à se retrouver dans des milliards de pages disponibles sur le net ou encore le commerce en ligne avec des sites comme Amazon et eBay qui font leurs apparitions. Cette seconde vague révolutionne le monde avec l'apparition du smart phone qui a apporté l'accès à l'internet mobile (Perera, Zaslavsky et al. 2014).
- La prochaine étape de l'IoT était de permettre aux objets qui nous entourent de se connecter les uns aux autres (par exemple, machine à machine) et de communiquer via Internet. L'internet comme spécifié va d'évolution en évolution. Déjà démocratisé avec le développement fulgurant des smart phones et tablettes, la 3ème ère de l'internet appelée web 3.0 ou web intelligent désigne un Internet qui s'affranchi de toute plateforme pour devenir accessible partout: c'est l'Internet des objets (IoT). Ainsi l'Internet des objets représente la 3ème évolution de l'internet. Son principe consiste à connecter toutes sortes d'appareils à Internet (Des téléphones portables aux montres en passant par les cafetières, et bien sûr, les frigos, les voitures, les maisons etc.).

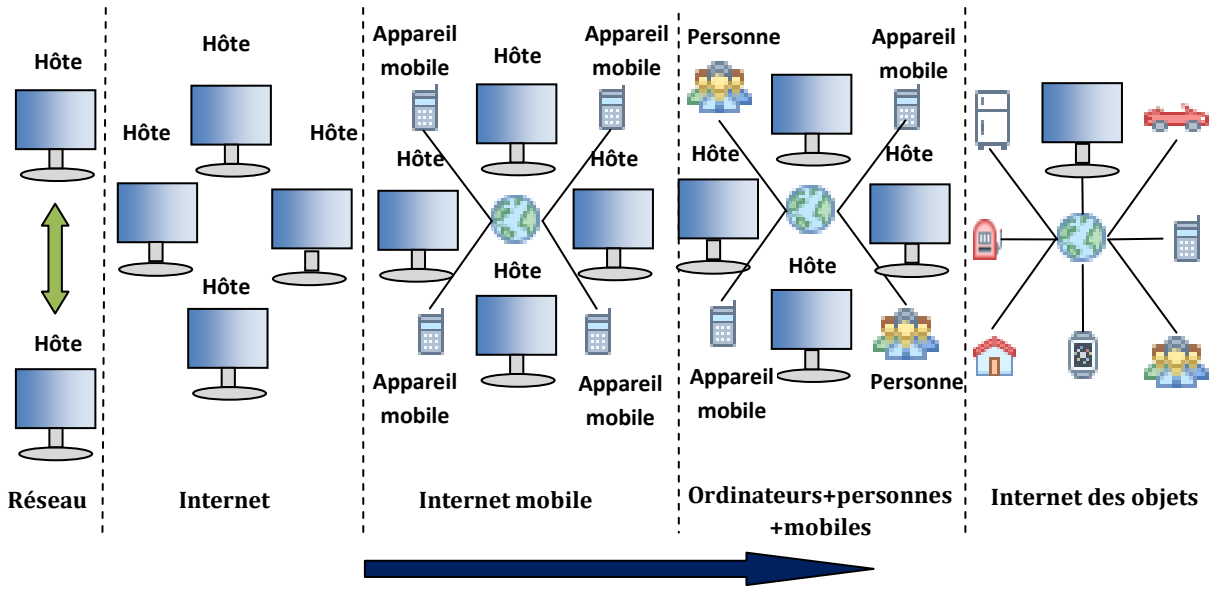

Figure 2 - Phases d'évolution de l'Internet (Perera, Zaslavsky et al. 2014)

Les premiers objets connectés n'apparaissent que dans les années 90. Il s'agit de grille-pain, machines à café ou autres objets du quotidien. En 2000, le fabricant Coréen LG est le premier industriel à parler sérieusement d'un appareil électroménager relié à Internet, Les années 2000 ont vu les premières expérimentations d'appareils connectés à Internet. En 2003, la population mondiale s'élevait à environ 6,3 milliards d'individus et 500 millions d'appareils étaient connectés à Internet. Selon la définition de Cisco IBSG, l'IoT n'existait pas encore en 2003 car le nombre d'objets connectés était faible. En raison de l'explosion des Smartphones et des tablettes, le nombre d'appareils connectés à Internet a atteint 12,5 milliards en 2010, alors que la population mondiale était de 6,8 milliards.

En ce qui concerne l'avenir, Cisco IBSG estime que 50 milliards d'appareils seront connectés à Internet d'ici 2020. Il est important de noter que ces estimations ne tiennent pas compte des progrès rapides d'Internet ni des avancées technologiques, mais reposent sur les faits avérés à l'heure actuelle (Evans 2011).

1.4. Défis autour de l'IoT

L'IoT offre de nombreuses nouvelles opportunités à l'industrie et aux utilisateurs dans de nombreux domaines d'application. Cependant, à l'heure actuelle, l'IoT lui-même manque de théories, d'architecture standards et de normes intégrant le monde virtuel et le monde physique dans un cadre unifié (Billet 2015). Les principaux défis posés par (Chen, Xu et al. 2014) sont listés ci-dessous:

- **Défi architectural**

L'IoT englobe un très large éventail de technologies. Il implique un nombre croissant de dispositifs et de capteurs intelligents interconnectés qui sont souvent non intrusifs, transparents et invisibles. Comme les communications entre ces dispositifs sont censées se dérouler à tout moment et en tout lieu pour tous les services associés, ces communications se font généralement de manière sans fil, autonome et ad hoc. De plus, les services deviennent beaucoup plus mobiles, décentralisés et complexes.

En IoT, les intégrations de données sur différents environnements sont difficiles et seront supportées par des composants modulaires interopérables. Les solutions d'infrastructure nécessiteront que les systèmes combinent des volumes de données provenant de diverses sources et déterminent les caractéristiques pertinentes, interprètent les données et montrer leurs relations, comparer les données aux informations utiles et faciliter la prise de décision. Une architecture de référence unique ne peut donc pas être un modèle pour toutes les applications.

Des architectures de référence hétérogènes doivent coexister dans l'IoT. Les architectures doivent être ouvertes et, conformément aux normes, elles ne doivent pas empêcher les utilisateurs d'utiliser des solutions fixes de bout en bout. Les architectures IoT doivent être flexibles pour prendre en charge des cas tels que l'identification (RFID, tags), les dispositifs intelligents et les objets intelligents.

- **Défi technique:**

La technologie IoT peut être complexe pour diverses raisons. Premièrement, il existe des architectures hétérogènes dans les technologies et applications réseau existantes, par exemple, des applications et des environnements différents ont besoin de technologies de réseau différentes, ainsi que d'autres caractéristiques de réseaux cellulaires, des réseaux locaux sans fil et des technologies RFID sont très différentes les unes des autres. Deuxièmement, les technologies de communication, y compris les systèmes de communication fixes et mobiles, les communications sur ligne électrique, les communications sans fil et les technologies de communication sans fil à courte portée, pour les dispositifs fixes et mobiles, simples ou complexes, devraient être peu coûteuses et avec une connectivité fiable. Enfin, il existe des milliers d'applications différentes; Il est naturel

que différentes exigences soient imposées aux parties en matière de communication, au type de solutions de sécurité appropriées, etc.

- **Défi matériel**

Les appareils intelligents dotés d'une communication améliorée entre appareils mèneront à des systèmes intelligents dotés d'une intelligence élevée. L'autonomie des appareils intelligents permet un déploiement rapide d'applications IoT et la création de nouveaux services. Par conséquent, les recherches sur le matériel se concentrent sur la conception de systèmes identifiables sans fil avec une taille réduite, de faible coût et ayant des fonctionnalités suffisantes. La bande passante des terminaux IoT pouvant varier de kbps à mbps, de la simple détection de valeur au flux vidéo, les besoins sur le matériel sont divergents. Cependant, deux exigences ont néanmoins été essentielles: l'une est la consommation électrique extrêmement faible en mode veille et l'autre est un coût extrêmement faible.

Des milliards de terminaux IoT seront utilisés; le coût d'un terminal IoT doit être ultra bas. Cependant, jusqu'à présent, il n'y avait pas de solution de positionnement à faible coût pour l'IoT, en particulier la précision de positionnement d'un terminal IoT à courte portée devait être élevée.

Une faible puissance active constitue également un défi pour les terminaux à faible coût. Traditionnellement, un faible coût équivaut à une performance inférieure ou à une latence de processus plus longue. Une latence de traitement plus longue aboutit à une consommation d'énergie plus élevée. D'une autre manière, le futur IoT devra peut-être utiliser une bande de spectre très étroite non encore utilisée entre deux bandes utilisées. Pour utiliser une bande très étroite avec des voisins puissants, le coût de la composante passive ne sera pas faible et cela constituera certainement un défi potentiel à l'avenir.

- **Défi de la confidentialité et de la sécurité**

Par rapport aux réseaux traditionnels, les problèmes de sécurité et de confidentialité de l'IoT deviennent plus importants. La plupart des informations incluent la vie privée des utilisateurs, de sorte que la protection de la vie privée devienne un problème de sécurité important dans l'Internet des objets. En raison des combinaisons d'éléments, de services et de réseaux, la sécurité de l'IoT doit couvrir davantage d'objets et de niveaux de gestion que la sécurité réseau traditionnelle. L'architecture de sécurité existante est conçue à partir du

point de vue de la communication humaine, peut ne pas être appropriée et directement appliquée au système IoT. L'utilisation de mécanismes de sécurité existants bloquera la relation logique entre les objets dans l'IoT.

IoT a besoin de solutions techniques à faible coût et orientées M2M pour garantir la confidentialité et la sécurité. Dans de nombreux cas d'utilisation, la sécurité d'un système a été considérée comme une fonctionnalité générale. Les recherches connexes doivent porter sur le contrôle de la vie privée. Les algorithmes de cryptographie à faible coût, à faible temps de latence et à faible consommation d'énergie, ainsi que le matériel flexible associé, seront essentiels pour les capteurs ou les périphériques.

- **Défi standard**

Les normes jouent un rôle important dans la formation de l'IoT. Une norme est essentielle pour permettre à tous les acteurs d'accéder et d'utiliser de manière égale. Les développements et la coordination des normes et des propositions favoriseront le développement efficace des infrastructures, des applications, des services et des dispositifs IoT. En règle générale, les normes élaborées par des multipartis ayant coopéré, ainsi que les modèles et protocoles d'information inclus dans les normes, doivent être ouverts. Le processus d'élaboration des normes doit également être ouvert à tous les participants, et les normes résultantes sont accessibles au public et librement. Dans le monde des réseaux d'aujourd'hui, les normes mondiales sont généralement plus pertinentes que les accords locaux.

- **Défi métier**

Pour une application mature, son modèle commercial et son scénario d'application sont clairs et faciles à intégrer aux exigences techniques. Les développeurs n'ont donc pas besoin de passer beaucoup de temps sur les aspects liés aux affaires. Mais pour l'IoT, il existe trop de possibilités et d'incertitudes dans les modèles commerciaux et les scénarios d'application. Elle est donc inefficace en termes d'alignement technologie-entreprise et une solution unique ne conviendra pas pour tous. L'IoT est un modèle économique traditionnel ambitieux. Bien que les applications à petite échelle aient été rentables dans certaines industries, elles ne sont pas durables lorsqu'elles sont étendues à d'autres industries. Au début du développement de l'IoT, les aspects commerciaux doivent être pris en compte pour réduire le risque d'échec.

1.5. Caractéristiques des systèmes IoT

Nous explorons dans cette section les caractéristiques de l'IoT. En se basant sur les recherches précédentes, (Perera, Zaslavsky et al. 2014) ont identifié sept caractéristiques principales de l'IoT y compris: l'intelligence, l'architecture, complexité, considérations de la taille, considérations temporelles, considérations spatiales et tout en tant que service. Ces caractéristiques doivent être prises en compte lors du développement de solutions IoT au cours de toutes les phases de la conception, du développement, de la mise en œuvre et de l'évaluation.

- **L'intelligence**

Cela signifie l'application de la connaissance. Tout d'abord, les connaissances doivent être générées par la collecte de données et leur raisonnement. La transformation des données brutes collectées en connaissances (informations de haut niveau) peut être réalisée en collectant, en modélisant et en raisonnant le contexte. Le contexte peut être utilisé pour fusionner les données de capteurs afin d'inférer de nouvelles connaissances. Une fois que nous avons des connaissances, elles peuvent être appliquées pour une interaction et une communication plus intelligentes.

- **L'architecture**

L'IoT devrait être facilité par une architecture hybride qui comprend de nombreuses architectures différentes. Il y aurait principalement deux architectures: événementielle et temporelle. Certains capteurs produisent des données lorsqu'un événement survient, les autres produisent des données en continu, sur la base de délais spécifiés (par exemple, un capteur de température). Les règles événement-condition-action (ECA) sont couramment utilisées dans de tels systèmes.

- **Système complexe**

L'IoT comprend un grand nombre d'objets (capteurs et actionneurs) qui interagissent de manière autonome. Les nouveaux objets commenceront à communiquer et les objets existants disparaîtront. Actuellement, des millions de capteurs sont déployés dans le monde. Les interactions peuvent différer considérablement en fonction des capacités des objets. Certains objets peuvent avoir très peu de fonctionnalités et, à ce titre, stockent des informations très limitées et ne font aucun traitement. En revanche, certains objets peuvent

avoir des capacités de mémoire, de traitement et de raisonnements plus importants, ce qui les rend plus intelligents.

- **Considération de taille**

Il est prévu qu'il y aura 50-100 milliards d'appareils connectés à Internet d'ici 2020. L'IoT doit faciliter l'interaction entre ces objets. Les chiffres vont augmenter continuellement et ne vont jamais diminuer. Similaire au nombre d'objets, le nombre d'interactions peut aussi augmenter de manière significative.

- **Considérations temporelles**

L'IoT pourrait gérer des milliards d'événements parallèles et simultanés, en raison du nombre considérable d'interactions. Le traitement des données en considération temporelles est essentiel.

- **Considérations spatiales**

La localisation géographique précise de l'objet est essentielle car l'emplacement joue un rôle important dans l'informatique contextuelle. Lorsque le nombre d'objets augmente, le suivi devient une exigence essentielle. Les interactions dépendent fortement de leurs emplacements, de leurs environnements et de la présence d'autres entités (objets et personnes, par exemple).

- **Tout en tant que service**

En raison de la popularité du Cloud Computing, consommant des ressources en tant que service, telles que PaaS (Platform-as-a-Service), IaaS (Infrastructure-as-a-Service), SaaS (Software-as-a-Service)), est devenu le flux principal. Le modèle Tout en tant que Service est hautement efficace, évolutif et facile à utiliser. L'IoT exige la mise en place de nombreuses infrastructures pour concrétiser sa vision, en suivant une approche basée sur la communauté ou la foule. Par conséquent, le partage serait essentiel, dans le cas où un modèle «tout en tant que service» conviendrait essentiellement à la détection en tant que service (sensing-as-a-service).

1.6. Architecture des systèmes IoT

L'IoT est capable d'interconnecter des milliards d'objets via Internet. Il est très difficile de gérer des objets entiers sur Internet. Donc, il existe un besoin pour une architecture normalisée (Ray 2018).

L'architecture IoT peut être traitée comme un système physique, virtuel ou hybride des deux, consistant en un ensemble de nombreux objets physiques actifs, capteurs, actionneurs, services Cloud, protocoles IoT spécifiques, couches de communication, utilisateurs, développeurs, etc. et couche entreprise (Ray 2018).

L'architecture de base de l'IoT consiste en trois couches (présentée à la Figure 3). En analysant cette structure de l'Internet des objets, (Wu, Lu et al. 2010) suggèrent que l'architecture à trois couches décrit la structure de l'IoT du point de vue technique, ce qui est raisonnable au stade initial de développement. Cette structure ne peut pas exprimer toutes les caractéristiques et les connotations de l'Internet des objets.

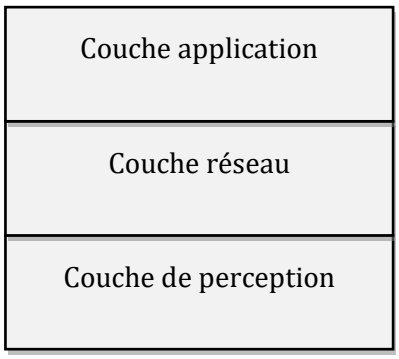

Figure 3 - Architecture de base de l'IoT(Wu, Lu et al. 2010)

Selon (Khan, Khan et al. 2012), l'IoT est construit sur cinq couches principales (Figure 4) permettant son fonctionnement, y compris la couche de perception constituée d'objets physiques et de capteurs. Cette couche s'occupe de l'identification d'objets et la collecte de données contextuelles, la couche réseau ayant pour rôle de transférer les informations des capteurs au système de traitement de l'information par le biais de supports de transmission câblés ou sans fil et de technologies telles que la 3G, l'infrarouge, le Wi-Fi, ZigBee, etc. La couche middleware traite la gestion des services mis en œuvre par les appareils, la couche application gère les applications mises en œuvre par l'IoT, telles que la santé intelligente, l'agriculture intelligente, la maison intelligente, la ville intelligente, le transport intelligent,

etc. et enfin la couche métier qui est responsable de la gestion du système global IoT, y compris des applications et des services.

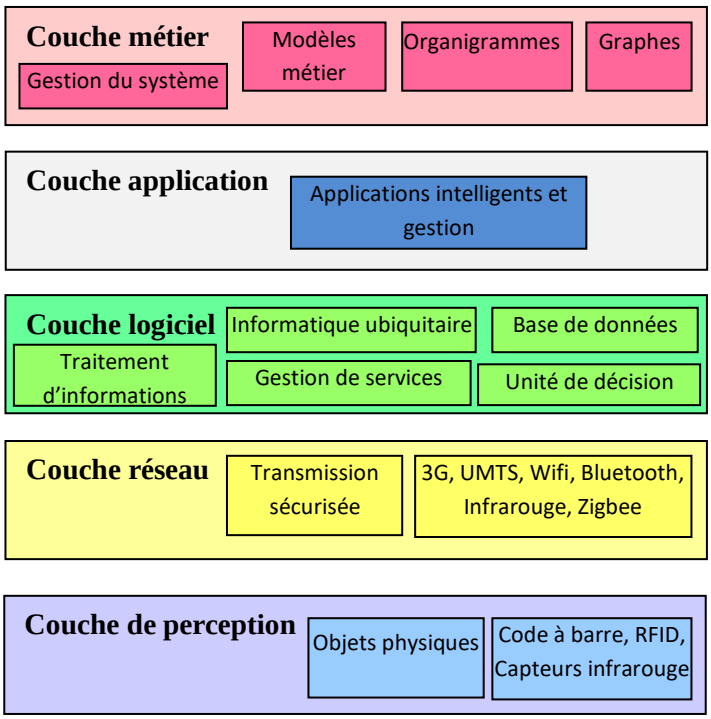

Figure 4 - Architecture de l'Internet des Objets (Khan, Khan et al. 2012)

Selon (Guthikonda, Chitta et al. 2014), les architectures précédemment proposées pour l'IoT ne suffisent pas pour une meilleure utilisation et un déploiement facile surtout pour les architectures des systèmes multi-industries. Pour y remédier, les auteurs proposent l'architecture suivante:

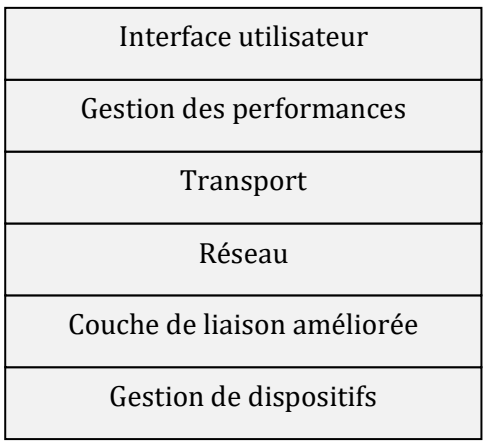

Figure 5 - Architecture de l'IoT (Guthikonda, Chitta et al. 2014)

A travers le module Gestion des performances, cette architecture supporte la reconfiguration selon le critère qualité de service en négociant avec l'utilisateur. Néanmoins, il n'y a pas de garantie sur le fonctionnement du système après la reconfiguration.

Scholze et al (Scholze and Barata 2016) ont examiné une approche de reconfiguration dynamique appliquée aux systèmes IoT. La proposition suit trois étapes pour permettre l'adaptation dynamique du système, notamment la surveillance et l'agrégation des données de capteur brutes, l'évaluation des données surveillées, l'extraction de la situation actuelle à l'aide du modèle de contexte ontologique et enfin la mise à jour du comportement du système pour l'adapter à son nouveau contexte. Cette solution fournit une architecture spécifique (Figure 6) appliquée uniquement au domaine de la fabrication. Il ne fournit aucune garantie sur le maintien de la qualité des services offerts par l'application après adaptation. Une autre lacune est que la reconfiguration est exécutée directement au niveau du système sans abstraction. En outre, le stockage et le traitement des informations contextuelles sont effectués localement.

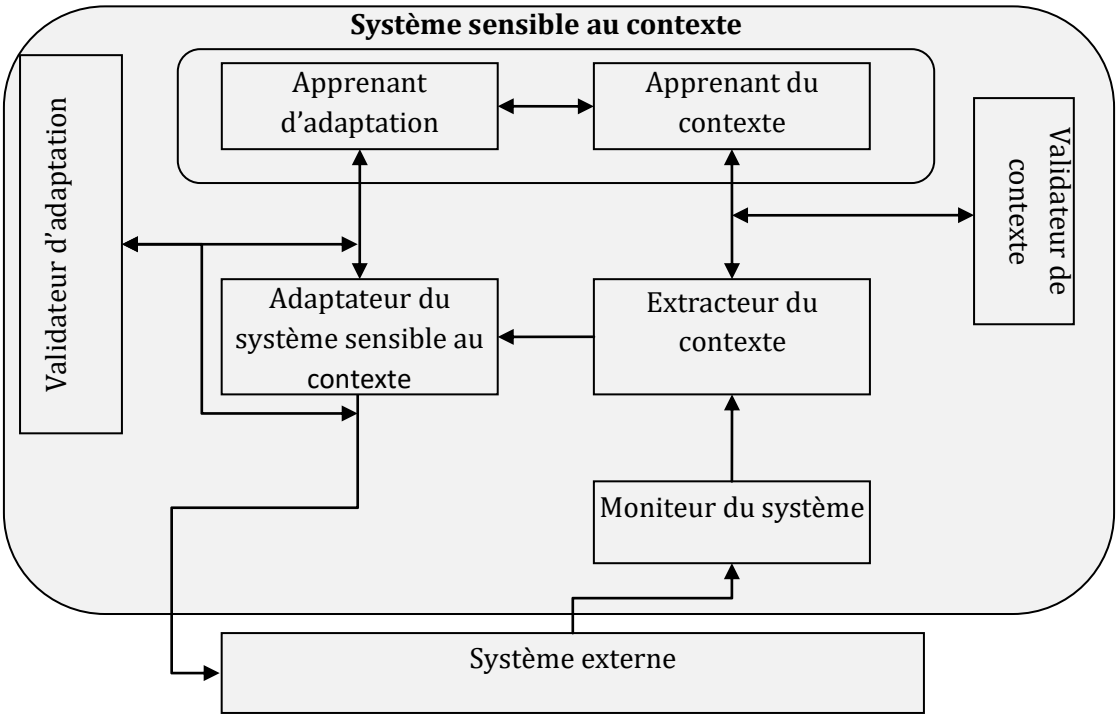

Figure 6 - Architecture d'un système sensible au contexte(Scholze and Barata 2016)

1.7. Applications

Les potentialités offertes par l'IoT permettent le développement d'un grand nombre d'applications, dont seule une très petite partie est actuellement disponible pour notre

société. Nombreux sont les domaines et les environnements dans lesquels de nouvelles applications amélioreraient probablement la qualité de nos vies: à la maison, en voyage, en cas de maladie, au travail, etc. Ces environnements sont maintenant équipés d'objets dotés d'une intelligence primitive, la plupart du temps sans aucune capacité de communication. Donner à ces objets la possibilité de communiquer les uns avec les autres et d'élaborer les informations perçues de l'environnement implique de disposer d'environnements différents dans lesquels une très grande variété d'applications peuvent être déployées. Ceux-ci peuvent être regroupés dans les domaines suivants: infrastructure intelligente, santé, chaînes d'approvisionnement / logistique et applications sociales.

- **Infrastructure intelligente**

L'intégration d'objets intelligents dans l'infrastructure physique peut améliorer la flexibilité, la fiabilité et l'efficacité du fonctionnement de l'infrastructure. Ces avantages peuvent réduire les coûts et les besoins en main-d'œuvre, ainsi que renforcer la sécurité.

Les réseaux intelligents utilisent la technologie IoT pour collecter des données sur la consommation d'énergie et les rendre disponibles en ligne. Les données sont généralement incorporées dans des rapports indiquant des modèles d'utilisation et incluent des recommandations sur la manière de réduire la consommation d'énergie et les coûts(Liu, Li et al. 2011). Les technologies IoT sont également utilisées dans les maisons et les bureaux. Les maisons et les bâtiments sont équipés de capteurs et d'actionneurs qui suivent la consommation des services publics, surveillent et contrôlent les infrastructures des bâtiments, tels que les systèmes d'éclairage, et assurent la surveillance des besoins de sécurité (Darianian and Michael 2008).

À plus grande échelle, les technologies IoT peuvent être utilisées pour rendre les villes plus efficaces. L'objectif des villes intelligentes est de tirer parti de l'IoT pour améliorer la vie des citoyens en améliorant le contrôle du trafic, en surveillant la disponibilité de places de stationnement, en évaluant la qualité de l'air et en fournissant même une notification lorsque les conteneurs à déchets sont pleins (Schaffers, Komninos et al. 2011).

- **Transport et logistique**

Les voitures, les trains, les autobus, les vélos, ainsi que les routes et / ou les rails sont de plus en plus instrumentés avec des capteurs, des actionneurs et une puissance de traitement. Les routes elles-mêmes et les marchandises transportées sont également

équipées d'étiquettes et de capteurs qui envoient des informations importantes aux sites de contrôle de la circulation et aux véhicules de transport afin de mieux orienter le trafic, d'aider à la gestion des dépôts, de fournir aux touristes les informations de transport appropriées et de surveiller le statut des lieux.

- **Santé**

L'IoT est proposé pour améliorer la qualité de la vie en automatisant certaine des tâches de base que l'homme doit effectuer. En ce sens, la surveillance et la prise de décision peuvent être déplacées du côté humain au côté machine. L'une des principales applications de l'IoT dans les soins de santé concerne les scénarios de vie assistée. Les capteurs peuvent être placés sur un équipement de surveillance de la santé utilisé par les patients. Les informations recueillies par ces capteurs sont mises à la disposition des médecins, des membres de la famille et d'autres parties intéressées sur Internet afin d'améliorer le traitement et la réactivité(Dohr, Modre-Opsrian et al. 2010). De plus, les dispositifs IoT peuvent être utilisés pour surveiller les médicaments actuels d'un patient et évaluer le risque de nouveaux médicaments en termes de réactions allergiques et d'interactions indésirables(Jara, Belchi et al. 2010).

- **Chaînes d'approvisionnement/logistique**

Les réseaux RFID et de capteurs ont déjà des rôles bien établis dans les chaînes d'approvisionnement. Les capteurs sont utilisés depuis longtemps dans les chaînes de montage des usines de fabrication, et la RFID est fréquemment utilisée pour suivre les produits tout au long de la chaîne d'approvisionnement contrôlée par une entreprise spécifique. Bien que l'utilisation de ces technologies dans les chaînes d'approvisionnement ne soit pas nouvelle, l'omniprésence promise par l'IoT permettra l'utilisation de ces technologies au-delà des frontières organisationnelles et géographiques.

Plus précisément, l'IoT peut encore améliorer l'efficacité de la logistique et de la chaîne logistique en fournissant des informations plus détaillées et plus actualisées (Flügel and Gehrmann 2008)que celles actuellement disponibles, atténuant ainsi l'effet coup de fouet(Yan and Huang 2009), réduisant la contrefaçon (Yan and Huang 2008)et l'amélioration de la traçabilité des produits (Zhengxia and Laisheng 2010).

- **Applications sociales**

Étant donné que les appareils IoT sont susceptibles d'être connectés à de nombreux objets et même aux personnes elles-mêmes, il convient d'examiner le potentiel des impacts sociétaux et personnels de l'IoT. Les appareils IoT offrent un certain nombre de fonctionnalités susceptibles de favoriser les interactions sociales et les besoins personnels. Une application possible de l'IoT dans un contexte social est l'interaction des périphériques de l'Internet des objets avec les services de réseaux sociaux existants tels que Facebook ou Twitter (Vazquez and Lopez-De-Ipina 2008)

L'utilisation d'appareils IoT pour fournir des informations sur les activités et l'emplacement d'un individu peut faire gagner du temps à l'utilisateur. De plus, les applications qui collectent et intègrent automatiquement ces informations peuvent informer les individus lorsqu'ils se trouvent à proximité d'amis, d'événements sociaux ou d'autres activités susceptibles de les intéresser (Guo, Zhang et al. 2011). En outre, les téléphones mobiles compatibles IoT peuvent se connecter directement à d'autres téléphones portables et transférer des informations de contact lorsque des profils de rencontre ou d'amitié prédéfinis sont compatibles (Guo, Yu et al. 2012).

1.8. Conclusion

Nous avons présenté dans ce chapitre le contexte global dans lequel s'inscrit cette thèse à savoir le domaine des objets connectés. Nous avons visité les stations historiques qui ont marqué l'émergence du domaine. Par la suite, pour mieux comprendre ce que recouvre l'Internet des Objets nous avons synthétisé les définitions notables proposées dans la littérature ainsi que les éléments de base constituants ces systèmes. Pour bien cerner le domaine, nous avons présenté les défis autour de l'IoT ainsi que leurs caractéristiques, cela nous permettra d'établir l'Internet des Objets comme une bonne pratique. Enfin, quelques domaines d'applications de l'IoT sont dénombrés.

Dans le chapitre suivant, une étude sur le contexte et la sensibilité au contexte sera envisagée. Cette étude est argumentée par le fait que le contexte est le responsable principal derrière l'évolution des systèmes à objets connectés. L'étude sera enrichie avec un état d'art sur les travaux existants dans le domaine.

CHAPITRE 2

Sensibilité au Contexte

2.1. Introduction

Les terminaux mobiles envahissent notre vie quotidienne, ils sont de plus en plus accessibles et présents. Ces objets communicants et mobiles nous suivent partout et introduisent une forte variabilité dans les environnements d'exécution. Cela a mené Mark Weiser en 1991 (Weiser 1991) à poser la notion d'informatique ubiquitaire. Dans son idée, les outils informatiques sont embarqués dans des objets de la vie quotidienne qui sont utilisés aussi bien au travail qu'à la maison. Ces objets interagissent et coopèrent dans un environnement commun partageant un contexte.

Lorsque des humains communiquent, ils peuvent utiliser des informations de situation implicites, ou un contexte, pour augmenter la largeur de bande de conversation. Malheureusement, cette capacité à transmettre des idées ne se transmet pas bien aux humains qui interagissent avec les ordinateurs. Par conséquent, les ordinateurs ne sont pas actuellement en mesure de tirer pleinement parti du contexte du dialogue homme-machine. En améliorant l'accès de l'ordinateur au contexte, nous augmentons la richesse de la communication en interaction homme-machine et permettons de produire des services informatiques plus utiles.

Dans ce chapitre, nous présentons les connaissances relatives à la sensibilité au contexte. Nous illustrons quelques exemples de systèmes sensibles au contexte. Par la suite, la structure du gestionnaire de contexte ainsi que le processus qu'il implémente sont décrits.

2.2. Concepts de base

Nous étudions dans cette section la sensibilité au contexte, les notions de contexte existantes et les usages de l'information contextuelle.

2.2.1. Le contexte

Dans le travail qui a introduit pour la première fois le terme «sensible au contexte», Schilit et Theimer (Schilit, Adams et al. 1994) se réfèrent au contexte en tant qu'emplacement, aux identités des personnes et des objets proches et aux modifications apportées à ces objets.

Dans une définition similaire, Brown et al (Brown, Bovey et al. 1997) définissent le contexte comme l'emplacement, l'identité des personnes entourant l'utilisateur, l'heure du jour, la saison, la température, etc.

Dey (Dey 1998) désigne le contexte par l'état émotionnel de l'utilisateur, son centre d'attention, son emplacement et son orientation, la date et l'heure, les objets et les personnes présents dans l'environnement de l'utilisateur.

Tandis que Schilit et al (Schilit, Adams et al. 1994) affirment que les aspects importants du contexte sont les suivants: où vous vous trouvez, avec qui vous êtes et quelles ressources sont à proximité. Ils définissent le contexte comme étant l'environnement d'exécution en constante évolution. Ils incluent les éléments suivants de l'environnement: processeurs disponibles dans l'environnement informatique, périphériques accessibles pour la saisie et l'affichage par l'utilisateur, capacité du réseau, connectivité et coûts de traitement, emplacement de l'environnement utilisateur, regroupement des personnes à proximité et situation sociale, éclairage de l'environnement physique et niveau de bruit.

Dey (Dey 2001) définit par contexte: *« toute information pouvant être utilisée pour caractériser la situation d'une entité (personne, objet physique ou informatique). Et plus généralement tout élément pouvant influencer le comportement d'une application.»*

Les applications contextuelles examinent les entités, qui sont, où, quand et ce qu'elles font (c'est-à-dire ce que fait l'utilisateur) et utilisent ces informations pour déterminer la raison pour laquelle la situation se produit.

2.2.2. La sensibilité au contexte

L'informatique mobile est caractérisée par un changement de contexte permanent, il est alors important que le système informatique réagisse de manière automatique.

Schilit et Theimer (Schilit, Adams et al. 1994) ont introduit pour la première fois en 1994 l'informatique contextuelle en tant que logiciel qui s'adapte en fonction de son lieu d'utilisation, de la collection de personnes et d'objets proches, ainsi que des modifications apportées à ces objets au fil du temps. Cependant, il est généralement admis que la première étude de recherche sur l'informatique contextuelle était le travail d'Olivetti « Active Badge » (Want, Hopper et al. 1992) en 1992.

Selon (Dey 2001), Un système est sensible au contexte s'il utilise le contexte pour offrir des informations ou des services pertinents pour l'utilisateur. La sensibilité au contexte est devenue liée à d'autres termes: système adaptatif, réactif, situé, et orienté environnement.

Plusieurs fonctions sont nécessaires pour qu'un système sensible au contexte puisse fonctionner: la collecte des informations de contexte, la détection des changements de situations de contexte et la réalisation des adaptations.

2.3. Exemples de systèmes sensibles au contexte

Nous citons dans ce qui suit les exemples qu'a donné (Taconet 2009) sur des scénarii des systèmes sensibles au contexte:

- Choix d'un algorithme de réconciliation en fonction des temps de latence
- Démarrage d'une synchronisation lorsque le réseau est rétabli
- Avertissement lorsqu'une personne est à proximité
- Affichage d'une position géographique
- Ajustement de la vitesse en fonction de la distance du véhicule qui est devant
- Modification de la température de la pièce en fonction des personnes présentes
- Arrêt de la radio lors de la réception d'un appel téléphonique.

L'auteur énumère ainsi dans le même support les premières applications sensibles au contexte:

- **L'assistant commercial personnel (Asthana, Crauatts et al. 1994)**

Ce système utilise la localisation de l'utilisateur pour le guider dans un magasin. Ils ont développé cette idée en un service basé sur deux produits: un dispositif de communication

sans fil très volumineux, le PSA (Personal Shopping Assistant), que le client possède, et un serveur centralisé situé dans le centre commercial auquel le client communique à l'aide du PSA. Le serveur centralisé gère la base de données client, la base de données du magasin et fournit des réponses audiovisuelles aux demandes de renseignements de dizaines à des centaines de clients en temps réel sur un réseau sans fil de petite taille.

- **Cyber Guide (Abowd, Atkeson et al. 1997)**

Il offre des informations touristiques sur une carte interactive: sites à visiter selon la localisation de l'utilisateur et l'historique des visites. Des prototypes d'un guide touristique mobile sensible au contexte sont construits. La connaissance de l'emplacement actuel de l'utilisateur, ainsi que l'historique des emplacements précédents, sont utilisés pour fournir davantage de types de services du guide touristique.

- **Assistant de conférences (Dey, Salber et al. 1999)**

C'est un prototype d'application mobile sensible au contexte qui aide les participants à la conférence en suggérant par exemples des présentations à laquelle assister; affichage automatique des présentations en cours. L'application utilise une grande variété de contextes et améliore les interactions des utilisateurs avec l'environnement et les autres utilisateurs.

- **Téléphone GSM adaptatif et Assistant Personnel (Schmidt, Aidoo et al. 1999)**

Assiste les utilisateurs de terminaux mobiles en changeant la taille d'écriture en fonction de l'activité de l'utilisateur; sélection automatique des profils de téléphones portables en fonction du contexte (sonner, vibrer ou rester silencieux). Deux types de capteurs sont distingués: les capteurs physiques et logiques, qui fournissent des informations à partir des paramètres d'environnement et des informations sur l'hôte. Un assistant numérique personnel (PDA) et un téléphone portable ont été utilisés avec le prototype pour démontrer la connaissance de la situation. Dans le PDA, la taille de la police et le rétro éclairage ont été modifiés en fonction des contextes présentés tandis que dans le téléphone mobile, le profil utilisateur actif a été modifié.

- **Transfert d'appel (Want, Hopper et al. 1992)**

Grâce au système de localisation « Active Badge », le réceptionniste utilise la localisation de l'utilisateur pour faire suivre les appels téléphoniques vers le téléphone le plus proche de l'utilisateur. Le badge actif a été développé entre 1989 et 1992. Son but est de fournir un

moyen de localisation direct du personnel dans un immeuble pour acheminer automatiquement les appels téléphoniques. Pour ce faire, chaque personne porte un badge qui transmet périodiquement des signaux vers un système centralisé de localisation. Le plus grand système de badges actifs actuellement utilisé se trouve au laboratoire d'informatique de l'université Cambridge, comptant plus de 200 badges et 300 capteurs utilisés quotidiennement.

2.4. Structure d'un système sensible au contexte

De nombreux framework et systèmes sensibles au contexte ont été développés au cours des dernières années. La plupart d'entre eux diffèrent par la plage fonctionnelle, l'emplacement et le nom des couches, l'utilisation d'agents facultatifs ou d'autres préoccupations architecturales. Outre ces adaptations et modifications, une architecture commune dans les applications modernes sensibles au contexte est identifiable lors de l'analyse des différentes approches de conception. Une séparation entre la détection et l'utilisation du contexte est nécessaire pour améliorer l'extensibilité et la réutilisabilité des systèmes. Dans ce contexte, (Baldauf, Dustdar et al. 2007) proposent l'architecture multicouches présentée dans la Figure 7.

La première couche consiste en une collection de différents capteurs. Il convient de noter que le mot "capteur" ne désigne pas seulement le matériel de détection, mais également toutes les sources de données pouvant fournir des informations contextuelles utilisables.

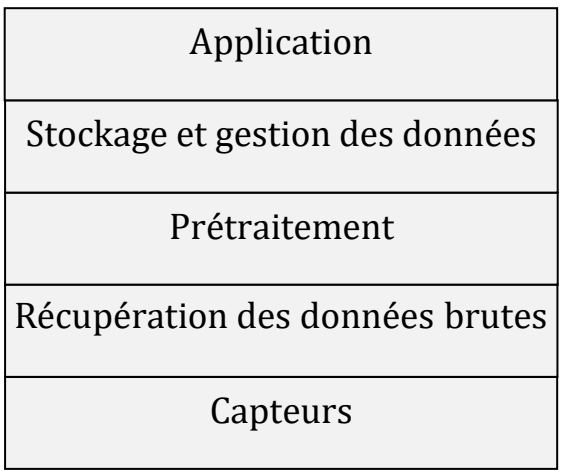

Figure 7 - Cadre conceptuel en couches pour les systèmes sensibles au contexte (Baldauf, Dustdar et al. 2007)

Nous expliquerons dans ce qui suit les spécifications et les fonctionnalités de chaque couche du framework tels que proposé par (Baldauf, Dustdar et al. 2007).

2.4.1. Capteurs

Concernant la manière dont les données sont capturées; les capteurs peuvent être classés en trois groupes:

1. Capteurs physiques: Ce sont les capteurs les plus fréquemment utilisés. De nos jours, de nombreux capteurs matériels sont capables de capturer presque toutes les données physiques.
2. Capteurs virtuels: Les capteurs virtuels génèrent des données de contexte à partir d'applications logicielles ou de services. Par exemple, il est possible de déterminer l'emplacement d'un employé en utilisant non seulement des systèmes de suivi (capteurs physiques), mais également un capteur virtuel, par exemple, en parcourant un calendrier électronique, un système de réservation de voyages, des courriers, etc. pour obtenir des informations de localisation. D'autres attributs de contexte pouvant être détectés par des capteurs virtuels incluent, par exemple, l'activité de l'utilisateur en vérifiant le mouvement de la souris et la saisie au clavier.
3. Capteurs logiques: Ces capteurs utilisent plusieurs sources d'informations et combinent des capteurs physiques et virtuels avec des informations supplémentaires provenant de bases de données ou de diverses autres sources afin de résoudre des tâches plus complexes. Par exemple, un capteur logique peut être construit pour détecter la position actuelle d'un employé en analysant les connexions aux ordinateurs de bureau et en mappant une base de données des périphériques aux informations de localisation.

2.4.2. Récupération des données brutes

La deuxième couche est responsable de l'extraction des données de contexte brutes. Elle utilise des pilotes appropriés pour les capteurs physiques et des API pour les capteurs virtuels et logiques. La fonctionnalité de requête est souvent implémentée dans des composants logiciels réutilisables qui rendent transparents les détails de bas niveau de l'accès au matériel en fournissant des méthodes plus abstraites telles que getPosition(). En utilisant des interfaces pour les composants responsables de types de contexte identiques,

ces composants deviennent échangeables. Par conséquent, il est possible de remplacer un système RFID par un système GPS sans aucune modification majeure des couches actuelles et supérieures.

2.4.3. Prétraitement des données contextuelles

Elle peut fournir des informations utiles si les données brutes sont trop grossières. La couche de prétraitement est responsable du raisonnement et de l'interprétation des informations contextuelles. Les capteurs interrogés dans la couche sous-jacente renvoient le plus souvent des données techniques que les concepteurs d'applications ne peuvent utiliser. Par conséquent, cette couche élève les résultats de la couche deux à un niveau d'abstraction plus élevé. Les transformations comprennent des opérations d'extraction et de quantification. Par exemple, la position GPS exacte d'une personne peut ne pas être utile pour une application, mais le nom de la pièce où elle se trouve peut être.

2.4.4. Stockage et gestion des données contextuelles

La quatrième couche, organise les données collectées et les propose via une interface publique au client. Les clients peuvent avoir accès de deux manières différentes, synchrone et asynchrone. De manière synchrone, le client interroge le serveur pour connaître les modifications effectuées via des appels de méthode distants. Par conséquent, il envoie un message demandant une sorte de données proposée et se met en pause jusqu'à ce qu'il reçoive la réponse du serveur. Le mode asynchrone fonctionne via des abonnements. Chaque client s'abonne à des événements spécifiques qui l'intéressent. Lors de l'un de ces événements, le client est simplement averti ou une méthode du client est directement impliquée par un rappel. Dans la majorité des cas, l'approche asynchrone est plus appropriée en raison de changements rapides du contexte sous-jacent.

2.4.5. Application

Le client est réalisé dans cette cinquième couche. La réaction réelle sur différents événements et instances de contexte est mise en œuvre ici. Parfois, la récupération d'informations et la gestion du contexte et du raisonnement spécifiques à une application sont encapsulés sous la forme d'agents, qui communiquent avec le serveur de contexte et

agissent en tant que couche supplémentaire entre le prétraitement et la couche application. Un exemple de contexte logique coté client est l'affichage sur les appareils mobiles: puisqu'un capteur de lumière détecte un mauvais éclairage, le texte peut être affiché avec un contraste de couleur plus élevé.

2.5. Utilisation du contexte

Pour bien exploiter les données contextuelles dans une situation, le cycle de vie du contexte montre comment ces données se déplacent d'une phase à l'autre dans le système. (Bernardos, Tarrio et al. 2008) ont identifié trois phases dans un système de gestion de contexte typique: acquisition du contexte, traitement de l'information, raisonnement et décision.

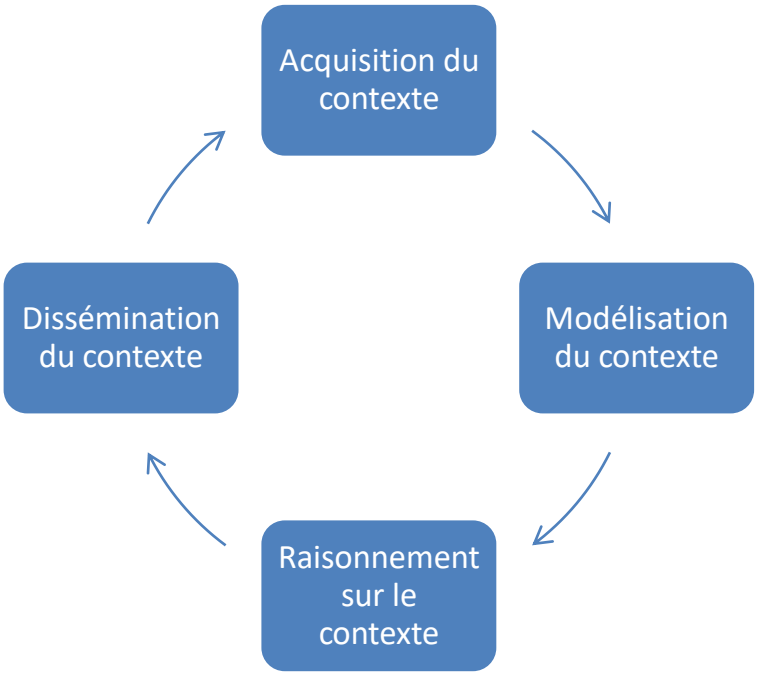

Figure 8 - Cycle de vie du contexte (Perera, Zaslavsky et al. 2014)

Ce cycle de vie de contexte tels que décrit par (Perera, Zaslavsky et al. 2014) comprend quatre phases. Premièrement, le contexte doit être acquis de différentes sources. Les sources peuvent être des capteurs physiques, virtuels ou logiques (acquisition de contexte). Deuxièmement, les données collectées doivent être modélisées et représentées de manière significative (modélisation du contexte). Troisièmement, les données modélisées doivent être traitées pour dériver des informations de contexte de haut niveau à partir des données de capteur brutes de bas niveau (raisonnement de contexte). Enfin, les contextes de haut

niveau et de bas niveau doivent être distribués aux consommateurs intéressés par le contexte (diffusion du contexte). Nous expliquerons dans ce qui suit le processus implémenté dans chaque phase de la Figure 8.

2.5.1. Acquisition du contexte

Au cours de cette phase, toutes les données du contexte utilisateur (telles que le bruit ambiant, la température actuelle et les préférences de l'utilisateur) et le contexte système (telles que les ressources de calcul disponibles et les périphériques partagés) sont collectées et souvent traduites en événements de contexte de haut niveau.

Selon (Perera, Zaslavsky et al. 2014), Il existe trois types de contexte selon la méthode de son acquisition: contexte détecté, dérivé ou fourni manuellement.

- Contexte détecté: Les données sont détectées par des capteurs, y compris les données détectées stockées dans des bases de données (par exemple, récupérer la température d'un capteur, extraire les détails d'un rendez-vous d'un calendrier).
- Contexte dérivé: Les informations sont générées en effectuant des opérations de calcul sur les données de capteurs. Ces opérations peuvent être aussi simples que des appels de service Web ou aussi complexes que des fonctions mathématiques exécutées sur des données détectées (par exemple, calculer la distance entre deux capteurs à l'aide de coordonnées GPS). Les données nécessaires devraient être disponibles pour appliquer toute technique de raisonnement numérique ou logique.
- Contexte fourni manuellement: les utilisateurs fournissent manuellement des informations de contexte via des options de paramètres prédéfinies telles que les préférences (par exemple, comprenez qu'ils n'aiment pas recevoir de notifications d'événement entre 22h00 et 6h00). Cette méthode peut être utilisée pour récupérer tout type d'information.

Selon (Ferry, Lavirotte et al. 2008), le contexte peut être classé en quatre catégories permettant de définir les conditions pouvant déclencher une adaptation:

- Contexte utilisateur: son emplacement, son humour, le travail ou l'action qu'il effectue.

- Contexte temporel: moments, dates, historique…etc.

-Contexte de la machine: environnement d'exécution, ressources disponibles, surveillance…etc.

- Contexte environnemental: correspond à ce qui entoure le système: luminosité, bruit, température…etc.

Tandis que (Henricksen and Indulska 2004) ont trouvé des modèles de contexte plus riches intégrant des informations détectées, statiques, fournies par l'utilisateur (profilées) et dérivées, étant les plus utiles.

Les informations de contexte sont généralement acquises à l'aide de capteurs. Cet ensemble d'informations sera évalué dans le but de décider de déclencher ou ignorer une opération de reconfiguration. Par conséquent, le déclenchement d'une opération de reconfiguration est principalement lié à des modifications du contexte du système, c'est ce que l'on appelle la connaissance du contexte (Schilit, Adams et al. 1994).

2.5.2. Modélisation du contexte

Un modèle de contexte est nécessaire pour définir et stocker les données de contexte dans un formulaire pouvant être traité par une machine. Développer des ontologies de contexte flexibles et utilisables qui couvrent un large éventail de contextes possibles est une tâche ardue. Les approches de modélisation de contexte les plus pertinentes sont les suivantes (Perera, Zaslavsky et al. 2014):

1) Modèle clé-valeur: Cette approche modélise les informations de contexte sous forme de paires clé-valeur dans différents formats tels que des fichiers textes et des fichiers binaires. C'est la forme la plus simple de représentation du contexte parmi toutes les autres techniques. Ils sont faciles à gérer lorsqu'ils contiennent moins de données. Cependant, la modélisation clé-valeur n'est ni évolutive ni adaptée au stockage de structures de données complexes. En outre, les structures hiérarchiques ou les relations ne peuvent pas être modélisées à l'aide de paires clé-valeur. Par conséquent, le manque de capacité de structuration des données rend difficile l'extraction efficace des informations modélisées.

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

2) Modèle de balisage: Il modélise les données à l'aide de balises. Par conséquent, le contexte est stocké dans les balises. Cette technique est une amélioration par rapport à la technique de modélisation clé-valeur. L'utilisation des balises de marquage présente l'avantage de permettre une récupération efficace des données. En outre, la validation est

prise en charge via les définitions de schéma. Des outils de validation sophistiqués sont disponibles pour les techniques de balisage populaires telles que XML. Les schémas de balisage tels que XML sont largement utilisés dans presque tous les domaines d'application pour stocker temporairement des données, transférer des données entre applications et transférer des données entre des composants d'application. En revanche, les langages de marquage ne fournissent pas de fonctionnalités expressives avancées permettant le raisonnement.

3) Modèle graphique: il modélise le contexte avec des relations. En termes de richesse expressive, la modélisation graphique est meilleure que la modélisation par balisage et clé-valeur car elle permet de capturer des relations dans le modèle de contexte. La représentation réelle de bas niveau de la modélisation graphique pourrait être variée. Par exemple, cela pourrait être une base de données SQL, base de données noSQL, XML, etc. Beaucoup d'autres extensions ont également été proposées et mis en œuvre en utilisant cette technique. Les bases de données peuvent contenir d'énormes quantités de données et fournir des opérations de récupération de données simples, qui peuvent être effectuées rapidement. En revanche, il existe des limitations sur les mécanismes de récupération de données tels que SQL. En outre, des exigences sophistiquées de récupération de contexte peut exiger l'emploi de requêtes SQL très complexes. Les requêtes peuvent être difficiles à créer, à utiliser et à gérer même avec les outils sophistiqués qui existent aujourd'hui. L'ajout d'informations contextuelles et la modification de la structure des données sont également difficiles à réaliser ultérieurement.

4) Modèle basé sur les objets: Les concepts orientés objets sont utilisés pour modéliser les données à l'aide d'hiérarchies et de relations de classes. Le paradigme orienté objet favorise l'encapsulation et la réutilisation. Comme la plupart des langages de programmation de haut niveau prennent en charge les concepts orientés objets, la modélisation peut être facilement intégrée à des systèmes sensibles au contexte. Par conséquent, la modélisation basée sur les objets convient pour être utilisée en tant que mécanisme de modélisation, de manipulation et de stockage du contexte d'exécution interne, non partagé, basé sur du code. Cependant, il ne fournit pas de capacités de raisonnement intégrées. La validation des conceptions orientées objet est également difficile en raison de l'absence de normes et de spécifications.

5) Modèle logique: les faits, les expressions et les règles sont utilisés pour représenter des informations sur le contexte. Les règles sont également utilisées par d'autres techniques de modélisation, telles que les ontologies. Les règles sont principalement utilisées pour exprimer des stratégies, des contraintes et des préférences. Il offre une richesse beaucoup plus expressive par rapport aux autres modèles discutés précédemment.

Par conséquent, le raisonnement est possible jusqu'à un certain niveau. Les structures et les langages spécifiques pouvant être utilisés pour modéliser le contexte à l'aide de règles sont variés. Cependant, le manque de standardisation réduit les possibilités de réutilisation et d'applicabilité. De plus, des techniques graphiques hautement sophistiquées et interactives peuvent être utilisées pour développer des représentations basées sur la logique ou sur des règles.

Ainsi, même les utilisateurs non techniques peuvent ajouter des règles et une logique aux systèmes pendant l'exécution. La modélisation logique permet d'extraire de nouvelles informations de contexte de haut niveau en utilisant un contexte de bas niveau. Par conséquent, il est capable d'améliorer d'autres techniques de modélisation de contexte en faisant office de complément.

6) Modèle basé sur des ontologies: le contexte est organisé en ontologies à l'aide de technologies sémantiques. Un certain nombre de normes différentes (RDF, RDFS, OWL) et des capacités de raisonnement peuvent être utilisées en fonction des besoins.

Une large gamme d'outils de développement et de moteurs de raisonnement est également disponible. Toutefois, la récupération de contexte peut nécessiter beaucoup de temps de calcul lorsque la quantité de données augmente. Selon de nombreuses enquêtes, en informatique contextuelle et en gestion de données de capteurs, les ontologies constituent le mécanisme privilégié de gestion et de modélisation du contexte, malgré ses faiblesses.

2.5.3. Raisonnement sur le contexte

(Bikakis, Patkos et al. 2007) définissent le raisonnement contextuel comme une méthode permettant de déduire de nouvelles connaissances et de mieux comprendre en fonction du contexte disponible. L'exigence de raisonnement est apparue en raison de deux caractéristiques du contexte brut: les imperfections (données inconnues, ambiguës,

imprécises ou erronées) et les incertitudes. (Nurmi and Floréen 2004) décrivent le raisonnement contextuel par les trois étapes suivantes:

1. Le prétraitement des données de contexte vise à faciliter le traitement ultérieur en sélectionnant les attributs de contexte pertinents, en traitant les attributs manquants et en nettoyant les données, par exemple en supprimant les valeurs aberrantes.

2. La fusion de données fait l'objet de nombreuses recherches dans le domaine des réseaux de capteurs, dont la principale motivation est de réduire les coûts de communication en intégrant des sources de données similaires.

3. L'inférence de contexte est une tâche très difficile. Tout d'abord, il est nécessaire de reconnaître de nouveaux contextes. Deuxièmement, il doit y avoir un mécanisme sous-jacent permettant de mapper le contexte de niveau inférieur à un contexte de niveau supérieur. Nous pouvons utiliser des ontologies dans le processus et utiliser un raisonnement logique pour la phase de mappage. L'autre approche consiste à utiliser le raisonnement probabiliste.

2.5.4. Dissémination du contexte

La distribution du contexte est une tâche assez simple. Il fournit des méthodes pour fournir un contexte aux consommateurs. Du point de vue du consommateur, cette tâche peut être appelée acquisition de contexte. Il existe deux méthodes couramment utilisées dans la distribution du contexte:

Requête: Le consommateur de contexte effectue une requête afin que le système de gestion de contexte puisse l'utiliser pour produire des résultats.

Abonnement (également appelé publish/subscribe): Le consommateur de contexte peut être autorisé à s'abonner à un système de gestion de contexte en décrivant les exigences. Le système renverra ensuite les résultats périodiquement ou lorsqu'un événement survient (dépassement du seuil). En d'autres termes, les consommateurs peuvent s'abonner à un capteur spécifique ou à un événement. Toutefois, dans les implémentations soulignées, les requêtes peuvent également être utilisées pour définir des abonnements. En outre, cette méthode est généralement utilisée dans le traitement en temps réel.

2.6. Etat de l'art sur les systèmes sensibles au contexte

L'informatique sensible au contexte constitue l'un des développements technologiques les plus importants de la dernière décennie. Il a le potentiel d'affecter profondément notre style de vie. Le domaine reste encore frais, ouvert et vaste puisqu'il revient aux débuts des années 2000. La recherche couvre plusieurs aspects envers l'investigation et la stabilisation de ces systèmes dans le but d'avoir des objets intelligents pouvant être déployés dans n'importe quel environnement avec un réseau interopérable leur permettant de se fondre avec les autres objets qui les entourent.

Plusieurs études sur les systèmes sensibles au contexte ont été présentées (Baldauf, Dustdar et al. 2007; Gubbi, Buyya et al. 2013; Guthikonda, Chitta et al. 2014; Perera, Zaslavsky et al. 2014). Les auteurs introduisent les systèmes IoT sensibles au contexte existants en s'appuyant sur les middlewares et les framework, ce qui facilite le développement des applications sensibles au contexte. Nous citons dans ce qui suit les travaux les plus notés dans le domaine.

Context toolkit (Dey 2001): Vise à faciliter le développement et le déploiement d'applications contextuelles. Il s'agit de l'un des premiers efforts visant à fournir un framework pour le développement d'applications contextuelles. Context Toolkit contient une combinaison de fonctionnalités et d'abstractions permettant de prendre en charge les développeurs d'applications contextuelles. Comme indiquée dans la Figure 9, il présente trois abstractions principales: le widget de contexte (pour récupérer des données à partir de capteurs), un interpréteur de contexte (pour raisonner des données de capteur à l'aide de techniques de raisonnement différentes) et un agrégateur de contexte. La recherche autour de Context Toolkit est toujours active et un certain nombre d'extensions ont été développées pour améliorer ses capacités tenant compte du contexte.

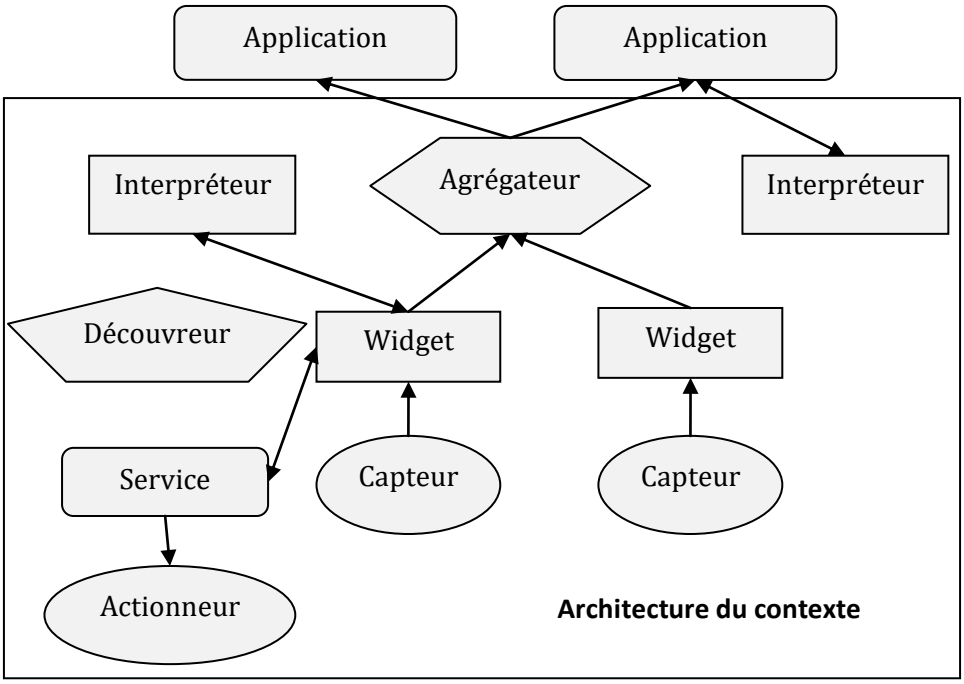

Figure 9 - Architecture du Context Toolkit (Dey 2001)

Aura (Garlan, Siewiorek et al. 2002): Est un système orienté tâches basé sur une architecture distribuée qui se concentre sur différents dispositifs informatiques utilisés par les utilisateurs humains tous les jours. L'objectif est d'exécuter un ensemble d'applications dans tous les périphériques afin de gérer en douceur les tâches de l'utilisateur de manière contextuelle, sur tous les périphériques. Aura répond à deux défis majeurs; Premièrement, elle permet à un utilisateur de préserver la continuité de son travail lorsqu'il se déplace entre différents environnements. Deuxièmement, il est capable de s'adapter au calcul en cours d'un environnement particulier en présence d'une variabilité dynamique des ressources. Aura comprend quatre composants principaux illustrés dans la Figure 10: observateur de contexte (collecte le contexte et l'envoie aux gestionnaires de tâches et d'environnement), gestionnaire de tâches, gestionnaire d'environnement (gère les fournisseurs contextuels et les services associés) et les fournisseurs contextuels (fournit des informations contextuelles). Les schémas de balisage basés sur XML sont utilisés pour décrire les services.

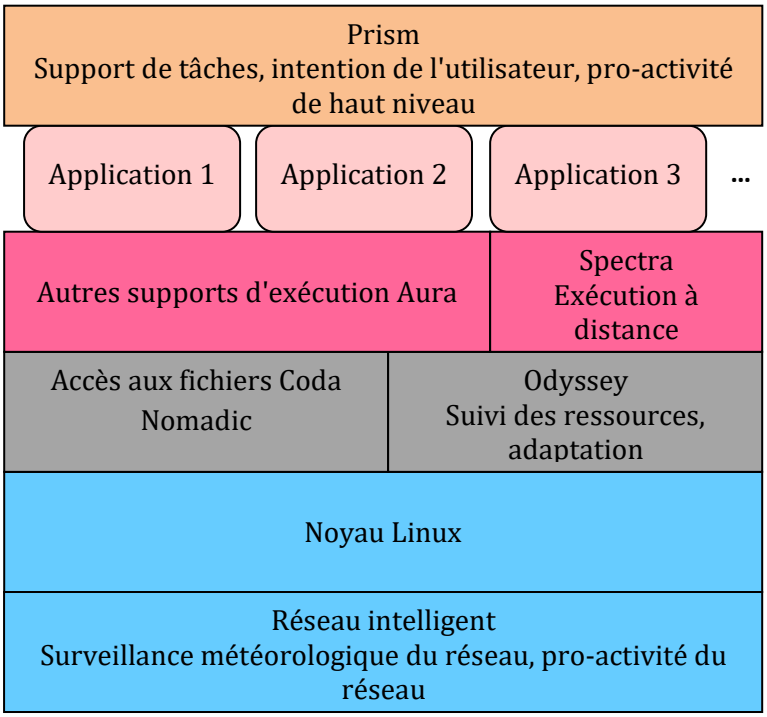

Figure 10 - Architecture Aura (Garlan, Siewiorek et al. 2002)

e-SENSE (Gluhak and Schott 2007): Permet à l'intelligence ambiante d'utiliser des réseaux multi capteurs sans fil (Wireless Sensor Network) pour rendre les informations riches en contexte accessibles aux applications et aux services. e-SENSE combine des réseaux de capteurs corporels (Body Sensor Network), des réseaux de capteurs d'objets (Object Sensor Network) et des réseaux de capteurs d'environnement (Environment Sensor Network) pour capturer le contexte dans le paradigme IoT (Figure 11). Les fonctionnalités requises par les solutions de middleware IoT sensibles au contexte sont identifiées comme étant la capture de données de capteurs, le pré-filtrage de données, l'intégration de sources de données d'abstraction de contexte, l'extraction de contexte, le moteur de règles et l'adaptation.

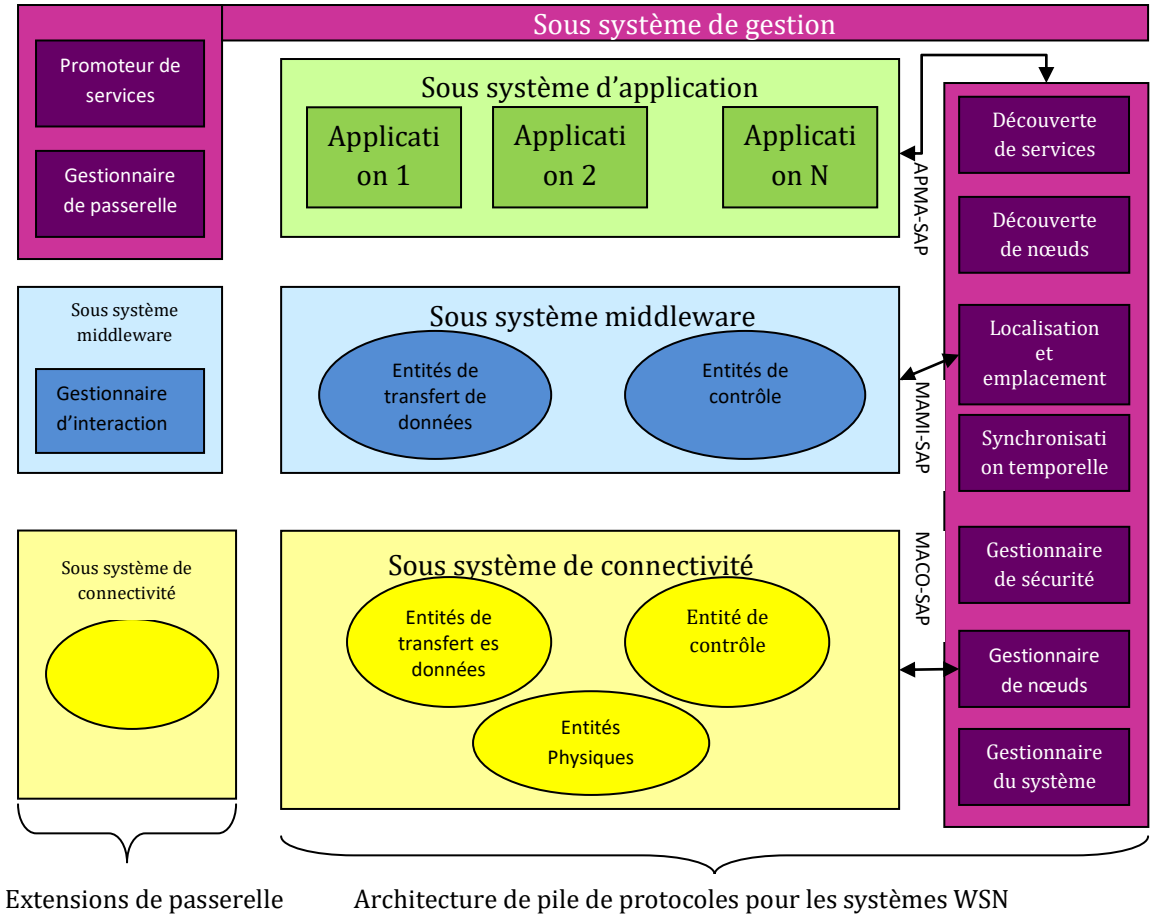

Figure 11 - Architecture eSense (Gluhak and Schott 2007)

ACoMS (Autonomic Context Management System) (Hu, Peizhao, Indulska et al. 2008): Peut configurer et reconfigurer de manière dynamique ses fonctionnalités d'acquisition et de prétraitement d'informations de contexte afin d'assurer le provisionnement à tolérance de panne des informations de contexte. L'architecture ACoMS (Figure 12) comprend des magasins de gestionnaires de souscription de contexte d'application (gère les demandes d'informations de contexte des applications à l'aide d'un mécanisme d'abonnement), gestionnaire des sources de contexte (effectue des actions telles que la communication de bas niveau avec des sources de contexte, la découverte, l'enregistrement et la configuration de sources de contexte) et le gestionnaire de reconfiguration (effectue des tâches de surveillance telles que le mappage de sources de contexte sur des informations de contexte).

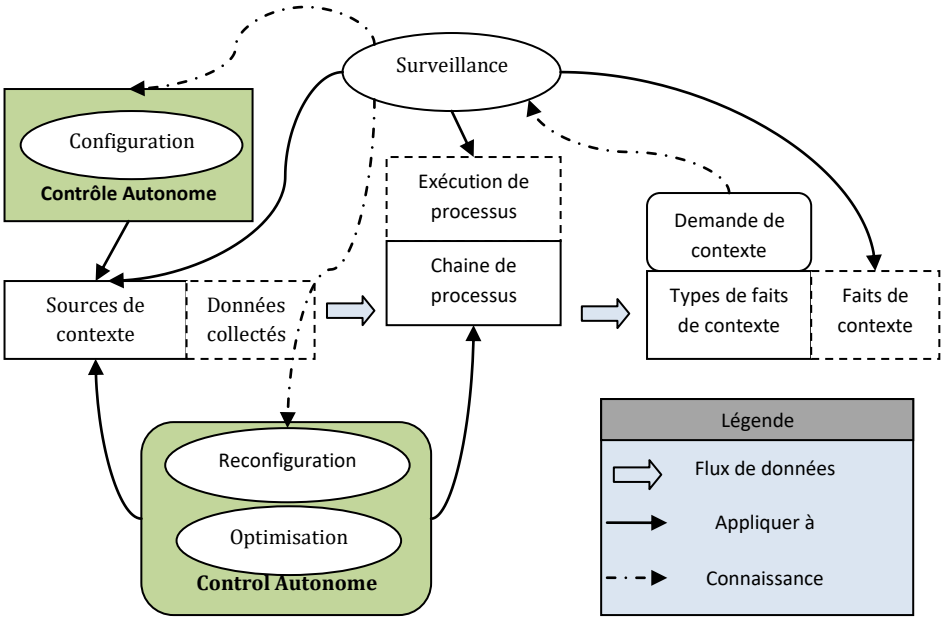

Figure 12 - Architecture ACoMS (Hu, Peizhao, Indulska et al. 2008)

Feel@Home(Guo, Sun et al. 2010) est un framework de gestion de contexte qui prend en charge les interactions entre différents domaines. L'approche proposée est démontrée en utilisant trois domaines: maison intelligente, bureau intelligent et mobile. Les informations de contexte sont stockées en utilisant OWL. Feel@Home prend en charge deux interactions différentes: intra-domaine et inter-domaine. L'interaction entre domaines est essentielle dans le paradigme de l'IoT. De plus, c'est l'une des différences majeures entre les réseaux de capteurs et l'IoT. Les réseaux de capteurs traitent généralement un seul domaine. Cependant, l'IoT exige la capacité de gérer plusieurs domaines. En outre, les framework de gestion de contexte ne doivent pas être limités à un nombre spécifique de domaines. Feel@Home se compose de trois parties (Figure 13): les requêtes des utilisateurs, le serveur d'administration globale (GAS) et le gestionnaire de contexte de domaine (DCM). Les requêtes des utilisateurs sont d'abord reçues par le GAS. Il décide de ce que le domaine pertinent doit être contacté pour répondre à la requête de l'utilisateur. Ensuite, GAS redirige la requête de l'utilisateur vers les gestionnaires de contexte de domaine appropriés. Deux composants résident dans GAS, le gestionnaire d'entrée de contexte (CEM) et le moteur d'entrée de contexte (CEE), qui effectue la tâche ci-dessus. DCM se compose de composants typiques de la gestion du contexte tels que le wrapper de contexte (rassemble le contexte à partir de capteurs et d'autres sources), l'agrégateur de contexte (raisonnement de contexte de déclencheurs), le raisonnement de contexte, la base de connaissances (contexte de

magasins) et plusieurs autres mécanismes d'abonnement. Les réponses à la requête de l'utilisateur seront renvoyées en utilisant le même chemin que lors de la réception.

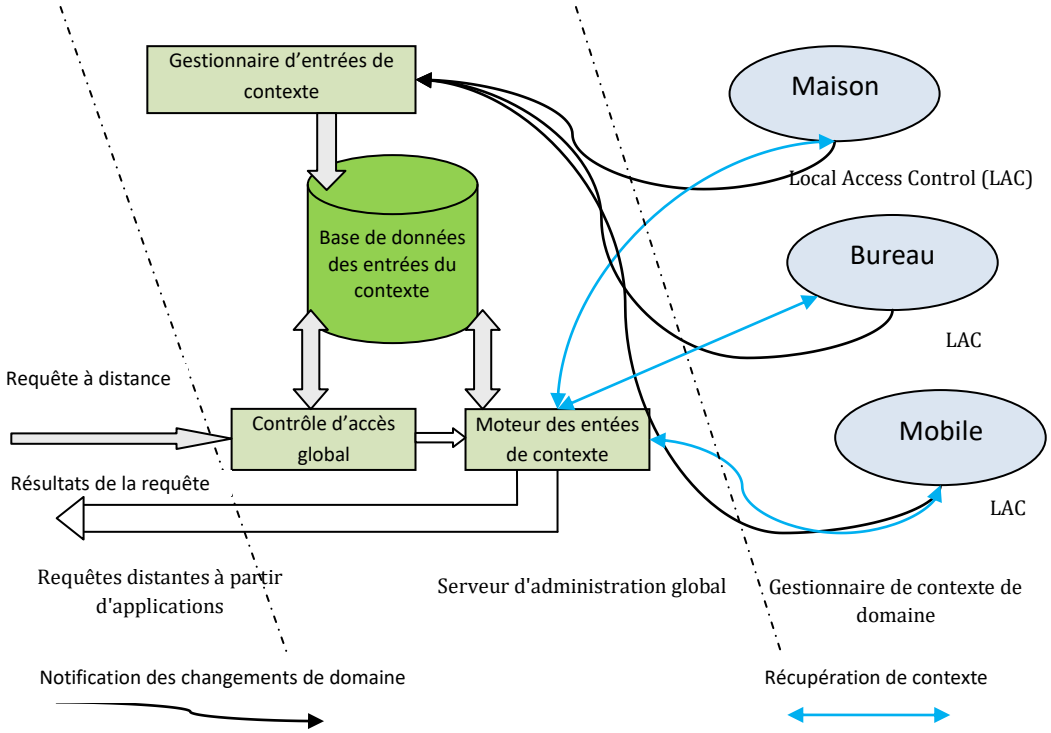

Figure 13 - Framework de gestion de contexte Feel@Home (Guo, Sun et al. 2010)

2.7. Conclusion

Le nombre d'appareils mobiles augmente du jour à l'autre, nous sommes donc face a l'ère de l'informatique ubiquitaire. Dans cette ère, l'utilisateur a à sa disposition une gamme de petits appareils informatiques tels que le smart phone ou l'assistant personnel, et leur utilisation fait partie de sa vie quotidienne. Pour une bonne utilisation et une meilleure exploitation de ces dispositifs, la notion de sensibilité au contexte s'impose pour faciliter l'interaction entre l'homme et la machine et les interactions des machines entre eux (ou ce qu'on appelle le M2M).

Nous avons présenté dans ce chapitre toutes les notions et concepts relatifs au contexte, son utilisation, la structure d'un système sensible au contexte. Nous avons ainsi cité des exemples sur les applications contextuelles existantes.

Nous présentons dans ce qui suit l'évolution des systèmes en réponse aux changements de contexte. Nous expliquons le processus correspondant en témoignant par les modèles d'évolution existants.

CHAPITRE 3

Evolution des architectures logicielles

3.1. Introduction

Les systèmes sont de plus en plus dynamiques, leur topologie change afin de pouvoir répondre rapidement à de nouveaux besoins et d'alléger les ressources matérielles.

Le terme évolution est utilisé depuis les débuts des années 60 pour caractériser la dynamique de croissance des logiciels. Dey et al (Dey, Abowd et al. 1998) commencent à introduire la notion d'adaptation en définissant la conscience au contexte comme un travail menant à l'automatisation d'un logiciel basé sur la connaissance du contexte de l'utilisateur.

Nous avons présenté dans le chapitre précédent les notions de base relatives à l'Internet des Objets et à la sensibilité au contexte, nous abordons dans ce chapitre la problématique de l'évolution dans les architectures logicielles. Nous nous intéressons plus spécifiquement aux caractéristiques d'une évolution et à son impact sur les architectures des systèmes IoT.

Bien que l'idée de faire participer un système logiciel à la prise en charge des changements qu'il subit soit intéressante, la manière dont ces changements sont appréhendés, codés puis injectés dans le système, pour être concrétisés au moment opportun, demeure un défi considérable. C'est un défi aussi bien sur le plan méthodologique (absence d'une démarche adéquate), que sur le plan de modélisation et plateformes supports.

L'évolution des architectures logicielles est étudiée dans ce chapitre, notamment ses définitions, ses types, les dimensions qui forment une évolution. Une étude sur les modèles d'évolution existants est donnée selon différents critères. La notion de style d'évolution est ainsi introduite.

3.2. Définitions de l'évolution

Nous parlons d'évolution pour désigner soit une opération de maintenance, d'adaptation, de reconfiguration ou de versionnement d'un système. Les adaptations sont le résultat de l'évolution. Un système est capable d'évoluer par lui même et automatiquement grâce aux auto-reconfigurations.

La norme IEEE sur la maintenance logicielle définit l'évolution par *« La modification d'un logiciel, après sa livraison, afin de corriger des défaillances, d'améliorer sa performance ou d'autres attributs ou de l'adapter suite à des changements d'environnements »* (1219-1993 1993).

La norme ISO-IEC 12207 pour les processus de cycles de vie décrit essentiellement la maintenance comme l'un des processus primaires du cycle de vie d'un produit logiciel subissant une modification du code et de la documentation associée due à un problème ou au besoin d'amélioration. L'objectif est de modifier le produit logiciel existant tout en préservant son intégrité (Singh 1996).

(Sicilia, Cuadrado et al. 2005) définissent l'évolution par: *« La totalité des activités qui sont requises afin de procurer un support, au meilleur coût possible, d'un logiciel. Certaines activités débutent avant la livraison du logiciel, donc pendant la conception initiale, mais la majorité des activités ont lieu après sa livraison finale (quand l'équipe de développement considère qu'elle a terminé son travail) »*.

La définition donnée par (FIPS 1984) est la suivante: *« L'ensemble des activités requises afin de garder le logiciel en état d'opération suite à sa livraison opérationnelle ».*

3.3. Types d'évolution

Tout système est susceptible d'évoluer, qu'il soit déployé dans son site final ou encore en cours de développement. Cette évolution agit soit sur la structure du système i.e. comment sont organisés ses composants ou sur son comportement i.e. la façon dont il réagit avec son extérieur. On distingue deux types d'évolution selon son impact sur le système: «évolution structurelle» et «évolution comportementale».

Une évolution est dite structurelle si elle agit sur les composants internes du système, on distingue des opérations d'ajout, suppression, modification des composants ou des liens

entre eux. Elle est comportementale si elle influe sur les fonctionnalités du système ou sur la façon dont il réagit avec son environnement.

L'évolution d'une architecture logicielle peut être réalisée à l'étape de spécification ou de conception du système qu'elle décrit (évolution statique) ou en cours de l'exécution de ce système (évolution dynamique). Principalement, il y a quatre genres d'adaptations: adaptation du contenu, adaptation du comportement (service), adaptation de présentation (ou interface) et adaptation de composants logiciels.

Selon (Gandrille 2014), nous pouvons adapter l'architecture conceptuelle de l'application en ajoutant, supprimant ou modifiant les composants. Nous pouvons ainsi adapter l'architecture de déploiement qui correspond à la migration d'un composant d'un site à l'autre dans le contexte de mobilité des applications.

Une classification d'évolution en douze types est donnée par Chapin et al (Chapin, Hale et al. 2001): évaluation, consultation, formation, mise à jour, réformation, adaptation, performance, prévention, arrangement, enrichissement, correction et réduction.

3.4. Lois de l'évolution

Selon (Lehman, Ramil et al. 1997), le système doit être continuellement mis à jour pour conserver sa validité et s'adapter aux nouvelles circonstances.

Constatant que la plupart des logiciels sont susceptibles de changer au cours de leur existence, Lehman et Beladi ont entrepris de déterminer les lois auxquelles ces changements obéiront ou devront obéir pour que le logiciel puisse survivre. Depuis 1947, ces lois ont été révisées et étendues à plusieurs reprises. Nous citons dans ce qui suit les huit lois qui ont été formulées.

1. Changement continu: Un programme utilisé et qui reflète une réalité, subit des changements continus ou perd progressivement de son utilité. Le processus de changement ou de désintégration se poursuit jusqu'à ce qu'il soit jugé plus rentable de remplacer le programme par une version recréée.

2. Augmentation de la complexité: Comme un programme en évolution change constamment, sa complexité, reflétant une structure en détérioration, augmente à moins que des travaux ne soient faits pour le maintenir ou le réduire.

3. Autorégulation: L'évolution du programme est soumise à une dynamique qui rend le processus de programmation, et donc les mesures d'attributs globaux de projet et de

système, s'autorégulant avec des tendances et des invariances statistiquement déterminables.

4. Conservation de la stabilité organisationnelle: Pendant la vie d'un programme, son taux d'activité est approximativement invariant sur la durée de vie du produit.

5. Conservation de la familiarité: Toutes les personnes qui sont associées au système susceptible d'évoluer doivent maintenir la maîtrise de son contenu et son comportement pour obtenir une évolution satisfaisante. L'incrément de changement dans chaque version est approximativement constant.

6. Croissance continue: Le contenu fonctionnel du système de doit être continuellement augmenté pour maintenir la satisfaction des utilisateurs tout au long de son cycle de vie.

7. Qualité décroissante: La qualité du système diminue à moins qu'il ne soit rigoureusement maintenu et adapté aux changements de son environnement de fonctionnement.

8. Système de rétroaction: Les processus d'évolution de type E constituent des systèmes de rétroaction à plusieurs niveaux, à plusieurs boucles et à plusieurs agents ; et ils doivent être traités comme tels pour obtenir des améliorations significatives par rapport à toute base raisonnable.

3.5. Dimensions d'une évolution

Le but principal de cette étude est de proposer une approche efficace pour effectuer des actions de reconfiguration de manière transparente et dynamique sur des systèmes basés sur des objets connectés. Tout d'abord, nous devons travailler sur le sujet en essayant de répondre aux quatre dimensions qui définissent l'évolution dans ces systèmes et qui vont mener à construire le processus correspondant. Les caractéristiques qui nous paraissent importantes pour aborder l'évolution dans les architectures logicielles sont introduites sous forme d'interrogations: qui, quoi, quand et comment.

3.5.1. Objet de l'évolution

Le quoi correspond à l'objet de l'évolution. Cette dimension précise les éléments architecturaux sur lesquels porte l'évolution. Dans cette étude, nous nous intéressons aux systèmes à base d'objets connectés, l'évolution peut donc porter sur les objets, les connexions entre eux ou sur les deux.

Dans cette recherche, nous nous intéressons aux systèmes logiciels complexes basés sur des objets connectés. Ces systèmes contiennent un ensemble d'entités reliées entre elles par des connexions réseau dans le but de communiquer, de partager des données et des ressources pour une meilleure exploration d'Internet. Ces entités peuvent être des personnes, des lieux ou des objets du monde réel (Abowd, Atkeson et al. 1997). Les connexions entre ces entités diffèrent selon les intervenants (connexion homme-machine, connexion homme à homme, connexion machine à machine (Tan and Wang 2010)).

D'un point de vue structurel et fonctionnel (Figure 14), chaque objet connecté contient un identifiant unique, un nom, un emplacement et d'autres attributs décrivant ses propriétés. Il contient plusieurs capteurs en fonction de son type et de ses caractéristiques. Ces capteurs servent d'acquéreurs de données contextuelles utilisant des fonctionnalités communes, notamment l'acquisition, l'interprétation et la réaction aux données collectées (Kortuem, Kawsar et al. 2010). Les objets intelligents sont connectés entre eux par des connexions réseaux et technologiques.

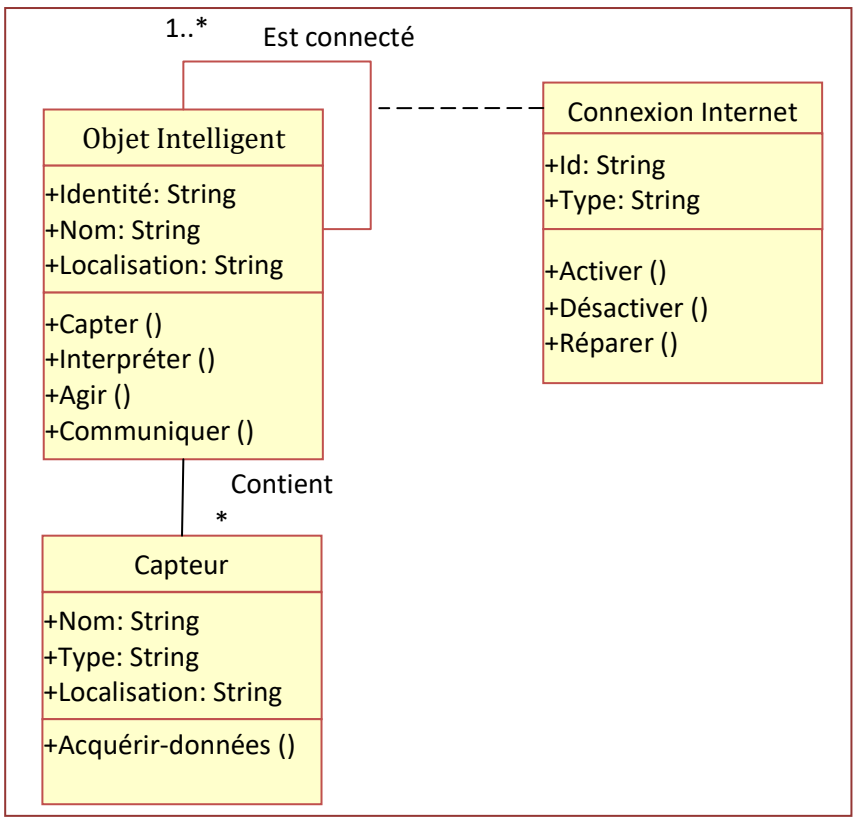

Figure 14 Structure des objets connectés

Comme le montre la figure, chaque objet connecté contient un identifiant unique, un nom, un emplacement et d'autres attributs décrivant ses propriétés. Il contient plusieurs capteurs en fonction de son type et de ses fonctionnalités. Ces capteurs servent d'acquéreurs de données contextuelles utilisant des fonctionnalités communes, notamment l'acquisition, l'interprétation et la réaction aux données collectées. Les objets intelligents sont connectés par des connexions réseau et technologiques telles que Wi-Fi, Bluetooth, NFC, RFID, ZigBee, Z-wave, 6lowPan, etc.

3.5.2. Déclencheur de la reconfiguration

Cette dimension correspond à qui intervient dans ces objets. Autrement dit, quels sont les responsables de déclenchement d'une évolution ou encore pourquoi évoluer un système. Dans notre cas, l'évolution est résultante des changements de contexte du système. La nature des systèmes étudiés fait que les évolutions sont imprévus et doivent donc être effectués dynamiquement de façon non intrusive.

Définir ce qu'on entend par contexte est une question complexe, puisque cette notion est utilisée par plusieurs domaines de recherche et pas seulement l'informatique qui l'appréhendent sous différentes perspectives.

Les informations de contexte brutes seront traduites et évaluées dans le but de décider de déclencher ou ignorer une opération de reconfiguration. Par conséquent, le déclenchement d'une opération de reconfiguration est principalement lié à des modifications du contexte du système, c'est ce qu'on appelle la sensibilité du contexte (Schilit, Adams et al. 1994).

Trois difficultés majeures concernant l'acquisition de contexte sont énumérées par (Dey and Abowd 2000; Dey 2001):

(1) Le contexte est dynamique: les changements dans l'environnement doivent être détectés en temps réel et les applications doivent s'adapter a ces changements continus;

(2) Le contexte est capturé à partir de multiples sources, souvent hétérogènes et reparties;

(3) Le contexte est obtenu à travers la manipulation des périphériques non conventionnels (autres que la souris ou le clavier).

Dans les environnements d'intelligence ambiante, les appareils communiquent et coopèrent de manière autonome, sans intervention de l'utilisateur. Les appareils

détermineront les choix en fonction de divers facteurs, notamment l'inclination des utilisateurs et la proximité d'utilisateurs différents dans le voisinage immédiat (Preuveneers, Van den Bergh et al. 2004).

L'ordinateur devrait devenir invisible pour l'utilisateur. Au lieu de cela, l'utilisateur doit communiquer naturellement avec un environnement intelligent, ce qui le soutient dans ses objectifs. L'interaction naturelle est obtenue en fournissant un support d'interface pour la langue, le mouvement, le geste et le pointage (Coen 1998). La connaissance du contexte est également importante pour soutenir l'interaction naturelle et fournir des services proactifs (Wojciechowski et Xiong 2008).

3.5.3. Techniques d'intervention

Cette dimension correspond à comment intervenir en cas de changements de contexte, c'est-à-dire quelles sont les techniques de reconfiguration utilisées pour garder le système dans une situation stable ou encore comment fonctionne un système sensible au contexte.

Principalement, toute opération de reconfiguration passe par trois phases, à savoir:

- Détection/notification des variations de contexte: Cela nécessite un mécanisme permettant de détecter et de notifier les variations pertinentes de l'environnement, telles que la diminution de la bande passante du réseau, l'action directe d'un utilisateur humain sur une interface homme-machine.
- Prise de décision : Après la notification d'une variation du contexte, l'adaptateur doit décider en fonction d'une politique d'adaptation quelles actions entreprendre.
- Exécution: L'adaptateur exécute les actions de modifications décidées. Elles doivent être appliquées sur l'application concernée par les variations (sur ses composants et / ou sa structure) ou sur l'infrastructure sous-jacente et / ou sur les applications simultanées.

En ce qui concerne les techniques et les concepts de programmation utilisés pour l'adaptation, nous avons selon (Chefrour 2005),(Ketfi and Belkhatir 2004; Aissaoui, Atil et al. 2013):

Technique de réflexion: La réflexion était utilisée pour développer des systèmes ouverts et extensibles. Un système logiciel est réflexif s'il est capable d'accéder à son propre comportement, de le raisonner et de le modifier. Un système de réflexion est composé de deux niveaux principaux: le niveau de base contenant le système et le méta-niveau qui

manipule la représentation de soi. Ces deux niveaux sont liés de façon causale, c'est à dire que tout changement dans l'un d'eux se répercute sur l'autre. La technique de réification est utilisée pour accéder au méta-niveau en exposant certains aspects de la représentation de soi de telle sorte qu'ils puissent être accédés par l'application (le niveau de base).

L'avantage majeur apporté par la technique de réflexion est la séparation entre le code métier des applications et le code qui gère l'adaptation. Cependant, cette technique engendre un surcoût en temps d'exécution à cause de la réification. Un système réflexif possède de deux caractéristiques complémentaires:

- L'introspection est la capacité d'un système à observer et raisonner sur son propre état;
- L'intercession est la capacité d'un programme à modifier son propre état d'exécution.

Programmation orientée aspects: L'idée principale de l'AOP est de considérer qu'un système est mieux développé lorsque l'implémentation de ses fonctionnalités est séparée de celles qui concernent des fonctionnalités techniques et transverses. Les composants ou entités développés dans une application ont généralement besoin de plusieurs fonctionnalités transverses purement techniques : log, habilitation, gestion de transactions. Ces fonctionnalités ajoutent du code dans les traitements ce qui les alourdit. La programmation orientée aspect propose d'externaliser ces fonctionnalités sous la forme d'aspects. Cela signifie que les aspects non fonctionnels de l'application traitant de la reconfiguration du système sont implémentés séparément du code de l'application de base et que le tissage est effectué à l'exécution.

Programmation basée ADL: Il est possible d'adapter une application dynamiquement en reconfigurant son architecture. L'idée commune des ADLs dynamiques consiste en la réaction aux évolutions logicielles et aux changements de l'environnement d'exécution par créer ou détruire dynamiquement les composants et les connexions entre eux.

Programmation orientée composants: Elle facilite la construction des systèmes complexes, leur déploiement, administration et control de leurs évolutions à travers l'utilisation des composants logiciels, patrons de conception, framework et middlewares.

3.5.4. Moments d'intervention

Cette dimension correspond au moment de l'intervention, une évolution peut être déclenchée pendant la phase de développement, de déploiement ou pendant l'exécution du système (Ketfi and Belkhatir 2004; Chefrour 2005).

- Temps de conception (statique): Le changement de logiciel concerne le code source du système. Par conséquent, le logiciel doit être recompilé pour que les modifications deviennent disponibles. La technique la plus simple mais la plus coûteuse est la réécriture de tout ou d'une partie du code pour accommoder l'application aux conditions particulières de son environnement.
- Temps de chargement: La modification logicielle se produit lorsque les éléments logiciels sont chargés dans un système exécutable. C'est notamment le cas quand il s'agit de porter une application vers une plate-forme différente de celle prévue initialement
- Temps d'exécution (dynamique): Le changement de logiciel a lieu pendant son exécution. Afin d'assurer le bon fonctionnement des applications en dépit des variations de leurs environnements il faut les adapter en cours d'exécution. Cette adaptation qualifiée de dynamique peut être difficile à mettre en œuvre car, en plus d'écrire le code métier de l'application, les développeurs doivent écrire le code qui observe les variations des ressources et réalise les modifications nécessaires tout en préservant l'intégrité de l'application.

3.6. Travaux connexes (Evolution des architectures logicielles)

Beaucoup d'approches ont été proposées dans la littérature afin de reconfigurer dynamiquement des composants et des applications conçues à base de composants.

Garlan et al (Garlan, Siewiorek et al. 2002) proposent Aura, un système visant à minimiser les distractions de l'attention de l'utilisateur, en créant un environnement qui s'adapte à son contexte et à ses besoins. Les détails sur le fonctionnement et la structure d'Aura sont données dans la section 2.6.

Par ailleurs, en raison de l'échec des infrastructures existantes à traiter les informations de contexte de manière évolutive et efficace, (Conan, Rouvoy et al. 2007) proposent Cosmos. Un framework basé sur les nœuds de contexte et de règles de gestion de nœuds traduites en composants logiciels dans une architecture logicielle.

Le framework de gestion de contexte Cosmos est divisé en trois couches: la couche collection de contexte, la couche de traitement de contexte et la couche d'adaptation de contexte.

En outre, (Floch, Hallsteinsen et al. 2006) proposent MADAM, un middleware fournissant trois fonctions élémentaires à savoir la détection des changements de contexte, raisonner sur les changements et décider de l'adaptation à effectuer, et la mise en œuvre les choix d'adaptation.

Grâce à la modularisation et à la flexibilité nécessaire à l'intégration de nouveaux composants et modèles, l'approche MADAM prend en charge l'évolution à la fois au temps de la conception et de l'exécution.

Parlant des systèmes cyber physiques (CPS), la puissance de calcul et la durée de vie de la batterie de chaque nœud associées à la nature très distribuée de ces systèmes, compliquent considérablement la gestion de la couche logicielle distribuée. (Acosta Padilla, Weis et al. 2014) proposent d'adapter le modèle @runtime pour l'utiliser sur les CPS pour permettre la gestion du déploiement de logiciels et la reconfiguration dynamique de ces systèmes. Ce middleware sera chargé de recevoir les nouveaux modèles définissant le nouvel état ciblé du système, définir l'ensemble des adaptations locales nécessaires pour atteindre ce nouvel état, en appliquant enfin les diverses adaptations locales produites par l'étape précédente.

Nous pouvons classifier les travaux répertoriés précédemment suivant différents critères que nous énonçons ci-après:

Critères relatifs aux raisons de la reconfiguration dynamique contextuelle

Etant donné que la majorité des travaux actuels se positionnent dans des environnements ubiquitaires et mobiles, la plupart des approches visant une reconfiguration dynamique contextuelle d'une application sont conçues à base de composants et ont pour objectif de la faire évoluer en fonction de son environnement d'exécution ou à la rendre plus performante sous certaines conditions.

Cependant, peu de travaux s'intéressent à rendre une application réactive plus flexible ou à améliorer la réutilisabilité des composants logiciels pour leur intégration dans des applications destinées à être exécutées dans des environnements totalement différents de ceux pour lesquels ils ont été conçus (par exemple, un composant qui a été conçu pour être déployé sur un ordinateur standard peut ne pas être réutilisable pour la conception d'une application destinée à être exécutée dans un environnement à ressources limitées (PDA, téléphone portable, etc.).

Critères relatifs à la cible de la reconfiguration dynamique contextuelle

La reconfiguration dynamique contextuelle d'une application peut concerner son comportement ou sa structure. Cependant, nous avons constaté que la majorité des approches existantes a pour objectif de faire évoluer le comportement en fonction de son environnement d'exécution. Peu de travaux se consacrent à reconfigurer dynamiquement suivant le contexte la structure de l'application afin de la rendre plus flexible Or, cette stratégie est grandement encouragée par le fort développement de l'informatique ubiquiste où la flexibilité est indispensable pour permettre le déploiement d'application dans certaines conditions (ressources limitées par exemple) et garantir la continuité et la qualité de service.

Critères relatifs à l'environnement d'exécution

A cause de la démocratisation des terminaux mobiles tels que les téléphones portables, PDA et autres, depuis quelques années, la majorité des travaux portant sur la reconfiguration d'applications en général et plus récemment, d'applications conçues à base de composants logiciels se positionnent dans le cadre d'environnements ubiquitaires et mobiles. En effet, l'informatique ubiquitaire et mobile dispose de caractéristiques telles que l'hétérogénéité des terminaux et des moyens de communication, qui rendent le domaine de recherche unique et fertile.

Dans le domaine des composants logiciels, les projets existants tentent de proposer des approches visant à introduire certaines propriétés liées à l'ubiquité telles que la prise en compte du contexte ou bien la gestion des connexions et des déconnexions des ressources d'un tel réseau, à des applications existantes.

Critères relatifs au moment de la reconfiguration dynamique contextuelle

La plupart des approches proposées dans la littérature ont pour objectif de reconfigurer les applications au moment de leur exécution et ce, sans l'arrêter. Ceci est dû, comme nous l'avons évoqué précédemment, au fort développement de l'informatique ubiquitaire qui nécessite la prise en compte permanente du contexte d'exécution par l'application de manière à lui garantir une continuité de service et une qualité de service.

Critères relatifs au niveau d'automatisation de la reconfiguration dynamique contextuelle

La plupart des travaux existants ont pour objectif de concevoir des applications auto-reconfigurables (i.e. la reconfiguration est réalisée par l'application elle-même) dés lors qu'ils se positionnent dans un contexte de reconfiguration dynamique. En effet, dans de

nombreux environnements tels que les environnements ubiquitaires, le contexte d'exécution est en perpétuelle évolution. De ce fait, une reconfiguration manuelle, réalisée par l'administrateur de l'application ne peut être envisagée. Généralement, une couche non-fonctionnelle est intégrée aux composants afin de les rendre auto-reconfigurables. Cette couche peut prendre la forme d'un service non-fonctionnel intégré aux composants, d'un composant non-fonctionnel ajouté au modèle ou bien d'un niveau méta permettant d'agir sur l'application pour la reconfigurer (i.e. déclenchement de la reconfiguration, prise de décisions et réalisation de la reconfiguration).

Dans la majorité des approches, la reconfiguration doit être prévue dés la conception et le développement de l'application. Par exemple, certaines stratégies de reconfiguration utilisant le patron de conception « Strategy » doivent faire appel aux programmeurs qui sont chargés d'implémenter différentes versions d'un même composant de manière à ce qu'il exécute celle la mieux adaptée au contexte courant. Cependant, ce type de reconfiguration est très limitatif car il ne permet pas de réagir à des événements non prévus par les programmeurs. En effet, toutes les situations doivent être prévues dés la conception de l'application.

De plus, les politiques de reconfiguration sont généralement définies par l'administrateur de l'application sous forme de règles ECA (Evénement Condition Action). Ces politiques peuvent dans certains cas être insérées pendant l'exécution de l'application. De ce fait, le rôle de l'administrateur reste prépondérant dans la majorité des approches proposées.

Critères relatifs aux techniques de reconfiguration dynamique contextuelle utilisées

Les techniques de reconfiguration utilisées dans le cadre des approches existantes sont étroitement liées à la cible de la reconfiguration et au moment de la reconfiguration.

Concernant les techniques permettant de reconfigurer le comportement d'une application ou d'un composant, les plus utilisées dans la littérature sont la réflexion et la mise en œuvre de patron de conception. En effet, la réflexion offerte par la plupart des infrastructures logicielles permet de reconfigurer facilement le comportement d'une application ou des entités qui la composent de par l'utilisation des propriétés d'intercession qui permettent d'agir sur ces entités. Concernant les patrons de conception, les plus utilisés sont le patron d'observation (Observer) et le patron Strategy. Ces derniers permettent d'observer le contexte d'une application et de reconfigurer le comportement des composants en sélectionnant celui qui est le plus adapté à la situation.

Concernant les techniques permettant de reconfigurer la structure d'une application ou d'un composant, elles permettent de réorganiser la structure d'une application ou d'un composant de par la modification des connexions entre ses composants ou bien l'ajout, le remplacement ou la suppression de composants. Cependant, cette technique ne permet de reconfigurer que des assemblages de composants déjà existants (assemblages à plat ou bien assemblages hiérarchiques).

La reconfiguration de la structure a cependant été plus étudiée dans le domaine de l'orienté objet où de nombreuses approches permettant de partitionner des applications existantes ont été proposées.

Tableau 1 Framework de classification des approches de reconfiguration dynamique contextuelle

	Raison de reconfiguration	Cible de reconfiguration	Environnement d'exécution	Moment de reconfiguration	Niveau d'automatisation	Technique de reconfiguration
Aura (Garlan, Siewiorek et al. 2002)	Minimiser les distractions de l'attention de l'utilisateur	Comportement	Sensible au contexte	En exécution	Non intrusif	Proactive / auto-ajustement
Cosmos (Conan, Rouvoy et al. 2007)	Traiter les informations de contexte de manière évolutive et efficace	Comportement	Nœuds de contexte sensible au contexte	En exécution	Les nœuds de contexte extraient automatiquement tous les attributs disponibles	Patrons de conception(Composite, Factory, Flyweight, Singleton)
MADAM (Floch, Hallsteinsen et al. 2006)	Fonctionne avec des paramètres où les besoins des utilisateurs et les conditions de fonctionnement varient de manière dynamique	Structure et comportement	Environnement sensible au contexte/ changeant	En exécution et en conception	Auto-adaptation	Stratégies d'adaptation / politiques d'objectif étendu exprimées en "fonctions d'utilité"
Model @runtime for CPS (Acosta Padilla, Weis et al. 2014)	Puissance de calcul et autonomie de la batterie des nœuds CPS + nature distribuée des CPS	Déploiement et reconfiguration sur un CPS	Système distribué dynamique	En déploiement	Reconfiguration automatique	Model @runtime

3.7. Styles d'évolutions

La notion de style a été abordée dans plusieurs domaines comme la mode, l'urbanisme, la littérature. En architecture logicielle, David Garlan et Mary Shaw (Garlan and Shaw 1993) définissent un style architectural par: *"an architectural style determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of constraints on how they can be combined".*

En traduisant, un style architectural détermine le vocabulaire des composants et des connecteurs pouvant être utilisés dans les instances de ce style, ainsi qu'un ensemble de contraintes sur la façon dont ils peuvent être combinés.

Un style architectural caractérise une famille de systèmes liés par des propriétés structurelles et sémantiques communes. Plus spécifiquement, les styles architecturaux fournissent généralement quatre éléments (Monroe, Kompanek et al. 1997) y compris 1) un vocabulaire d'éléments de conception, 2) des règles de conception ou contraintes qui déterminent la composition autorisée de ces éléments, 3) une interprétation sémantique, dans laquelle les compositions d'éléments de conception, convenablement contraintes par les règles de conception, ont des significations bien définies. 4) Les analyses pouvant être effectuées sur des systèmes construits dans ce style. L'intérêt principal derrière l'utilisation des styles est la réutilisation des conceptions précédentes, les solutions trouvées peuvent être appliquées à de nouveaux problèmes en toute confiance.

Par analogie aux architectures logicielles, Olivier le Goaer *(Le Goaer 2009)* introduit le terme de style d'évolution architecturale. Il le définit comme suit: *« Un style d'évolution caractérise une famille d'évolutions dépendantes d'un domaine et technologiquement neutres. Il définit un vocabulaire d'activités et les moyens de les orchestrer pour atteindre un objectif donné.»*

L'objectif principal derrière l'utilisation des styles architecturaux est de thésauriser les savoirs et savoir-faire des personnes ayant déjà fait face à des évolutions, en vue de les réutiliser dans des situations similaires récurrentes.

3.8. Evolution des systèmes à objets connectés

La communauté de recherche envisage des efforts considérables envers l'adaptation des systèmes à objets connectés à leurs contextes. Cependant, il est important de signaler que c'est un défi majeur par rapport à la nature de ces systèmes. Les objets sont hautement dynamiques, ils sont hétérogènes et ils sont nombreux générant une énorme masse de données.

Il est à noter que le développement d'un bon logiciel revient à garantir son ouverture et la possibilité de l'étendre pour prendre en charge les nouveaux besoins de l'utilisateur et les changements qui affectent l'environnement d'exécution du système.

Nous dénombrons dans cette section quelques projets relatifs à l'évolution des systèmes à objets connectés.

Le framework Rainbow (Garlan, Cheng et al. 2004) fournit des mécanismes permettant de surveiller l'environnement d'un système, de l'analyser pour lancer le processus d'adaptation, de sélectionner la stratégie d'adaptation requise et d'apporter les modifications nécessaires au système en cours d'exécution. Pour capturer les réactions du système aux changements de contexte, ils utilisent un langage appelé Stitch. Figure 15 montre la boucle de contrôle du framework Rainbow pour l'auto-adaptation.

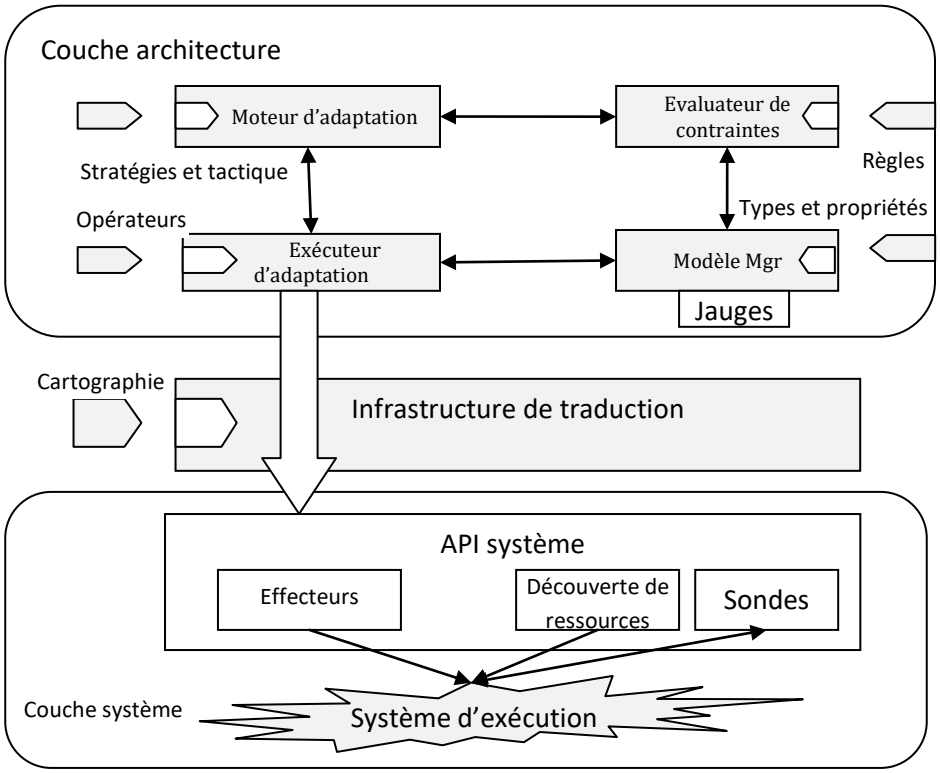

Figure 15 - Framework Rainbow (Garlan, Cheng et al. 2004)

Le projet SOCAM (Service-Oriented Context-Aware Middleware) (Gu, Pung et al. 2005) a introduit une architecture permettant de construire des systèmes logiciels adaptatifs. Il utilise un serveur central pour collecter des informations de contexte auprès de fournisseurs de contexte distribués. Ces informations sont ensuite traitées, de manière à pouvoir être utilisées par la fonctionnalité du système. L'architecture SOCAM illustrée dans la (Figure 16) se compose des composants qui agissent en tant que composants de service indépendants: les fournisseurs de contexte, l'interprète de contexte, la base de données contextuelle, les services sensibles au contexte et les services de localisation.

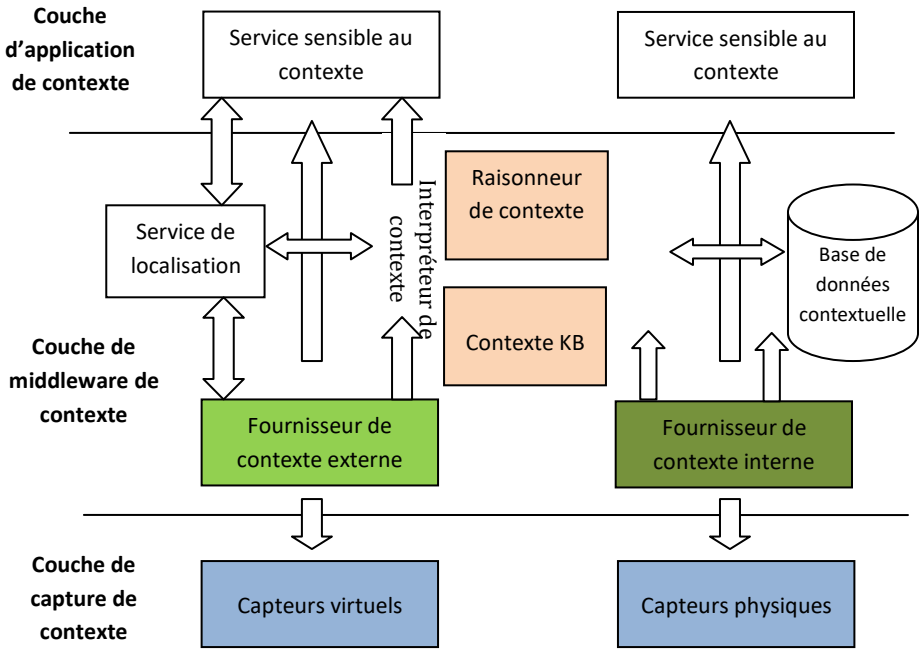

Figure 16 - Architecture SOCAM(Gu, Pung et al. 2005)

Le projet MUSIC (Rouvoy, Barone et al. 2009) est un framework à base de composants utilisé pour optimiser l'utilitaire global du système en réponse aux changements de contexte. Ils ont un modèle de qualité de service (QoS) décrivant la composition du système ainsi que les dimensions de QoS pertinentes, ainsi que la manière dont elles sont affectées lorsque le système passe d'une configuration à une autre. Le modèle de qualité de service est utilisé pour sélectionner une nouvelle configuration dotée du meilleur utilitaire et capable de gérer les changements de contexte. Figure 17 indique les composants impliqués dans le processus de reconfiguration y compris le gestionnaire de contexte, le contrôleur d'adaptation, le responsable de l'adaptation, le référentiel de plans, le gestionnaire de qualité de service ainsi que l'exécuteur de configuration.

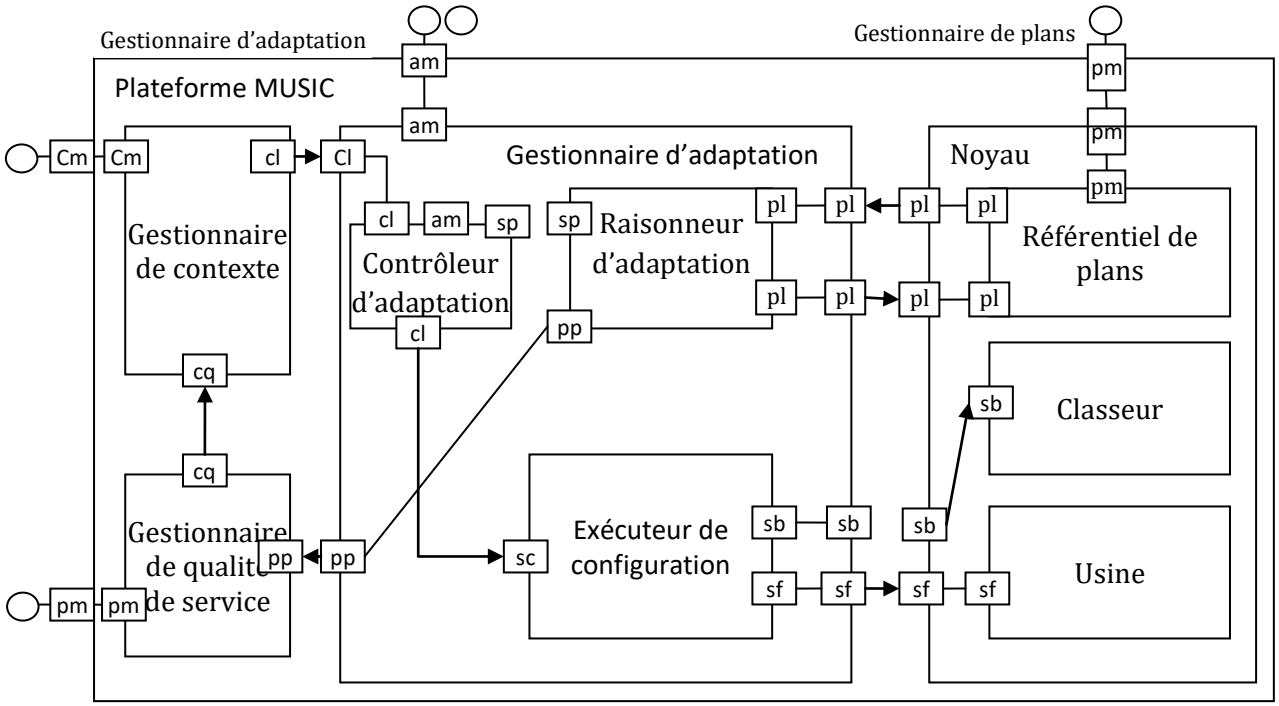

Figure 17 Architecture MUSIC(Rouvoy, Barone et al. 2009)

3.9. Conclusion

Dans le troisième chapitre de la thèse, nous avons discuté le sujet de l'évolution des systèmes logiciels. Les connaissances de base du sujet ont été bien cernées pour pouvoir par la suite les utiliser dans la proposition. Nous avons vu que l'évolution des logiciels est inévitable si on veut que le logiciel reste utile et s'améliore d'une configuration à l'autre pour satisfaire les nouveaux besoins de l'utilisateur et pour corriger les erreurs et améliorer les performances du système en question.

La notion de style d'évolution est accentuée dans ce chapitre. Ce concept a pour but de réutiliser les compétences d'une ancienne évolution dans des situations récurrentes.

Dans le chapitre suivant, nous mettons l'accent sur les détails de l'approche proposée en vu de remédier à la problématique posée dans cette thèse qu'est la reconfiguration dynamique contextuelle des systèmes à objets connectés.

CHAPITRE 4

Reconfiguration dynamique contextuelle des systèmes à objets connectés

4.1. Introduction

Face à un environnement changeant, aux nouvelles exigences et à la mobilité des usagers des systèmes informatique (Smartphone, tablette, ordinateurs portables, dispositifs sans fil, etc.), la prise en charge de l'aspect reconfiguration dynamique des systèmes est un aspect très important et sa prise en charge constitue un atout important. Par conséquent, la reconfiguration dynamique est indispensable lorsqu'on change du contexte d'exécution là où certains composants doivent disparaitre et d'autres qui seront détectés et qu'il faut prendre ces changements en considération d'une manière dynamique. Ceci justifie les motivations et les travaux de recherche sur la reconfiguration dynamique des systèmes avec une prise en compte dynamique du contexte, les problèmes qu'elle cause et les solutions possibles.

La mise à jour des systèmes complexes pour des raisons d'améliorations et d'adaptations aux changements de contextes est considérée une tache très difficile et couteuse. Pour cela, la notion de reconfiguration dynamique a vu le jour pour permettre d'apporter des changements aux systèmes en cours d'exécution.

Ce chapitre est consacré à la description de la solution proposée pour envisager la problématique principale de la thèse. C'est-à-dire nous discutons le modèle proposé et le fonctionnement du processus de reconfiguration contextuelle.

4.2. Concept de reconfiguration

Le verbe “reconfigurer” est composé d’un verbe en racine «configurer» et d’un préfixe «re». Le dictionnaire français Larousse (Larousse 1865) définit le verbe «configurer» comme : *”verbe transitif: Régler les paramètres d’un logiciel ou d’un matériel pour le faire fonctionner dans des conditions données. ”.* Quant au préfixe «re», selon le dictionnaire libre Wikitionary (dictionary 2015), il peut indiquer *une réaction, un acte de réponse en retour à un acte donné.*

Donc, nous constatons que la définition du terme « reconfiguration » revient à modifier une configuration initiale pour la faire fonctionner dans des conditions données. (Ketfi and Belkhatir 2004) définissent une reconfiguration par toute opération dont le rôle est le changement de la configuration courante.

Une reconfiguration est qualifiée de dynamique si elle est réalisée à l'exécution en assurant la cohérence et l'intégrité de l'application reconfigurée, elle est statique sinon.

Deux catégories de systèmes peuvent particulièrement bénéficier des reconfigurations dynamiques pour leur évolution: les systèmes à haute disponibilité et les systèmes adaptatifs. En effet, une reconfiguration dynamique ne nécessite pas l'arrêt ou le redéploiement complet du système pour le modifier, l'interruption de service est minimisée et la disponibilité du système est donc préservée. De plus, un système est capable d'évoluer par lui même et automatiquement grâce aux reconfigurations dynamiques.

Selon (Léger 2009), deux types de reconfigurations dynamiques existent, les reconfigurations programmées, et les reconfigurations ad-hoc.

Les reconfigurations programmées désignent des reconfigurations qui sont prévues au moment de la conception du système. Il est par exemple possible de prévoir plusieurs implémentations d'un même composant qui joueront le rôle d'alternatives possibles pour adapter le système au cours de son exécution.

Au contraire, les reconfigurations ad-hoc ou non-anticipées ne sont pas connues lors de la conception, elles consistent en des modifications arbitraires, peuvent intervenir à un moment quelconque, et sont initiées depuis l'extérieur du système. Ces reconfigurations sont par conséquent plus susceptibles de mettre le système dans un état invalide.

Les deux types de reconfigurations peuvent éventuellement être supportés en même temps dans un modèle de composant.

4.3. Architecture proposée

En vu d'adapter les systèmes à objets connectés aux changements de contexte, nous proposons un modèle d'architecture générique. L'architecture multicouche est inspirée de celle présentée par Khan, R et al (Khan, Khan et al. 2012) (Figure 4) en termes d'éléments et de technologies indispensables à l'IoT. Ce modèle d'architecture réflexive implémente un processus de reconfiguration dynamique contextuelle non-intrusive des systèmes à objets connectés. Elle est composée de quatre couches principales comprenant:

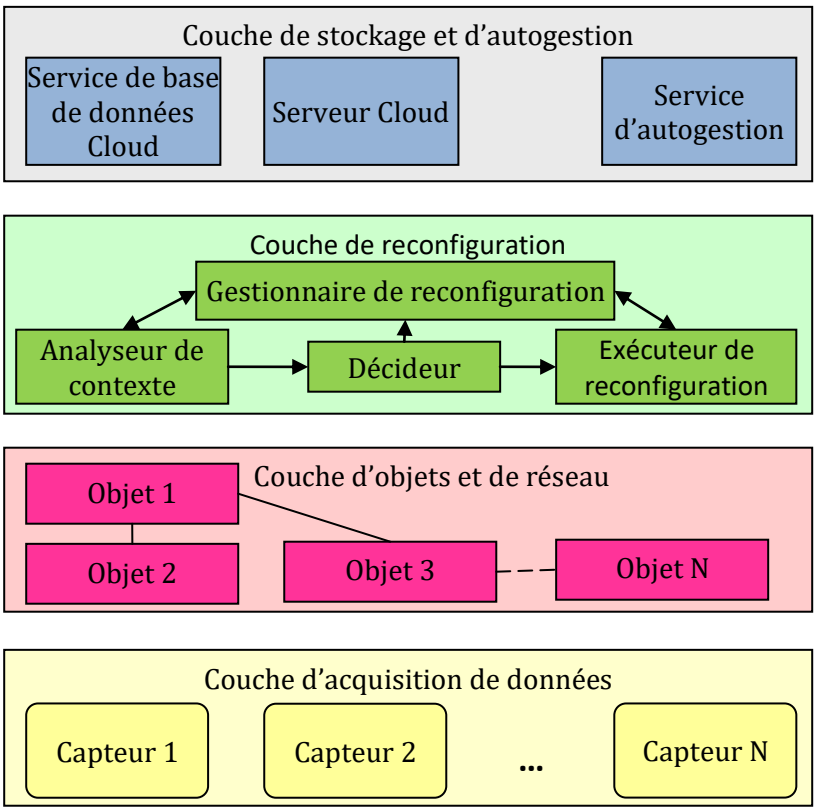

Figure 18 - Architecture de reconfiguration des systèmes IoT (Hakim, Amirat et al. 2018)

• Couche d'acquisition de données: Elle contient de nombreux capteurs qui servent d'acquéreurs de données de contexte brutes dynamiques. La première couche représente le contexte d'exécution des objets connectés, ce contexte peut être catégorisé selon (Ferry, Lavirotte et al. 2008) en quartes classes liés soi a l'utilisateur du système c.-à-d. il concerne toutes les informations qui peuvent être relatives à un utilisateur, sa localisation, son humeur, la tâche qu'il est en train de réaliser. Il peut aussi être lié à l'aspect temporel qui traite tout ce qui est relatifs aux instants, date, historique. Il concerne aussi les propriétés et variations de la machine qui s'intéresse à l'environnement d'exécution, aux ressources

disponibles pour les dispositifs du système. Le contexte environnemental est relatif à tout ce qui entoure le système, luminosité, bruit, température etc.

Cette catégorisation nous permettra par la suite de définir les grandes familles de conditions pouvant déclencher une adaptation.

Les informations relatives au contexte sont représentées dans la première couche de façon brute, elles seront par la suite disséminées vers les couches supérieurs pour les analyser et prendre en considération les informations les plus pertinentes et les plus appropriées a la situation.

• Couche d'objets et de réseau: représente l'aspect fonctionnel du système. Ce dernier est composé d'un ensemble d'objets intelligents liés entre eux. Ces objets sont représentés simplement par un diagramme de classes ou chaque objet est définit par les valeurs de ces attributs qui sont l'identité, le nom, la localisation et l'ensemble des capteurs qu'il contient. Les objets ont des fonctionnalités (Kortuem, Kawsar et al. 2010) communes qui caractérisent leurs intelligences y compris l'acquisition, l'interprétation des données contextuelles puis la réaction selon une politique donnée. Néanmoins, chaque objet a d'autres fonctionnalités et activités qui dépendent de sa nature et de son rôle. Les objets sont connectés entre eux par des liaisons et des technologies réseaux le plus souvent sans fils comme le wifi, le Bluetooth, le NFC (Near Field Communication), le ZigBee, le 6LowPan et le Z-Wave etc.

• Couche de reconfiguration: elle gère les opérations de reconfiguration. Elle contient quatre outils principaux: gestionnaire de reconfiguration, analyseur de données de contexte, décideur et exécuteur de reconfiguration.

Une énorme masse de données contextuelles est acquise à travers les capteurs ou les sondes, ces données sont envoyés au moteur ou gestionnaire des données contextuelles pour les analyser et les filtrer et partager les données les plus pertinentes au moteur de décisions qui, a son tour analyse ces données et décide selon une certaine politique de déclencher ou ignorer une opération de reconfiguration. Si la décision est positive et que le moteur de décision juge importantes les changements de contexte, la décision est envoyée au re-configurateur qui va exécuter l'opération et agir sur la couche des objets connectés soit en ajoutant, supprimant ou modifiant l'implémentation d'un objet ou un connecteur.

• Couche de stockage et d'autogestion: Elle est responsable du stockage et du traitement des données de contexte transmises par l'analyseur de données de contexte au Cloud. Elle contient également la base de connaissances sur le contexte statique. Cette couche fournit

ainsi un service d'autogestion (réflexivité) ayant pour rôle de maintenir la cohérence et de l'efficacité de l'ensemble du système après la reconfiguration.

Nous considérons que la couche réseau de l'architecture de Khan (Khan, Khan et al. 2012) est intégrée dans la couche application. Nous visons à améliorer l'architecture susmentionnée en ajoutant:

- Deux services Cloud, l'un délégué pour stocker les informations contextuelles et l'autre pour leur traitement.
- Outils exécutant un processus de reconfiguration dynamique contextuel (gestionnaire de reconfiguration, analyseur de données de contexte, décideur, exécuteur de reconfiguration).
- Un service pour assurer la réflexivité du système et son autogestion (le service d'autogestion dans la couche supérieure).
- Des mécanismes permettant de vérifier les propriétés non fonctionnelles du système après sa reconfiguration via le service de gestion automatique.
- Utilisation de styles d'évolution pour rendre les évolutions architecturales réutilisables afin d'exploiter les compétences d'un architecte.

4.4. Processus de reconfiguration

La mécanique opératoire de l'approche proposée consiste en un processus de reconfiguration appliqué à un élément architectural du système, quelle que soit sa granularité (objet, ensemble d'objets ou l'ensemble du système).

Par défaut (définition d'un objet dans la vision orientée objet), chaque élément d'architecture est défini par sa structure (champs et attributs) et son comportement (méthodes et procédures).

Dans cette recherche, nous nous intéressons aux systèmes IoT, qui sont intrinsèquement évolutifs en raison des changements fréquents dans leur contexte. Nous proposons d'associer à chaque élément architectural une partie "Evolution" comme illustré dans la Figure 19. Cette partie sera responsable de la gestion des transitions des systèmes tout en effectuant des évolutions, en particulier des reconfigurations.

Objet classique= structure+ comportement

Objet connecté= structure +comportement+évolution

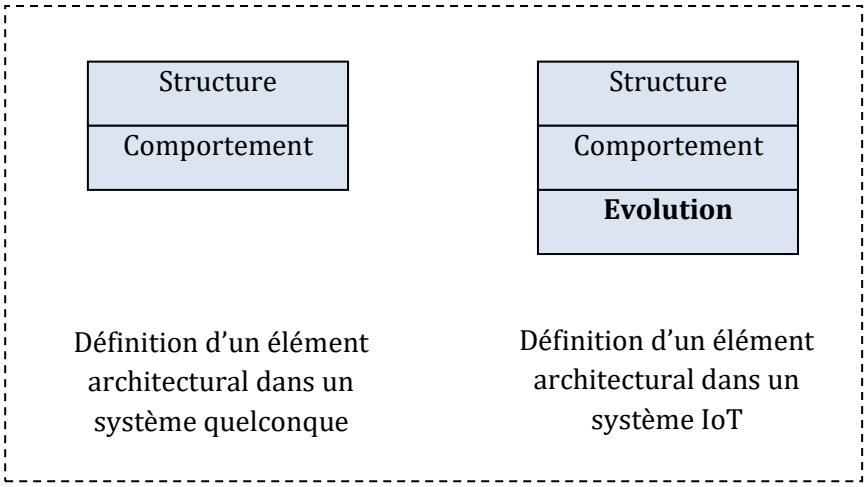

Figure 19 - Notre vision à la structure de l'objet

Il convient de noter que la reconfiguration s'applique au niveau architectural du système et influence l'évolution de son architecture car, selon Hassan, A et al (Hassan, Queudet et al. 2016), l'architecture logicielle est devenue la base du cycle de vie d'un système logiciel et constitue le modèle qui pilote le processus d'ingénierie. Par conséquent, l'évolution de l'architecture logicielle a été une question clé de la recherche sur l'évolution des logiciels.

Kakousis et al (Kakousis, Paspallis et al. 2010) ont été inspirés par la boucle MAPE-K dans les environnements d'informatique autonome; ils définissent la reconfiguration dans les systèmes omniprésents contextuels par une boucle fermée comprenant les étapes successives suivantes: en premier lieu, percevoir et traiter le contexte: au cours de cette phase, le contexte utilisateur (bruit, température, préférences utilisateur) et le contexte des données système sont collectés et traduits en événements contextuels de haut niveau pouvant déclencher une adaptation du système. Deuxièmement, raisonner et planifier l'adaptation: l'outil auto-adaptatif est invoqué pour raisonner sur le nouveau contexte et décider des actions à entreprendre. Et enfin, l'exécution de l'adaptation, où des mécanismes d'adaptation appropriés sont appliqués pour mettre en œuvre les décisions prises par le processus de raisonnement.

La boucle d'adaptation générale d'un système logiciel adaptatif comprend selon (Dobson, Denazis et al. 2006): l'observation de l'environnement, la sélection des adaptations et leur exécution. C'est la boucle CADA (Collection, Analyse, Décision et Action). Dans une telle boucle d'adaptation, le système peut être autonome ou impliquer l'humain.

4.4.1. Structure du gestionnaire d'évolution des systèmes IoT

Parlons de l'évolution des systèmes à objets connectés, il s'agit d'un processus hiérarchique, où la première couche est représentée par l'architecture du système IoT inspirée de celle de Khan et al (Khan, Khan et al. 2012). La deuxième couche c'est la mécanique opératoire du processus qui est la reconfiguration dynamique contextuelle. La troisième couche garantie la réflexivité du système et son autonomie. Les éléments clés de ce processus sont identifiés ainsi que les relations entre eux dans le méta-modèle présenté à la Figure 20.

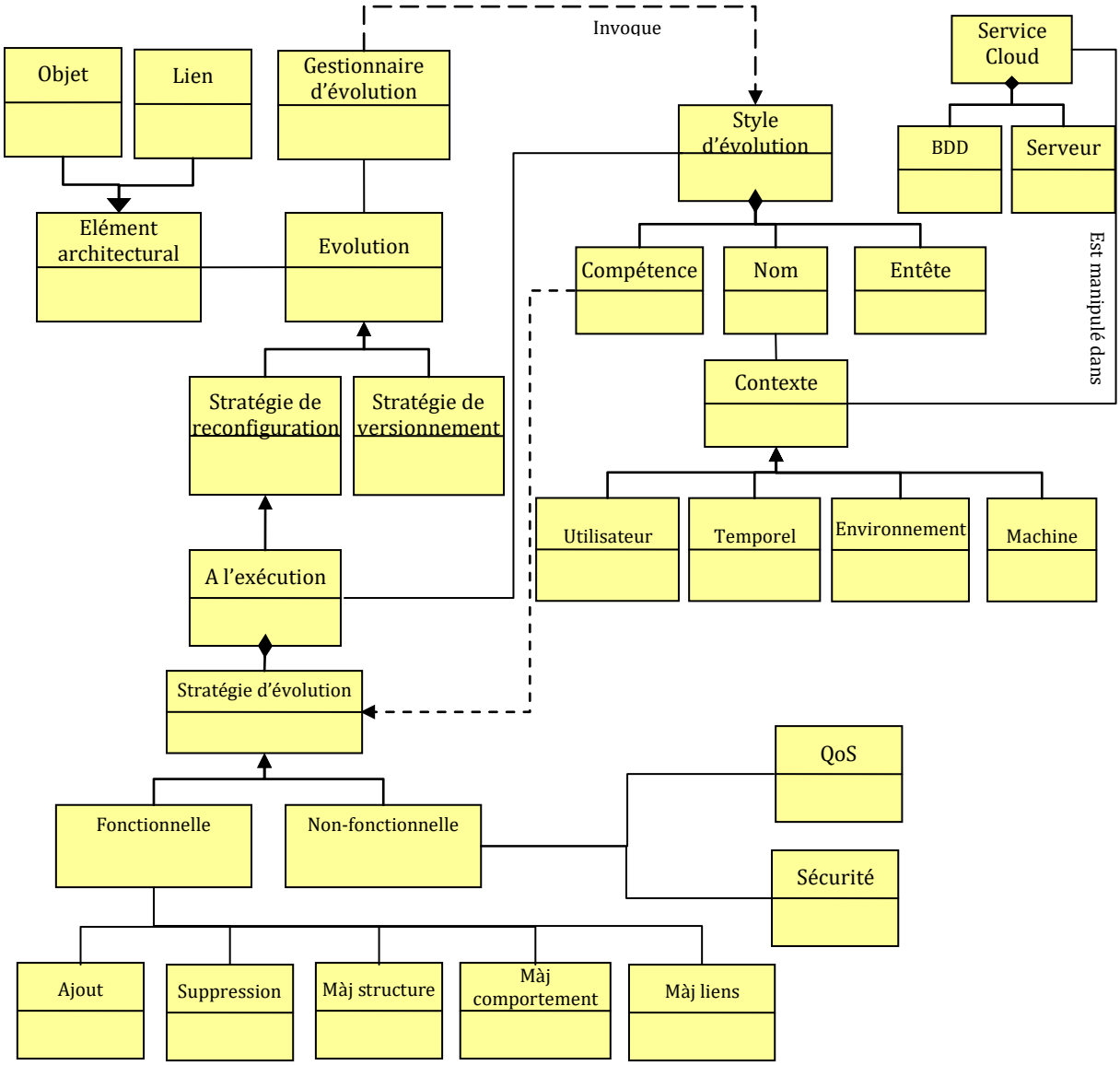

Figure 20 - Structure du gestionnaire d'évolution des systèmes à objets connectés

Figure 20 met en évidence la structure interne du gestionnaire d'évolution. Chaque élément architectural est associé à une ou plusieurs évolutions. Une évolution peut être une

stratégie de reconfiguration, une stratégie de gestion des versions ou tout autre type d'évolution. Dans ce travail, nous nous concentrerons sur la stratégie de reconfiguration dynamique. Par dynamique, nous voulons dire qu'elle peut être achevée à l'exécution sans nécessiter de coupures ni d'arrêts du système. Les actions de reconfiguration prises en charge par notre processus incluent: l'ajout d'un nouvel élément, la suppression d'un élément existant, la mise à jour de la structure / du comportement des objets, la mise à jour des liens, la révision des propriétés non fonctionnelles du système.

La nouveauté de notre travail réside dans l'utilisation de styles d'évolution. Le concept, introduit pour la première fois par Le Goaer et al (Le Goaer, Oussalah et al. 2007; Le Goaer 2009), par analogie avec les styles architecturaux, vise à rendre les évolutions architecturales réutilisables pour exploiter les compétences d'un architecte. Un style d'évolution est composé d'un nom, d'un entête et d'un ensemble de compétences. L'entête possède une description informelle de l'objectif et publie une liste de paramètres et d'assertions. Un compartiment de contexte est décrit dans l'entête d'un style d'évolution pour exprimer le contexte de l'élément évolutif. L'entête déclare les éléments descriptifs de l'évolution, y compris ses conditions pré et post évolution. Les compétences du style d'évolution indiquent comment procéder avec l'évolution.

4.4.2. Etapes de la reconfiguration dynamique contextuelle

Afin de promouvoir les systèmes IoT avec plus de flexibilité et de confort d'utilisation, nous visons à assurer une reconfiguration dynamique contextuelle. Pour ce faire, nous devons connaître les propriétés et les caractéristiques de ces systèmes. L'IoT est par nature lié à l'intelligence ambiante et au contrôle autonome, ce qui signifie la capacité du système de se contrôler soi-même. Le processus de reconfiguration des systèmes IoT consiste principalement à manipuler leur contexte en appliquant la boucle MAPE/K qui consiste à surveiller les données de contexte, à les analyser, à planifier et à exécuter la reconfiguration de manière appropriée. Nous discutons dans la section suivante le fonctionnement du processus étape par étape.

4.4.2.1. Acquisition des données contextuelles

Un système IoT est un système sensible au contexte, ce qui signifie qu'il est fortement influencé par les changements et les variations du contexte. Chaque objet connecté contient

un ou de plusieurs capteurs lui permettant d'acquérir des données de contexte et de prendre conscience de l'environnement auquel il appartient.

Les informations contextuelles peuvent provenir d'une grande variété de sources. Bien qu'une grande partie de la recherche précédente dans le domaine de sensibilité au contexte se concentre uniquement sur les informations détectées, d'autres modèles de contexte ont été trouvé, ils sont plus riches: ils intègrent les informations détectés, statique, et des informations dérivées fournis par l'utilisateur.

Selon N. Ferry et al. (Ferry, Lavirotte et al. 2008), le contexte peut être classé en quatre catégories pour définir les conditions pouvant déclencher une adaptation: contexte utilisateur, contexte machine, contexte temporel et contexte environnemental. Tandis que Bettini et al (Bettini, Brdiczka et al. 2010)ont trouvé que des modèles de contexte riches intégrant des informations détectées, statiques, fournies par l'utilisateur et dérivées étaient les plus utiles.

D'autres sources d'informations contextuelles peuvent être liées aux caractéristiques du système ou, en d'autres termes, à la connaissance de sa propre structure, de son architecture, de l'ajout ou de la suppression soudaine de nœud / entité existants.

En se référant à la structure du système sensible au contexte illustrée à la Figure 7, les données brutes détectées doivent être traitées et filtrées. Ici, un format de données standard est requis pour unifier la grande quantité de données provenant de sources variables.

Pour faciliter la recherche d'informations contextuelles et l'identification de la situation actuelle, le format XML est adapté (Figure 21). Les informations contextuelles des systèmes IoT sont divisées en une partie interne qui inclut les objets eux-mêmes, chaque objet étant caractérisé par un identifiant unique, un type et un ensemble de capteurs. Et une partie externe décrivant les informations de contexte relatives à l'utilisateur, à l'heure, à l'environnement et à la machine.

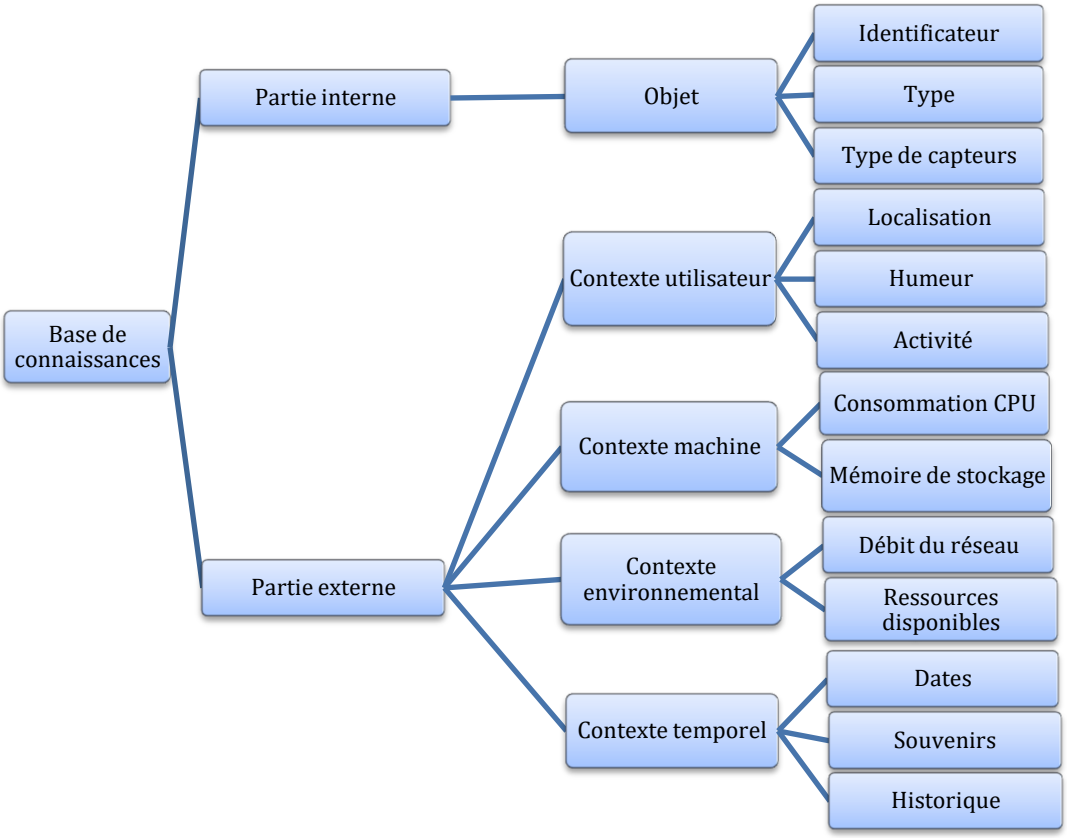

Figure 21 - Représentation des données contextuelles en utilisant le schéma XML

4.4.2.2. Prise de décision

À la réception d'une notification indiquant un changement de contexte du système, résultante de la mise à jour de la base de connaissances (fichier XML), le décideur analyse ces données dans l'algorithme suivant pour identifier la situation selon laquelle le gestionnaire de reconfiguration déclencherait ou ignorer l'action de reconfiguration.

Le décideur écoute en permanence le gestionnaire de la reconfiguration. Lorsque le fichier de la base de connaissances change, le gestionnaire de reconfiguration envoie une notification au décideur. La première étape à effectuer par ce dernier consiste à interroger la partie fonctionnelle du système en comparant le nombre actuel d'objets aux versions précédentes (qui sont stockées et triées par date dans le service de base de données du Cloud). Il y a trois possibilités:

Cas 1. Si le nombre d'objets augmente, l'exécuteur de reconfiguration invoque la stratégie d'ajout. Ici, un nouvel objet sera ajouté à l'ensemble des objets connectés. Les objets se distinguent par un numéro d'identification, les liens entre les anciens et le nouvel objet sont à régler.

Cas 2. Si le nombre d'objets diminue, il y a deux possibilités:

1. Un objet ne fait plus partie du système: dans ce cas, la stratégie de suppression d'objet est exécutée.

2. La bande passante du réseau est en train de diminuer ou un objet non fonctionnel a déclenché une anomalie. Nous donnons comme exemple les problèmes d'extensibilité qui influencent le fonctionnement du système.

Cas 3. Si le nombre d'objets ne change pas, il y a deux possibilités:

1. Certains objets existants modifient leurs structures ou leurs comportements,

2. Les préoccupations non fonctionnelles du système ont été affectées.

Algorithme de prise de décision

```
Begin
1.  While true do {
2.  Listen_to_the_reconfiguration_manager()
3.  Receive_notification ()
4.  Int nb:= old_number_of_objects
5.  Int count=0
6.  foreach (xml_node n in
knowledge_base.functional_part) {
7.  count += 1
8.   if (count < nb) then
9.     functional_reconf. add_object ()
10. End if
11.  else
     if (count > nb) then
12.   if network_bandwidth. IsOk () continue;
13.   else functional_reconf.remove_object();
14. End if
15. else if (count==nb) then
16.    functional_reconf. check_objectsStructure ()
17. End if
18.non_functional_reconf. CheckAllparameters ()
End.
```

Après exécution, l'algorithme fournit une solution de décision ou un plan d'adaptation dans un temps donné. L'algorithme peut être changé dynamiquement en runtime (da 2014).

4.4.2.3. Exécution de la reconfiguration

Le processus de reconfiguration sera déclenché en réponse aux changements de contexte. Une fois notifié par un événement de reconfiguration, à partir de l'analyseur de

contexte, le planificateur sélectionne une stratégie pour répondre aux nouvelles exigences à l'aide de la base de connaissances partagée. Dès réception du plan de reconfiguration, l'exécuteur adapte le système à son contexte actuel. Les actions de reconfiguration prises en charge par notre processus incluent:

– Ajouter de nouveaux objets: Créer de nouveaux objets et leur attribuer un identifiant unique, en connectant des capteurs aux objets en fonction de leurs fonctionnalités.

– Supprimer des objets existants: Cette action entraînera la suppression de ses liens avec d'autres objets.

– Modification de la structure d'un objet: La modification de la structure d'un objet consiste principalement à modifier sa visibilité, à ajouter ou à supprimer des attributs. Pour ce faire, supprimez temporairement l'objet, remplacez-le par un objet de sauvegarde, modifiez l'objet concerné, puis remplacez l'objet backup par l'objet modifié.

– Modification du comportement d'un objet: Le comportement d'un objet est décrit par Kortuem et al. (Kortuem, Kawsar et al. 2010): «*They sense, log, and interpret what's occurring within themselves and the world, act on their own, intercommunicate with each other, and exchange information with people*». Sur la base de cette définition, le changement de comportement peut affecter la manière dont l'objet capture les informations contextuelles ou comment il partage les informations qu'il capture. Cela peut également affecter l'interprétation des événements qu'il reçoit. La prise en charge de cette action se fait de la même manière que la modification de la structure.

– Modification de liens: les objets intelligents sont connectés par réseau et par des connexions technologiques telles que Wi-Fi, Bluetooth, NFC, RFID, ZigBee, Z-wave, 6lowPan, etc. Les changements à cette phase sont les suivants: changement du type de connexion utilisé pour connecter les objets système. Le changement peut également affecter la qualité du signal ou la bande passante des liens entre les objets connectés.

– Maintien des propriétés non fonctionnelles du système (qualité de service, sécurité, flexibilité, etc.): Une fois le processus de reconfiguration terminé, une vérification de la cohérence du système en termes de propriétés non fonctionnelles est effectuée. Cette dernière étape obligatoire garantit les performances du système et préserve la qualité de service et, enfin, le bon fonctionnement des objets connectés.

Chaque action de reconfiguration correspond à une stratégie. Par exemple, la maintenance d'un objet nécessite de le remplacer par un objet de sauvegarde afin de

garantir la transparence du système pour ses utilisateurs lors de ses opérations de maintenance.

En résumé, notre processus de reconfiguration dynamique consiste en: d'abord, l'activation du gestionnaire de l'analyse du contexte par le gestionnaire de reconfiguration. Le gestionnaire de l'analyse du contexte surveille continuellement l'environnement pour acquérir des données dynamiques de contexte. Ensuite, ces données seront analysées et traduites en information de contexte symbolique. Elles seront ensuite envoyées au gestionnaire de reconfiguration. Le décideur reçoit de nouvelles informations de contexte et analyse ces données selon l'algorithme définie pour identifier la situation selon laquelle le gestionnaire de reconfiguration pourrait déclencher ou ignorer l'action de reconfiguration.

Dans le cas où la décision est favorable, le gestionnaire de reconfiguration active l'exécuteur de reconfiguration en déclenchant une action de reconfiguration. Ce dernier, tout en terminant le processus de reconfiguration, attache un rapport et le remet au gestionnaire qui, à son tour, envoie le rapport correspondant au service d'autogestion afin de vérifier les aspects non-fonctionnels de la reconfiguration tels que la sécurité, l'exactitude et l'efficacité du système après la reconfiguration.

Les étapes du processus sont décrites à l'aide du diagramme de séquence illustré dans la Figure 22.

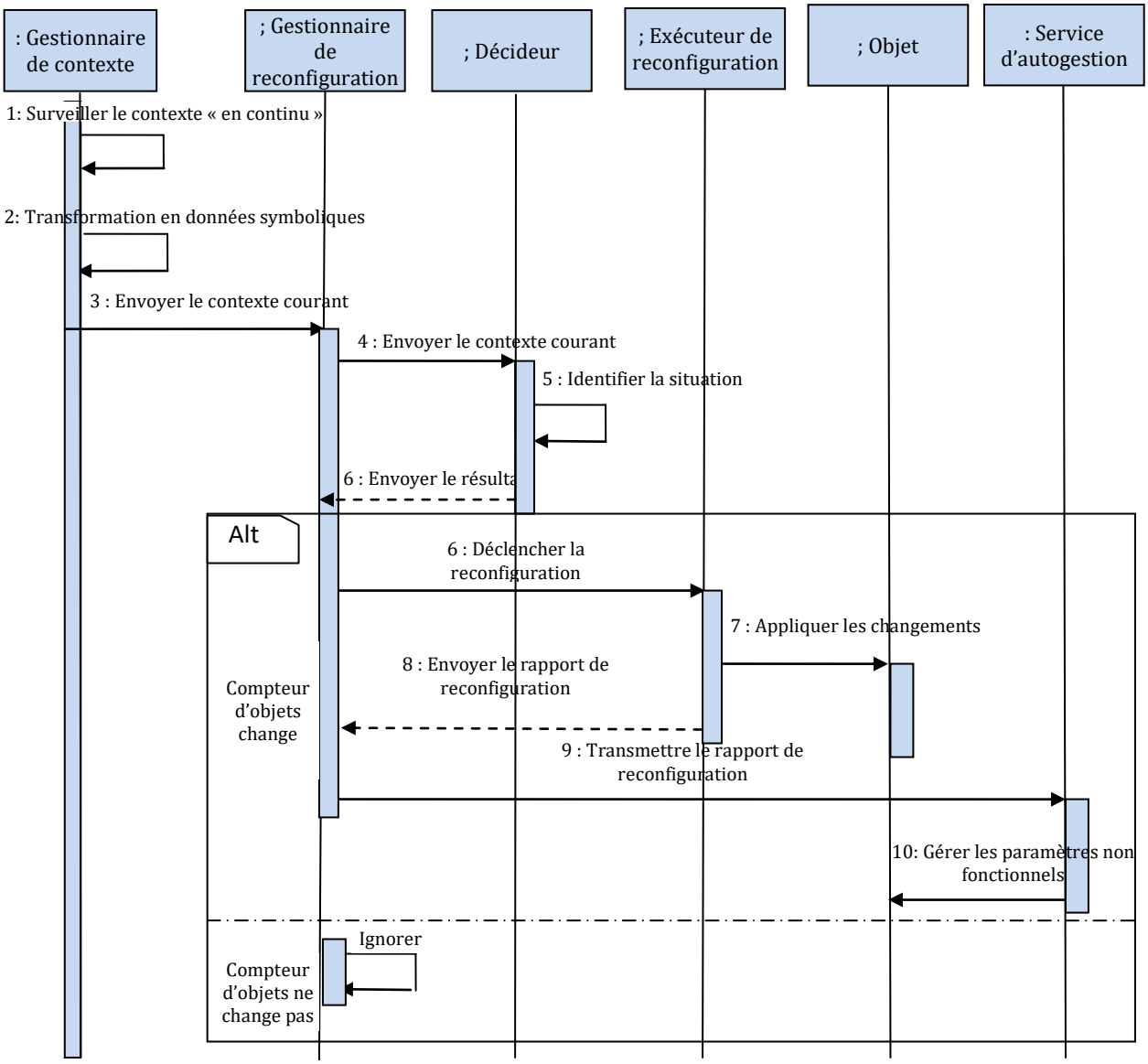

Figure 22 - Fonctionnement du processus de reconfiguration

4.5. Conclusion

Dans ce chapitre, la mécanique opératoire du processus proposé a été décrite. La solution au problème posé dans cette thèse est donnée dans ce chapitre où un modèle d'architecture générique à été proposé. L'architecture multicouche implémente un processus de reconfiguration dynamique contextuelle. La technique utilisée étant la réflexion, de ce fait, le système s'adapte automatiquement sans intervention externe.

La structure du gestionnaire de reconfiguration et son fonctionnement ont été abordés dans ce chapitre. Nous avons montré comment exploiter les informations contextuelles pour adapter le système aux paramètres changeants afin de satisfaire les besoins de l'utilisateur

et d'obéir à la nature des systèmes en question. Il nous reste à évaluer la proposition dans le chapitre qui suit.

Le chapitre suivant se concentre sur la réalisation du modèle proposé, les techniques utilisées pour la mise en marche du processus de reconfiguration et la proposition d'une étude de cas pour concrétiser par l'exemple.

Une simulation de l'environnement décrit dans le scénario de l'étude de cas est abordée à la fin du chapitre suivant. La simulation est effectuée dans un environnement informatique avant le développement réel du système proposé.

CHAPITRE 5

Expérimentations et réalisation

5.1. Introduction

Au cours des chapitres précédents, nous avons expliqué le besoin d'avoir un mécanisme qui assure l'ouverture et l'autonomie des systèmes à objets connectés c'est-à-dire la capacité d'évoluer dynamiquement. Un système à objet connecté est un système qui présente une haute disponibilité et qui se caractérise par une forte dynamique, les objets apparaissent et disparaissent à la volée. Les changements ne peuvent pas être prévus au temps de la conception, il est donc nécessaire d'avoir un processus de reconfiguration dynamique contextuelle.

Nous avons proposé un processus de reconfiguration dynamique contextuelle. Le fonctionnement du processus et ses principes ont été détaillée dans le chapitre précédent. Il nous reste à proposer une réalisation possible et à mener une expérimentation autour de notre proposition.

Nous avons opté vers une étude d'un cas particulier pour étudier le comportement du processus proposé face à un scénario concret dans le domaine de santé électronique. Ce chapitre est divisé en trois parties, la première étant l'étude de cas et le scénario approprié, la deuxième partie pour les détails de réalisation du processus et la troisième partie concerne la simulation du modèle proposé.

5.2. ComfortCareIoT: Etude de cas

Dans cette section, nous évaluerons notre approche sur un système mobile dans le domaine de santé électronique. Le système est chargé de surveiller le niveau glycémique d'un patient diabétique et de lui indiquer de procéder à son injection d'insuline à temps.

Un patient atteint du diabète type 1 âgé de soixante ans vivant seul a besoin d'un système lui permettant de s'occuper de sa vie, indépendamment de sa localisation et de son état de santé, c'est-à-dire quand il est chez lui ou à l'extérieur.

ComfortCareIoT est un système que nous proposons pour gérer la vie du patient diabétique susmentionné en lui rendant la maison plus intelligente, en prenant soin de lui en lui rappelant ses injections d'insuline et en prévenant les dommages pouvant survenir chez lui pendant son sommeil (Il est à noter que les diabétiques se caractérisent par un sommeil profond).

Il est arrivé que le système envoie une notification au patient pour lui rappeler son injection d'insuline à 9 heures du soir. La notification a été envoyée sur son Smartphone et le système attend la confirmation de sa réception.

Cinq minutes plus tard, le système inhabituellement n'a pas reçu de confirmation du patient. Ainsi, un appel a été transféré sur son Smartphone. En répondant, il a confirmé qu'il n'avait reçu aucune notification. Dans ce cas, le responsable du système rappelle au patient de prendre son injection et déclenche une action de reconfiguration.

Les actions de reconfiguration consistent à remplacer temporairement le Smartphone défectueux par la montre intelligente, en demandant au patient d'apporter le téléphone au service de maintenance du centre de santé en ligne.

La maison intelligente que nous proposons appartient à une personne âgée malade qui vit seule. Le fonctionnement du système sous un style sensible au contexte garantit que l'intervention de l'utilisateur n'est pas requise. Les utilisateurs ayant des capacités limitées peuvent facilement utiliser le système. Dans le scénario que nous proposons, le patient peut être endormi ou inconscient, mais sa vie reste toujours surveillée automatiquement.

5.3. Modélisation et réalisation

Pour implémenter le processus de reconfiguration dynamique sur les systèmes à objets connectés, nous avons proposé un modèle d'architecture générique (Figure 18).

En utilisant EMF, nous avons créé deux types de modèles, d'un côté un méta modèle (Figure 23) définissant les concepts de l'architecture discutée précédemment et de l'autre côté un modèle instanciant ses concepts. Ce dernier est décrit à l'aide d'une palette générée en utilisant l'outil Emfatic d'Eugenia sous Eclipse.

Notre ambition derrière la démarche proposée était de construire un modèle d'architecture générique et extensible. Pour cela, plusieurs propriétés sont visées par notre modèle:

La généricité: Le modèle que nous proposons est conçu d'une manière à lui permettre de s'adapter au domaine d'application dans lequel le système va être déployé. Cette caractéristique fournie par l'utilisation de la méta-modélisation permet de ne pas limiter notre approche à un simple langage ou à une notation de modélisation.

La réflexivité: Cette propriété fait que le système s'occupe lui-même de son adaptation. La détection du changement pertinent requérant une adaptation, nous visons à doter le système de la capacité d'auto configuration, auto optimisation et auto protection. Cette caractéristique diminuera les couts liés aux interventions externes en termes de temps et d'efforts. La réflexivité est assurée par le module d'autogestion implémenté dans la quatrième couche de l'architecture proposée.

L'invariance à l'échelle: Une des caractéristiques de l'internet des objets est le grand nombre d'objets connectés. La nature de ces objets implique qu'elle doit être flexible à l'ajout de nouveaux objets et de nouveaux services aux objets existants. Grace à la reconfiguration dynamique, l'ajout des objets en exécution n'influence pas la disponibilité du système.

La réutilisation: Le haut niveau d'abstraction de l'architecture proposée, ainsi que l'utilisation des styles d'évolution permettent la réutilisation des savoir faire concernant les stratégies d'évolution possible sur le système que ce soit par exemple l'ajout, la suppression et la modification d'objets connectés.

La sensibilité au contexte: Le modèle proposé permet l'adaptation du système aux changements de son environnement, cela est permis grâce à la qualité de sensibilité au contexte offerte par les systèmes IoT. Les informations contextuelles sont très importantes pour l'amélioration des performances des applications.

Le Cloud computing pour le stockage d'information de contexte: Les objets connectés ont des ressources limités en termes de calculs et de stockage: la durée de vie de la batterie est limitée, les objets consomment de l'énergie ce qui réduit les performances de l'application. Nous proposons que le stockage et le traitement des données se fait au niveau du Cloud.

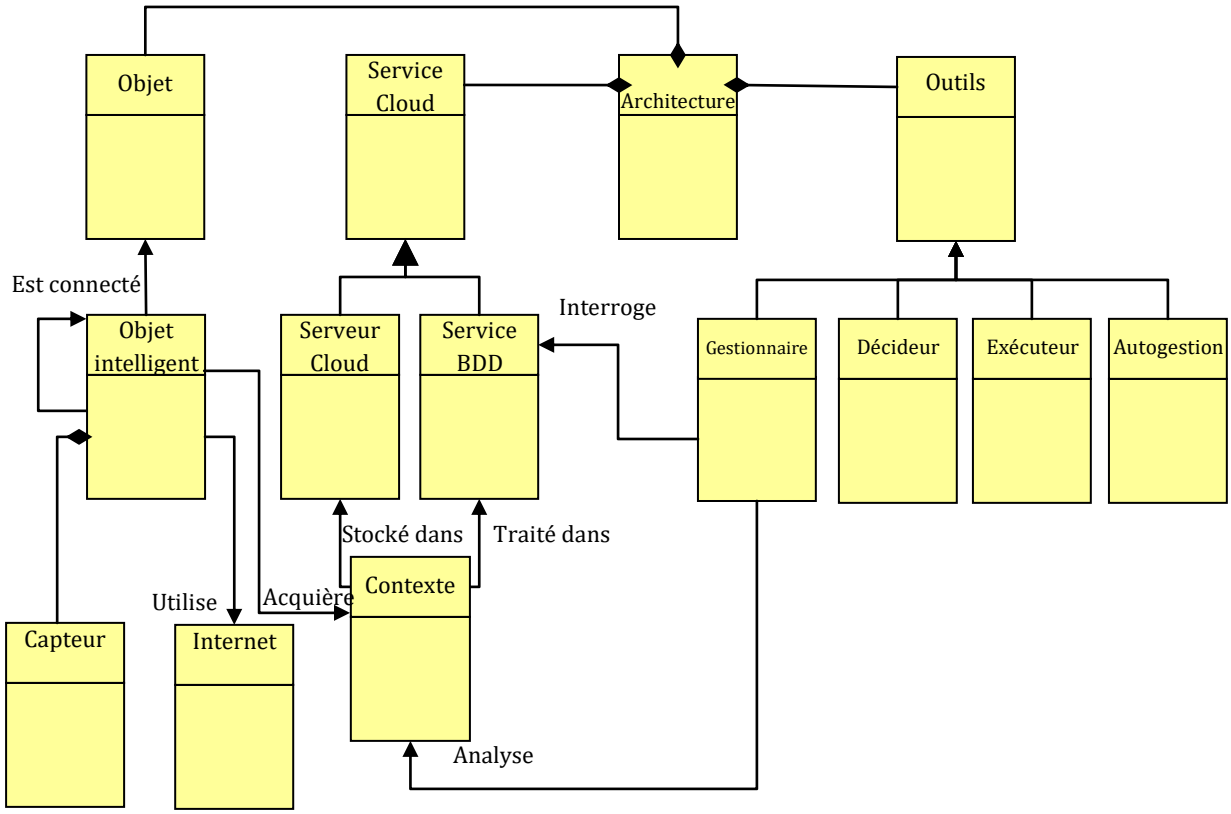

Figure 23 - Méta modèle de l'architecture proposée

En instanciant ce modèle, nous obtenons un langage spécifique au domaine de l'Internet des objets dans le domaine de santé électronique sous forme d'un plugin Eclipse (Figure 24) et une palette contenant les instances des éléments du méta-modèle précédent (éléments conformes a ce méta-modèle).

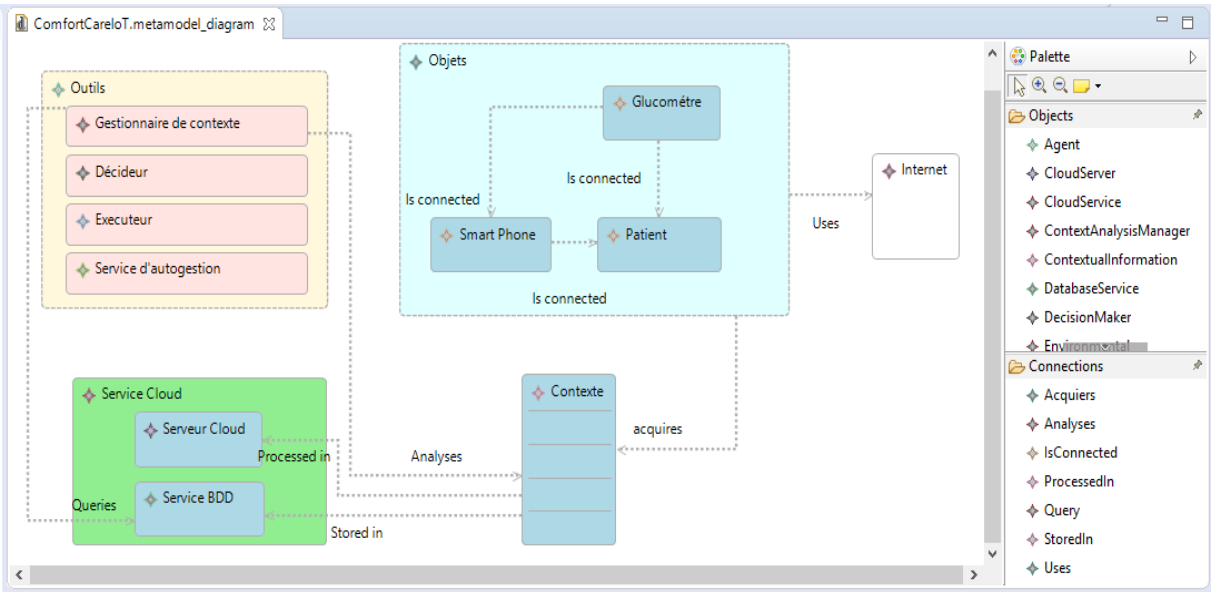

Figure 24 - Plugin Eclipse du modèle ComfortCareIoT

Concernant le comportement ou le fonctionnement du processus décrit par l'exemple concret de la panne arrivée au téléphone portable du patient, la stratégie de maintenance est mise en évidence dans le diagramme de séquence suivant. Ce type de reconfiguration déclenche plusieurs autres actions (addition, suppression) qui constituent un scénario efficace.

En réponse à la défaillance de l'objet Smart Phone, le gestionnaire de reconfiguration invoque la stratégie de mise à jour (Figure 25). Il consiste à supprimer temporairement l'objet défectueux pendant sa réparation et à le remplacer par un objet de sauvegarde afin de préserver la stabilité du système et sa transparence pour l'utilisateur final.

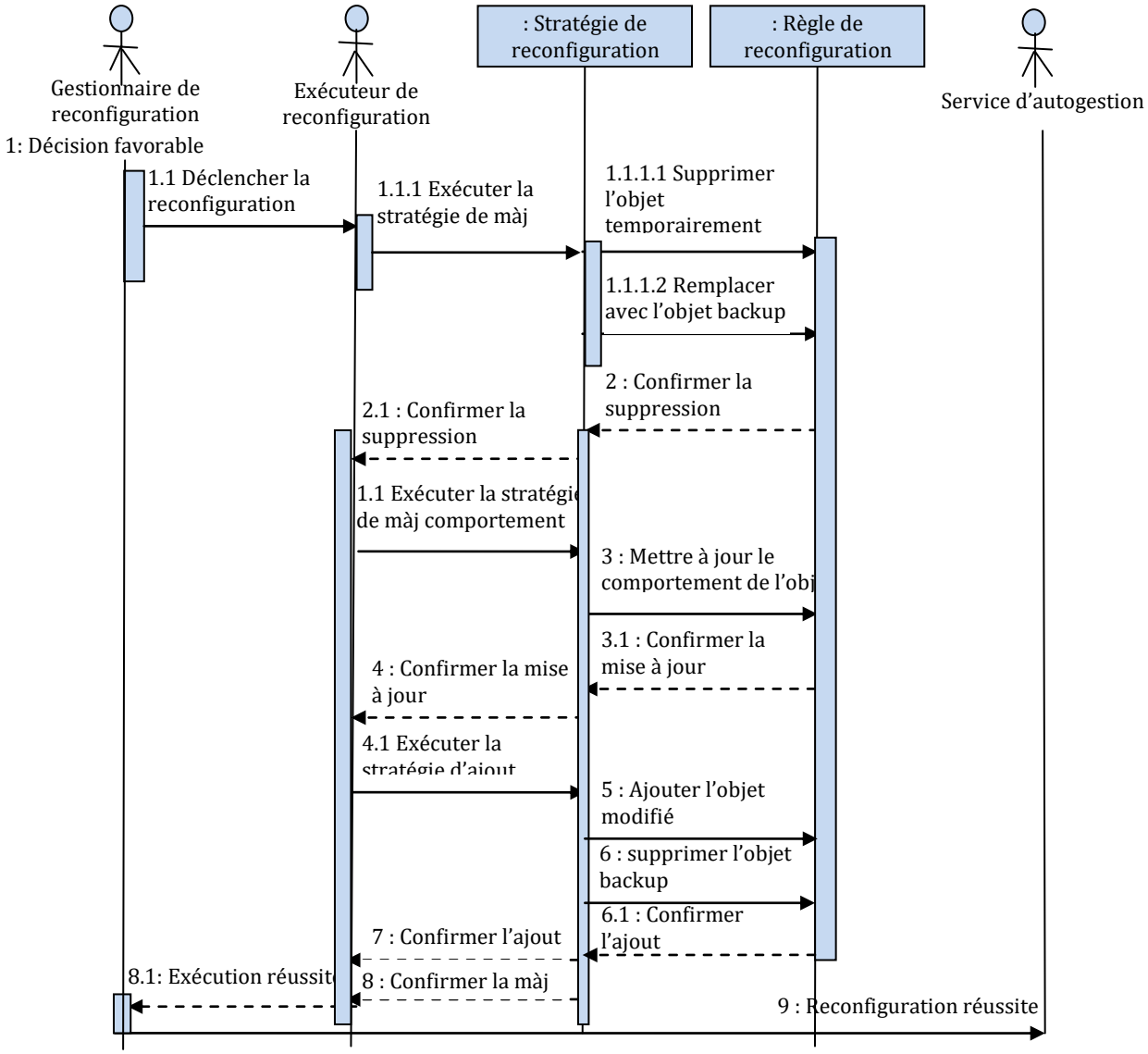

Figure 25 Diagramme de séquence expliquant l'action de maintenance d'un objet (Hakim, Amirat et al. 2018)

Pour capturer le savoir faire concernant la stratégie de maintenance d'un objet, nous recommandons l'utilisation d'un style d'évolution correspondant à la stratégie de reconfiguration « Mise à jour du comportement d'un objet ». Le style est comme expliqué dans la (section 4.4.1) est composé d'un nom, d'un entête et d'un ensemble de compétences. Nous illustrons dans la (Figure 26) le style d'évolution correspondant au scénario de l'étude de cas proposé.

Mise à jour comportementale de l'objet Smartphone
Contexte: Utilisateur, Environnemental, Temporel, machine Pré: Le smart phone ne reçoit plus de notifications Post: Le smart phone est reconnecté au système
• Créer un objet backup pour remplacer le smart phone • Supprimer l'objet smart Phone • Remplacer le smart phone par la smart Watch (l'objet backup) • Réparer l'objet défectueux • Supprimer l'objet backup • Reconnecter l'objet Smart phone au système • Vérifier les propriétés non fonctionnelles du système

Figure 26 Style d'évolution « Maintenance d'un objet »

5.4. Simulation

Une validation du scénario proposé est effectuée dans un environnement simulé avant de procéder au développement réel. Une maison intelligente est mise en œuvre (présentée à la Figure 27) à l'aide du logiciel de simulation Cisco Packet Tracer, qui comprend différents objets intelligents utilisés pour la domotique, tels qu'un ventilateur intelligent, une fenêtre intelligente, une porte intelligente, une lumière intelligente, un extincteur d'incendie, un extincteur de gazon, etc.

Cisco Packet Tracer est un outil très puissant qui permet d'émuler une topologie de réseau. Il ne nécessite pas de matériel coûteux ni des heures de câblage. Packet Tracer simule le fonctionnement d'un réseau en échangeant et en visualisant des trames Ethernet. Deux modes de simulation sont proposés par l'outil suscité.

– Simulation en temps réel: permet de visualiser immédiatement toutes les séquences qui se déroulent en temps réel.

– Simulation détaillée: permet de visualiser les séquences au ralenti entre deux ou plusieurs équipements.

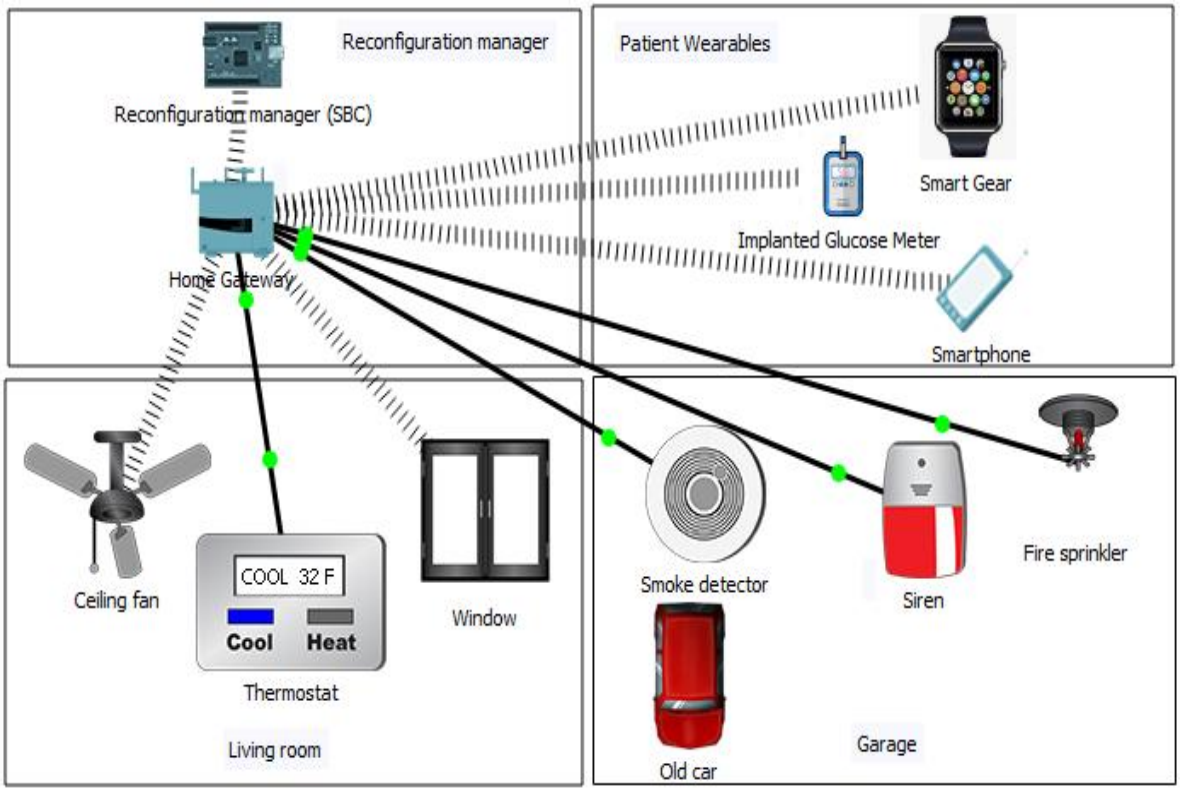

Figure 27 - Simulation de la maison intelligente en utilisant Cisco Packet Tracer 7.1

5.4.1. Dispositifs utilisés dans la maison intelligente

Une variété de périphériques et de capteurs sont installés à l'intérieur de la maison intelligente, à l'aide de connexions sans fil et d'un mode DHCP permettant d'attribuer des adresses IP par la passerelle domestique. Ces dispositifs sont: une passerelle domestique utilisée pour enregistrer des objets intelligents et attribuer des adresses IP, un ventilateur de plafond pour ventiler l'environnement domestique en fonction de certaines conditions, une fenêtre, un thermostat, une vieille voiture, un extincteur automatique qui rejette de l'eau détecté par un détecteur de fumée, une sirène émettant un son en cas de danger et un téléphone intelligent appartenant à la personne lui permettant de contrôler les objets intelligents qui y sont enregistrés. Le patient dispose également d'un engrenage intelligent et d'un glucomètre implanté.

Tableau 2 Dispositifs utilisés dans la maison intelligente

Dispositif	Utilité
Home gateway	Utilisé pour enregistrer des objets intelligents et leurs attribuer des adresses IP
ventilateur de plafond (seiling fan)	Utilisé pour ventiler l'environnement de la maison en fonction de certaines conditions
Fenêtre (window)	Utilisé pour contrôler la fenêtre à distance
Thermostat	Utilisé pour détecter la température de la maison
Ancienne voiture (old car)	Utilisé pour simuler différents scénarios dans la conception d'une maison, dans la mesure où il affecte, le niveau de CO, de CO2 et de fumée
Extincteur d'incendie (fire sprinkler)	Rejette de l'eau lorsque les effets d'un incendie ont été détectés
Détecteur de fumée (smoke detector)	Utilisé pour détecter le niveau de fumée
Sirène (siren)	Fournir du son pour certains événements à la maison
Smartphone	Pour contrôler les objets intelligents enregistrés et fournir différentes fonctionnalités de serveur
Montre intelligente (Smart gear)	Utilisé pour recevoir des notifications à la place du Smartphone
Glucomètre implanté	Utilisé pour surveiller le niveau glycémique du patient
Ordinateur à carte unique (Single Board Computer SBC)	Simule un Raspberry Pi, le gestionnaire de reconfiguration doit être installé dans le SBC

Le scénario proposé invoque de nombreuses actions de reconfiguration, notamment l'ajout d'objets, la modification de la structure et du comportement des objets. Voici des exemples d'actions mentionnées:

•La sirène et l'extincteur d'incendie sont activés une fois que la fumée est détectée.

•La fenêtre s'ouvre ou se ferme en fonction de la température détectée par le thermostat.

•Le ventilateur de plafond s'allume si la température dépasse les 15 ° C.

•Le téléphone intelligent doit être remplacé par son équipement intelligent.

Le gestionnaire de reconfiguration est installé à l'intérieur de l'objet ordinateur mono-carte (simule un Rasperry Pi). Pour mettre en œuvre la stratégie de maintenance, nous avons travaillé sur la réalisation du scénario susmentionné, c'est-à-dire que le gestionnaire de reconfiguration devait envoyer une notification au Smartphone du patient sous forme de tâche planifiée écrite en Java.

```
	public static void Send_notification () {
		Calendar calendar = Calendar.getInstance() ;
		calendar.set (Calendar.HOUR_OF_DAY, 21);
		calendar. set (Calendar.MINUTE, 00);
		calendar.set (Calendar.SECOND, 00);
		Date time = calendar.getTime() ;
		Timer timer = new Timer();
		timer.schedule(new MaTask(), time);
}
```

Figure 28 - Mèthode Java pour notifier le patient

Toutefois, si le téléphone intelligent est défectueux, ce dernier sera remplacé par l'objet de sauvegarde (la montre intelligente) pour que le système fonctionne.

```
	public static void main (String args [] ) {

		if (!SmartPhone.Send_notification()){
		System.out.println("It seems that an error occurred on your mobile,
please contact the maintenance department");
		SmartGear.Send_notification();
		}
	}
```

Figure 29 - Invocation de la méthode "Send_notification"

Au lieu d'écrire des programmes, une autre solution consiste à utiliser l'assistant Google pour assurer une bonne communication entre le système et le patient par le biais d'une saisie au clavier ou d'une voix naturelle. L'assistant virtuel lancé par Google peut rechercher sur Internet, planifier des événements et des alarmes, régler les paramètres matériels du périphérique de l'utilisateur et afficher des informations provenant de son compte Google. L'assistant Google s'étend à peu près à tous les appareils Android modernes. Il est également disponible sur les appareils Wear OS, Android TV et Nvidia Shield ainsi qu'Android Auto.

Le gestionnaire de reconfiguration doit être associé à Google Assistant, qui est intégré de manière distribuée dans les appareils intelligents des utilisateurs, conformément aux fonctionnalités IoT.

En ce qui concerne l'interface, l'assistant de Google sera utilisé tel qu'il est. L'utilisateur est libre de l'utiliser via la voix naturelle ou via une entrée au clavier.

L'utilisation de Google Assistant en collaboration avec le gestionnaire de reconfiguration renforcera l'interactivité du système en facilitant la communication entre l'utilisateur et le système. Cela rendra également efficace le processus de collecte des données relatives à l'utilisateur par rapport à son compte Google.

5.5. Conclusion

Dans ce chapitre, nous avons expliqué le processus de reconfiguration dynamique contextuelle des architectures logicielles des systèmes à objets connectés. Nous avons présenté l'architecture qui implémente ce processus, sa conception, sa structure et le fonctionnement des différents outils qu'elle contient.

Nous avons ainsi évalué le processus contre une étude de cas que nous avons proposé dans le domaine de santé électronique. Pour vérifier l'efficacité du processus proposé, nous avons simulé le scénario de l'étude de cas dans le simulateur réseau Cisco Packet Tracer.

Les résultats de la simulation ont renforcé nos arguments derrière l'utilité du processus de reconfiguration que nous avons proposé dans cette thèse.

Conclusion et perspectives

1. Bilan

Dans cette thèse, nous avons circonscrit le sujet de l'évolution des architectures logicielles des systèmes à objets connectés et plus spécifiquement leurs reconfigurations. L'objectif principal étant de proposer une solution aux problèmes que rencontre un système à objet connecté en réagissant aux changements de contexte.

L'émergence de l'Internet des objets dans les différents secteurs de la vie quotidienne améliore la façon dont l'homme interagit avec son environnement. Le but ultime est de créer un monde meilleur pour les êtres humains, où les objets qui nous entourent savent ce que nous aimons, ce que nous voulons et ce dont nous avons besoin et agissent ainsi sans instructions explicites.

Nous avons présenté une approche de reconfiguration dynamique des systèmes logiciels à base d'objets connectés. L'objectif principal derrière cette approche est de permettre à un système IoT de continuer de fonctionner en s'évoluant.

L'approche que nous avons proposée consiste en la proposition d'un modèle d'architecture générique pour les systèmes à objets connectés. Ensuite, la proposition d'un processus qui va être implémenté par ce modèle afin de permettre une reconfiguration contextuelle dynamique.

Le modèle est décrit à l'aide d'une architecture réflexive à quatre couches, comprenant une couche contextuelle, une couche d'objets et de réseau, une couche de reconfiguration dynamique et une couche de contrôle automatique.

La couche contextuelle permet d'acquérir les données contextuelles brutes à travers les capteurs qu'elle contienne, elle représente donc le contexte d'exécution des objets connectés. La couche d'objets et de réseau représente la couche métier du système. Ce dernier est basé sur un ensemble d'objets connectés reliés entre eux par des liaisons et des technologies réseaux le plus souvent sans fils. La couche de reconfiguration gère les opérations de reconfiguration à travers quatre outils principaux, un gestionnaire de reconfiguration, un analyseur de données de contexte, un décideur et un exécuteur. La couche de stockage et de d'autogestion est responsable du stockage et du traitement des

données de contexte. Cette couche fournit ainsi un service d'autogestion qui assure la cohérence de l'application après l'exécution de la reconfiguration.

La variabilité de contexte est le principal responsable du déclenchement des actions de reconfiguration sur les entités système. A l'exécution, les systèmes IoT reçoivent des notifications relatives à un changement de contexte inattendu. La réaction contre ces notifications s'effectue de manière transparente et dynamique, c'est-à-dire sans nécessiter l'arrêt du système tout entier. Les actions de reconfiguration incluent les ajouts et suppressions d'objets; modifier la structure ou le comportement d'un objet; établir des connexions et des liens entre les objets systèmes; maintien des propriétés non fonctionnelles du système telles que (qualité de service, précision, flexibilité…).

Le processus de reconfiguration implémenté par l'architecture proposée est inspiré de la boucle de l'informatique autonome; Il comprend quatre étapes: acquisition du contexte, raisonnement, planification et exécution. Après avoir effectué une reconfiguration, il est important de vérifier les propriétés non fonctionnelles du système en question pour s'assurer qu'il conserve toujours le même niveau d'efficacité et de précision.

Pour valider l'approche proposée, une étude de cas dans le domaine de santé électronique a été proposée. Le modèle correspondant à été réalisé en utilisant le framework de modélisation d'Eclipse (EMF) et son instanciation à travers l'outil Emfatic d'Eugenia sous Eclipse. Une simulation est réalisée à l'aide du simulateur Cisco Packet Tracer associé au langage Java pour évaluer le système avant de procéder au développement réel. L'assistant Google est ainsi utilisé par l'utilisateur pour intégrer et gérer la maison proposée.

Pour conclure, nous devons admettre que les résultats de cette étude doivent être considérés à la lumière de certaines limitations sur le plan de modélisation ainsi que sur le plan de réalisation. Au meilleur de notre connaissance, le modèle que nous avons proposé manque des mécanismes permettant d'assurer la délivrance des résultats en temps réel. Nous devons aussi avoir recours à l'aspect sécurité qui est un point très important dans ce genre de systèmes; le partage d'information exige un certain niveau de sécurité, les utilisateurs finaux doivent bénéficier d'une vie privée confidente et sécurisée.

2. Perspectives

Le travail de cette thèse a permis la proposition d'un modèle pour l'évolution des architectures logicielles des systèmes à objets connectés. Il consiste en une architecture

implémentant un processus de reconfiguration dynamique contextuelle des systèmes IoT. Néanmoins, cette thèse peut s'étendre pour assurer d'autres aspects des systèmes à objets connectés. Nous citons dans ce qui suit quelques voies d'exploration ouvertes pour enrichir notre approche.

Reconfiguration en temps réel: Nous visons à borner la réaction du système par un très court délai d'un ordre de millisecondes. Néanmoins, le système ne doit pas simplement délivrer des résultats dans des délais imposés, il doit les délivrer avec un haut nivaux d'exactitude.

Une meilleure exploitation du Cloud: Il est bien clair que l'utilisation du Cloud revient par de nombreux bénéfices au sujet de la consommation des ressources des objets tels que la batterie, l'espace de stockage et les performances des dispositifs. Cependant, le fait d'envoyer les données contextuelles sur le Cloud pour leurs traitements et stockage est couteux en termes de temps. La mise en ouvre des techniques temps réel sera alors requise dans ce cas pour respecter les délais et l'aspect temporel.

Développement d'autres styles d'évolution: Les savoirs faire que nous capturons a chaque opération de reconfiguration sont sauvegardés (par analogie avec le concept de Components Off the Shelf (COTS) pour pouvoir les réutiliser dans des situations similaires, nous visons a enrichir la bibliothèque de styles pour prendre en charge toutes les situations de changements de contexte pouvant influencer un ou plusieurs objets connectés.

Maintien des propriétés non fonctionnelles: Dans notre proposition, nous avons insisté sur la vérification des propriétés fonctionnelles de l'application après son évolution. Cependant, nous nous somme concentré seulement sur la cohérence de l'application et plus exactement sur la qualité des services qu'offre chaque objet. Nous visons à donner plus d'importance à l'aspect sécurité et confidentialité. D'un coté, les données transmises vers le Cloud doivent être sécurisée pour éviter la perte de trames de données. De l'autre coté, la vie privée des utilisateurs doit être respectée, nous devons pour cela gérer la visibilité des propriétés des objets dans le réseau.

Publications

Publications dans des journaux internationaux (avec comité de lecture)

1. Hakim, A., Amirat, A., & Oussalah, M. C. (2018). Non-intrusive contextual dynamic reconfiguration of ambient intelligent IoT systems. Journal of Ambient Intelligence and Humanized Computing, 1-12.

Conférences internationales (avec comité de lecture)

1. Hakim, A., Abdelkrim, A., & Oussalah, M. C. (2017) "Towards a dynamic reconfiguration approach of Internet of Things systems", The 2nd International Conference on Pattern Analysis and Intelligent Systems (PAIS'2016), Khenchela, Algeria, November 2016.

2. Hakim, A., Abdelkrim, A., & Oussalah, M. C. (2017). Towards a dynamic reconfiguration approach of Internet of Things systems. International Conference on Internet of Things, Data and Cloud Computing, ACM - International Conference Proceedings Series. Cambridge, United Kingdom.

Références bibliographiques

1219-1993, S. M. I. S. (1993). "IEEE Standard for Software Maintenance." IEEE Std 1219-1993 1-45.

Abdmeziem, M. R., D. Tandjaoui and I. Romdhani (2016). Architecting the internet of things: state of the art. Robots and Sensor Clouds, Springer: 55-75.

Abowd, G. D., C. G. Atkeson, J. Hong, S. Long, R. Kooper and M. Pinkerton (1997). "Cyberguide: A mobile context-aware tour guide." Wireless networks 3(5): 421-433.

Acosta Padilla, F. J., F. Weis and J. Bourcier (2014). Towards a model@ runtime middleware for cyber physical systems. Proceedings of the 9th Workshop on Middleware for Next Generation Internet Computing, ACM.

Aissaoui, O., F. Atil and A. Amirat (2013). Towards a Generic Reconfigurable Framework for Self-adaptation of Distributed Component-Based Application. Modeling Approaches and Algorithms for Advanced Computer Applications, Springer: 399-408.

Al-Fuqaha, A., M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash (2015). "Internet of things: A survey on enabling technologies, protocols, and applications." IEEE Communications Surveys & Tutorials 17(4): 2347-2376.

Ashton, K. (2009). "That 'internet of things' thing." RFiD Journal 22(7): 97-114.

Asthana, A., M. Crauatts and P. Krzyzanowski (1994). An indoor wireless system for personalized shopping assistance. Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on, IEEE.

Atzori, L., A. Iera and G. Morabito (2010). "The internet of things: A survey." Computer networks 54(15): 2787-2805.

Baldauf, M., S. Dustdar and F. Rosenberg (2007). "A survey on context-aware systems." International Journal of Ad Hoc and Ubiquitous Computing 2(4): 263-277.

Bernardos, A. M., P. Tarrio and J. R. Casar (2008). A data fusion framework for context-aware mobile services. Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on, IEEE.

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

Bikakis, A., T. Patkos, G. Antoniou and D. Plexousakis (2007). A survey of semantics-based approaches for context reasoning in ambient intelligence. European Conference on Ambient Intelligence, Springer.

Billet, B. (2015). Système de gestion de flux pour l'Internet des objets intelligents, Université de Versailles-Saint Quentin en Yvelines.

Brown, P. J., J. D. Bovey and X. Chen (1997). "Context-aware applications: from the laboratory to the marketplace." IEEE personal communications 4(5): 58-64.

Chapin, N., J. E. Hale, K. M. Khan, J. F. Ramil and W. G. Tan (2001). "Types of software evolution and software maintenance." Journal of software maintenance and evolution: Research and Practice 13(1): 3-30.

Chefrour, D. (2005). Plate-forme de composants logiciels pour la coordination des adaptations multiples en environnement dynamique, Université Rennes 1.

Chen, S., H. Xu, D. Liu, B. Hu and H. Wang (2014). "A vision of IoT: Applications, challenges, and opportunities with china perspective." IEEE Internet of Things journal 1(4): 349-359.

Commission, E. (2008). "Internet of things in 2020 road map for the future,." Working Group RFID of the ETP EPOSS, Tech. Rep.

Conan, D., R. Rouvoy and L. Seinturier (2007). Scalable processing of context information with COSMOS. IFIP International Conference on Distributed Applications and Interoperable Systems, Springer.

da, k. (2014). Kalimucho-A : Autonomic Knowledge-based context-driven adaptation platform. France, UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR. PhD.

Darianian, M. and M. P. Michael (2008). Smart home mobile RFID-based Internet-of-Things systems and services. Advanced Computer Theory and Engineering, 2008. ICACTE'08. International Conference on, IEEE.

Dey, A. K. (1998). Context-aware computing: The CyberDesk project. Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments.

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

Dey, A. K. and G. D. Abowd (2000). "Providing architectural support for building context-aware applications."

Dey, A. K., G. D. Abowd and A. Wood (1998). "CyberDesk: A framework for providing self-integrating context-aware services." Knowledge-based systems 11(1): 3-13.

Dey, A. K., D. Salber, G. D. Abowd and M. Futakawa (1999). The conference assistant: Combining context-awareness with wearable computing. Wearable Computers, 1999. Digest of Papers. The Third International Symposium on, IEEE.

dictionary, w. t. f. (2015). re-. wiktionary le dictionnaire libre.

Dobson, S., S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt and F. Zambonelli (2006). "A survey of autonomic communications." ACM Transactions on Autonomous and Adaptive Systems (TAAS) 1(2): 223-259.

Dohr, A., R. Modre-Opsrian, M. Drobics, D. Hayn and G. Schreier (2010). The internet of things for ambient assisted living. Information technology: new generations (ITNG), 2010 seventh international conference on, Ieee.

Evans, D. (2011). "L'Internet des objets. Comment l''evolution actuelle d'Internet transforme-t-elle le monde ?" Livre Blanc, Cisco IBSG, USA.

Ferry, N., S. Lavirotte, G. Rey and J.-Y. Tigli (2008). "Adaptation Dynamique d'Applications au Contexte en Informatique Ambiante." Rapport Technique, Laboratoire I3S (Universite de Nice Sophia Antipolis/CNRS), numero I3S/RR-2008-20-FR, Sophia Antipolis, France.

FIPS, P. (1984). "183: Integration definition for Function Modeling (IDEF0)." Federal Information Processing Standards, United States National Institute of Standards and Technology (NIST).

Floch, J., S. Hallsteinsen, E. Stav, F. Eliassen, K. Lund and E. Gjorven (2006). "Using architecture models for runtime adaptability." IEEE software 23(2): 62-70.

Flügel, C. and V. Gehrmann (2008). Scientific Workshop 4: Intelligent objects for the Internet of Things: Internet of Things–application of sensor networks in logistics. European Conference on Ambient Intelligence, Springer.

Gandrille, E. (2014). Adaptation autonomique d'applications pervasives dirigée par les architectures, Université de Grenoble.

Garlan, D., S.-W. Cheng, A.-C. Huang, B. Schmerl and P. Steenkiste (2004). "Rainbow: Architecture-based self-adaptation with reusable infrastructure." Computer 37(10): 46-54.

Garlan, D. and M. Shaw (1993). An introduction to software architecture. Advances in software engineering and knowledge engineering, World Scientific**:** 1-39.

Garlan, D., D. P. Siewiorek, A. Smailagic and P. Steenkiste (2002). "Project aura: Toward distraction-free pervasive computing." IEEE Pervasive computing 1(2): 22-31.

Gluhak, A. and W. Schott (2007). A wsn system architecture to capture context information for beyond 3g communication systems. Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on, IEEE.

Gu, T., H. K. Pung and D. Q. Zhang (2005). "A service-oriented middleware for building context-aware services." Journal of Network and computer applications 28(1): 1-18.

Gubbi, J., R. Buyya, S. Marusic and M. Palaniswami (2013). "Internet of Things (IoT): A vision, architectural elements, and future directions." Future generation computer systems 29(7): 1645-1660.

Guillaume Plouin, N. C. (2011) "Modèles d'architectures de l'Internet des Objets."

Guo, B., L. Sun and D. Zhang (2010). The architecture design of a cross-domain context management system. Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on, IEEE.

Guo, B., Z. Yu, X. Zhou and D. Zhang (2012). Opportunistic IoT: Exploring the social side of the internet of things. Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on, IEEE.

Guo, B., D. Zhang and Z. Wang (2011). Living with internet of things: The emergence of embedded intelligence. 2011 IEEE International Conferences on Internet of Things, and Cyber, Physical and Social Computing, IEEE.

Guthikonda, R. T., S. S. Chitta, S. Tekawade and T. Attavar (2014). "Comparative analysis of IoT architectures." TLEN 5710 Capstone, Tech. Rep.

Hakim, A., A. Amirat and M. C. Oussalah (2018). "Non-intrusive contextual dynamic reconfiguration of ambient intelligent IoT systems." Journal of Ambient Intelligence and Humanized Computing (1-12).

Henricksen, K. and J. Indulska (2004). A software engineering framework for context-aware pervasive computing. Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on, IEEE.

Hu, G., W. P. Tay and Y. Wen (2012). "Cloud robotics: architecture, challenges and applications." IEEE network 26(3).

Hu, P., J. Indulska and R. Robinson (2008). An autonomic context management system for pervasive computing. Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on, IEEE.

Jain, R. (2015). Internet of Things: Challenges and Issues, IEEE CS Keynote at 20th Annual Conference on Advanced Computing and Communications.

Jara, A. J., F. J. Belchi, A. F. Alcolea, J. Santa, M. A. Zamora-Izquierdo and A. F. Gómez-Skarmeta (2010). A Pharmaceutical Intelligent Information System to detect allergies and Adverse Drugs Reactions based on internet of things. Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on, IEEE.

Kakousis, K., N. Paspallis and G. A. Papadopoulos (2010). "A survey of software adaptation in mobile and ubiquitous computing." Enterprise Information Systems 4(4): 355-389.

Ketfi, A. and N. Belkhatir (2004). A metamodel-based approach for the dynamic reconfiguration of component-based software. Software Reuse: Methods, Techniques, and Tools, Springer**:** 264-273.

Ketfi, A. and N. Belkhatir (2004). A metamodel-based approach for the dynamic reconfiguration of component-based software. International Conference on Software Reuse, Springer.

Khan, R., S. U. Khan, R. Zaheer and S. Khan (2012). Future internet: the internet of things architecture, possible applications and key challenges. Frontiers of Information Technology (FIT), 2012 10th International Conference on, IEEE.

Kortuem, G., F. Kawsar, D. Fitton and V. Sundramoorthy (2010). "Smart objects as building blocks for the internet of things." Internet Computing, IEEE 14(1): 44-51.

Larousse, P. (1865). Grand dictionnaire universel du XIXe siècle: Français, historique, géographique, mythologique, bibliographique, littéraire, artistique, scientifique, etc., etc, Larousse & Boyer.

Le Goaer, O. (2009). Styles d'évolution dans les architectures logicielles, Université de Nantes; Ecole Centrale de Nantes (ECN)(ECN)(ECN)(ECN).

Le Goaer, O., M. Oussalah, D. Tamzalit and A.-D. Seriai (2007). Evolution dirigée par les styles. Atelier RIMEL (Rétro-Ingénierie, Maintenance et Evolution des Logiciels).

Lee, I. and K. Lee (2015). "The Internet of Things (IoT): Applications, investments, and challenges for enterprises." Business Horizons 58(4): 431-440.

Léger, M. (2009). Fiabilité des reconfigurations dynamiques dans les architectures à composants, École Nationale Supérieure des Mines de Paris.

Lehman, M. M., J. F. Ramil, P. D. Wernick, D. E. Perry and W. M. Turski (1997). Metrics and laws of software evolution-the nineties view. Proceedings Fourth International Software Metrics Symposium, IEEE.

Liu, J., X. Li, X. Chen, Y. Zhen and L. Zeng (2011). Applications of Internet of Things on smart grid in China. Advanced Communication Technology (ICACT), 2011 13th International Conference on, IEEE.

Miorandi, D., S. Sicari, F. De Pellegrini and I. Chlamtac (2012). "Internet of things: Vision, applications and research challenges." Ad hoc networks 10(7): 1497-1516.

Monroe, R. T., D. Kompanek, R. Melton and D. Garlan (1997). "Stylized architecture, design patterns, and objects." IEEE software 14(1): 43-52.

Nurmi, P. and P. Floréen (2004). "Reasoning in context-aware systems." Helsinki Institute for Information Technology, Position paper.

Perera, C., A. Zaslavsky, P. Christen and D. Georgakopoulos (2014). "Context aware computing for the internet of things: A survey." IEEE Communications Surveys & Tutorials 16(1): 414-454.

Pirotte, J. (juin 2014). "Les quatre défis à surmonter pour l'Internet des Objets." from http://www.objetconnecte.net/les-quatre-defis-surmonter-l-internet-des-objets/.

Ray, P. P. (2018). "A survey on Internet of Things architectures." Journal of King Saud University-Computer and Information Sciences 30(3): 291-319.

Rouvoy, R., P. Barone, Y. Ding, F. Eliassen, S. Hallsteinsen, J. Lorenzo, A. Mamelli and U. Scholz (2009). Music: Middleware support for self-adaptation in ubiquitous and service-oriented environments. Software engineering for self-adaptive systems, Springer**:** 164-182.

Schaffers, H., N. Komninos, M. Pallot, B. Trousse, M. Nilsson and A. Oliveira (2011). Smart cities and the future internet: Towards cooperation frameworks for open innovation. The future internet assembly, Springer.

Schilit, B., N. Adams and R. Want (1994). Context-aware computing applications. Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on, IEEE.

Schmidt, A., K. A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven and W. Van de Velde (1999). Advanced interaction in context. International Symposium on Handheld and Ubiquitous Computing, Springer.

Scholze, S. and J. Barata (2016). Context awareness for flexible manufacturing systems using cyber physical approaches. Doctoral Conference on Computing, Electrical and Industrial Systems, Springer.

Sicilia, M., J.-J. Cuadrado, E. García, D. Rodríguez and J. R. Hilera (2005). The evaluation of ontological representation of the SWEBOK as a revision tool. 29th Annual International Computer Software and Application Conference (COMPSAC), Edinburgh, UK.

Singh, R. (1996). "International Standard ISO/IEC 12207 software life cycle processes." Software Process: Improvement and Practice 2(1): 35-50.

Sundmaeker, H., P. Guillemin, P. Friess and S. Woelfflé (2010). "Vision and challenges for realising the Internet of Things." Cluster of European Research Projects on the Internet of Things, European Commision 3(3): 34-36.

Taconet, C. (2009). Sensibilité au contexte Modélisation et adaptation, Département INF université de Paris sud.

Tan, L. and N. Wang (2010). Future internet: The internet of things. Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on, IEEE.

Vazquez, J. I. and D. Lopez-De-Ipina (2008). Social devices: autonomous artifacts that communicate on the internet. The Internet of Things, Springer**:** 308-324.

Vermesan, O., P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. S. Jubert, M. Mazura, M. Harrison and M. Eisenhauer (2011). "Internet of things strategic research roadmap." Internet of Things-Global Technological and Societal Trends 1(2011): 9-52.

Want, R., A. Hopper, V. Falcao and J. Gibbons (1992). "The active badge location system." ACM Transactions on Information Systems (TOIS) 10(1): 91-102.

Weiser, M. (1991). "The computer for the twenty-first century (pp. 94-100)." Scientific American, September Issue.

Weiser, M. (1999). "The computer for the 21st century." Mobile Computing and Communications Review 3(3): 3-11.

Wu, M., T.-J. Lu, F.-Y. Ling, J. Sun and H.-Y. Du (2010). Research on the architecture of Internet of Things. Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on, IEEE.

Yan, B. and G. Huang (2008). Application of RFID and Internet of Things in Monitoring and Anti-counterfeiting for Products. Business and Information Management, 2008. ISBIM'08. International Seminar on, IEEE.

Yan, B. and G. Huang (2009). Supply chain information transmission based on RFID and internet of things. Computing, Communication, Control, and Management, 2009. CCCM 2009. ISECS International Colloquium on, IEEE.

Zhengxia, W. and X. Laisheng (2010). Modern logistics monitoring platform based on the Internet of things. Intelligent Computation Technology and Automation (ICICTA), 2010 International Conference on, IEEE.