الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
جامعة أبي بكر بلقايد تلمسان
Université Aboubakr Belkaïd– Tlemcen –
Faculté de TECHNOLOGIE

THESE

Présentée pour l'obtention du **grade** de **DOCTEUR EN SCIENCES**

En : Télécommunications

Spécialité : Télécommunications

Par : ABDELLAOUI Ghouthi

Sujet

La reconfiguration dynamique des systèmes pervasifs à base des systèmes multi-agents

Soutenue publiquement le 27 septembre 2018, devant le jury composé de :

MERIAH Sidi Mohammed	Professeur	Univ. Tlemcen	Président
BENDIMERAD Fethi Tarik	Professeur	Univ. Tlemcen	Directeur de thèse
MERAD Lotfi	Professeur	ESSA-Tlemcen	Examinateur
DEBBAT Fatima	Professeur	Univ.Mascara	Examinateur

Dédicaces

Je dédie ce modeste travail en guise d'amour, de respect et de reconnaissance :

À la mémoire de mon père que Dieu l'accueille dans son vaste paradis,
Qui m'a donné un magnifique modèle de labeur et de persévérance.
Qui a su m'entourer d'attentions, m'inculquer les valeurs nobles de la vie, m'apprendre le sens du travail, de l'honnêteté et de la responsabilité.

À ma chère mère, l'être le plus cher au monde. Que Dieu tout-puissant te protéger du mal, te procurer longue vie, santé et bonheur pour que je puisse te rendre un minimum de ce que je te dois.

À mes frères et sœurs, que je ne pourrais d'aucune manière exprimer ma profonde affection et mon immense gratitude pour tous les sacrifices consentis, votre aide et votre générosité extrêmes ont été pour moi une source de courage, de confiance et de patience.

À mes neveux et mes nièces, les anges de ma vie.

À tous mes amis, c'est une grande fierté pour moi d'être parmi vous. Et je vous remercie pour la confiance dont vous m'avez fait part.

À ceux que j'aime et m'aiment, à tous ceux qui me sont chers, ...

Ghouti

Remerciements

Louange a Dieu qui m'a permis, en cette heureuse occasion, d'exprimer ma reconnaissance et ma profonde gratitude à tous ceux et celles qui m'ont fait, qui m'ont formé et qui ont suivi patiemment mes pas jusqu'à la réalisation de cette étape, concrétisée par ce modeste travail.

A ceux-là, j'adresse mes remerciements les plus chaleureux pour les efforts qu'ils n'ont cessés de me prodiguer tout au long de ce périple sinueux et bien difficile.

Il s'agit, bien sûr de mes parents qui m'ont vu naître et grandir et qui, bien souvent, se sont privés pour satisfaire mes exigences et mon désir d'apprendre.

Il s'agit, surtout, de mon directeur de thèse **Monsieur Fethi Tarik BENDIMERAD** qui, malgré ses lourdes et bien nombreuses charges, n'a pas hésité à puiser de son précieux temps pour m'orienter dans mon projet de thèse.

Il s'agit, également du président de jury **Monsieur Sidi Mohammed MERIAH** pour l'intérêt qu'il a porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par ses propositions.

Il s'agit, encore, de remercier respectueusement les membres de mon jury d'examen **Monsieur Lotfi MERAD** et **Madame Fatima DEBBAT**, qui ont accepté d'examiner cet humble travail que j'espère à la hauteur de mes ambitions bien modestes.

Il s'agit, aussi ,de tous les membres des équipes de recherche du Laboratoire de Télécommunication de Tlemcen qui m'ont ouvert les portes pour me permettre l'accès à la connaissance quelle qu'elle soit et où qu'elle soit.

Je ne serai oublié, enfin, de mes enseignants qui m'ont inculqué les bases de la science et les principes de la vie.

الملخص

لقد أصبحت تكنولوجيا الأنظمة السائدة نموذجًا جديدًا للعقود القادمة في مجال المعلوماتية. حاليًا، نجد هذا النوع من الأنظمة في كل مكان في حياتنا اليومية، مما أدى إلى خلق بيئة معلوماتية في غاية الديناميكية، وبالتالي فإن التطبيق الذي يجب أن يدير هذا النوع من النظام يتوجب عليه أن يغير من سلوكه ليتكيف مع الشروط الجديدة. ولحل هذه المشكلة ، نقترح منهجًا جديدًا لإعادة الهيكلة ، مما يسمح بإعادة تشكيل الأنظمة السائدة في الوقت الفعلي.و يعتمد نموذجنا على منهجية العملاء، بحيث نفترض أن العملاء هم أصغر الكيانات المستقلة لأي نظام. و بهذا يمكننا تلخيص إعادة الهيكلة الديناميكية في طريقة إضافة أو إزالة عميل من النظام. يتيح التنظيم الجديد للنظام، تحقيق أهدافه بفعالية تامة من خلال دمج الخدمات الجديدة التي يقدمها العملاء الجدد، وضمان حسن سير النظام بشكل عام في ظل مختلف الظروف التي تفرضها البيئة. من أجل التحقق من هذا المنهاج ، تم اختيار LPWAN كمنصة اتصال لربط جميع الأجهزة السائدة ، وذلك بسبب تغطيتها الواسعة من حيث الشبكة.

منهجيتنا هي نتيجة العديد من الدراسات حول مختلف الرؤى لإعادة الهيكلة، المتواجدة في عدة مجالات كالبرمجيات، الأجهزة الإلكترونية والأنظمة الصناعية، بهدف إيجاد منهاج عام ينطبق على معظم الأنظمة المدروسة.

كلمات البحث: إعادة الهيكلة الديناميكية، الأنظمة السائدة، منهجية العملاء، LPWAN.

Abstract

Pervasive systems technology has become a new paradigm for the coming decades. Currently, we find this type of system everywhere in our daily life, while creating a very dynamic pervasive environment, therefore the application that must manage this kind of system have to change thier behavior to adapt to new situations. To solve this problem, a new approach of reconfiguration is proposed, allowinga real time reconfiguration of the pervasive systems. This new technique is based on the agent-oriented approach.Because agents are the smallest elemental and autonomous entities for any system. With this approach, dynamic reconfiguration is the automation of adding or removing an agent from the system. This configuration change, allows the system to achieve its objectives effectively by integrating new services offered by the new agents,it also helps to ensure the proper functioning of the system in different situations. In order to validate our approach, LPWANs have been chosen as a communication platform to interconnect all pervasive devices, due to its extensive network coverage.

Keywords: Dynamic Reconfiguration; Pervasive system; Multi-agent system; LP-WAN.

Résumé

La technologie des systèmes pervasifs est devenue un nouveau paradigme pour les prochaines décennies. Actuellement, nous trouvons ce type de système partout dans notre vie quotidienne, tout en créant un environnement pervasif très dynamique, par conséquent l'application qui gère ce genre de systèmes doit changer de comportement afin de s'adapter aux nouvelles situations. Pour résoudre ce problème, une nouvelle approche de reconfiguration est proposée, permettant la reconfiguration en temps réel des systèmes pervasifs. Cette nouvelle technique est basée sur l'approche orientée agent, en considérant que les agents sont les plus petites entités élémentaires et autonomes pour n'importe quel système. Par cette approche, la reconfiguration dynamique se résume à l'automatisation de l'ajout ou de la suppression d'un agent du système. Ce changement de configuration, permet au système d'atteindre ses objectifs efficacement par l'intégration des nouveaux services offerts par les nouveaux agents, il permet aussi d'assurer le bon fonctionnement du système dans les différentes situations. Afin de valider notre approche, les réseaux LPWANs ont été choisis comme plate-forme de communications pour interconnecter tous les périphériques pervasifs, en raison de sa grande couverture en matière de réseau.

Mots clés **:** Reconfiguration dynamique; Système pervasif; Système multi-agent; LPWAN.

Table des matières

Table des figures

Introduction générale

1 Introduction

L'évolution technologique à créé un nouveau monde dédié au traitement de l'information, où n'importe quel composant réel est caractérisé par deux ensembles (données et traitements). Ce dernier peut être vu comme un environnement global d'observation pour plusieurs systèmes dans les différents domaines comme le contrôle des équipements médicaux, l'assistance autonome, le contrôle et la sécurité du trafic, les systèmes avancés automobile, le contrôle du traitement, la conservation d'énergie, le contrôle de l'environnement, le contrôle des infrastructures critiques (l'énergie électrique, les système de communications par exemple: télé-présence, télé-médecine), les systèmes militaires, les systèmes de manufacturing, et toutes les structures intelligentes, etc. La majorité de ces systèmes sont critiques et varient d'une manière chaotique dans le temps, ce qui nécessite un changement continue de leur représentation dans le monde de l'information.

Ce changement peut être effectué pour plusieurs raisons (correction, adaptation, amélioration, évolution, etc.), ceci implique l'intégration ou la suppression des données d'un état du système et par conséquent nous allons obtenir un nouvel état pour ce dernier; ce passage d'un état à un autre donne un aspect dynamique au système et donc une bonne réflexion pour le système réel. Étant donné que chaque état du système est représenté par une configuration, et l'ensemble de configurations possibles pour un système représente un

espace de variation de ce dernier, par conséquent, le passage entre deux états peut être vu comme une reconfiguration du système.

Actuellement, le plus grand souci des développeurs est de faire adapter leurs applications aux différents changements qui peuvent être produits dans le monde réel. Ces changements peuvent impliquer une maintenance continue ou des mises à jour des applications. Ceci est parfois très difficile vue que les systèmes sont critiques et ne permettent pas un temps de maintenance ou de mise à jour ce qui complique la tâche pour les développeurs. Cependant, le processus classique de maintenance ne peut être appliqué à certaines catégories d'applications, où l'interruption complète du service ne peut être tolérée. Ces applications, qualifiées de "non-stop", doivent être adaptées d'une manière dynamique avec le minimum de perturbations. Dans ce contexte un nouveau domaine de recherche et d'innovation a été créé et qui s'en charge d'étudier l'interaction entre le monde physique et le monde de l'information, tel qu'un CPS (*Cyber Physical System*) qui fait l'intégration du traitement de l'information avec le traitement physique. Un autre exemple des systèmes de ce type nous avons les LPWAN (Low Power Wide Aerea Network) et les WSN (Wireless Sonsor Network). Mais, la grande difficulté pour ce type de systèmes est comment gérer tous les composants ou les terminaux qui peuvent être ajouté à n'importe quel moment et d'une manière automatique, dans ce cas nous devons avoir un processus d'intégration automatique de tous les nouveaux composants et prendre en considération les services offerts par les nouveaux arrivés, ce processus peut être vue comme une auto-configuration ou une reconfiguration dynamique (RD) du système.

Dans cette thèse, nous proposons une nouvelle approche pour effectuer la reconfiguration dynamique d'un système. Et qui se charge de superviser, contrôler et actualiser le système dynamiquement sans l'intervention de l'utilisateur humain. Notre solution peut être applicable à une grande variété de systèmes car elle contient les règles générale de la RD unifiée et simplifiée.

2 Problématique

Les périphériques pervasifs occupent notre environnement quotidien, commençant le matin par le réveil passant par la machine à café jusqu'à la voiture et le milieu de travail sans oublié les Smartphones, smart-TV, etc. Cette remarquable augmentation des systèmes, complique le processus d'intégration de tous ces périphériques où l'utilisateur doit à chaque fois installé manuellement et reconfiguré son système afin d'exploiter tous les équipements présents dans son environnement. Le nombre important des équipements et leur dynamicité exige que les systèmes pervasif doivent être de plus en plus autonomes et reconfigurables dynamiquement pour qu'ils peuvent prendre en considération tous ces périphériques et les rendre accessibles et exploitables efficacement par le simple utilisateur humain. Un des problèmes qui se posent pour la reconfiguration dynamique des systèmes pervasif est l'hétérogénéité des périphériques, ce qui complique l'implémentation de leur intégration automatique dans le système.

Actuellement, il n'existe aucun moyen pour modéliser un système *ouvert* qui prend en charge n'importe quelle configuration possible, vu que chaque système est conçu à la base d'un ensemble fini de configurations. Par conséquent, la reconfiguration sera limité contrairement à la réalité. Ceci influe négativement sur la fiabilité du système numérique.

Pour résoudre ce problème nous avons décomposé le système global en sous-systèmes, où chacun de ces derniers représente un module de composition pour la configuration globale.

Dans ce cas, une reconfiguration sera la différence entre deux configurations successives due à :

1. Un ajout d'un :
    - Module : Parfois le système fait appel à des nouveaux modules pour accomplir de nouvelles fonctionnalités au système.
    - Lien : La présence d'un nouveau module dans le système nécessite l'ajout d'un

ou plusieurs liens afin de garantir la communication du nouveau module avec le système, parfois il est nécessaire de créer de nouveaux lien entre des modules déjà existants dans le système, en vue de mieux les exploiter.

2. Une suppression d'un :

   - Module : La défaillance d'un module peut être la principale cause de sa suppression du système; ainsi la reconfiguration de ce dernier se produira afin de s'adapter à la nouvelle situation.
   - Lien : L'abolition d'un module implique forcément la suppression de ces liens voir la présence d'un lien superflu, sa suppression sera jugé utile.

3. Une modification : Peut être considérée à la fois comme ajout et/ou suppression d'un module ou d'un lien.

Sachant que l'opérande dans ces trois opérations c'est le module qui représente l'unité de base de composition d'un système selon notre vision, et par conséquent ce denier doit être autonome, indépendant, communiquant et réutilisable. À travers cette thèse, nous devons trouvé un moyen pour modéliser un module tout en garantissant ces caractéristiques. Aussi nous devons répondre aux deux questions cruciales pour effectuer la reconfiguration dynamique (Quand et Comment).

1. Quand : Cette question tente à déterminer à quel moment et pour quelle raison la reconfiguration doit se faire.

2. Comment : Cette question tente à déterminer la méthode ou la procédure à suivre pour faire une reconfiguration lors d'un ajout ou d'une suppression, cette procédure peut être vue comme un ensemble de règles pour la reconfiguration. À savoir, un scénario à suivre lors d'une reconfiguration.

La réponse à ces questions nécessite l'utilisation d'un moyen de modélisation et

de développement qui prend en charge toute sorte de modification dans n'importe quelle configuration.

3 Démarche

Afin de comprendre le phénomène de la reconfiguration dynamique, nous avons établi plusieurs études sur les différents systèmes qui peuvent introduire cette nouvelle notion comme une fonctionnalité de base dans leur fonctionnement, ces études ont comme objectif, déduire les différents exigences et contraintes imposés sur la RD, pour cela nous avons divisé l'ensemble des systèmes reconfigurables en trois grandes classes (software, hardware, industriel) et étudié chaque classe séparément et soustrait les spécifications et les outils utilisés pour effectuer la RD pour chaque classe.

L'ensemble des spécifications déduit de chaque classe de système sera l'objet de notre cahier de charge de ce projet de thèse que nous devons respecter pour pouvoir proposé une approche généraliste et applicable à un grand nombre de systèmes quel que soit sa nature.

4 Organisation de la thèse

Après une brève introduction, nous avons réalisé un état de l'art dont le premier chapitre présente une étude sur les systèmes perversif pour pouvoir comprendre de quoi s'agit-il?, et l'intérêt qu'ils portent dans notre vie quotidienne, puis dans le chapitre suivant, nous allons étudier la reconfiguration dynamique à travers plusieurs types de système et comprendre comment pouvons-nous l'appliquer pour améliorer la perception des systèmes pervasif de leur environnement. À la fin de l'état de l'art, et dans le chapitre trois, nous présentons les systèmes multi-agent comme une solution pour notre problématique, pour cela nous réalisons une étude sur ces derniers afin d'extraire les caractéristiques principales qui peuvent être utilisées pour effectuer la reconfiguration dynamique.

Dans le chapitre quatre, nous présentons notre approche pour la reconfiguration

dynamique et la démarche à suivre pour implémenter notre solution. Pour cela, nous choisissons le système pervasif des réseaux LPWANs comme exemple d'application, à la fin de ce chapitre nous présentons et discutons les résultats obtenus.

Et à la fin de cette thèse, nous synthétisons le travail réalisé en montrant l'importance de la reconfiguration dynamique dans les systèmes pervasif, puis nous citons quelques perspectives à ce projet de thèse considéré comme une étude de base pour plusieurs futurs projets.

Chapitre 1

Les systèmes pervasifs

Le développement des systèmes informatiques classiques "Desktop" a donné plusieurs visions pour ces systèmes à travers le temps. La dernière vision, nous guide vers des systèmes informatiques perversifs dont le but est de généraliser l'utilisation des systèmes informatiques et les émerger dans la vie quotidienne des utilisateurs. Dans ce chapitre nous étudierons les systèmes perversifs afin de les mieux comprendre et étudier l'intérêt de les reconfigurer dynamiquement dans le temps.

1.1 Introduction

Depuis la naissance des systèmes informatiques, plusieurs visions sont proposées et exploitées selon l'évolution technologique Hardware et software.

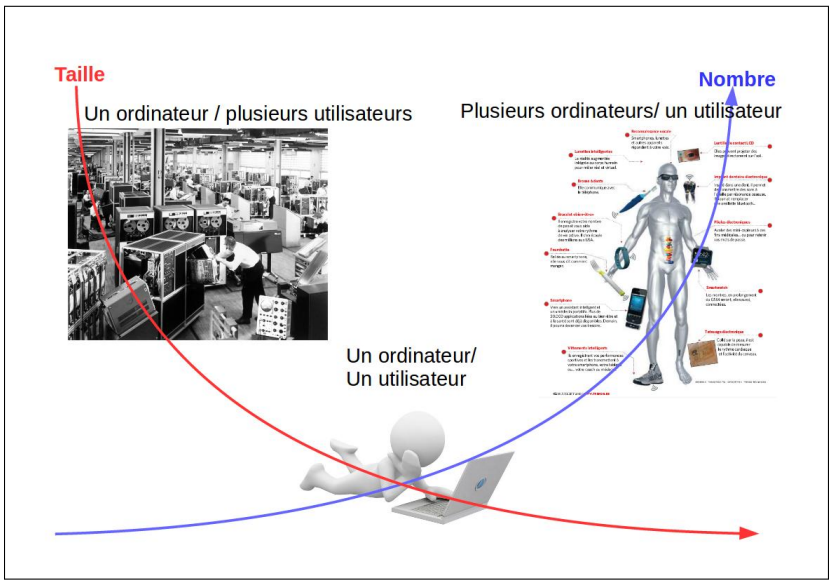

Figure 1.1: Évolution des ordinateurs

La figure 1.1 représente l'évolution des ordinateurs depuis les premiers mainframes développés en 1960, où le principe était une machine pour plusieurs utilisateurs, jusqu'aux les ordinateurs actuels qui se présentent sous forme de mini ou micro systèmes informatiques dédiés à une tâche bien précise où nous trouvons plusieurs machines de ce type pour la même personne.

En parallèle avec cette évolution Hardware une évolution software se fait développée, à partir des premiers systèmes informatiques classiques mono-tâche, passant par les systèmes distribués et les systèmes mobiles jusqu'à la nouvelle vision avec les systèmes pervasifs. La figure 1.2 illustre ce passage avec les nouvelles contraintes et spécifications pour chaque type de système.

Les systèmes pervasifs sont conçus à base des espaces intelligents dotés d'un ensemble de senseurs pour avoir une certaine perception à l'environnement, ceci permet aux utilisateurs une meilleurs exploitation de leur environnement en leurs offrant des nouvelles fonctionnalités, ce qui garantir d'une part, l'invisibilité du fonctionnement du système, et d'autre part une efficacité de la solution.

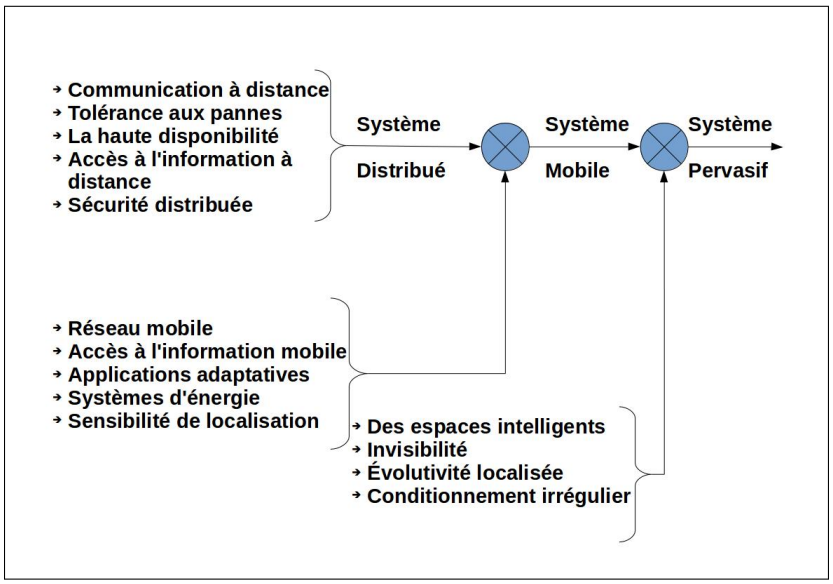

Figure 1.2: Évolution des systèmes informatiques

1.2 Présentation des systèmes pervasifs

En 1991, Mark Weiser présentait sa vision futuriste de l'informatique du 21ème siècle, c'est une vision qui a donné l'espoir au monde des machines de s'ouvrir sur le monde des humains. Initialement, cette vision était sous forme des systèmes ubiquitaires qui seront développés plus tard en systèmes pervasifs.

Mark Weiser a exprimé sa vision comme suit:

Définition *"A new way of thinking about computers in the world, one that takes into account the natural human environment and allows the computers themselves to vanish in the background"[3]*

En d'autre termes, il voulait créer un environnement doté d'une capacité de calcul et de communication, afin de développer des entités de traitement facilement utilisable et exploitable par le simple utilisateur, ce qui augmente l'efficacité des systèmes informatiques en les émergents dans la vie quotidienne des utilisateurs.

Pour arriver aux systèmes pervasifs actuels, la vision de Mark Weiser a été développée et évoluée à travers plusieurs étapes suivant les besoins du marché et les technologies

disponibles dans le temps. Chaque étape, dans cette évolution représente une classe de systèmes, que nous résumons dans les points suivants:

- Ubiquitaire : Accessible de n'importe où,
- Mobile : Qui intègre des terminaux mobiles
- Context-Aware: Qui prend en charge le contexte d'exécution,

Ces étapes sont considérées comme des sous-systèmes dans l'architecture générale d'un système pervasif, en lui offrant ces principales caractéristiques et propriétés. Chaque sous-système représente une classe indépendante de technologie informatique, qui a ses propres contraintes et objectifs, et son développement se fait d'une manière indépendante par rapport aux autres sous-systèmes. Sachant qu'un système pervasif est un système ubiquitaire, mobile et context-aware, son évolution dépend directement de l'évolution de ces sous-systèmes

La jonction de ses caractéristiques a élargie les champs d'application des systèmes pervasifs dans les activités journalières des individus. Ils sont présents à travers un ensemble de dispositifs intelligents et connectés entre eux dont le but est d'offrir un ensemble de services à l'utilisateur final. Ces services peuvent être exploités dans plusieurs domaines tels que:

- Le commerce : comme les codes à bar, QrCode ou les RFID, qui sont généralement utilisés dans les systèmes de production et chez les détaillants pour codifier leurs produits, dont le but est de simplifier l'opération de vente et les mises à jour lors d'un inventaire.
- L'industrie automobile : A l'aide d'un ensemble de systèmes embarqués invisibles tels que les freins pilotés par ordinateur, le régulateur de vitesse et les alarmes automatiques lorsque la route est glissante, etc.

- Les systèmes de santé : Il existe de nombreuses applications dans ce domaine tels que, les pansements intelligents qui permettent de savoir comment la blessure se produit ou des consultations vidéo avec des médecins, qui peuvent traiter les patients à la maison, les T-shirt intelligent, etc.

- Les systèmes de technologie de l'information : Généralement sont utilisés pour collecter et stocker l'information et la rendre accessible à ceux qui ont le droit, tels que les carte à puce, les documents biométrique et tous les dispositifs qui permettent de lire ce genre de documents.

Dans la section suivante, nous verrons comment ses caractéristiques sont structurées pour former un système pervasif ainsi que les particularités obtenues en les regroupant.

1.3 Architecture des systèmes pervasifs

Un système pervasif comme nous l'avons déjà vu dans la section précédente, est un système ubiquitaire, mobile et context-aware, c'est pour cette raison nous trouvons des modules de gestion pour chaque type de ces systèmes dans son architecture générale. La figure 1.3 montre comment ces modules sont regroupés et organisés afin de bénéficier de tous les avantages offerts par ces derniers.

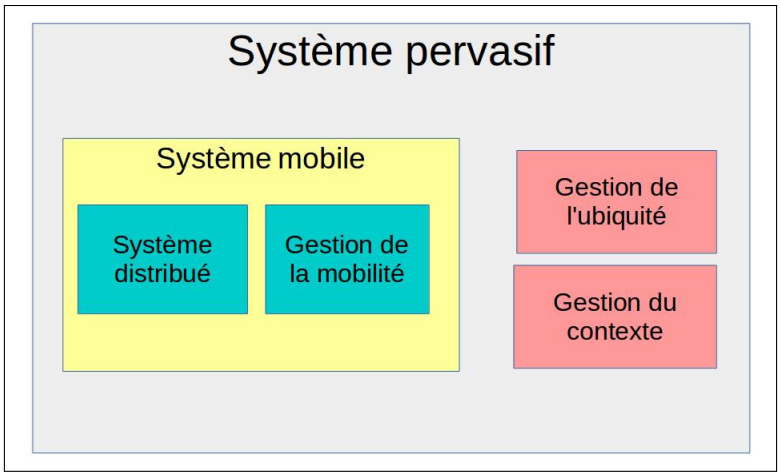

Figure 1.3: Les éléments d'un système pervasif [1]

Les modules de gestion intégrés dans l'architecture générale d'un système pervasif, serrent à assurer la coordination entre les sous-systèmes impliqués dans cette technologie et gérer leurs particularités d'une manière cohérente, vu que leur conception initiale n'avait pas comme objectif de coopérer avec d'autres systèmes. Dans les sous-sections suivantes nous allons étudier le rôle de chaque module dans cette architecture.

1.3.1 Gestion de l'ubiquité

Dérivé du latin *ubique* qui signifie partout. En informatique, le terme ubiquitaire désigne un environnement dans lequel les systèmes informatiques et réseaux sont "enfouis", "intégrés" et "omniprésents" dans le monde réel. Ceci est possible grâce à des milliers de dispositifs intelligents auto-communicants tels que les téléphones intelligents, les systèmes embarqués prenant un rôle actif dans chaque domaine des activités quotidienne de la vie humaine.

Pour pouvoir accomplir cette tâche, les systèmes pervasifs sont dotés d'un ensemble de capteurs intelligents miniatures, créant ainsi un environnement intelligent. Les étiquettes RFID permettent une identification et une intégration subtile d'objets physiques (par exemple un produit dans un magasin, ou des livres dans une bibliothèque) dans l'environnement informatique.

La gestion de l'ubiquité dans ce genre de système sert à adapter ces dispositifs aux besoins des utilisateurs en leur offrant l'ensemble de capteurs et effecteurs nécessaires pour pouvoir d'une part, les intégrer d'une manière fluide dans l'environnement réel, et d'autre part, simplifier leur utilisation à l'aide des interfaces homme-machine intelligentes qui présentent l'information dans sa forme naturelle et n'ont pas numérique (sons, hologrammes, etc.).

1.3.2 Gestion de la mobilité

L'apparition des ordinateurs portables et les réseaux locaux sans fil dans les années 1990 a crée des nouveaux domaines de recherche afin d'engendrer les nouvelles contraintes

pour les clients mobiles. Vu que les systèmes pervasifs cherchent à offrir des services et assurer la connectivité de ce genre de client, un module de gestion de mobilité devient un élément crucial dans l'architecture de base de ces systèmes.

La mobilité permet de garantir la communication en mouvement, grâce au développement des réseaux mobiles, tels que 2G/2.5G/3G/LTE/5G/WiFi, etc. Ceci joue un rôle très important pour garantir une des caractéristiques les plus indispensables dans les systèmes pervasifs "l'informatique n'importe où, n'importe quand".

Dans ce cadre, la gestion de mobilité d'un dispositif en mouvement dans un système pervasif lui permet de se connecter au réseau et partager les informations avec le reste des dispositifs, ceci offre aux utilisateurs une liberté de déplacement (client mobile) tout en restant connecté et rattaché aux différents services disponibles dans le réseau.

Cette gestion dépend évidemment de la technologie de déploiement du réseau, et elle bénéfice directement des avantages offerts par cette dernière, par exemple si le réseau sur lequel l'application a été développée est un réseau cellulaire (2G, 3G, LTE, etc.), nous pouvons avoir une large surface de mobilité puisque ce genre de réseau à une grande couverture surtout dans les zones urbaines, par contre dans certaines zones où nous avons une faible couverture, le déploiement d'un autre type de réseau est nécessaire tel que le WiFi à une grande portée ou LPWANs.

De plus, la gestion de la mobilité intègre d'autres contraintes qui peuvent affecter négativement la qualité du système. Parmi ces contraintes nous avons: la variation imprévisible de la qualité de réseau, le déclin de la confiance et la robustesse des éléments mobiles, la limitation des ressources locales imposées par les contraintes de poids et la taille du dispositif, ainsi que les soucis de la consommation de l'énergie des batteries.

La gestion de la mobilité est encore un domaine de recherche très actif et en évolution, dont les corpus de connaissances attendent d'être codifiés dans les manuels. Les résultats obtenus jusqu'à présent peuvent être regroupés dans les grands domaines suivants:

- Les réseaux mobiles : Les IP mobile (IPv6), les protocoles des réseaux ad-hoc, les techniques d'amélioration de l'utilisation du TCP dans les réseaux sans fil.

- L'accès à l'information mobile, y compris l'accès au fichier adapté en bande passante, et le contrôle sélectif de la cohérence des données.

- La prise en charge d'applications adaptatives, y compris le transcodage par proxy [4] et la gestion adaptative des ressources [5].

- Les techniques d'optimisation de la consommation d'énergie au niveau du système, telles que la planification du processeur à vitesse variable [6] et la gestion de la mémoire sensible à l'énergie [7].

- La sensibilité de l'emplacement, y compris la détection de l'emplacement (techniques de localisation par GPS, GSM, etc.) et le comportement du système en fonction de l'emplacement (par exemple choix automatique de la langue).

1.3.3 Gestion du contexte (Context-Aware)

Le développement de l'informatique sensible au contexte constitue l'un des développements majeurs de la vision de Weiser [3]. La sensibilité au contexte est définie comme étant la capacité d'un système à découvrir et à réagir à des changements dans l'environnement où il se trouve. Certaines recherches signalent également l'importance de l'adaptation du système à ces changements.

Pour certaines applications, le contexte est défini comme un élément de l'environnement de l'utilisateur que l'ordinateur de l'utilisateur connaît. D'autres définissent le contexte comme un sous-ensemble des états physiques et conceptuels. La gestion du contexte permet d'ajouter le comportement de l'utilisateur et les interactions actuelles avec le système pervasif dans la conception des applications. Elle souligne aussi l'importance de capteurs intégrés dans l'environnement afin de détecter l'emplacement et le mouvement actuel de l'utilisateur et l'ajouter aux propriétés d'un système sensible au contexte.

La prise en compte du contexte d'utilisation dans les applications est un domaine de recherche d'actualité connu sous le nom de "sensibilité au contexte". Une application sensible au contexte doit percevoir la situation de l'utilisateur dans son environnement et adapter par conséquent son comportement à la situation en question nécessaire afin de fournir un service satisfaisant et proactif aux utilisateurs.

En résumé, les limites des systèmes pervasifs ont été élargies par le développement de nouvelles technologies, ainsi que l'utilisation intensive des technologies existantes, telles que l'Internet, les communications mobiles et sans fil, les réseaux de capteurs et la technologie RFID.

1.4 Les technologies utilisées par les systèmes pervasifs

La croissance rapide de l'utilisation des systèmes pervasifs fait face à de nombreux problèmes de recherche due au développement des applications critiques. La grande hétérogénéité des ressources matérielles et logicielles ainsi que la structure du réseau posent de véritables problèmes de coordination et exige une connaissance approfondie des différentes technologies. La figure 1.4 visualise l'ensemble de ces technologies et montre la position des systèmes pervasifs entre eux, ceci peut illustrer clairement la définition d'un système pervasif et les éléments qui peuvent être impliqués pour développer un tel système.

Essentiellement, l'informatique pervasif utilise la technologie Web, les appareils portables et la communication sans fil. Elle fait appel aussi aux quelques protocoles de communication telle que le HTTP comme un standard sur lequel repose la navigation dans le réseau Internet, car il facilite l'accès aux différentes informations et il peut être mis en œuvre facilement sur des appareils divers. Cela offre aux utilisateurs mobiles un accès transparent aux ressources hors de leur environnement actuel.

Dans les sous-sections suivantes, nous allons résumer quelque technologie de base pour construire un système pervasif minimal.

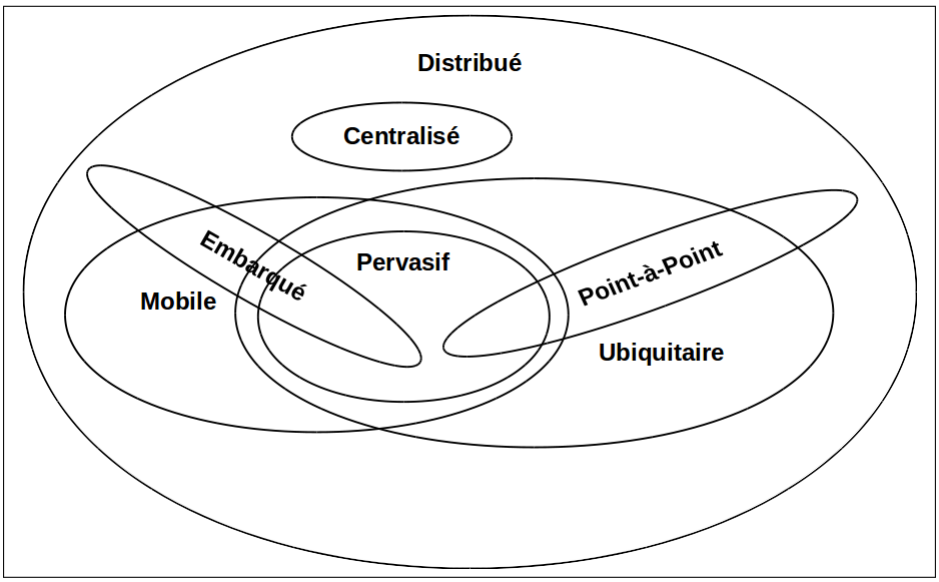

Figure 1.4: Les technologies impliquées dans les systèmes pervasifs

1.4.1 Les systèmes embarqués

La contrainte d'intégration des dispositifs informatiques dans des objets connectés miniaturisés cause un problème majeur pour les systèmes pervasifs. Cette dernière exige un taux d'intégration très élevé de ces objets dans l'environnement réel. Par conséquent, la nécessité d'utiliser les technologies des systèmes embarqués devient cruciale, ces derniers offrent une plate-forme informatique de traitement modulaire et communicante, qui a pour objectif de réaliser un ensemble de tâches bien définies dans l'intention est de créer un environnement pervasif personnalisable selon les besoins des utilisateurs.

Un système embarqué est définit comme un système à la fois Hardware et software, cette fusion a comme objectif de concevoir un dispositif électronique doté d'une capacité de traitement de données selon un programme informatique embarqué à l'intérieur du dispositif lui-même. C'est pourquoi, un tel système doit contenir au minimum les composants électroniques suivants:

- CPU : Généralement représenté par un microcontrôleur, mais nous pouvons trouver dans certains systèmes un microprocesseur. Elle joue le rôle du cerveau dans le système car c'est à elle d'exécuter la suite d'instructions définies par les programmes

informatiques.

- Mémoire : Il existe plusieurs types de mémoires (vives, mortes, etc.) mais le choix est conditionné par la nature des données et la consommation d'énergie, ainsi par la quantité d'informations à gérer et à stocker par ce genre de circuits.

- Source d'énergie locale : Ce sont généralement des batteries, nous pouvons aussi équiper le dispositif par une source d'énergie renouvelable comme les panneaux solaires pour garantir une autonomie d'énergétique durable.

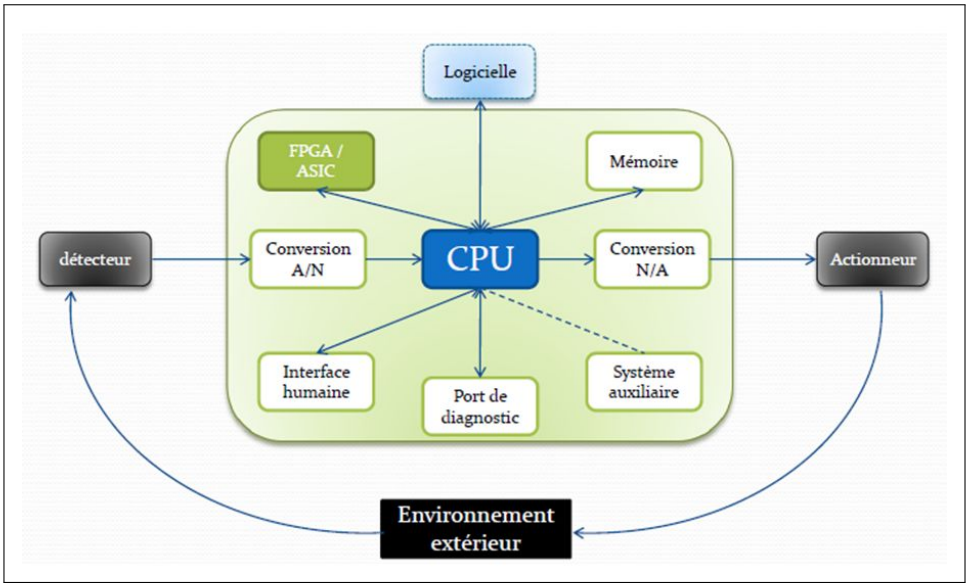

Figure 1.5: L'architecture générale d'un système embarqué

Ces trois composants électroniques permettent à un système embarqué de garantir un traitement local de données, et un fonctionnement autonome en matière d'énergie ce qui lui offre une certaine mobilité dans un champ parfois limité par la couverture du réseau. Dans certains cas, il est nécessaire de lui ajouter un ensemble de capteurs et/ou effecteurs pour pouvoir communiquer avec son environnement. Tout ce Hardware doit être géré par un ou plusieurs programmes software selon les tâches définies lors de la conception du système. La figure 1.5 illustre clairement tous les éléments qui peuvent former un système embarqué.

Un système embarqué joue le rôle de l'acteur principal dans un système pervasif, puisqu'il contient l'interface de communication avec l'environnement extérieur (utilisateur humain et/ou les phénomènes naturels tels que la température, la pression, la lumière, etc.)

1.4.2 Les systèmes distribués

Certains systèmes embarqués doivent effectuer un traitement lourd qui nécessite des capacités de calcul très élevées en matière de CPU et de mémoire. Dans ce cas, et sachant que ce type de système est très limité en ressources, il doit utiliser des techniques de calcul à distance telles que le Cloud ou les grilles de calcul, ce qui est possible grâce aux systèmes distribués qui permettent une utilisation efficace des ressources disponibles. réseau et une meilleure coopération pour mener à bien un ensemble de tâches bien coordonné.

Dans ce cas, un système distribué peut être vue comme un ensemble d'entités autonomes de calcul (ordinateurs, PDAs, Smart phone, etc.) interconnectées et qui peuvent communiquer pour réaliser les objectifs suivants :

- uniformité des accès locaux et distants,
- localisation des ressources non perceptibles,
- migration des ressources possibles sans interférences avec la localisation physique,
- réplication de ressources non visibles,
- concurrence d'accès aux ressources non perceptibles,
- invisibilité du parallélisme offert par l'environnement d'exécution,
- tolérance aux pannes permettant à un utilisateur de ne pas s'interrompre (ou même se rendre compte) à cause d'une panne d'une ressource.

Comme tout système informatique, les systèmes distribués sont développés à la base d'une couche software appelée *interlogiciel* (middleware), c'est une couche logicielle

située entre les couches basses (systèmes d'exploitation, protocoles de communication) et les applications dans un système informatique réparti (CORBA, EJB, COM, etc.). La figure 1.6 montre la position de la couche middleware dans les systèmes software.

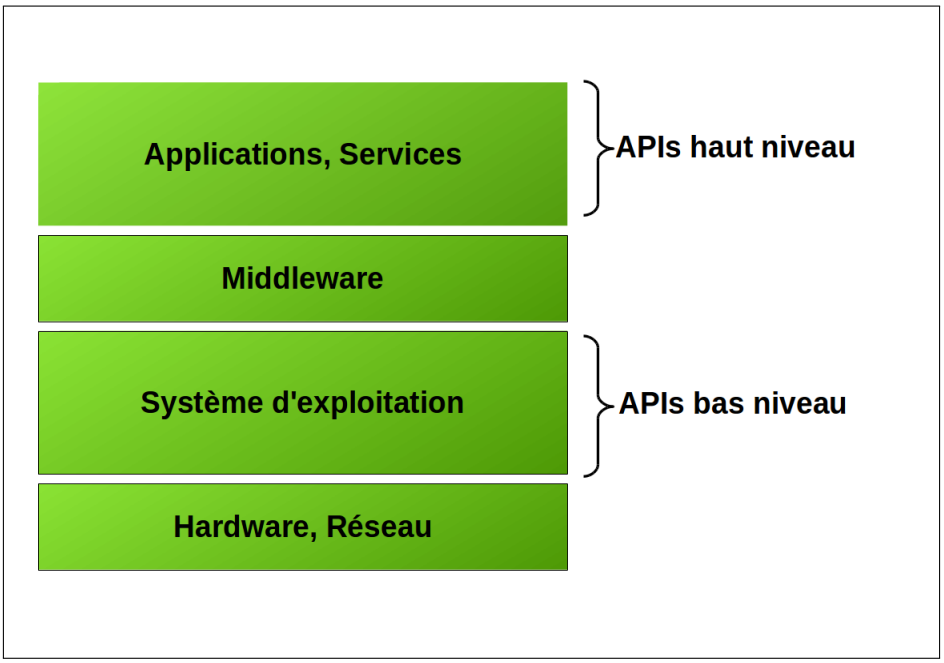

Figure 1.6: Les couches software et hardware dans un système distribué

Les objectifs de cette couche se résument dans les points suivants :

- Fournir une interface ou API de haut niveau aux applications,
- Masquer l'hétérogénéité des systèmes matériels et logiciels sous adjacents
- Rendre la répartition aussi invisible (transparente) que possible,
- Faciliter la programmation répartie (développement, évolution, réutilisation, portabilité des applications),

1.4.3 Les systèmes peer-to-peer

Dans un système distribué, la communication et le partage de données entre les différents dispositifs nécessitent une plate-forme réseau solide et facilement extensible.

Contrairement à l'architecture client/serveur, les systèmes peer-to-peer (P2P) décentralisés offrent comme avantage d'être plus fiable, disponible et efficace. Les systèmes P2P sont basés sur des machines informatique autonomes en terme de ressource hardware et software appelées "peers".

Ces homologues partagent des données (ainsi que la gestion des données) stockées de manière conventionnelle sur un serveur central. Habituellement, les peers peuvent être autonomes ou sous le contrôle des utilisateurs individuels qui allument ou éteignent leurs machines à tout moment. Ce qui implique que les peers rejoignent et quittent le système P2P à des taux élevés, un problème qui n'existe pas dans les systèmes classiques. En d'autres termes, un système P2P est constitué uniquement de composants non fiables. Néanmoins, le système P2P devrait fournir un service fiable et efficace.

D'un côté, l'avantage des systèmes (P2P) est qu'ils peuvent évoluer vers des millions de dispositifs avec du matériel à faible coût, alors que l'approche classique d'utilisation de systèmes basés sur serveur ne s'adapte pas bien, sauf si de puissants serveurs sont fournis. D'un autre côté, les systèmes basés sur serveur peuvent fournir des garanties sur leurs services. C'est pourquoi, ils sont préférables pour les applications critiques qui nécessitent un haut niveau de fiabilité. Pour cette raison, nous pouvons trouver dans les systèmes pervasifs un couplage entre les deux approches dans l'intention de partager leurs avantages sans partager leurs inconvénients.

Les systèmes pervasifs font appel à des technologies de type P2P pour former une plate-forme réseau sur laquelle les données et les informations peuvent être communiqué, sans le but de donner une liberté aux périphériques pervasifs d'intégrer le réseau ou le quitter à tout moment sans influencer sur le fonctionnement globale du réseau (les autres dispositifs).

1.5 Caractéristiques des systèmes pervasifs

La forte demande des systèmes pervasifs dans la vie quotidienne des utilisateurs crée des nouveaux besoins engendrés par la prolifération d'objets connectés, et par conséquents de nouveaux défis sont soulevés par les chercheurs pour pouvoir garantir tous les propriétés indispensables qui permettent de développer un système pervasif fiable. Ces propriétés se sont une conséquence de la combinaison des différentes technologies avec les éléments de base de l'architecture générale des systèmes pervasifs, ce qui leur donne un nombre important de caractéristiques que nous pouvons les résumer dans les points suivants :

1.5.1 Dynamique

Dans un système informatique ubiquitaire, la disponibilité des objets communicants peut varier au cours du temps. En effet, ces derniers peuvent être dynamiquement ajoutés ou retirés du système, de manière intentionnelle ou non dû à la portée limitée des technologies réseaux sans fil, ressources limitées en énergie ou des défaillances matérielles et logicielles, ce qui peut les rendre indisponibles pour un temps indéfini.

Les systèmes pervasifs gèrent cette caractéristique par le biais des réseaux P2P pour manager la présence de tous les dispositifs qui peuvent apparaître instantanément dans le réseau et les rendre accessibles et facilement exploitables par les utilisateurs.

1.5.2 Interopérabilité

L'interopérabilité compte parmi l'une des principales priorités des dispositifs pervasifs et s'inscrit dans le cadre de l'initiative pour l'informatique de confiance [8]. Elle est définie comme la capacité de deux ou plusieurs systèmes d'échanger des informations et d'utiliser les informations échangées avec des différents langages d'implémentation, d'environnement d'exécution ou de modèles d'abstraction.

L'interopérabilité dans les systèmes pervasifs a pour objectif de permettre à plusieurs systèmes logiciels, de même nature ou hétérogènes, de communiquer et de coopérer afin de minimiser les coûts d'intégration de ces systèmes. L'interopérabilité repose sur des

normes techniques (à la base des standards) qui définissent des exigences accompagnées de recommandations permettant à deux systèmes qui satisfont aux exigences de dialoguer et d'évoluer librement tant qu'ils respectent la norme définissant leurs interfaces.

L'interopérabilité est de plus en plus requise du fait du déploiement d'un grand nombre d'applications et de leur potentiel d'interaction et de coopération. Elle peut être abordée de façon statique (compilation de deux applications à travers un bus CORBA par exemple) ou de manière dynamique (invocation de service par exemple).

Trois niveaux d'interopérabilité peuvent être distingués dans un environnement pervasif :

- Le niveau technologique: La diversité des technologies utilisées pour développer les dispositifs pervasifs cause des problèmes d'interopérabilité matérielle ainsi que la personnalisation des protocoles utilisés chez certains constructeurs ne permet pas la communication avec le reste de dispositifs développés par d'autres constructeurs.

- Le niveau syntaxique: La représentation syntaxique de l'information différent d'un système à un autre ce qui complique les échanges de données, et implique une faible interopérabilité dans le système global.

- Le niveau sémantique: La représentation de la même donnée sous plusieurs formes qui semble être différentes complique la recherche et l'interprétation de l'information par des systèmes étrangers.

1.5.3 Hétérogénéité

La transparence des inégalités d'environnement, malgré l'hétérogénéité matérielle, logicielle et protocoles de communication entre les objets communicants qui forment les systèmes pervasifs. Les équipements utilisés sont très variés, fonctionnant avec divers systèmes d'exploitation, adoptant différentes technologies sans fil. Au sein d'un même réseau, des stations de travail aux capacités de calcul et de stockage important peuvent

coexister avec d'autres appareils à faibles ressources. Un environnement hétérogène est crée, qui doit collaborer pour offrir une diversité de services à l'utilisateur final.

Cette hétérogénéité peut être dû à la grande différence entre le degré d'intelligence des systèmes dans le même environnement, ceci peut causer aussi un problème d'invisibilité.

1.5.4 Reconfigurabilité

Les applications s'exécutant dans l'environnement pervasif doivent être capables de s'adapter à ces changements imposés par les variations pseudo-chaotiques du mode réel, dont l'objectif est de garantir une utilisation efficace des ressources dans ces nouveaux environnements, ce qui augmente la fiabilité des applications pervasifs.

La nécessité de reconfigurer un système pervasif est une conséquence des autres propriétés de ce système, tels que la dynamicité, l'interopérabilité et l'invisibilité. Dans ce contexte, la reconfiguration sert à exploiter efficacement les ressources présentes dans l'environnement pervasif. Selon les raisons de la reconfiguration, nous pouvons la localiser dans les niveaux suivants :

- Terminal : Généralement représenté par un système embarqué autonome, dans ce cas le terminale doit reconfigurer son :
  - Matériel : Pour prendre en considération tout périphérique qui lui est lié (capteur, effecteur, interface de communication, etc.
  - Système : Pour prendre en considération tout service qui sera lui proposé et jugé être utile pour son fonctionnement.

- Réseau :
  - Organisation des nœuds : Réseau ad hoc mobile,
  - Routage : routage proactif, routage réactif,
  - Multicast dans les réseaux mobile.
- Système d'information: La collecte et l'organisation de toutes les informations présentes dans le système,
- Utilisateur: Adapter l'application au profil de l'utilisateur (gestion de session).

1.5.5 Smart Spaces

L'interconnexion d'un ensemble de dispositifs pervasif dans le même environnement crée un espace intelligent qui permet d'avoir une projection numérique du mode réel des utilisateurs à travers des objets communicants dotés d'une capacité pour capturer et collecter les informations présentes dans le monde physique, ensuite, ces dernières serons interprétées, filtrées et agrégées par diverses applications, afin d'enrichir les données contextuelles du système dans le but de fournir des services adaptés aux besoins des utilisateurs. En résumé, ceci impose qu'un dispositif pervasif doit être :

- **Context-awareness** : Perception de l'environnement pour interagir plus "naturellement" avec l'utilisateur,
- **Smartness** : Utiliser efficacement les informations du contexte,
- Exemple : Maison intelligente, Classe intelligente, etc.

La figure 1.7 représente un exemple concret d'un espace intelligent créé pour des objectifs éducatifs, ce dernier se présente sous forme d'une salle de cours intelligente dotée

Figure 1.7: Exemple d'un smart space (smart classroom)

d'un ensemble de dispositifs communicants (système de visioconférence, tableau intelligent interactif, micro-ordinateurs, tablettes, etc.) qui permettent aux utilisateurs (enseignants, étudiants) une utilisation efficace et cohérente de ces ressources ce qui implique une meilleurs exploitation de ces derniers afin d'augmenter les performances et la qualité d'enseignements.

1.5.6 Invisibilité

Avec l'augmentation du nombre de dispositifs informatisés autour des utilisateurs, le nombre des applications qui gères ces dispositifs augmente aussi, ce qui rend le simple utilisateur perdu entre toutes ces applications, ceci complique sa vie au lieu de la simplifier. Pour cela, les systèmes pervasifs doivent fonctionner discrètement et d'une façon invisible pour l'utilisateur afin qu'il puisse se concentrer sur une tâche bien précise.

Un système qui nécessite une intervention humaine minimale offre une approximation raisonnable de l'invisibilité. Les humains peuvent intervenir pour régler des environnements intelligents lorsqu'ils ne répondent pas automatiquement aux attentes des utilisateurs. Une telle intervention pourrait également faire partie d'un cycle d'apprentissage continu pour l'environnement.

L'invisibilité d'un système peut se résumer dans les points suivants :

- Nécessité d'un minimum d'intervention humaine,
- Adaptation seulement aux changements,
- Auto-apprentissage,
- Proactive : Suggérer, proposer des actions correctives à l'utilisateur en fonction du contexte présent ou prédit,
- Exemple : Reconfiguration dynamique des caractéristiques réseaux d'un équipement,

1.5.7 Scalabilité

À mesure que les espaces intelligents gagnent en sophistication, l'intensité des interactions entre l'espace informatique personnel d'un utilisateur et son environnement augmente. Ceci a de graves implications en termes de bande passante, d'énergie et de distraction pour un utilisateur mobile sans fil. La présence de plusieurs utilisateurs compliquera encore ce problème. L'extensibilité, au sens le plus large du terme, est donc un problème critique dans les systèmes pervasifs.

Indispensable dans les systèmes pervasifs, le passage à l'échelle permet de déceler les limitations du système lors d'une montée en charge du nombre important de composants entrant en jeu. Le nombre d'interactions dans les espaces intelligents fait que l'utilisation des ressources devient critique, il faut donc que les systèmes pervasifs soient capables d'appréhender un grand nombre d'équipements dynamiques.

En général, la scalabilité exige les points suivants :

- Passage à l'échelle,
- Développement des applications indépendantes du volume,
- Gestion des volumes de plus en plus grand de (l'utilisateur, application, dispositifs connectés)

- Utilisation des techniques d'adaptation pour pouvoir répondre à chaque cas,

1.5.8 Autonomie

Les systèmes pervasifs visent à diminuer le rôle de l'homme dans le contrôle et la maintenance des systèmes informatiques où son intervention sera concentrée sur quelque tâche critique qui exige vraiment une intelligence humaine. Cette autonomie est possible grâce à l'ensemble de technologies et techniques utilisées par les systèmes pervasifs telle que les systèmes embarqués et la reconfiguration.

Ces systèmes sont conçus pour être capables d'agir sur leur propre comportement afin de s'adapter et se reconfigurer aussi bien à des conditions d'exécution changeantes qu'à une erreur survenue en leur sein. Ceci leur permet une meilleure intégration dans leur environnement et une durée de vie beaucoup plus importante de celle d'un simple système.

1.5.9 La distribution

Les systèmes distribués sont considérés comme le noyau des systèmes pervasifs. En effet, ils peuvent fonctionner dans des environnements homogènes via un réseau de communication sans fil, dispersés dans l'environnement physique et peuvent prendre en charge des nouveaux contextes et ressources sans que cela ne soit visible par un utilisateur.

1.6 Conclusion

En regroupant toutes ses caractéristiques, nous avons constaté qu'un système perversif joue le rôle inversé d'un système de réalité virtuel, en d'autres termes si la réalité virtuelle permet aux humains de vivre l'expérience du monde virtuel en fusionnant le monde réel dans le virtuel, les systèmes perversifs force la machine de vivre dans le monde réel à travers un ensemble de capteurs qui leurs permettent une bonne perception de leur environnement, et un autre ensemble d'effecteurs qui leurs donnent la possibilité de réagir aux différents événements.

De plus l'interconnexion des différents dispositifs pervasifs forment un réseau d'objets connectés dont le but est de gérer la complexité du monde réel, cela va créer une énorme quantité d'information dans le réseau qui nécessite des techniques et des outils de gestion et d'analyse automatique et en temps réel pour garder un œil sur l'environnement extérieur. Pour cette raison un mécanisme de reconfiguration dynamique est nécessaire pour prendre en considération tout changement dans l'environnement.

Chapitre 2

La reconfiguration dynamique

La reconfiguration dynamique est une technique qui peut être utilisée dans plusieurs applications dans des différents domaines. Dans ce chapitre nous allons citer quelques exemples d'utilisation de cette technique à fin de comprendre les principes de base de la reconfiguration dynamique.

2.1 Introduction

Actuellement, plusieurs systèmes nécessitent une reconfiguration dynamique. Ce besoin devient crucial quand le redémarrage ou l'arrêt total du système n'est pas autorisé. Cette caractéristique "non-stop" est généralement exigée par les systèmes industriels possédant des systèmes embarqués complexes, tels que les systèmes d'exploitation des voitures modernes vues que leur environnement impose plusieurs changements qui doivent être pris en considération par le système d'exploitation. Dans ce cas, nous pouvons prendre la définition suivante pour la reconfiguration dynamique:

Définition *La reconfiguration dynamique est la capacité d'un système logiciel à permettre la modification d'une sous-partie du système durant son exécution, sans interruption de service.[9]*

Il existe plusieurs approches pour reconfigurer dynamiquement un système donné, et chaque approche dépend directement de la nature du système et la raison pour laquelle la reconfiguration dynamique est nécessaire. Dans ce qui suit, nous présentons quelques approches déjà proposées pour des domaines différents et étudions leurs points communs en vue d'extraire les caractéristiques principales d'une reconfiguration dynamique.

2.2 Reconfiguration dynamique software

Dans le domaine software, reconfigurer une application i.e. changer sa configuration, est une tâche commune dans tous les domaines de l'ingénierie du logiciel. Actuellement, plusieurs applications nécessitent une reconfiguration dynamique pendant l'exécution, tel que l'exemple du système d'exploitation des voitures modernes, ce dernier est basé sur l'approche des composants softwares, car cela facilite le contrôle de tous les sous-systèmes d'une voiture moderne, vu qu'elle doit assurer un bon fonctionnement dans les différents modes associés aux différentes situations, dans lesquels plusieurs configurations son possible.

La commutation entre les différents modes implique que tous les composants doivent être mis à jour de façon synchrone, tout en préservant la stabilité de l'application i.e. du système. Ceci est une contrainte très importante dans le domaine de la reconfiguration dynamique.

Nous pouvons avoir aussi un autre cas pour la reconfiguration dynamique dans les applications orientées services qui produisent autres services par composition, ces derniers ne peuvent être contrôlé par l'application appelante donc elle n'a aucun pouvoir pour redémarrer le service, pour cela elle exige une nouvelle configuration pour sa nouvelle situation. La reconfiguration dans les applications orientées services peut devenir indispensable dans le cas où un de ces services devient inaccessible ou sa qualité devient inacceptable, dans ce cas la reconfiguration peut intervenir pour changer vers un autre service ou pour modifier complètement l'architecture de l'application.

Dans certains domaines il est largement suffisant d'avoir un ensemble fini de configurations possibles comme le cas des voitures cité, mais dans la majorité des autres domaines, nous ne pouvons pas prévoir toutes les configurations possibles pour cela l'utilisation d'une approche basée sur des règles de configurations est plus appropriée.[10]

Parmi les applications qui ont eu une tendance vers la reconfiguration dynamique nous avons :

2.2.1 La reconfiguration dynamique des applications à base de composants

L'approche par composant [11] est devenue la méthode la plus utilisée dans le monde du développement logiciel vu ces avantages qui offrent aux développeurs une meilleure solution pour modéliser le monde réel et développer des applications qui respectent la plupart des caractéristiques du génie logiciel tels que la maintenabilité, la réutilisation, la modularité, etc.

L'élément de base dans cette approche est le composant qui peut-être défini comme suite :

Définition *un composant logiciel est une unité de composition avec des interfaces spécifiées contractuellement et des dépendances de contexte explicites. Un composant peut être déployé indépendamment et composé par des tiers.*

Définition *Un composant est une unité non triviale, presque indépendante, et une partie remplaçable d'un système qui accomplit une fonction claire dans le contexte d'une architecture bien définie. Un composant se conforme à un ensemble d'interfaces et fournit leur réalisation physique[12].*

Pour mieux comprendre l'utilité des composants logiciels dans le mode de développement software, nous allons étudier les différentes caractéristiques de ces derniers et qu'elle approche doit-on suivre pour reconfigurer un système à base de composants.

Caractéristique d'un composant logiciel

Un composant est la définition abstraite d'une unité logicielle; il est caractérisé par trois éléments comme le montre la figure 2.1:

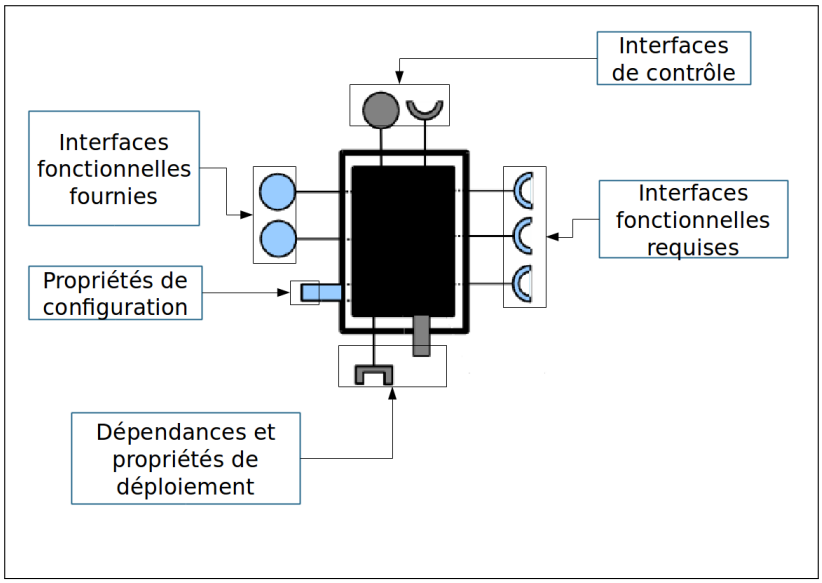

Figure 2.1: L'architecture générale d'un composant software

1. Les interfaces : Il existe trois types d'interface dans un composant:
   - Interfaces requises : ce sont des interfaces en entrée,
   - Interfaces fournies : elles définissent les services fournis par le composant,
   - Interfaces de contrôle : elles permettent le contrôle du composant,
2. Les modes de coopération avec les autres : Ils définissent de quelle manière une interface peut communiquer avec une autre. Il existe trois modes de communication:
   - le mode synchrone,
   - le mode asynchrone,
   - le mode de diffusion en continu.
3. Les propriétés de configuration.
4. Dépendances et propriétés de déploiement.

Les modèles de composants:

Actuellement, il existe plusieurs modèles de composants selon la technologie utilisée, mais ils partagent les mêmes principes : la construction d'un composant, assemblage des composants, déploiement, exécution et aspect dynamique d'un composant.

La reconfiguration dynamique d'un système à composant:

Il existe plusieurs propositions de définition pour la reconfiguration dynamique; dans le cas des systèmes logiciels nous avons choisi celle qui nous a paru la plus adéquate:

Définition *La reconfiguration dynamique est la capacité d'un système logiciel à permettre la modification d'une sous-partie du système pendant son exécution, sans interruption du service.[9]*

La défaillance ou l'obsolescence d'un composant logiciel peut être la principale raison de la modification d'une application informatique, ce qui nécessite une reconfiguration de celle-ci afin d'appliquer les mises à jour nécessaires pour prendre en compte les nouveaux changements. Ceci est possible grâce à des interfaces requises ou fournit des composants, qui peuvent être connectés à d'autres composants qui offrent des interfaces compatibles.

Intuitivement une reconfiguration se déroule de la manière suivante[9]:

1. Identifier et délimiter la partie du système à reconfigurer [ident].
2. Suspendre son exécution (pour éviter de corrompre le système)[suspd].
3. Modifier la configuration du système (ajout, suppression de parties, etc.)[modif]
4. Transférer l'état vers les nouvelles parties [transf].
5. Reprendre l'exécution de la partie interrompue du système [resum].

Les cinq étapes proposées par [9] exigent une intervention humaine pour effectuer chacune d'entre eux. Ce qui représente une faiblesse pour cette approche envers les sys-

tèmes automatiques ainsi que les systèmes qualifiés par la caractéristique "non-stop", car le passage d'une configuration à une autre exige l'arrêt total de l'application.

2.2.2 La reconfiguration dynamique des Services Web

L'approche des services ou les Services Web sont devenu une nécessité dans le monde des entreprises. Elle leur offre une meilleure présentation et une visibilité via le Web pour tous les clients dans le monde. Actuellement, elle tire sa puissance de la notion de composition de services, d'où elle cherche à trouver chaque jour des nouveaux services pour les utilisateurs; pour cela elle a eu aussi une tendance vers la reconfiguration dynamique des services.

Plusieurs travaux ont été effectués, d'autres sont toujours en cours de réalisation et qui cherchent à améliorer la qualité de service offert à travers le Web en composant à chaque fois un nouveau service.

Un service est représenté généralement par les spécifications d'ISC (interfaces, scénario et ses contraintes)Figure 2.2 [13].

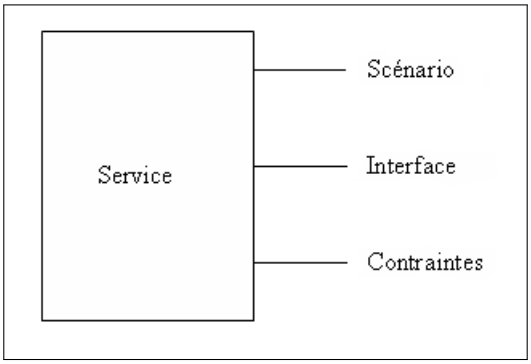

Figure 2.2: Représentation Graphique d'un service web

- Interfaces:
- Scenario:
- Contraintes:

Cette représentation offre la possibilité d'avoir un service formé ou bien composé de plusieurs sous services comme le montre la figure 2.3, dans ce cas le service composé (résultant) aura ses propres spécifications ISC.

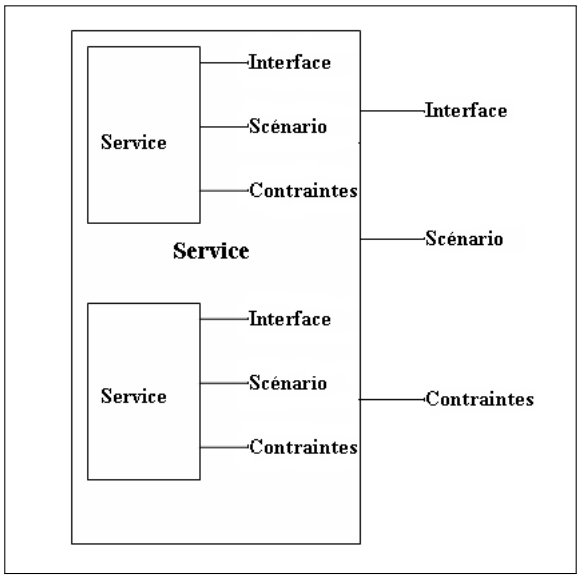

Figure 2.3: Représentation Graphique d'un service web composé

La composition des nouveaux services à base des services existants exige une reconfiguration dynamique dont le but est de les publier dans le web et faciliter leur exploitation par l'utilisateur final.

Dans la DRS (Dynamic Reconfiguration Services) les modules à configurer sont les services; elle utilise les spécifications ISC comme des interfaces pour configurer les services à composer afin de former une application web.

Le rôle de la DRS est d'assurer les fonctions suivantes:

- Dynamic Service lookup, service publication, construction du service, description du service.
- Registration et De-registration des services.
- Vérification du service en exécution.

- La DRS change le comportement du service en cours d'exécution ce qui veut dire changement des spécifications ISC.

Dans le but d'assurer toutes ses fonctions, la DRS comprend quatre sous services qui coopèrent pour composer dynamiquement des nouveaux services et les rendre publics, la figure 2.4 illustre l'ensemble de fonctionnalités de ces sous services, et qui ont comme rôle :

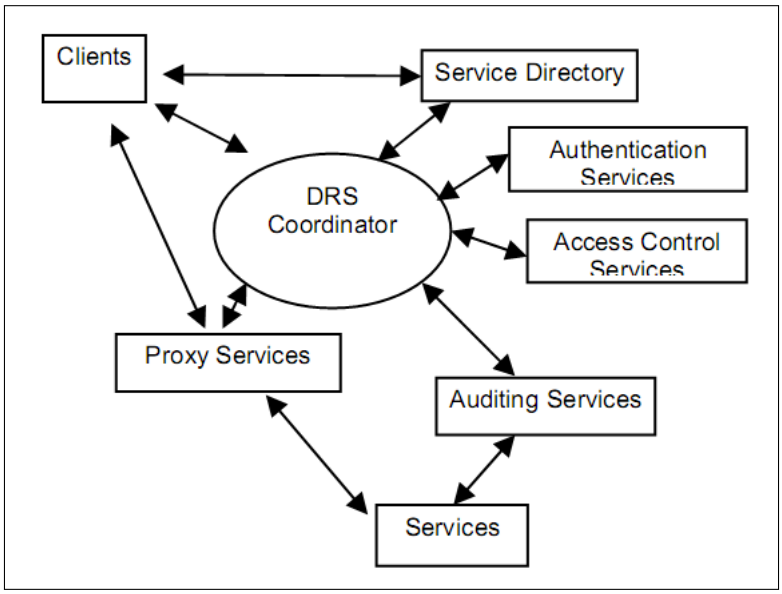

Figure 2.4: Architecture de la Dynamic Reconfiguration Service

- Service Directory(SD) : Il enregistre et organise à la fois les groupes de services relatifs dans un arbre hiérarchique.

- Standard Service Naming Directory(SSND) : Il sauvegarde tous les noms de services enregistrés par ordre alphabétique.

- Proxy Agents(PA) : Il est le responsable de l'interopérabilité et l'intégration entre les DRS et les clients ainsi il renforce la sécurité des accès.

- Auditing Agents(AA) : Il vérifie les performances et les propriétés d'utilisateurs pour les services participants en cours d'exécution et met à jour leur profil.

L'utilisation des services comme l'unité de base pour la reconfiguration offre la possibilité d'ajouter, supprimer ou modifier à n'importe quel moment un service dans l'ensemble des services du système, ceci donne la possibilité d'étendre les fonctionnalités initiales de ce dernier.

Nous pouvons aussi remarquer l'utilisation des agents afin d'assurer non seulement les différentes fonctionnalités de la DRS mais aussi la coordination entre ses différents éléments.

2.3 Reconfiguration dynamique hardware

Certains systèmes Hardware possèdent la possibilité d'être reconfigurer dynamiquement au cours de son fonctionnement, cela peut être dûe à un changement dans l'environnement ou pour améliorer ses performances. Parmi les systèmes qui utilisent cette technique nous avons les systèmes à base de circuits FPGAs.

2.3.1 La reconfiguration dynamique des circuits FPGAs

Les circuits FPGA (Field Programmable Gate Arrays) sont conçus pour pouvoir simplifier le développement des applications embarquées dans l'intention de les miniaturiser et les rendre beaucoup plus fiable et flexible, ces derniers peuvent être utilisés dans plusieurs domaines comme le spatial, le médical et l'automobile, etc.

La puissance des FPGA est offerte par plusieurs caractéristiques obtenues à partir de leur architecture et qui sont généralement basées sur la possibilité de reconfigurer dynamiquement.

2.3.2 Architecture des FPGAs

L'architecture simplifiée d'un circuit FPGA [14] montre qu'il est constitué généralement de trois éléments illustrés par la Figure 2.5:

- CLB (configurable logical blocks) : c'est les éléments de base dans un FPGA, ils implémentent les fonctions logiques déterminées par l'application, ils jouent générale-

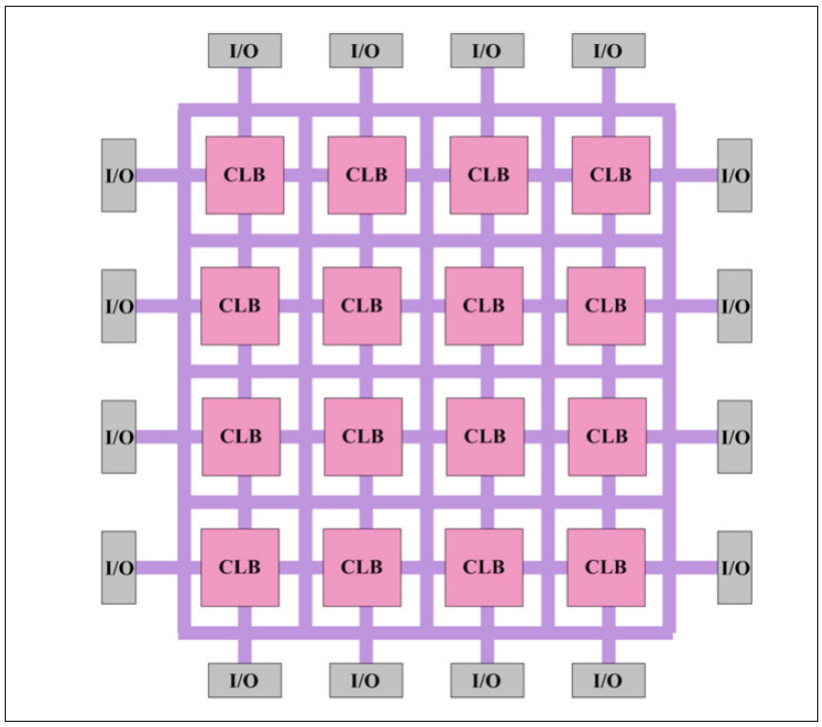

Figure 2.5: L'architecture générale d'un circuit FPGA

ment le rôle du processeur dans un FPGA. Ils sont généralement constitués de :

- une table de transcodage LUT,
- des bascules D,
- des Multiplexeurs.

- IOB (input/output blocks) : c'est des blocs d'entrée/sortie pour le circuit FPGA, ils se servent pour programmer, reprogrammer (reconfigurer), alimenter et réceptionner les données.

- Connections (lignes/matrices) : ils sont utilisées pour relier les différents blocs CLB et IOB à travers des lignes ou des matrices programmables, ce qui donne la notion de routage entre blocs. Nous pouvons distinguer plusieurs types de connexions :
  - connections directes,
  - connections générales à travers des matrices de routage,

- connections à long distance,
- distribution d'horloge spécifique.

L'architecture générale d'un circuit FPGA comme le montre la figure 2.5, est représenté par un ensemble de CLB organisé dans une matrice entouré par un ensemble de IOB et interconnecté par des connections.

2.3.3 Principe de la reconfiguration dynamique pour les FPGAs

La reconfiguration dynamique des circuits FPGAs [14] permet de changer une partie du circuit FPGA en lui chargeant les nouvelles données, pendant que l'autre partie reste en exécution avec les anciennes données. Pour cette raison nous parlons ici d'une reconfiguration dynamique partielle [15], c'est-à-dire touche juste une partie du circuit et non pas l'ensemble du circuit.

Cette caractéristique est possible grâce à la façon de programmer ce genre de circuit, qui consiste à activer ou désactiver certains blocs (CLB ou IOB) [14] à l'effet de construire un chemin de traitement qui permet d'acheminer les données en entrée vers des résultats aux sorties, ceci veut dire que pendant l'exécution d'un programme nous ne trouvons pas tous les blocs actifs au même temps, et par conséquent nous pouvons modifier les chemins initiaux en activant et désactivant les blocs CLB.

Donc la reconfiguration dynamique des circuits FPGAs est possible grâce à cette logique de programmation et qui offre la possibilité de reprogrammer certains blocs (CLB ou IOB) qui sont jugés inconfortables pour l'application, vu que chaque bloc ou groupe de blocs peut implémenter une ou plusieurs tâches dans l'application. Ceci rend cette dernière modulaire et donc l'entretien partiel est possible en cours de l'exécution de l'application.

Approche pour la reconfiguration dynamique des FPGAs

La reconfiguration dynamique d'un circuit FPGA se base sur un ensemble d'éléments qui représente les acteurs principaux de toute approche de la reconfiguration. La figure 2.6 représente l'ensemble de ces éléments et l'interconnexion entre eux.

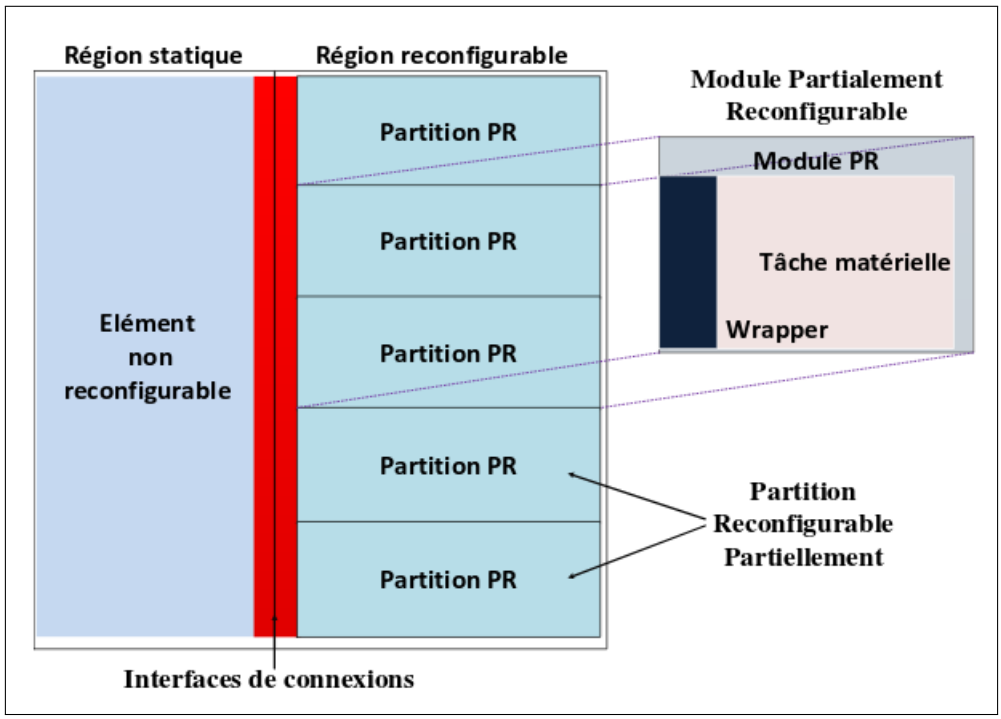

Figure 2.6: Les acteurs principaux de la reconfiguration dynamique des FPGAs

- Région reconfigurable : Région délimitée initialement dans le circuit FPGA, elle est destinée à être le siège des reconfigurations dynamiques,
- Région statique : Région délimitée qui, à l'inverse de la région reconfigurable, est destinée à un fonctionnement purement statique,
- Partition reconfigurable Partiellement (Partial Reconfigurable Partition) (Partition PR) : est une partition de la région reconfigurable. Une région reconfigurable peut être composée de plusieurs partitions PR,
- Tâches matérielles ou Intellectual Property (IP core) : Blocs fonctionnels complexes réutilisables,
- Module Partialement Reconfigurable (Module PR) : tâche matérielle qui peut peupler une région reconfigurable. Il peut y avoir plusieurs modules reconfigurables par région. Un module PR peut aussi occuper une ou plusieurs partitions PR,

- Wrapper : Circuit matériel permettant de connecter une tâche matérielle au reste du système reconfigurable,

- Interfaces de connexions : Point de connexion d'une région reconfigurable.

La première technique pour la reconfiguration d'un circuit FPGA est considérée comme une reconfiguration externe (par la demande d'un utilisateur) vu qu'un micro-ordinateur est nécessaire pour l'effectuer, ceci consiste à connecter ce dernier avec le circuit FPGA via le port JTAG(Joint Test Action Group) [16], et comme ça les bitstreams de la reconfiguration partielle ou complète seront transmis et installés. Mais cette technique n'est pas adéquate aux systèmes auto-adaptatifs vu qu'elle exige une intervention externe.

Pour cela, une deuxième technique de la reconfiguration interne est proposée pour les FPGAs Xilinx [17], ce dernier est possible grâce au Port d'Accès de Configuration Interne (ICAP) [18], qui est un sous-ensemble de l'interface parallèle à 8-bits SelectMap [15]. Le bitstream reconfigure alors les zones spécifiées du FPGA à l'aide d'un microprocesseur embarqué (PowerPC, Microblaze [19]).

Il existe plusieurs approches qui permettent la conception d'un système reconfigurable dynamiquement et partiellement avec un circuit FPGA, la méthode EAPR [20] (Early Access Partial Reconfiguration) est l'une des méthodes actuelles utilisées comme un flot d'accès rapide à la reconfiguration partielle. Elle permet aux signaux dans la conception de base de traverser une région partiellement reconfigurable sans l'utilisation d'un bus-macro[20]. Pour cela, un nouveau système de communication appelé slice bus-macro a été développé par [21] comme une alternative illustrée par la figure 2.7. Cela améliore les performances de synchronisation et simplifie le processus de construction d'une conception PR.La figure 2.8 illustre l'architecture de cette méthode.

Afin d'effectuer la reconfiguration dynamique dans un FPGA, il est nécessaire de passer par les étapes suivantes :

- créer une partition reconfigurable,

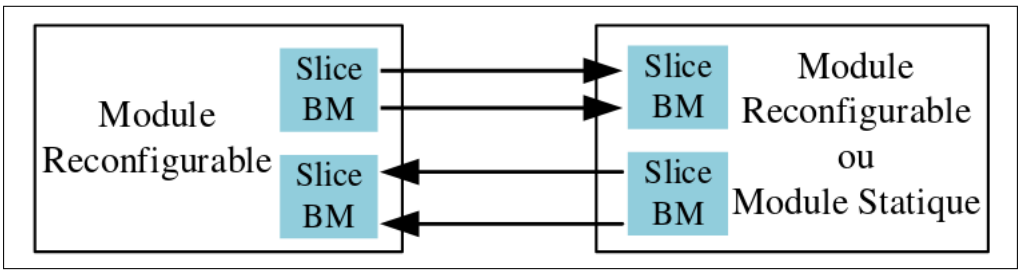

Figure 2.7: Slice BusMacro

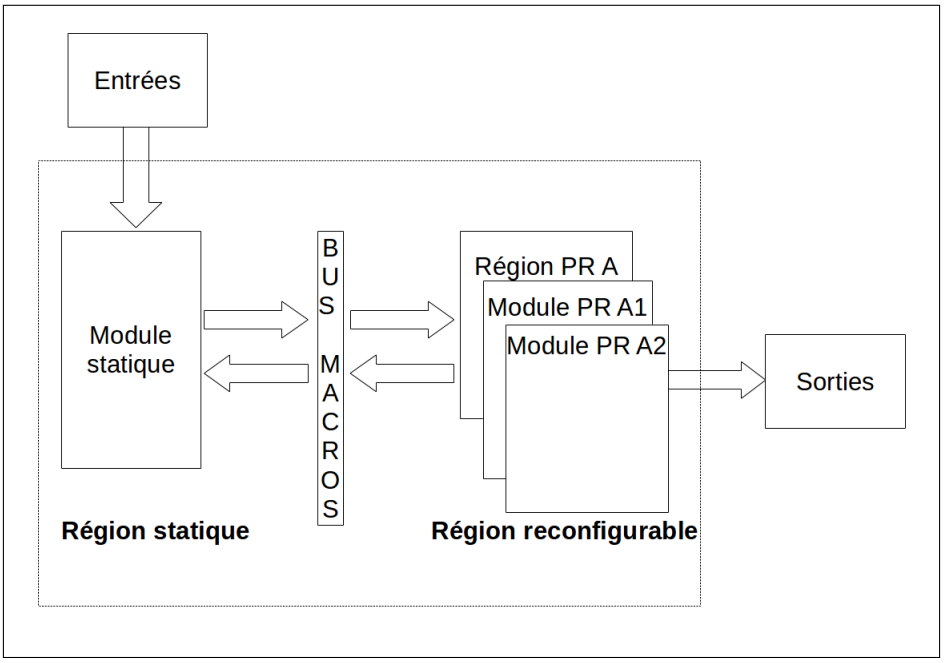

Figure 2.8: L'architecture EAPR

- ajouter un module reconfigurable,
- définir la rangée de P bloque pour la partition reconfigurable,
- implémenter la configuration,
- vérifier la configuration.

Le flot de conception d'un système FPGA est décrit principalement selon les quatre phases suivantes :

- Vue logiciel : elle est déclarée par un fichier MSS (Microprocessor Software Specification)[22], qui contient des librairies, des pilotes et parfois des systèmes d'exploitation utilisés

pour les composantes matérielles. Généralement, le microprocesseur exécute des applications écrites en C/C++ ou assembleur, et un fichier ELF sera le résultat de la compilation.

- Vue matériel : elle est déclarée par un fichier MHS(Microprocessor Hardware Specification) [22]utilisé par les outils Xilinx pour déclarer les composants à instancier dans le système. La description matérielle est écrite soit en langage matériel (HDL) tels que VHDL ou Verilog [23] pour générer un fichier bitstream.

- Association : lors de cette phase, le bitstream correspondant à la partie matérielle du système est fusionné avec le fichier exécutable du logiciel. Le fichier bitstream résultant *download.bit* peut, par la suite, être chargé sur l'FPGA à partir du PC.

- Simulation du flot : la phase de simulation peut être effectuée à deux niveaux différents lors de la conception : avant la synthèse (simulation comportementale) ou après la synthèse (simulation structurelle).

2.3.4 Les modes de la reconfiguration

La reconfiguration dynamique d'un FPGA peut être effectuée sur plusieurs échelles en fonction des besoins ou les raisons pour lesquelles nous devons reconfigurer le système. Et selon l'impact de la reconfiguration sur l'application, nous pouvons distinguer trois modes de configuration comme le montre la figure 2.9 :

- Single contexte : Le changement de la configuration se fait sur la totalité de l'application,

- Multi contexte : Le changement de la configuration se fait sur un module de l'application,

- Partiellement reconfigurable : Le changement de la reconfiguration se fait sur une partie du module de l'application.

Avec la naissance des circuits FPGAs, la reconfiguration dynamique d'un système Hardware sans avoir l'obligation d'arrêter le système ou le redémarrer est devenue possible

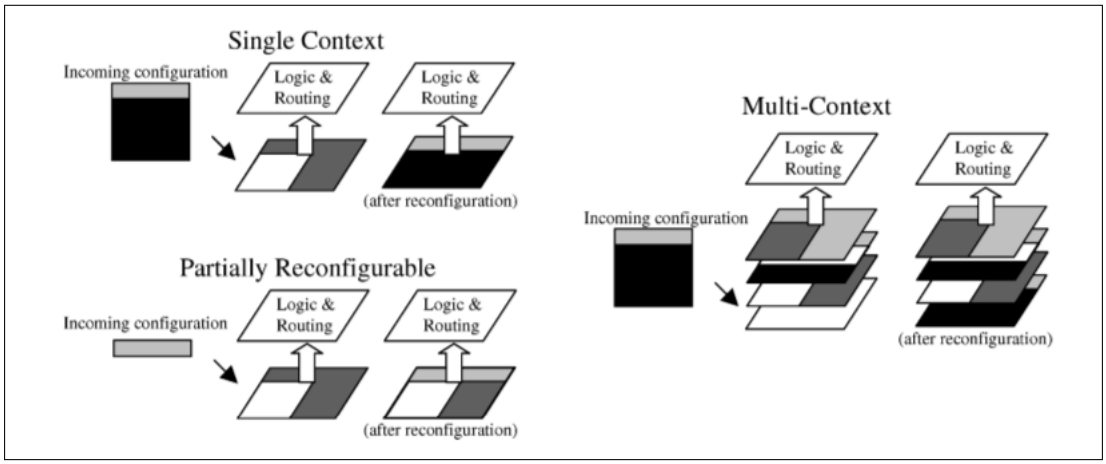

Figure 2.9: Les différents modes de la reconfiguration [2]

à implémenter, sachant qu'elle doit être prise en compte lors de la conception du système pour pouvoir la exploiter après sa mise en œuvre, ceci a influencé positivement sur le développement des systèmes embarqués et par conséquent sur les systèmes pervasifs d'une manière générale.

2.4 Reconfiguration dynamique dans l'industrie

Au 21ème siècle, les entreprises manufacturières doivent faire face à des changements de marché de plus en plus, imprévisibles et fréquents causés par la concurrence mondiale.

Pour rester compétitives, les entreprises doivent développer des systèmes de production qui produisent non seulement des produits à faible coût et de haute qualité, mais qui permettent également de répondre rapidement aux besoins du marché et des clients. Le système de manufacturiers reconfigurables ou RMS est un concept ou un paradigme qui est né de ce fait.

2.4.1 Les systèmes manufacturiers reconfigurables (RMS)

Comme tout système de fabrication, le RMS pourrait être considéré comme un système complexe. Il est conçu pour pouvoir s'adapter à son environnement. Pendant sa durée de vie, le système prendra différentes configurations. Il s'adaptera et évoluera, cette fonctionnalité d'adaptation et d'éveil spécifique rendra difficile la conception et introduira

plus de complexité dans le système.

Dans la littérature, le RMS a reçu de nombreuses définitions, mais la plupart d'entre elles utilisent les mêmes mots-clés tels que changement rapide, faible coût, famille de produits, etc. Le RMS est conçu dès le départ pour un changement rapide de la structure (physique et logique), pour ajuster rapidement la capacité de production et la fonctionnalité autour d'une famille de produits en réponse aux changements soudains du marché, ou aux changements intrinsèques du système. Il peut être créé en incorporant les modules de processus de base qui peuvent être réarrangés ou remplacés rapidement et de manière fiable.

La reconfiguration permet d'ajouter, de supprimer ou de modifier les capacités d'un processus particulier, d'un système de contrôle ou de la structure de la machine pour ajuster la fonctionnalité et la capacité de production en réponse aux demandes ou aux technologies changeantes du marché. Ce type de système sera une architecture ouverte qui pourra être améliorée, mise à jour et reconfigurée plutôt que remplacée [24].

2.4.2 Caractéristiques des systèmes manufacturiers reconfigurables

Les systèmes reconfigurables doivent être conçus au départ pour être configurables et doivent être créés en utilisant des modules hardware et software qui peuvent être intégrés rapidement et fidèlement sinon le processus de la reconfiguration va être long et peu pratique.

Dans ce qui suit, nous présentons les six caractéristiques essentielles expliquées dans [24]:

Modularité

Tous les composants majeurs (composants structurels, axes, contrôle, logiciel et outillage) sont modulaires. Par cette technologie, le système peut être facilement reconfiguré en enlevant simplement, en changeant, les unités constituantes ou les modules.

Scalabilité

La faculté à changer facilement la capacité de la production existante, en réarrangeant le système de production existant et/ou changer la capacité de production des composants reconfigurables (machines) dans ce système.

Convertibilité

La capacité à transformer facilement les fonctionnalités des systèmes, machines et commandes existants, pour les adapter à de nouveaux besoins.

Le mode de fonctionnement optimal est configuré en lots qui doivent être terminés en une journée, avec un court temps de conversion entre les lots. Il nécessite des programmes pièce, des outils et des accessoires changeants et nécessite également un ajustement du degré de liberté passif.

Personnalisation

Il a deux aspects: flexibilité personnalisée et contrôle personnalisé.

La flexibilité personnalisée offre uniquement la flexibilité nécessaire pour des pièces spécifiques en réduisant les coûts. Et le contrôle personnalisé est réalisé en intégrant des modules de contrôle avec l'ajout de la technologie de l'architecture ouverte.

Intégrité

La capacité d'intégrer des modules rapidement par un ensemble d'interfaces mécaniques, informationnelles et de commande qui offrent en même le temps l'intégration et la communication.

Les modules MACHINE sont conçus avec une interface pour l'intégration des composants. Cette performance est prédite par la performance donnée de ses composants et par les interfaces des modules matériels et logiciels.

Diagnostic

La capacité à lire automatiquement l'état courant du système et des commandes afin de détecter et diagnostiquer les causes à l'origine des défaillances, et par conséquent,

corriger rapidement les défauts opérationnels.

La détection d'une qualité de pièce inacceptable est essentielle pour réduire le temps d'accélération dans RMS.

2.4.3 Les phases de conception d'un RMS

La conception d'un RMS passe par plusieurs phases pour garantir efficacement les caractéristiques citées dans la section précédente. Pour discuter spécifiquement et concrètement ces phases critiques, un système de robot reconfigurable comme le montre la figure 2.10 est pris comme exemple. Ces phases comprennent la conception de l'architecture, la conception de la configuration et la conception des contrôles.

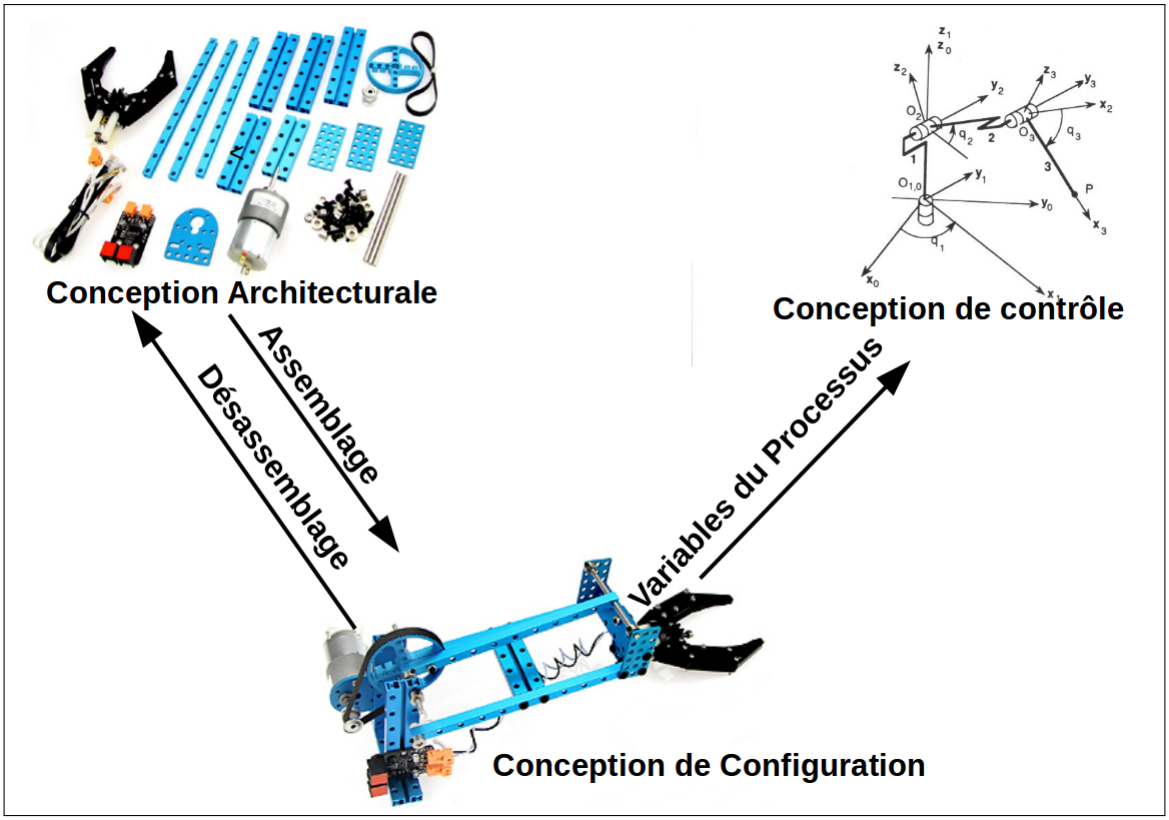

Figure 2.10: Les différents phases de conception d'un RMS (Exemple robot)

La conception de l'architecture détermine les composants du système et leurs interactions. Les composants du système sont des modules encapsulés. Les interactions sont les options lorsque les modules sont assemblés. L'architecture RMS doit être conçue pour

produire autant de variantes de systèmes que possibles, afin que le système puisse gérer les changements et les incertitudes de manière rentable. La conception de l'architecture est impliquée dans la phase de conception du système.

La conception de la configuration détermine la configuration du système sous une architecture système donnée pour une tâche spécifique. Une configuration est un assemblage des modules sélectionnés; une configuration peut remplir la tâche donnée de manière optimale. La conception de la configuration est impliquée dans la phase d'application du système.

La conception du contrôle détermine les variables de processus appropriées (déplacements et vitesses des articulations, etc., d'un module conjoint), de sorte qu'une configuration peut être utilisée pour remplir la tâche de façon satisfaisante. Conception de contrôle est impliquée à la phase de fonctionnement du système

2.4.4 La reconfiguration dynamique des systèmes manufacturiers

La reconfiguration dynamique dans les systèmes manufacturiers, est une reconfiguration software que plutôt hardware, vu qu'il est très simple de modifier les logiciels que modifier dynamiquement le hardware sauf dans certains cas ou la reconfiguration hardware s'applique sur les paramètres de la machine et non pas sur la machine elle-même, si la machine l'autorise.

Dans certains travaux la reconfiguration sert à rééquilibrer dynamiquement une ligne d'assemblage lorsqu'une perturbation survient.

Le processus de reconfiguration est un processus de réorganisation matérielle et/ou logiciel du système. L'objectif de cette réorganisation est de pouvoir assurer la production en réalisant un compromis entre les objectifs de production et l'état du système. Ce processus de reconfiguration peut être déclenché par deux catégories d'événements liés soit aux produits soit aux ressources de production.

La mise en œuvre du processus de reconfiguration dépend de deux paramètres : l'événement déclencheur et les contraintes temporelles s'exerçant sur le système lorsque

survient cet événement. Deux situations complémentaires peuvent être considérées : le cas du lancement d'une nouvelle production avec le système à l'arrêt et le cas d'une défaillance survenant sur un système en cours de fonctionnement.

La conception d'un RMS nécessite plusieurs techniques et méthodes. La modularité est l'une de ces techniques. Selon Baldwin et Clark [25], la modularité ou la modularisation d'un point de vue technique aide à gérer la complexité.

2.5 Objectifs et effets de la reconfiguration dynamique

2.5.1 Les objectifs de la reconfiguration dynamique

A) Reconfiguration correctionnelle

La reconfiguration correctionnelle est nécessaire si une application en cours d'exécution ne se comporte pas correctement ou comme prévu. Elle consiste d'abord, à identifier les composants de l'application qui sont responsables du mauvais comportement. Ensuite, à les remplacer par de nouveaux composants supposés avoir un comportement plus adéquat. Les nouveaux composants doivent fournir les mêmes fonctionnalités que les anciens et doivent se contenter simplement de corriger leurs défauts.

B) Reconfigration évolutive

Une application est initialement développée pour fournir un certain nombre de services à ses clients. Les besoins de ses clients peuvent changer, et de nouveaux services deviennent nécessaires. Pour répondre à cette évolution des besoins, l'application doit être étendue avec de nouvelles fonctionnalités. Dans une application à base de composants, l'évolution peut être réalisée :

- Soit en ajoutant de nouveaux composants pour assurer les nouvelles fonctionnalités requises.
- Soit en remplaçant les composants déjà existant avec d'autres composants of-

frant plus de fonctionnalités. Il faut garantir dans ce cas, que le remplacement ne met pas en cause l'intégrité de l'application.

C) Reconfiguration adaptative

Une application est en générale destinée à être déployée et exécutée dans un contexte particulier. Ce contexte, peut se traduire par l'environnement où l'application doit s'exécuter, peut changer au fil du temps. Certains changements dans le contexte peuvent rendre le comportement de l'application incohérent. Pour cette raison, l'application doit être reconfigurée pour prendre en compte les nouveaux paramètres de son contexte d'exécution.

D) Reconfiguration perfective

L'objectif de ce type de reconfiguration est d'améliorer les performances d'une application, même si son comportement est cohérent.

2.5.2 Les effets de la reconfiguration dynamique

La reconfiguration dynamique d'un système implique des modifications sur ce dernier, qui peuvent être classées en plusieurs catégories [10]:

A) Modification de l'architecture

Cette dernière peut se faire en ajoutant de nouvelles instances de composants, en supprimant des instances déjà existantes, ou en changeant les interconnexions entre les instances.

B) Modification de l'implémentation

Ce type de modification est généralement introduit pour des raisons de correction, d'adaptation aux changements de l'environnement ou d'amélioration de performances.

C) Modification des interfaces

La modification de l'interface d'un composant correspond à la modification de la liste des services qu'il fournit. Ceci se traduit par l'ajout de nouvelles interfaces, leur suppression initialement supportées par le composant, ou par la modification des opérations déclarées dans ces dernières.

D) Modification de la localisation (migration)

Elle correspond à la migration des instances d'un site d'exécution, vers un autre site pour la répartition des charges à titre d'exemple. Ce type de modification n'affecte pas l'architecture logique de l'application, cependant, la communication entre les instances migrées et les autres instances, doit être adaptée selon la nouvelle localisation des instances migrées.

2.6 Conclusion

À travers ces trois systèmes pris comme exemple nous pouvons constater que la reconfiguration dynamique est devenue une tâtche cruciale dans les différents systèmes hardware et software. Mais si nous allons voir les système par une vue beaucoup plus abstraite, nous pouvons comprendre que n'importe quel système reconfigurable est constitué d'un :

- Ensemble de modules,
- Ensemble de connecteurs,
- Ensemble d'interfaces,
- Ensemble de règles de gestion du système.

L'objectif principal de notre recherche est de permettre à un système de s'adapter aux différentes variations de son environnement, sachant que chaque variation nécessite

une configuration très spécifique. Pour cela, nous devrons trouver une approche pour la reconfiguration dynamique d'un système modulaire en vérifiant plusieurs critères tels que:

- L'unité de base est le module qui représente la plus petite unité indépendante du système.
- L'espace des configurations doit être ouvert ce qui veut dire qu'il engendre tous les états possibles du système même s'ils ne sont pas définis au préalable.
- L'ensemble des systèmes qui peuvent être reconfiguré doit être important vu que nous cherchons à mettre un moyen global et général.

Le passage d'une configuration à une autre est effectué à l'aide des différentes opérations d'ajout, de suppression et de modification, appliqué sur les modules du système. Pour mieux utiliser ces opérations nous devrons savoir quand et comment les appliquer.

Dans cette thèse nous allons donner des réponses à ces deux questions dans le but de compléter le modèle de reconfiguration dynamique.

Chapitre 3

Les systèmes multi-agents

Les systèmes multi agents (SMA) sont un paradigme de développement software qui offre plusieurs caractéristiques et des qualités impressionnantes, qui peuvent être exploitées pour résoudre le problème de la reconfiguration dynamique des systèmes pervasifs, dans ce chapitre nous étudierons et présenterons ce paradigme afin de pouvoir l'utiliser pour implémenter un système pervasif reconfigurable dynamiquement.

3.1 Introduction

La programmation par agent est considérée comme l'un des paradigmes les plus importants dans le monde du développement software, que ce soit dans le domaine de l'intelligence artificielle, des systèmes distribués, de la robotique, etc. Ce nouveau paradigme introduit la notion de l'autonomie des individus (les agents), qui permet de garantir les caractéristiques les plus importantes dans le génie logiciel, qui sont la modularité et la réutilisabilité, ce qui facilite la tâche pour les développeurs et augmente l'efficacité software par des taux très élevés, puisque ce paradigme offre la possibilité de modéliser les différents acteurs de l'application dans leur aspect statique et dynamique.

3.2 Les agents software

L'agent, ou l'agent software est l'unité fondamentale de composition des systèmes multi-agents, toutes les définitions se mettent d'accord qu'un agent est essentiellement un composant software autonome et offre des interfaces interopérables avec des systèmes arbitraires et qui se comporte plus ou moins comme un agent humain.
Dans la littérature nous pouvons trouver plusieurs définitions pour le terme agent. Dans [26]un agent est défini comme suit :

Définition *un agent est toute chose capable de percevoir son environnement à travers des capteurs et agir sur cet environnement à travers des effecteurs.*

Selon cette définition, un agent est une entité (physique ou virtuelle) qui détecte son environnement et agit par-dessus.

- Les entités physiques qui peuvent être considérées comme des agents, sont généralement des composants ou des dispositifs qui ont la capacité de fonctionner d'une manière indépendante des autres dispositifs. Dans notre cas, il s'agit d'un dispositif pervasif mis en œuvre par un système embarqué autonome.

- Les entités virtuelles qui peuvent être considérées comme un agent, sont généralement des éléments logiciels qui reçoivent des entrées à partir de leurs environnements et produisent des sorties qui déclenchent une action sur lui.

Cette perception permette de voir l'agent comme une entité intelligente qui réagit avec son environnement selon une base de connaissances propres à l'agent lui-même et qui lui permet aussi de prendre des décisions et être autonome dans son fonctionnement, la figure 3.1 illustre comment la perception de l'agent à son environnement peut créer une sorte de communication agent-environnement.

Mais la communication de ces entités n'est pas limitée seulement entre agent et son environnement, Il existe un autre type de communication entre les agents software, c'est

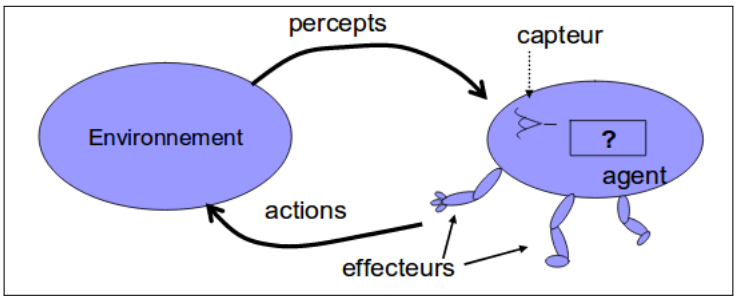

Figure 3.1: Communication agent-environnement

une communication de type agent-agent, cette dernière a comme but une meilleure modélisation des systèmes complexes nécessitant un traitement collaboratif entre l'ensemble des agents. Dans ce contexte les agents peuvent communiquer entre eux à l'aide d'un ensemble de standards dont l'objectif est de résoudre le problème global par la décomposition d'un système complexe en sous-systèmes autonomes interagissant, où chaque agent doit se focaliser sur une partie de ces sous-systèmes au lieu d'être obligé de considérer le système dans sa globalité, nous parlons ici d'un traitement collaboratif. Dans certains cas, nous trouvons un ensemble d'agents qui ont les mêmes tâches et qui se focalisent sur le même sous-système, ceci peut avoir comme objectif la recherche d'une meilleure solution parmi les solutions proposées par l'ensemble des agents, et dans ce cas nous parlons d'un traitement compétitif.

Il existe une autre définition plus générale et qui regroupe tous les caractéristiques d'un agent, elle se présente comme suit :

Définition *un agent est une entité réelle ou virtuelle, évoluant dans un environnement, capable de le percevoir et d'agir dessus, qui peut communiquer avec d'autres agents, qui exhibe un comportement autonome, lequel peut être vu comme la conséquence de ses connaissances, de ses interactions avec d'autres agents et des buts qu'il poursuit [27].*

Par cette définition nous pouvons dire que l'autonomie d'un agent signifie qu'il n'est pas dirigé par des commandes d'un utilisateur ou qui proviens d'un autre agent, par contre c'est une conséquence de son propre comportement qui lui a était attribué au

moment de son développement, ce dernier ne dépend pas seulement des valeurs des entrées capturées par l'ensemble des capteurs, mais aussi par les objectifs attribués à cet agent, cela lui donne une sorte de distinction par rapport aux autres agents. Mais l'autonomie n'est pas seulement comportementale, elle porte aussi sur les ressources telles que l'énergie, la CPU, la mémoire, l'accès à certaines ressources informatiques, etc. La figure 3.2 illustre le Framework générale d'un agent.

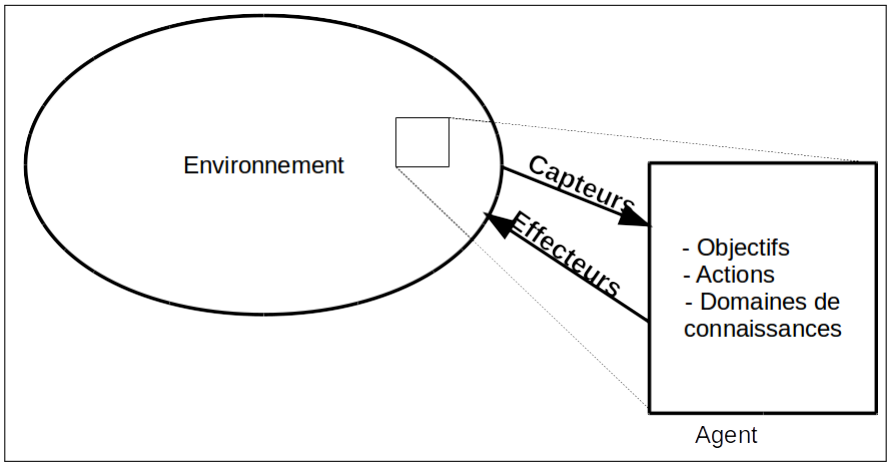

Figure 3.2: Agent framework

Un environnement dans ce cas, peut-être n'importe quel système physique réel, un système d'exploitation, un réseau Internet, ou même une combinaison de ces systèmes dont le but est de former un environnement pervasif pour l'agent dans lequel il va accomplir ces objectifs à travers les services qu'il offre.

Si un agent répond simplement, en temps réel, à des changements dans l'environnement et convertit de manière réactive ses entrées sensorielles en actions, cet agent est dit réactif. Les agents réactifs ne conservent généralement pas l'état interne (un simple exemple d'état interne consiste à conserver un ensemble d'entrées sensorielles antérieures) de l'agent et ne prédisent pas l'effet des actions. Par contre, si un agent maintient l'état interne et prédit les effets de ses actions, ou plus généralement inclut une sorte de raisonnement (se comporte plutôt comme "pensant"), cet agent se comporte délibérément et s'appelle agent délibératif (ou agent qui garde la trace de l'environnement).

3.2.1 propriétés des agents software

D'après les définitions précédentes, nous pouvons dire qu'un agent est une entité de traitement, autonome et qui a des objectifs. Ses caractéristiques offrent un ensemble de qualité et des avantages aux agents que nous pouvons citer :

- un agent peut communiquer avec n'importe quel autre agent,
- un agent fournit un ensemble de services qui sont disponibles pour tous les autres agents du système,
- il est de la responsabilité de chaque agent de limiter son accessibilité par rapport aux autres agents,
- il est de la responsabilité de chaque agent de définir sa relation, ses contrats, etc. avec d'autres agents. Ainsi, un agent "sait" directement (par l'intermédiaire de ses connaissances) l'ensemble des agents avec lesquels il peut interagir,
- chaque agent a son propre nom et son propre chemin d'accès depuis l'extérieur (la notion d'ID d'agent est bien connue chez tous ses concepteurs). Par conséquent, les agents sont supposés être autonomes et aucune contrainte n'est imposée sur la façon dont ils interagissent,
- chaque agent possède un cycle de vie illustré par la figure 3.3

Dans ce qui suit, nous étudierons le comportement d'un ensemble d'agents qui interagissent entre eux dans un système unique appelé *système multi-agent.*

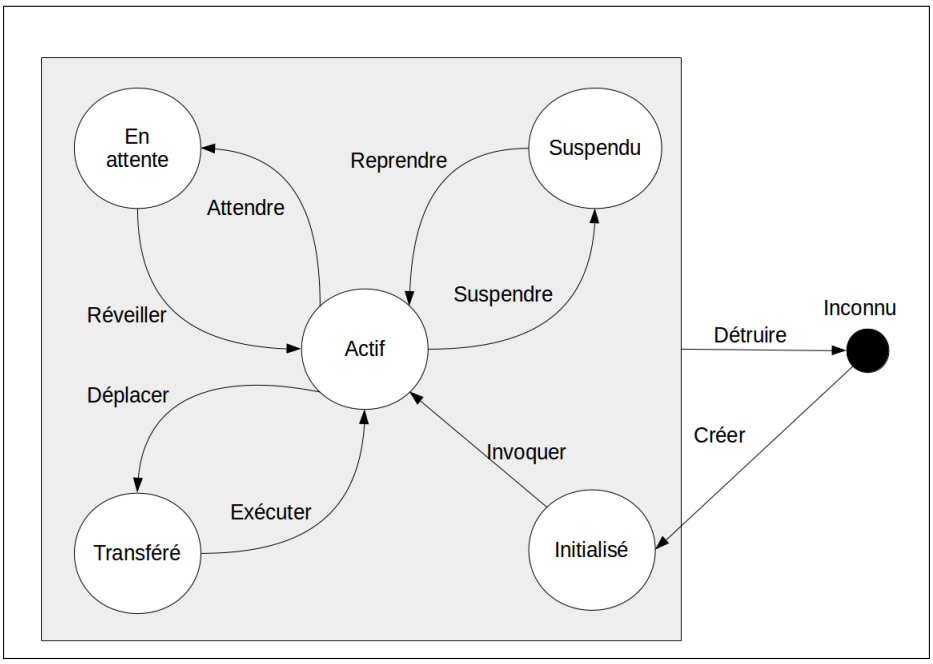

Figure 3.3: Cycle de vie d'un agent software

3.3 Approche Multi-agents

Le paradigme des systèmes multi-agents (SMA) devient progressivement l'un des paradigmes les plus efficaces pour le développement d'applications complexes, en particulier dans les scénarios distribués où la communication entre différentes entités est l'une des principales caractéristiques. Le paradigme repose principalement sur l'utilisation d'agents coopératifs, où chaque agent gère une connaissance particulière ainsi qu'un petit ensemble de tâches spécialisées, il est aussi capable de coopérer avec les autres agents afin d'atteindre certains objectifs au niveau du système, ce qui produit un degré élevé de flexibilité[28]. La coopération des agents, peut être un critère de classification pour les SMA, la figure 3.4 illustre les différentes classes obtenues à la base de ce critère.

Un SMA est indépendant si chaque agent individuel poursuit ses propres objectifs indépendamment des autres. Un SMA est discret s'il est indépendant, et si les objectifs des agents n'ont aucun rapport les uns avec les autres. Le SMA discret n'implique aucune coopération. Cependant, les agents peuvent coopérer sans avoir l'intention et si tel est le cas, la coopération est émergente. Le terme délibératif de la figure 3.4 ne doit pas être

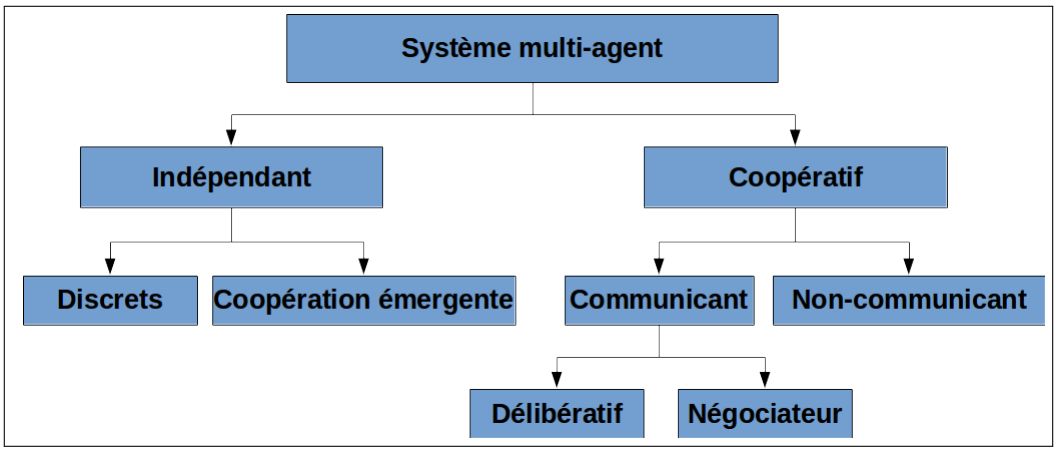

Figure 3.4: Classification des SMA selon le critère de coopération

confondu avec le terme d'agent délibératif, dans la figure il fait référence à l'idée que les agents planifient conjointement leurs actions de manière à coopérer les uns avec les autres. La coopération au sein d'un SMA peut être réalisée de trois manières :

- par une conception explicite (le concepteur des agents conçoit volontairement les comportements de l'agent pour que la coopération se produise),
- par adaptation (les agents individuels apprennent à coopérer),
- par l'évolution (la coopération des agents individuels évolue à travers une sorte de processus évolutif).

En fait, c'est le comportement social des agents et leur interaction, plus que leurs capacités individuelles, qui rendent les systèmes multi-agents si puissants et polyvalents dans de nombreux scénarios [29]. Dernièrement, les systèmes multi-agents sont devenus de plus en plus sophistiqués, avec un potentiel croissant de gestion de gros volumes de données et de coordination des opérations dans de nombreuses organisations. Dans ce contexte, l'un des problèmes liés au calcul distribué est l'échange d'informations au sein du système. Ainsi, l'architecture multi-agent doit nécessairement fournir une couche de communication robuste avec des mécanismes appropriés basés sur les messages qui permettent les processus

d'interaction, puisqu'ils conditionnent la façon dont les agents intelligents peuvent interagir et se coordonner les uns avec les autres.

Les systèmes multi-agents (SMA), sont un ensemble organisé d'agents qui interagissent entre eux dans un environnement commun, il est constitué d'une ou plusieurs organisations qui structurent les règles de cohabitation et de travail collectif entre agents.

3.3.1 Caractéristiques des systèmes multi-agents

- Sécurité des applications: la possibilité que tous les agents puissent communiquer sans contrôle externe peut entraîner des problèmes de sécurité. Lorsque tout agent peut interagir librement, il est de la responsabilité des agents (et donc du concepteur de l'application) de vérifier la qualification de ses interlocuteurs et d'implémenter des contrôles de sécurité. Parce qu'il n'y a pas de gestion de sécurité "générale", il est facile pour un agent d'agir en tant que pirate et d'utiliser le système frauduleusement.

- Modularité: en génie logiciel classique, il s'agit d'un ensemble d'entités qui travaillent étroitement ensemble, et qui sont regroupées en modules ou "packages". Pour chaque module, les règles de visibilité sont définies. Certaines entités peuvent être vues par d'autres paquets (et même par l'ensemble du logiciel) tandis que d'autres, dites entités privées, sont cachées et donc inaccessibles depuis l'extérieur du paquet. Il devrait être important de proposer un moyen de regrouper les agents qui doivent travailler ensemble. Cependant, cette proposition ne doit pas rester sur des bases statiques, mais proposer un moyen de regrouper des agents actifs qui travaillent ensemble.

- Framework / approche de composant: l'ingénierie logicielle moderne a montré l'importance du concept du Framework / composant. Un Framework est une architecture abstraite dans laquelle des plugins de composants sont développés. Il est souvent nécessaire de définir des sous Frameworks. Malheureusement, dans ACMAS, il n'y a qu'un seul Framework, la plate-forme elle-même, et il n'est pas possible de décrire un

sous Framework dans lequel des interactions spécifiques pourraient être construites.

L'une des caractéristiques les plus importantes dans un système multi-agent est la communication, vu qu'un agent a besoin de communiquer avec l'utilisateur humain, les ressources et les autres agents dont le but est de collaborer, négocier, coopérer avec eux. Cette interaction est possible grâce à des langages de communication précise tels que, le langage KQML(Knowledge Query and Manipulation Language) [30] qui définit un ensemble de protocoles et de standards permettant l'échange d'informations et de connaissances par le biais d'un nombre fini de verbes performatifs.

Actuellement, le langage de communication le plus utilisé est FIPA-ACL (Foundation for Intelligent Physical Agents - Agent Communication Langage) [31] qui offre la possibilité d'utiliser des différents messages et manager les conversations à travers des protocoles prédéfinis, dans ce cas un agent doit être capable de dialoguer avec les autres agents, pour cela il doit être :

- passif : il doit accepter les questions des autres agents et répondre à leurs questions,
- actif : il doit proposer et envoyer des interrogations.

Dans ce contexte, nous pouvons trouver deux modes de communication :

- communication directe : partage de l'information via environnement commun,
- communication indirecte : par envoi de messages.

3.3.2 Communication entre agents

Actuellement plusieurs systèmes physiques (Hardware) ont adopté la notion des agents dans leurs conception et développement, certain entre eux sont dotés d'un environnement dynamique, ce qui rend la communication des agents une caractéristique fondamentale dans le développement des agents software, car elle leur permet de publier

leurs services à travers le réseau et enrichir leurs connaissances à l'aide des informations acquises par l'interaction des autres agents avec leur monde physique.

Sans communication, l'agent n'est qu'un individu isolé, refermé sur sa boucle perception-délibération-action. Grâce à la communication, les agents peuvent coopérer, coordonner leurs actions, effectuer des tâches communes et fournir des solutions pour des systèmes complexes. La communication est exprimée comme une forme d'interaction dans laquelle la relation dynamique entre les agents s'exprime par l'intermédiaire de médiateurs, les signaux une fois interprétés, vont produire des effets sur ses agents. Cela nécessite généralement un langage commun, un langage de communication d'agent ou une liste de contrôle d'accès. Plusieurs travaux ont été réalisés pour développer des ACL qui soient déclaratives, syntaxiquement simples et lisibles par les gens. KQML et FIPA-ACL sont deux des ACL (Agent Communication Langage) les plus utilisées dans les systèmes multi-agents. Ces langues ont très bien réussi à faciliter la communication et la coordination des agents logiciels dans une variété de domaines, y compris la prise de décision organisationnelle [32], et même la maintenance des aéronefs [33]. Cette approche de la communication inter-agent, qui est bien adaptée aux communications liées à la négociation ou au transfert d'informations de haut niveau. Mais, elle présente un inconvénient important qui nécessite souvent le transfert de données ou de systèmes de bas niveau avec des limites de temps et de bande passante.

La définition d'un ACL inclut la définition des deux éléments suivants:

- la syntaxe, c'est-à-dire la manière dont les mots individuels sont assemblés,
- la sémantique, c'est-à-dire la signification des actes communicatifs.

La syntaxe d'un ACL

La syntaxe est un composant important dans une langue, elle décrit la manière dont les mots de la langue peuvent être assemblés pour former un énoncé. Il est possible de distinguer entre syntaxe abstraite et syntaxe concrète. La syntaxe abstraite décrit

à un niveau élevé la structure des énoncés et dépend de la complexité que présente le langage au niveau sémantique. La syntaxe concrète est la syntaxe implémentée réellement. Étant donné la syntaxe abstraite, il peut y avoir plusieurs façons pour la transformer en une syntaxe concrète. Ce qui est crucial, c'est que nous pouvons trouver des différentes syntaxes concrètes interchangeables, issues de la même syntaxe abstraite.

- Abstraction d'une syntaxe : une syntaxe abstraite valide pour un langage de communication d'agent doit inclure les paramètres suivants :
  - expéditeur : l'identificateur non ambigu de l'agent qui émet le message;
  - récepteur : l'identificateur non ambigu de l'agent recevant le message;
  - contenu : le sujet du message;
  - performatif : la fonction illocutoire des messages, elle peut être (assertifs, directifs, promissifs, expressifs, déclaratifs, interrogatifs);
  - langue : la syntaxe concrète adoptée;
  - ontologie : le dictionnaire des termes et des relations entre les termes adoptés;

  Il existe d'autres façons plus formelles de définir la syntaxe abstraite d'une langue. Par exemple, la syntaxe peut être définie à l'aide d'une formalisation algébrique : un message est un élément du produit cartésien de n ensembles; ou il est possible de définir la syntaxe par une formalisation logique : un message est une proposition faite par un prédicat à n arguments où les arguments appartiennent à des domaines différents.

- Syntaxe concrète : étant donné une syntaxe abstraite, il est possible de définir plusieurs syntaxes concrètes différentes. Il est important que les différentes syntaxes concrètes définies soient toutes interchangeables, car ce n'est pas une tâche facile de convaincre tous les concepteurs d'utiliser la même langue, en fait chacun a ses propres partisans.

La fondation pour les agents physiques intelligents (FIPA)[34], est une organisation internationale qui se consacre au développement des normes dans le domaine des agents, elle propose trois formes de représentation différentes pour les messages du langage de communication des agents.

- La "spécification de chaîne de caractères" qui est très similaire à une syntaxe abstraite. Un message est une séquence de chaînes de caractères, c'est-à-dire:

  message = "(" MessageType, MessageSlot*")"

  Où "MessageType" est l'un des actes de communication disponibles dans la bibliothèque FIPA.

  "MessageSlot" inclut: sender, receiver, content, reply-with, reply-by, in-reply-to, reply-to, language, encoding, ontology, protocol, conversation-id, UserDefineSlot.

- La "spécification XML" qui est définie à l'aide du langage XML (eXtensible Markup Language) pour la spécification de la structure et des données des documents sur le Web.

- La "Spécification Bit-Efficace" qui est un codage peu efficace de la syntaxe abstraite.

La sémantique d'un ACL

En supposant que la syntaxe du langage soit donnée, il est nécessaire de définir le langage pour le contenu, c'est-à-dire le vocabulaire avec les mots et leur signification, qui peuvent être utilisés par les agents pour interpréter les messages échangés. Le terme lexique vient de la tradition philosophique et équivaut à la notion de dictionnaire.

Un dictionnaire comprend un ensemble de symboles et leurs définitions ainsi qu'une définition de leurs relations. En logique, ces relations sont exprimées par un ensemble d'axiomes, alors qu'en Intelligence Artificielle, et au cours des dix dernières années, elles

sont définis par un ensemble des ontologies. Traditionnellement, le terme ontologie se réfère à la science de l'être. Cependant, en informatique et dans d'autres disciplines, les ontologies sont interprétées comme un dictionnaire de termes et de relations formels entre termes et dans certains cas, elles peuvent être interprétées et utilisées par des systèmes logiciels, en particulier les agents.

La fondation pour les agents physiques intelligents, propose différentes représentations pour le langage du contenu adopté par FIPA ACL, c'est-à-dire le langage de communication d'agent standard, basé sur les états mentaux des agents, proposé par FIPA. Tous ces langues du contenu expriment à l'aide d'une formalité choisie, les objets, les propositions, les relations et les fonctions disponibles dans la langue.

- L'un des langages des contenus possibles proposés par FIPA est le langage sémantique (SL). Dans ce langage une expression du contenu peut être: une proposition, c'est-à-dire une formule bien formée à laquelle nous pouvons attribuer une vraie valeur dans un contexte donné; une action qui peut être réalisée en une seule action ou en séquence ou en alternative à d'autres actions; une expression de référence d'identification (IRE) qui identifie un objet dans le domaine. Ces cas peuvent produire par composition d'autres expressions valides du contenu.

- Un autre langage du contenu possible est le FIPA Contraint Choice Language (CCL). Ce langage du contenu est basé sur la représentation de problèmes de choix tels que les problèmes de satisfaction de contraintes (CSP) et il permet : la représentation de problèmes, la collecte d'informations, la fusion d'informations et l'accès à des techniques de résolution de problèmes.

- Une description d'un autre langage de contenu FIPA est basée sur le Knowledge Interchange Format (KIF) [35]. Dans KIF, l'univers du discours, c'est-à-dire l'ensemble des objets supposés exister dans le monde, change sur la base des utilisateurs, mais chaque univers de discours doit inclure certains objets de base: tous les nombres,

tous les caractères ASCII , toutes les chaînes finies, les mots, toutes les listes finies d'objets dans l'univers du discours, un objet spécial pour traiter les cas où une fonction est appliquée à des arguments pour lesquels la fonction n'a aucun sens.

- Une autre description d'un langage du contenu FIPA peut être basée sur le Resource Description Framework (RDF). Les principaux points forts du langage RDF résident dans son extensibilité, sa réutilisabilité, sa simplicité et le fait qu'il s'agit d'une norme pour les applications web. Le modèle RDF propose le langage XML (eXtensible Markup Language) en tant que syntaxe de codage. Le langage de contenu RDF couvre la définition des objets, des propositions et des actions. En fait, bien que le FIPA n'exige pas qu'un langage de contenu puisse représenter des actions, beaucoup d'actes de communication nécessitent des actions dans leur contenu et le schéma RDF peut être étendu pour les exprimer.

3.3.3 Diagramme d'interaction entre agents

Dans les systèmes multi-agents la communication entre agents est définie par un ensemble des interactions, c'est le deuxième aspect de la conception architecturale d'un SMA après la définition des types d'agents. Elle consiste à spécifier l'interaction entre les agents, en capturant les aspects dynamiques du système. Cela s'appuie à la fois sur les types d'agent et sur les descripteurs de scénario défini par les spécifications du système.

Le développement de la spécification de l'interaction entre agents passe par les étapes suivantes :

1. développer des diagrammes d'interaction à partir de scénarios de cas d'utilisation,
2. généraliser les diagrammes d'interaction aux protocoles d'interaction,
3. développer des descripteurs de protocole et de message.

Comme les scénarios de cas d'utilisation, les diagrammes d'interaction ne décrivent

qu'un seul exemple du comportement du système. Les protocoles d'interaction entièrement spécifiés sont ensuite développés à partir de ceux-ci et sont l'artefact de conception final.

Diagramme d'interaction à partir des scénarios

Les diagrammes d'interaction sont empruntés directement à la conception orientée objet, mais montrent une interaction entre les agents plutôt qu'avec les objets. Le processus de développement de diagrammes d'interaction consiste à prendre les scénarios développés dans la phase de spécification et à construire des diagrammes d'interaction correspondants. La figure 3.5 illustre les étapes de ce processus.

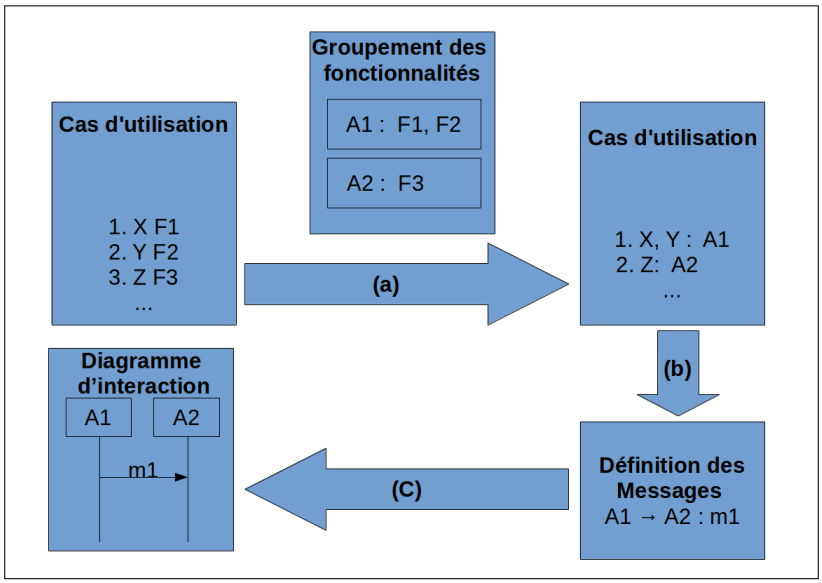

Figure 3.5: Développement du diagramme d'interaction à partir d'un cas d'utilisation

Cela implique

(a) de remplacer chaque fonctionnalité par l'agent qui l'inclut;

(b) l'insertion d'une communication entre les agents lorsque cela est nécessaire;

(c) l'expression du résultat sous la forme d'un diagramme d'interaction.

La notation utilisée pour les diagrammes d'interaction est standard. Chaque agent a une ligne de vie, représentée par une ligne verticale avec le nom de l'agent dans une case en haut de la ligne.

Les messages sont représentés par des flèches horizontales entre les lignes de vie avec une brève description du message au-dessus de la flèche.

Dans certains systèmes il est important de montrer les actions et les perceptions sur les diagrammes d'interaction. Par exemple, dans le cas d'un réseau de capteurs une interaction est souvent déclenchée par un percept (événement capturé par le capteur physique). Il y a un certain nombre de façons dont les actions et les percepts peuvent être représentés dans les diagrammes d'interaction. Ils peuvent être montrés comme des messages d'une ligne de vie invisible (la figure 3.6 (a)), ils peuvent être affichés comme texte sur la ligne de vie de l'agent concerné (la figure 3.6 (b)) ou peuvent être affichés comme messages dans un environnement explicite ligne de vie (la figure 3.6 (c)).

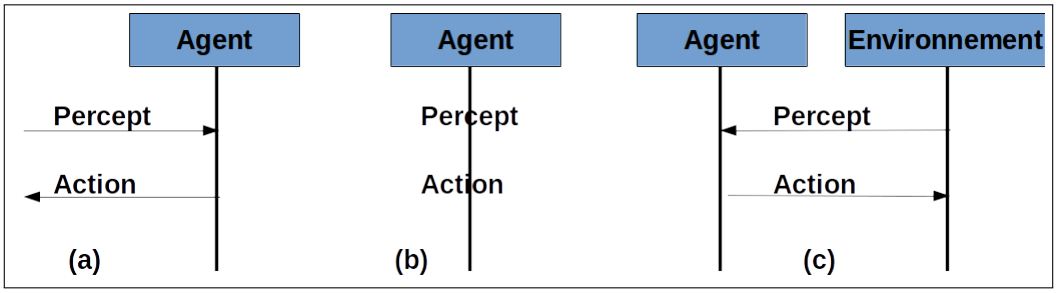

Figure 3.6: Les différentes représentation de l'interaction percept-action

Protocoles d'interaction à partir de diagramme d'interaction

Comme avec les scénarios, nous nous attendrions à avoir seulement un ensemble représentatif de diagrammes d'interaction, pas un ensemble complet. Afin d'avoir des interactions complètes et précisément définies, nous passons des diagrammes d'interaction aux protocoles qui définissent exactement quelles séquences d'interaction sont valides dans le système. Parce que les protocoles doivent montrer toutes les variations, ils sont souvent plus grands que le diagramme d'interaction correspondant et peuvent avoir besoin d'être scindés en plusieurs petits morceaux.

Afin de développer un protocole complet d'interaction, nous avons besoin d'une notation pour décrire les protocoles d'interaction. Puisque les protocoles d'interaction incluent le séquençage, les choix, l'itération et d'autres structures de contrôle, la notation

doit être très expressive. Une gamme de notations existe pour décrire les protocoles incluant les diagrammes d'activité UML, AUML (Agent UML) [36], et les réseaux de Pétri (par exemple, [37], [38]). Il existe aussi une version révisée de l'agent UML [39].

Le développement de protocoles se fait en considérant des alternatives. Pour chaque message (ou percept) qu'un agent reçoit, il faut définir tous les messages possibles que l'agent pourrait envoyer en réponse, ensuite il faut répéter le même processus pour ces messages. Plus généralement, il nous faut aussi prévoir les continuations possibles, c'est-à-dire les séquences de messages possibles.

Développement des descripteurs de protocoles et de messages

Comme pour chacune des entités de conception finales du système, nous avons des descripteurs pour les protocoles et pour les messages. Les descripteurs nous permettent à la fois de collecter des informations existantes dans d'autres endroits et de spécifier des détails supplémentaires sur une entité particulière.

Pendant le développement du protocole, un nombre important de messages seront identifiés, qui sont généralement des messages entre les agents du système. Dans certains cas, il existe des messages inter-agents qui ne nécessitent pas de protocole, car il s'agit d'un seul message ou d'une paire de réponses de message. Ils peuvent être spécifiés comme des protocoles dégénérés, mais ils peuvent également être spécifiés simplement comme des messages qui ne nécessitent pas de protocole.

Dans la conception détaillée, il y aura également des messages internes supplémentaires ajoutés à la conception.

Les descripteurs de messages doivent clairement contenir des informations sur le contenu du message. Il est également utile d'indiquer le but du message, qui peut être, par exemple, de transférer le contrôle à un autre agent ou une autre fonctionnalité, de demander un service ou de mettre à jour des informations.

3.3.4 Technologies de développement des systèmes multi-agent

La plupart des langages décrits dans ce chapitre ont une plate-forme sous-adjacente qui implémente la sémantique du langage pour la programmation d'un agent. Cette plate-forme est conçue selon le modèle standard défini par la FIPA et illustré par la figure 3.7.

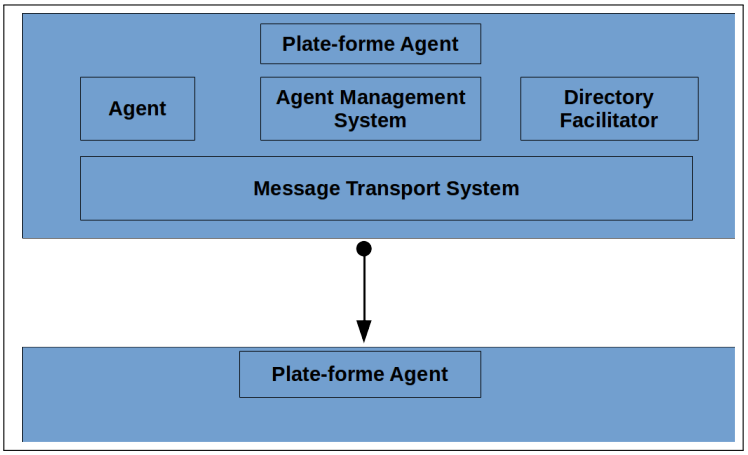

Figure 3.7: L'architecture de référence définie par la FIPA pour les plates-formes Agent

L'Agent Management System (AMS) est l'agent qui contrôle la supervision sur l'accès et l'utilisation de la plate-forme d'agent. Chaque plate-forme contient un seul AMS. L'AMS fournit un service de pages blanches et il gère le cycle de vie des agents, en maintenant un répertoire d'identificateurs d'agent (AID) et son état. Chaque agent doit s'inscrire auprès d'un AMS afin d'obtenir un AID valide.

Le Directory Facilitator (DF) est l'agent qui fournit le service de page jaune par défaut dans la plate-forme, en maintenant tous les services offerts par l'ensemble des agents inscrivent auprès de l'AMS.

Le Message Transport System, appelé aussi Agent Communication Channel (ACC), est le composant logiciel contrôlant tous les échanges de messages au sein de la plate-forme, y compris les messages vers / depuis les plates-formes distantes.

Nous pouvons trouver plusieurs Framework qui respectent ce modèle. Parmi eux, il existe certains Frameworks qui sont mis en œuvre mais qui ne sont pas si fortement liés

à un langage de programmation particulier. D'autres Framework sont plus soucieux de fournir un soutien pour le développement de certains aspects tels que la communication et la coordination des agents. Dans cette section, nous nous concentrons sur ces Framework, après avoir choisi TuCSoN, JADE et DESIRE comme exemples illustratifs.

TuCSoN

TuCSoN (Tuple Centre Spread over the Network) est un Framework chargé de la coordination SMA, basé sur un modèle et une infrastructure connexe fournissant des services programmables polyvalents pour soutenir la communication et la coordination des agents [40]. Le modèle est basé sur des centres de tuple comme des abstractions programmables d'exécution dont le comportement de coordination peut être dynamiquement spécifié avec un langage logique appelé ReSpecT [41]. Les centres de tuple sont un exemple d'artefacts de coordination [42], c'est-à-dire des entités de premier ordre peuplant l'environnement de travail coopératif des agents. Ces abstractions sont également utilisées dans la méthodologie SODA [43] comme éléments de base pour la conception du niveau social et de l'environnement dans un SMA.

La technologie TuCSoN est disponible en tant que projet open source (http://tucson.sourceforge.net). Il est entièrement basé sur Java et est composé de: une plate-forme d'exécution à installer sur les hôtes pour les transformer en nœuds de l'infrastructure; un ensemble de bibliothèques (API) pour permettre aux agents d'accéder aux services; et un ensemble d'outils principalement pour soutenir l'inspection et le contrôle de l'exécution (surveillance, débogage) du comportement, d'état et de coordination entre les centres de tuple.

Au cœur de la technologie TuCSoN se trouve la technologie tuProlog, un moteur Prolog entièrement intégré à l'environnement Java, disponible également en tant que bibliothèque et environnement autonome (la technologie tuProlog est disponible sur http://tuprolog.sourceforge.net) . En plus d'être adopté dans des projets de recherche (par exemple pour la gestion distribuée des flux de travail, la logistique et l'apprentissage en

ligne), TuCSoN est actuellement utilisé comme plate-forme de référence pour les systèmes d'agents dans les projets académiques.

JADE

JADE (Java Agent DEvelopment Framework) [44] [45] est un Framework Java pour le développement d'applications multi-agents distribuées. Il représente un intergiciel d'agent fournissant un ensemble de services disponibles et faciles à utiliser grâce à plusieurs outils graphiques utiles pour le débogage et les tests.

L'un des principaux objectifs de la plate-forme JADE est de soutenir l'interopérabilité en respectant rigoureusement les spécifications FIPA concernant l'architecture de la plate-forme ainsi que l'infrastructure de communication. En outre, JADE est très flexible et peut-être adapté pour être utilisé sur des appareils avec des ressources limitées telles que des PDA et des téléphones mobiles, ce qui lui fait le meilleur choix pour les systèmes pervasifs.

JADE a été largement utilisé au cours des dernières années par de nombreuses organisations académiques et industrielles, il est doté d'un nombre important des tutoriels pour le soutien à l'enseignement dans des cours universitaires liés aux agents au prototypage industriel. À titre d'exemple, Whitestein a utilisé JADE pour construire un système basé sur un agent pour la prise de décision dans les centres de transplantation d'organes [46].

La plateforme JADE est un logiciel open source, distribué par TILAB (Telecom Italia LABoratories) selon les termes de la licence LGPL et peut-être obtenu sur http://jade.tilab.com. Depuis mai 2003, le Conseil International JADE est chargé de superviser la gestion du projet. Actuellement, le conseil d'administration de JADE est composé de cinq membres: TILAB, Motorola, Whitestein Technologies AG, Profactor et France Télécom.

Jadex

Jadex [47] est un Framework pour la création d'agents orientés vers les buts suivant le modèle de croyance-désir-intention (BDI). Le Framework est réalisé en tant que couche d'agent rationnel qui se trouve au sommet d'une infrastructure d'agent middleware (intergiciel) telle que JADE, et prend en charge le développement d'agents avec des technologies bien établies telles que Java et XML.

Le moteur de raisonnement de Jadex aborde les limitations traditionnelles des systèmes BDI en introduisant de nouveaux concepts tels que les objectifs explicites et les mécanismes de délibération des objectifs.

Jadex a été utilisé pour construire des applications dans différents domaines tels que la simulation, la planification et les systèmes ubiquitaires. Par exemple, Jadex a été utilisé pour développer une application multi-agents pour la négociation des horaires de traitement dans les hôpitaux [48]. Jadex a également été utilisé avec succès dans plusieurs cours de génie logiciel à l'Université de Hambourg.

Le système Jadex, développé au sein du groupe Systèmes Distribués et Systèmes d'Information de l'Université de Hambourg, est disponible gratuitement sous licence LGPL et peut-être téléchargé sur http://jadex.sourceforge.net. Outre le Framework et les outils de développement supplémentaires, la distribution contient un didacticiel d'introduction, un guide de l'utilisateur et plusieurs exemples d'applications illustratives avec code source.

DESIRE

DESIRE (DEsign et Specification of Interacting REasoning components) est une méthode de développement de composition pour les systèmes multi-agents, basée sur une notion d'architecture de composition, développée par (Treur et al.) [49] à la Vrije Universiteit Amsterdam. Dans cette approche, la conception de l'agent repose sur les principaux aspects suivants :

- la composition du processus,

- la composition des connaissances,
- les relations entre la connaissance et la composition du processus.

Dans cette approche d'agent basée sur les composants, le processus de raisonnement complexe d'un agent est construit comme une interaction entre les composants représentant les sous processus du processus de raisonnement global [49]. Le processus de raisonnement est structuré en fonction d'un certain nombre de composants de raisonnement qui interagissent les uns avec les autres. Nous pouvons trouver des composants qui peuvent être composés d'autres composants comme nous pouvons trouver d'autres qu'ils ne sont pas décomposés et qui sont appelés composants primitifs. Le fonctionnement du système global d'agent est basé sur la fonctionnalité de ces composants primitifs plus la relation de composition qui coordonne leur interaction.

La spécification d'une relation de composition peut impliquer, par exemple, les possibilités d'échange d'informations entre les composants et la structure de contrôle qui active les composants.

L'approche DESIRE a été utilisée pour des applications telles que l'équilibrage de charge des systèmes de distribution et de diagnostic de l'électricité.

3.4 Les systèmes multi-agents distribués

La technologie multi-agents fournit un Framework qui permet à un système de se comporter comme un serveur ou un client selon la situation. Dans ce Framework, un système distribué est considéré comme une collection d'unités appelées agents qui offrent un ensemble de caractéristiques à ce dernier et qui peuvent se résumer dans les points suivants :

- Les agents sont autonomes et ont leurs propres objectifs.
- Les services d'un système distribué sont réalisés par la collaboration de ses agents afin d'atteindre leurs propres objectifs, où chaque agent offre des services et demande

à d'autres agents d'effectuer des tâches bien précises.

- L'autonomie des agents les rend indépendants l'un de l'autre ; ils ont leurs propres objectifs et comportements. Cette caractéristique offre la possibilité à un agent de refuser d'offrir ses services si les demandes d'autres agents sont susceptibles d'être impossibles ou ne correspondent pas à ses objectifs.

- Les agents ne se contentent pas d'envoyer des messages et des demandes les uns aux autres, mais négocient les uns avec les autres par la communication pour atteindre leurs objectifs.

Un système conçu à l'aide d'un Framework multi-agents se compose de nombreux types d'agents autonomes qui effectuent diverses tâches pour compléter leurs objectifs et offrir leurs services. Chaque agent travaille indépendamment sans être conscient de l'existence d'autres agents collaborateurs. Dans ce Framework, la gestion centralisée des données et des informations partagées entre agents n'est pas du tout nécessaire. Les agents doivent être conscients de l'existence d'autres agents uniquement lorsqu'ils demandent les services de cet agent. S'ils trouvent d'autres agents qui sont coopératifs, ils peuvent accéder aux services de ses autres agents. Même s'ils ne peuvent pas trouver d'autres agents, les agents proposent leurs propres services qui ne nécessitent pas de collaboration avec d'autres agents.

Lors du démarrage de la coopération, aucun agent et aucun système n'intercèdent avec les agents et aucune information n'est partagée entre les agents. Tout agent peut coordonner la collaboration entre les agents via la communication entre les agents impliqués.

Un agent établit dynamiquement des relations temporaires pour coopérer avec d'autres agents. Fondamentalement, un agent effectue ses tâches de manière autonome et indépendante. Lorsque la collaboration avec d'autres personnes s'avère nécessaire pour réaliser une tâche, un agent recherche des agents coopératifs en envoyant des messages. Si des coopérateurs sont trouvés, l'agent établit des relations de coopération avec eux.

Les agents sont évolutifs, c'est-à-dire, ils peuvent décider de façon dynamique avec quels agents ils vont collaborer et établir temporairement des relations de coopération avec ses agents à la demande. Cette évolution, permet aux nouveaux utilisateurs du système d'utiliser et de bénéficier facilement des services offerts par ce dernier en démarrant leurs propres agents, et les utilisateurs déjà existants peuvent facilement cesser d'utiliser un système en mettant fin à leurs propres agents, sachant qu'il n'y a pas de différence entre les agents précédemment existants et les nouveaux agents, et aucun agent n'est affecté par une augmentation ou une diminution du nombre d'utilisateurs. Autrement dit, l'ensemble du système est évolutif. L'évolutivité est une exigence importante pour un système qui est émergé dans le monde réel tel que les systèmes pervasifs.

3.4.1 Les systèmes distribués à la base de la plate-forme JADE

La plupart des plates-formes agent sont conçues pour garantir le bon fonctionnement des applications distribuées. Dans le cas de la plate-forme JADE, une plate-forme peut être distribuée sur plusieurs hôtes, dont chacun d'entre eux exécute sa propre machine virtuelle JAVA (JVM), afin de collaborer pour exécuter une seule application JAVA.

Chaque JVM exécute un conteneur d'agents de base qui fournit lui-même un environnement complet d'exécution pour l'agent, ce qui permet également à plusieurs agents de s'exécuter simultanément sur le même hôte.

La collaboration entre les agents se fait par le biais du registre RMI (Remote Method Invocation), qui est un mécanisme illustré par la figure 3.8 permettant à un objet résidant dans un système (JVM) d'accéder / invoquer un objet s'exécutant sur une autre JVM.

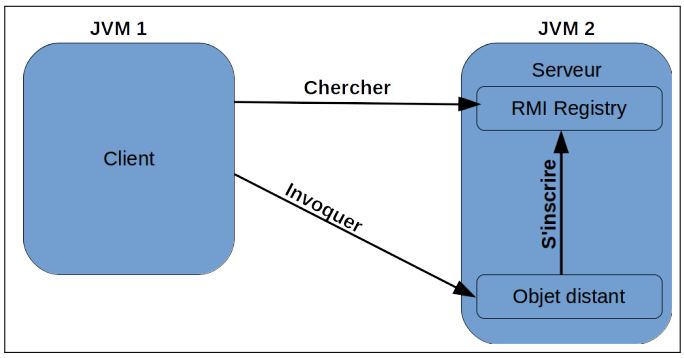

Figure 3.8: Principe de fonctionnement d'une invocation RMI

Le conteneur principal est un conteneur d'agent où réside AMS et DF et où le registre RMI , qui sont créés automatiquement par la plate-forme JADE. Les autres conteneurs, se connectent au conteneur principal pour former un environnement d'exécution complète pour tout l'ensemble d'agents JADE, comme le montre la figure 3.9.

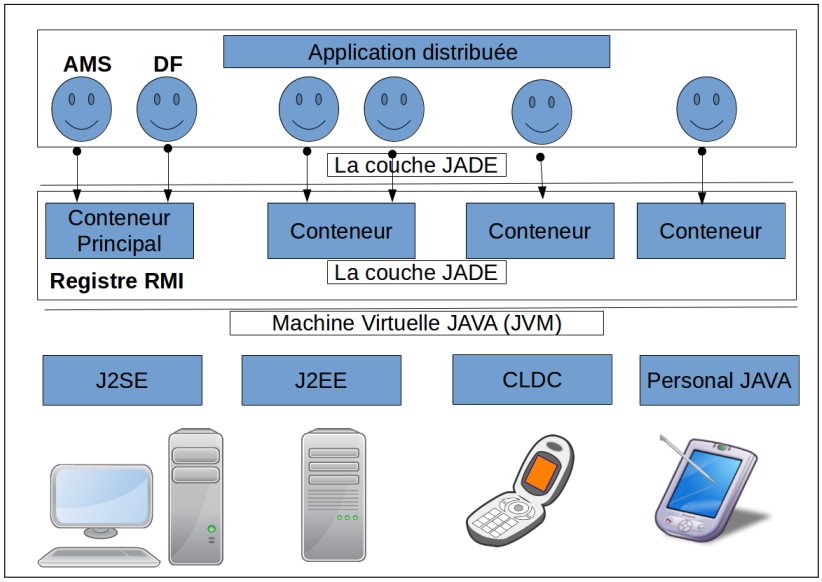

Figure 3.9: L'architecture d'un système distribué à la base de la plate-forme JADE

Au démarrage, le conteneur principal tente de localiser un registre RMI (RMI Registry) sur l'hôte local et lie sa référence d'objet à ce registre; Si aucun ne peut être trouvé, il crée un nouveau registre. En revanche, lorsqu'un conteneur non principal démarre, il localise le registre RMI sur l'hôte de conteneur principal spécifié et recherche la référence d'objet du conteneur principal. Il appelle ensuite la méthode distante *addNode()* du con-

teneur principal pour rejoindre la plate-forme et enregistre sa propre référence d'objet dans le conteneur principal.

Les messages d'agent et les informations de contrôle système échangées entre les conteneurs sont implémentés via un modèle de commande dans lequel le nœud demandeur (un conteneur) crée un objet *Command* approprié et transmet cet objet, avec des demandes d'exécution, au nœud exécutant.

Grâce à ses caractéristiques, les systèmes multi-agent offres une meilleure gestion des applications distribuées, sachant que ce type d'application est l'une des caractéristiques cruciales dans les systèmes pervasifs. Le choix des SMA est donc très favorable pour développer et garantir la reconfiguration dynamique de ce genre de système.

3.5 Conclusion

Dans ce chapitre, nous avons décrit ce que sont les agents, afin de comprendre pourquoi la technologie des agents est utile dans notre problématique. Il est important de réaliser que, à l'instar d'autres technologies logicielles telles que les objets, les agents ne sont pas magiques. Il s'agit simplement d'une approche de structuration et de développement de logiciels offrant certains avantages très bien adaptés aux systèmes pervasifs.

Les systèmes multi-agents (SMA) peuvent être utilisés pour gérer et modérer plusieurs types de systèmes y compris les systèmes distribués, en raison de la portabilité qu'ils offrent et la mobilité qui caractérise les agents software dans le système, ainsi que les services pouvant être déployés dans le réseau à travers la plate-forme des agents.

Un système multi-agent peut être facilement reconfiguré en ajoutant et en supprimant des agents de la plate-forme sans aucune influence sur le reste des agents (les autres agents actifs) cela nous donne la possibilité de déployer à n'importe quel moment un nouvel agent avec de nouveaux services, ce qui offre au système global de nouvelles fonctionnalités sans avoir besoin d'arrêter ou redémarrer le système, nous pouvons également intégrer une autre plate-forme secondaire et migré tous les agents ou les dupliquer pour une éventuelle

maintenance ou mise à jour de la plate-forme principale, afin de garantir la caractéristique du Non-stop du système. Ceci est possible grâce à un ensemble de propriétés qui caractérise un agent tels que :

- un agent peut être déployé indépendamment des autres agents,
- chaque agent à son propre comportement qui lui caractérise,
- chaque agent offre un ou plusieurs services,
- chaque agent utilise la norme FIPA-ACL pour communiquer avec les autres agents quelles que soient leurs plates-formes de développement.

Dans le prochain chapitre, nous verrons comment nous pouvons utiliser ses fonctionnalités pour concevoir un système pervasif reconfigurable dynamiquement basé sur des systèmes multi-agents.

Chapitre 4

La reconfiguration dynamique des systèmes pervasifs dans les réseaux LPWAN

Dans ce chapitre, nous proposerons une nouvelle approche pour la reconfiguration dynamique des systèmes pervasifs, basée sur les différentes approches étudiées dans le chapitre 2. Pour cela, nous prendrons comme exemple le système pervaisf du LPWAN (Low Power Wide Area Network) pour valider notre vision.

4.1 Introduction

L'objectif de notre travail est de structurer, organiser et réorganiser d'une manière automatique et dynamique tous les dispositifs présents dans un système pervasif, vu qu'il engendre un nombre important d'équipements, dans l'intention de bénéficier de tous les services disponibles dans l'environnement. Parmi les systèmes qui sont dotés d'une couche pervasive et qui ont une grande nécessité d'une reconfiguration dynamique, nous avons les réseaux LPWAN (Low Power Wide Area Network)[50].

Ce type de réseau est conçu pour connecter plusieurs entités dans un rayon pouvant atteindre 10 Km, ce gain de portée est possible grâce à une réduction de la quantité d'informations échangées. Cela fait de LPWAN le réseau le plus approprié pour les applications de télémétrie. Cette large couverture offerte par le LPWAN donne la possibilité

de connecter un grand nombre d'objets, qui eux-mêmes représentent des sous-systèmes autonomes et qui offrent plusieurs types d'informations dont le but est de concevoir une projection du monde réel dans le monde virtuel, ces informations peuvent être une température, une pression, un taux d'humidité, un mouvement, une vitesse d'un objet, etc.

Cette grande quantité d'informations entraîne des difficultés pour les différents systèmes associés à ce type de réseau à gérer tous les objets connectés et à s'adapter aux différents changements de l'environnement, c'est pourquoi un processus de reconfiguration automatique devient crucial pour les réseaux LPWAN.

Pour mieux comprendre cette difficulté, nous étudierons dans la section suivante l'architecture et les principaux acteurs des réseaux LPWAN.

4.2 Architecture générale d'un réseau LPWAN

Les réseaux LPWAN s'intègrent aisément aux objets connectés [50] grâce à des terminaux généralement miniatures qui permettent à l'objet de communiquer ses informations sur le réseau à travers des passerelles. Une fois les données récupérées par le serveur, via un canal hertzien, celles-ci seront directement retransmises par le biais du protocole HTTPs au serveur client pour les intégrer dans ses applications logicielles et les utiliser pour faire fonctionner les services adéquats, comme le montre la figure 4.1 :

Dans la figure 4.1 nous pouvons aussi remarquer que l'architecture LPWAN est typiquement mono-hop, où les périphériques terminaux sont directement connectés à la station de base (Passerelle), ce qui simplifie la topologie du réseau en lui offrant une robustesse et un contrôle centralisé. Quoique, cet accès massif à un seul canal influence négativement sur la fiabilité, l'évolutivité, la flexibilité et la qualité de service (QoS).

En effet, le mécanisme d'accès au canal de certaines technologies LPWAN recourt à ALOHA [51], un protocole de contrôle d'accès au support (MAC) aléatoire dans lequel les terminaux émettent sans effectuer de détection de porteuse pour vérifier l'état du canal. Cet accès incontrôlé aux médias conduit à des interférences ou à des collisions de

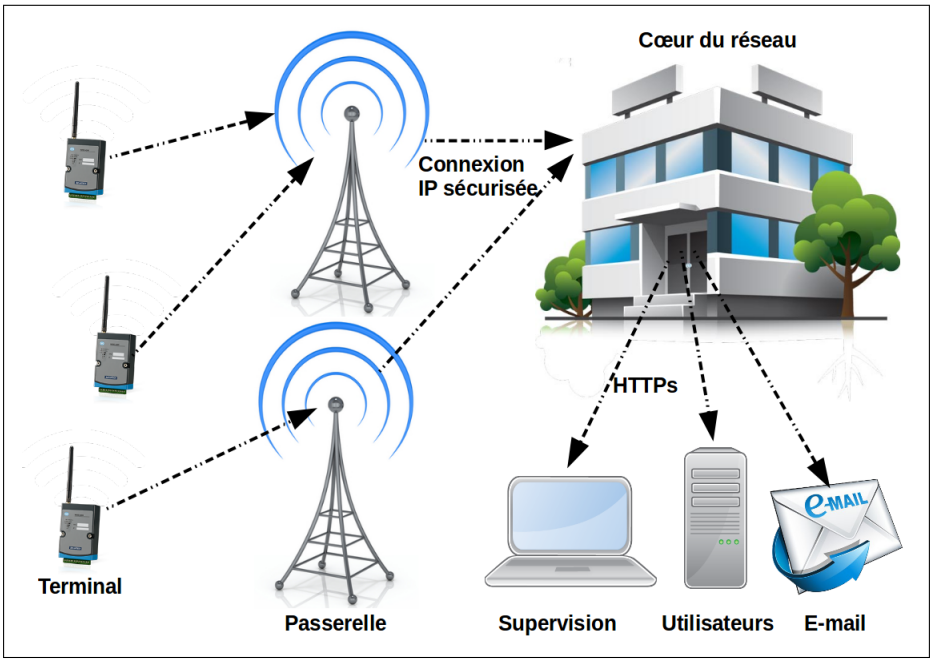

Figure 4.1: Architecture de base d'un réseau LPWAN

paquets entre des dispositifs non coordonnés, ce qui affecte grandement la fiabilité dans les réseaux denses. En outre, les dispositifs LPWAN situés loin de la station de base doivent utiliser des niveaux de puissance de transmission élevés, ce qui entraîne une importante consommation d'énergie et une durée de vie réduite de la batterie [52].

Il existe quelques implémentations des réseaux LPWAN tels que LoRaWAN[53] ou SigFox[54], qui permettent une meilleure exploitation de ce type de réseau, sachant que les deux implémentations respectent la même architecture représentée par la figure 4.1.

Dans le but de comprendre le fonctionnement des réseaux LPWAN, nous étudierons les caractéristiques de communication des réseaux LoRaWan, vu qu'il est le réseau le plus populaire d'aujourd'hui. LoRaWAN a jusqu'ici été principalement adopté par les pays européens, bien que récemment, plus de 100 villes aux États-Unis aient commencé à déployer des réseaux LoRaWAN à travers les villes. LoRaWAN à une portée radio annoncée allant jusqu'à 15 Km (en visibilité directe), et un débit de données allant jusqu'à 50kbps, et une autonomie de batterie d'environ 10 ans.

4.2.1 Communication dans les réseaux LoRaWan

LoRaWAN est une spécification pour le réseau LPWAN, qui définit l'architecture du système et les protocoles réseau pour les dispositifs compatibles LoRaWAN. Les réseaux LoRaWAN sont organisés selon de la topologie étoile, comme indiqué dans la figure 4.2.

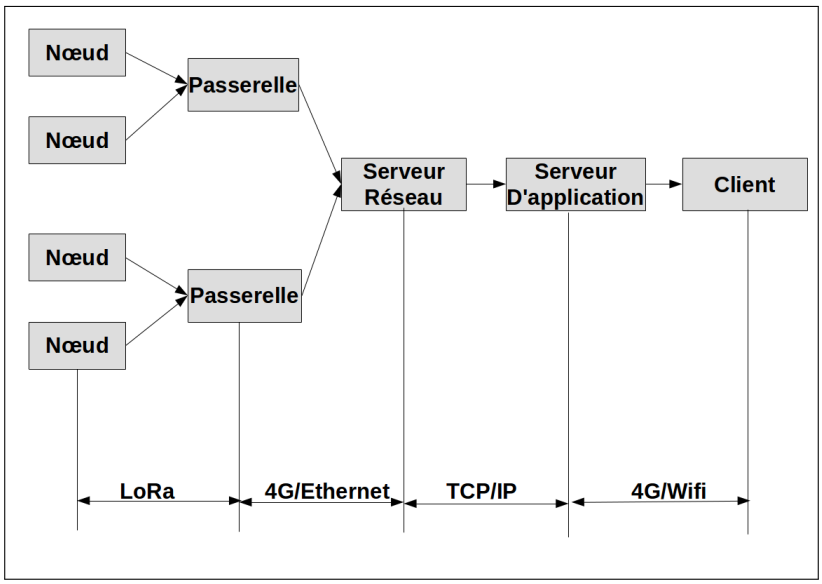

Figure 4.2: Les différentes technologies de communication dans le réseau LoRaWAN

Quatre types d'entités sont présents dans un LoRaWAN. Les nœuds de capteurs ou les nœuds d'extrémité qui envoient des paquets de données à une passerelle compatible LoRa. Une seule passerelle LoRa est capable de couvrir une ville entière (des centaines de kilomètres carrés). Les passerelles sont connectées à un serveur de réseau via un réseau backhaul[55][56] tel que 4G ou Ethernet.

Les serveurs réseau sont chargés d'identifier par des adresses IP tous les nœuds présents dans le réseau à travers le protocole 6LoWPAN[57]. L'attribution des adresses se fait d'une manière dynamique en utilisant le protocole DHCP. Les serveurs réseaux sont connectés à un serveur d'applications via TCP / IP qui permet la collecte d'informations et la réorganisation de tous les flux de données présentes dans le réseau, cela devient compliqué avec l'augmentation du nombre de nœuds dans le réseau. Notre objectif dans cette thèse est de simplifier cette complication grâce à d'un processus de reconfiguration

dynamique du réseau, ce processus sera détaillé dans les sections suivantes.

Les utilisateurs peuvent accéder aux données depuis les serveurs d'applications sur n'importe quel appareil disposant d'un accès à Internet, tels que les Smartphones ou les ordinateurs personnels.

4.3 Approche proposée pour la reconfiguration dynamique dans les réseaux LPWAN

Comme nous l'avons constaté précédemment, le point fort des réseaux LPWAN à couvrir un nombre important de nœuds devient un problème pour les serveurs d'applications, ce qui complique la gestion et la présentation de toutes les informations collectées à travers tout le réseau. Cette tâche devient trop lourde pour les serveurs d'applications, vu la dynamicité des nœuds et la diversité des informations qu'ils offrent. Tout ceci a une influence négative sur l'efficacité du réseau et cause plusieurs problèmes pour les utilisateurs. Pour résoudre ce problème nous avons proposé une nouvelle vision pour assurer une meilleure exploitations de tous les terminaux, pour cela nous proposons de libérer les serveurs d'applications de leur responsabilité de chercher et localiser les informations dans le réseau, et habiliter les terminaux à présenter et publier leurs services auprès des serveurs d'applications en utilisant un processus de coordination et de négociation qui se déroule selon les étapes suivantes[58] :

- le terminal se connecte automatiquement à la plate-forme mère à l'aide du protocole DHCP,
- le terminal cherche et localise d'une manière automatique le serveur d'applications qui sera pour lui comme une plate-forme mère,
- le terminal publie ces services au niveau de la plate-forme mère.

De son côté, le serveur d'applications (la plate-forme mère) doit réagir à la proposition du nouveau terminal comme suite :

- la plate-forme mère détecte l'arrivée d'un nouveau terminal et enregistre ses services,
- la plate-forme mère se reconfigure d'une manière automatique par la prise en considération des nouveaux services offerts par le nouveau terminal et les publie aux utilisateurs.

Dans notre vision, le terminal doit être une entité autonome et indépendante des autres entités du réseau avec la possibilité de se coopérer et offrir des services à travers le réseau. Avec ses propriétés, la reconfiguration dynamique peut être résumée dans les trois opérations suivantes :

- ajouter un terminal,
- supprimer un terminal,
- remplacer un terminal, qui peut être vu comme une combinaison des deux opérations (ajouter et supprimer).

La figure 4.3, illustre notre approche par un exemple d'un réseau de télémétrie, qui ne contient initialement aucune connaissance sur l'ensemble de capteurs utilisés par le réseau, mais qui dispose d'un noyau lui permettant de se reconfigurer dynamiquement lorsqu'un nouveau terminal est présent dans le réseau selon la procédure suivante :

(a) le terminal 3 (capteur d'humidité) se connecte d'une manière automatique au réseau via la passerelle et il s'identifie auprès du serveur réseau pour obtenir une adresse IP et un ID dans le réseau,

(b) le terminal 3 propose ses services et leur nature pour le serveur d'applications,

(c) le serveur d'applications enregistre les services offerts par le terminal 3 et se reconfigure pour les présenter aux utilisateurs en tant que nouveaux services présents sur le réseau.

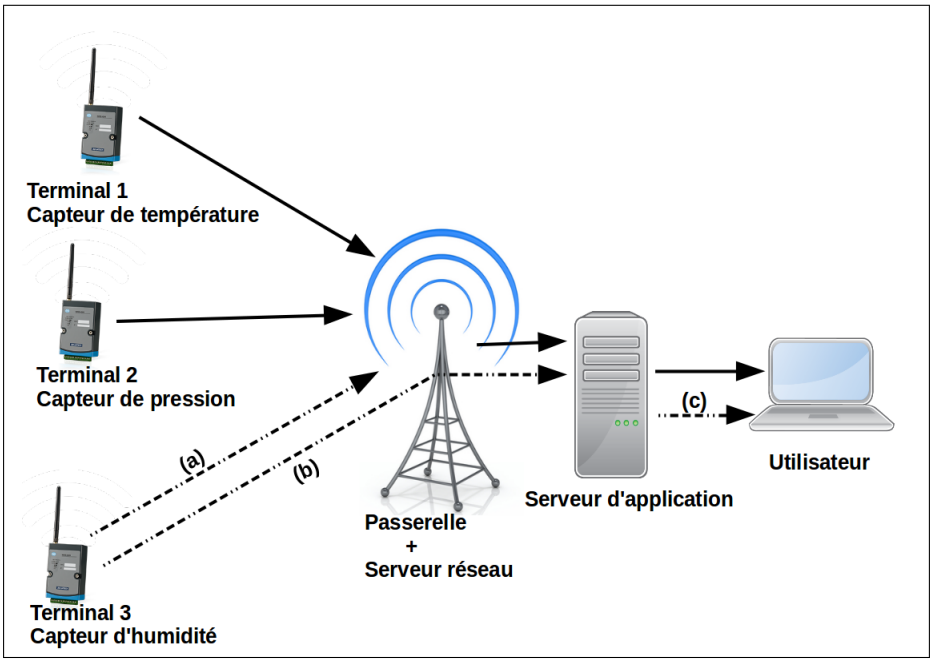

Figure 4.3: Comportement d'un système de télémétrie selon l'approche proposée

Selon l'étude réalisée au chapitre 2, et en comparant les différentes approches existant dans le domaine du développement software : orientée composant, orientée service et orientée agent, nous avons opté pour la dernière approche vu qu'elle englobe tous les avantages offerts par ses précédentes, de plus elle offre d'autres caractéristiques qui nous permettent de développer des terminaux autonomes et qui ont un comportement propre à eux.

L'utilisation de l'approche orientée agent pour implémenter la partie software du terminal, dans le but de lui donner une projection virtuelle (numérique) dans le serveur d'applications du système, nous permet de voir le système global comme une communauté d'agents, qui se collabore pour atteindre les objectifs globaux du système. Cette communauté peut offrir au système des nouveaux services qui n'ont pas été prévus lors de sa conception à condition que ces derniers soient jugés acceptables pour les objectifs généraux du système.

4.4 Reconfiguration dynamique à base d'agents software

Les systèmes multi-agent présents plusieurs avantages pour notre approche, vu qu'ils engendrent les applications distribuées et qui offrent la possibilité de mettre en œuvre des entités autonomes. Pour cela, nous identifierons dans un premier temps l'ensemble des agents nécessaires qui nous permet d'implémenter notre vision sur les réseaux LPWANs.

L'approche proposée est sous forme d'un ensemble de fonctionnalités permettant de concevoir des systèmes qui s'adaptent d'une manière automatique aux différents changements de leur environnement. Ces fonctionnalités seront par la suite la base d'un diagramme de cas d'utilisation sur lequel nous concevrons nos agents software.

Notre vision concerne les deux acteurs de l'architecture générale des réseaux LPWANs suivants :

- le terminal : il représente le nœud, le capteur ou le dispositif pervasif dans le réseau. Il contient l'agent suivant :
  - agent terminal : il est implémenté au niveau du terminal, et qui a comme tâches:
    - découvrir l'environnement du terminal et signaler sa présence dans le réseau,
    - publier les services offerts par le terminal,
    - rendre les services accessibles et exploitables dans le réseau.
- le serveur d'applications : dans lequel nous implémenterons notre plate-forme mère qui permet la gestion de tous les agents terminaux par le biais des agents suivants :
  - agent réceptionniste : il est implémenté au niveau de la plate-forme principale et qui a comme tâches :

    - réceptionner les nouveaux agents terminaux, et rendre leurs services exploitables à travers la plate-forme en interrogeant le nouvel agent à l'aide d'un système de messagerie,
    - reconfigurer la plate-forme pour rendre ces nouveaux services visibles pour les utilisateurs de la plate-forme.

  – Agent pages jaunes: il est implémenté dans la plate-forme, sa tâche principale est d'enregistrer tous les services offerts par les autres agents, dans le cas de la technologie JADE, cet agent est déjà implémenté et connu sous le nom DF (Directory Facilitator).

Comme nous l'avons déjà vu dans le chapitre 3, il existe plusieurs technologies qui nous permettent de concevoir un système multi agent, parmi lesquels nous avons choisi la plate-forme JADE. Dans la section suivante nous utiliserons cette plate-forme pour implémenter les trois agents et tester la communication entre eux afin de valider notre approche de la reconfiguration dynamique.

4.5 Implémentation et discutions

Notre solution se présente sous la forme d'une application JAVA à l'aide de la plate-forme JADE pour le développement des systèmes multi-agents. Ce choix a était fait pour les raisons suivantes :

- JAVA : pour garantir la portabilité de la solution sur plusieurs types de terminaux,
- JADE : c'est une plate-forme développée sous JAVA, et qui implémente agent page jaune "DF" sur laquelle est basée notre nouvelle approche.

La solution proposée représente une coordination entre l'agent réceptionniste de la plate-forme mère, et l'agent terminal, cette dernière est possible grâce aux différents messages définis par le standard FIPA-ACL. Ses deux agents plus l'agent page jaune qui

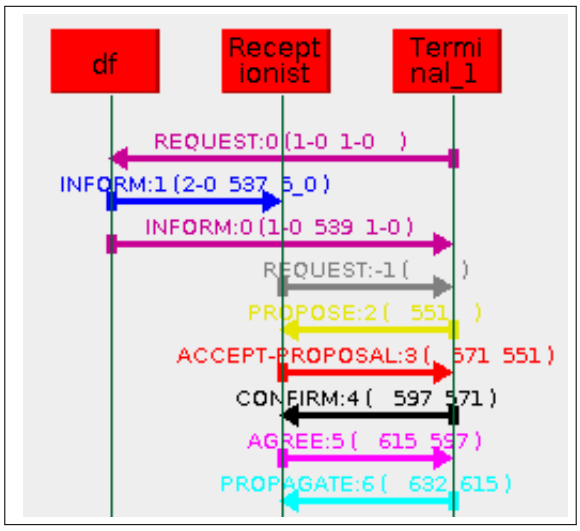

Figure 4.4: Scénario pour la reconfiguration dynamique à base des agents

est déjà défini dans la plate-forme JADE coopèrent afin de reconfigurer le système perversif du réseau LPWAN, tout en appliquant l'approche proposée dans la section précédente et selon le scénario illustré par la figure 4.4.

4.5.1 Agent terminal

Cet agent représente un nœud dans les réseaux LPWAN, physiquement c'est un simple système embarqué doté d'un capteur, une antenne pour assurer la connexion et un système d'exploitation simplifié et réduit, dans lequel nous implémenterons un agent propre à ce terminal et qui aura un accès aux informations fournisses par le capteur.

Le fonctionnement de ce terminal illustré par la figure 4.5 peut passer par les étapes suivantes :

- la connexion au réseau et l'obtention d'adresse IP : c'est possible grâce au service offert par le DHCP,
- la localisation et la connexion à la plate-forme mère de tous les agents,
- la publication des services au niveau de la plate-forme mère chez l'agent "DF",
- la communication des données captées par le capteur avec l'agent réceptionniste.

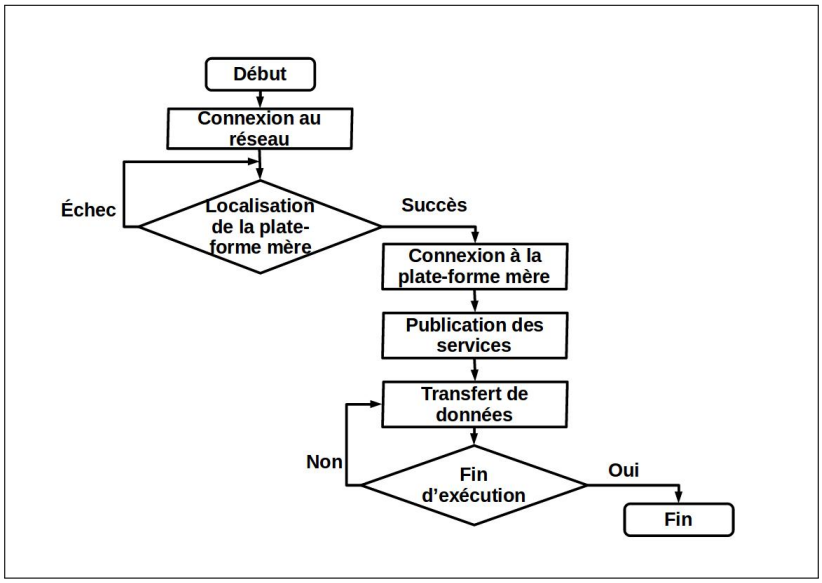

Figure 4.5: Organigramme de la démarche suivie par l'agent Terminal

4.5.2 Agent réceptionniste

Cet agent est implémenté au niveau de la plate-forme mère, il est chargé de la réception des nouveaux agents terminaux à l'aide d'une démarche décrite par l'organigramme illustré par la figure 4.6.

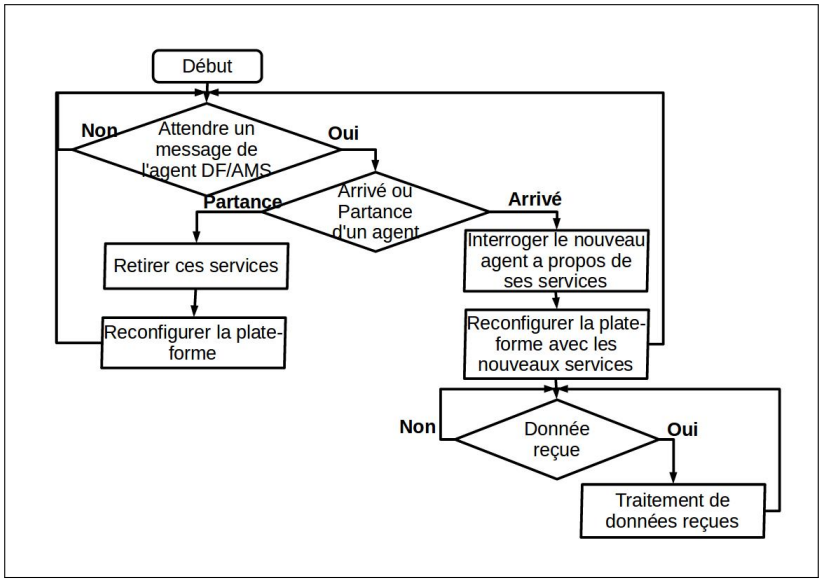

Figure 4.6: Organigramme de la démarche suivie par l'agent Réceptionniste

Conformément au scénario illustré par la figure 4.4, lorsqu'un agent terminal se

connecte à la plate-forme, son premier contact sera avec l'agent "DF" pour enregistrer ces services et annoncer sa présence à l'agent réceptionniste, qui est chargé d'interroger le nouvel agent terminal pour tirer parti de ces services et collecter les différentes données capturées par ce dernier et les rendre accessibles aux utilisateurs finaux à travers l'interface graphique incluse dans la plate-forme mère.

4.5.3 Communication entre agents

L'utilisation des systèmes multi-agents pour mettre en œuvre notre vision, présente le processus de la reconfiguration dynamique sous la forme d'une négociation entre l'agent terminal et la plate-forme mère, plus précisément l'agent réceptionniste. Cette négociation est basée sur l'échange d'un ensemble de messages du type FIPA-ACL entre les trois agents du scénario illustré par la figure 4.4, sachant que l'événement déclencheur de cette négociation est l'arrivée de l'agent terminal dans le réseau, puisque c'est à lui de signaler sa présence. Après cela, les autres agents doivent réagir à ce nouveau venu selon les messages suivants :

- l'agent terminal envoie un message REQUEST au "DF" pour lui informer sa présence et lui demander d'enregistrer ses services sur la plate-forme,
- l'agent "DF" envoie un message INFORM à l'agent terminal pour lui informer que sa demande a été acceptée,
- l'agent "DF" envoie un autre message INFORM à l'agent réceptionniste pour l'informer de la présence d'un nouvel agent terminal, ainsi que pour lui communiquer ses services,
- l'agent réceptionniste envoie un message REQUEST à l'agent terminal lui demandant de communiquer toutes les informations concernant les services proposés,
- l'agent terminal envoie un message PROPOSE à l'agent réceptionniste pour lui proposer le format de donné à envoyer,

- l'agent réceptionniste configure la plate-forme mère pour recevoir les données provenant du nouvel agent terminal et envoie un message ACCEPT-PROPOSAL à l'agent terminal pour lui informer que sa proposition a été accepté,
- l'agent terminal envoie un message CONFIRM à l'agent réceptionniste pour lui indiquer le début de l'envoi des données,
- l'agent réceptionniste envoie un message AGREE à l'agent terminal pour lui confirmer qu'il est prêt à recevoir les données,
- l'agent terminal envoie un message PROPAGATE à l'agent réceptionniste pour l'informer de l'envoi des premières données et ainsi de suite,

Lors d'une déconnexion de l'agent terminal, la plate-forme JADE la détecte automatiquement et informe tous les agents à travers un message INFORM envoyé par l'agent AMS, ce qui permet à l'agent réceptionniste de reconfigurer le système et de signaler l'absence et la rupture de données provenant de l'agent terminal déconnecté.

4.5.4 Discutions

En comparant notre approche avec les autres approches mentionnées dans le chapitre 2, nous pouvons constaté que la nôtre se base sur le principe qu'un objet du système ou un sous-système dans notre cas l'agent terminal, c'est à lui de découvrir son environnement, de chercher la plate-forme mère et de publier ces services. En revanche, les autres approches laissent cette tâche au système global, ce qui rend difficile la recherche et la découverte de tous les sous-systèmes présents dans environnement, et augmente le taux d'erreurs dans le système.

De plus, l'agent est une entité indépendante, ce qui signifie que l'ajout et la suppression d'un agent n'ont aucune influence sur le fonctionnement global du système, ce qui implique une continuité du fonctionnement du système malgré l'absence des services offerts par l'agent ou le terminal déconnecté, cela minimise le nombre de bugs dans l'application

et lui garantit le non-stop, puisque nous n'avons pas besoin de redémarrer le système à chaque fois qu'un nouveau terminal est ajouté ou supprimé.

4.6 Conclusion

L'application de notre nouvelle approche sur les réseaux LPWANs nous a simplifié la façon d'exploiter tous les terminaux présents dans le réseau, en créant une communauté d'agents qui gère l'ensemble du système à l'aide d'une suite de messages échangés entre eux. Cette nouvelle vision tente de se rapprocher plus au moins au comportement humain envers les systèmes complexes, qui englobe plusieurs entités à la fois. Ce comportement se présente sous la forme d'un ensemble de méthodes et de techniques de collaboration et de coordination où chaque individu offre un ensemble de services qui sera comme une boîte noire pour le reste des éléments de la communauté.

Avec cette nouvelle vision, le système global n'est pas censé connaître au préalable tous les services qui seront déployés dans le réseau, mais il doit savoir comment les exploiter au moment de leur arrivée dans le système, et comment les présenter au utilisateurs, en leur donnant le choix de les utiliser ou non.

Dans ce contexte, les systèmes multi-agents sont l'approche la plus appropriée pour mettre en œuvre notre vision, car elle offre plusieurs avantages que nous avons déjà mentionnés dans le chapitre 3 et que nous pouvons résumer dans les points suivants :

- l'autonomie de l'agent,
- la communication entre les agents,
- la possibilité d'implémenter des services au niveau de l'agent.

Notre approche, permet de simplifier la complexité impliquée par la dynamicité des systèmes pervasifs qui se réside dans l'ajout et la configuration des nouveaux dispositifs dans le système global.

Conclusion générale

La technologie des systèmes perversifs, est devenue un nouveau paradigme pour les prochaines décennies à venir. Penser à l'informatique pervasive signifie poser des questions fondamentales sur la façon par laquelle nous interagissons dans les sociétés dotées des systèmes artificiels et sur son influence sur nos vies.

Puisque que l'informatique pervasive est une technologie importante pour de nombreuses applications et les différents services, tels que les LPWANs et l'internet des objets, sa discussion est souvent liée à ces applications spécifiques. L'informatique pervasive engendre plusieurs domaines de recherche tels que la gestion de la mobilité, la gestion du contexte, l'autonomie, etc.

L'objectif de cette thèse, s'inscrit dans le domaine de gestion du contexte, il permet de résoudre l'un des problèmes majeurs des systèmes pervasifs, qui se présente dans les modes de gestion et d'utilisation de tous les dispositifs pervasifs qui entourent l'utilisateur souvent perdu autour de ce nombre important de dispositifs. Pour cela, un nouveau processus de reconfiguration dynamique est proposé, qui permet l'auto-intégration et l'auto-configuration de tous ces dispositifs. Ce processus consiste à créer un dialogue entre les différents dispositifs représentés par des agents software, qui doivent échanger entre eux une suite de messages du type FIPA-ACL pour négocier leurs positions dans le réseau, et la manière d'offrir leurs services aux utilisateurs, sachant que ses agents software doivent être implémentés au niveau du dispositif lui-même, ce qui veut dire, que ce dernier doit

être une sorte de systèmes embarqués qui contient une partie Hardware et une couche software dans laquelle l'agent s'exécute. De cette façon, il ne reste à l'utilisateur ordinaire que de bénéficier des services offerts par ces derniers.

L'application de la reconfiguration dynamique au niveau de la couche système pervasif du réseau LPWAN offre plus de flexibilité pour le réseau et une meilleure perception de l'environnement vu que les terminaux représentent un ensemble de capteurs. Une meilleure adaptation aux différents changements est réalisée par l'ajout ou la suppression d'un terminal. Notre approche garantit ainsi plusieurs caractéristiques :

- la flexibilité,
- l'indépendance,
- la continuité de fonctionnement (non-stop),
- la modularité.

Ces quatre caractéristiques peuvent être déployées dans plusieurs types de systèmes comme les réseaux de capteurs ou les maisons intelligentes ainsi les cyber physical system [59] où le système global est composé d'une couche système pervasif, qui est généralement composé de plusieurs composantes indépendantes.

L'une des perspectives de notre approche que nous considérons comme révolutionnaire dans les systèmes informatiques, se présente dans l'intégration des nouveaux périphériques hardwares dans les systèmes informatiques. La méthode utilisée consiste que chaque constructeur d'un périphérique développe son propre pilote qui doit être installé dans le système d'exploitation de la machine pour lui permettre d'utiliser et exploiter efficacement ce dernier. Ce qui implique souvent une saturation du système d'exploitation par tous les pilotes installés et qui vont prendre une part importante de l'utilisation du processeur.

Notre approche permet de simplifier ce processus d'intégration, en implémentant, d'une part le pilote du périphérique dans ce dernier lui-même qui sera sous forme d'un agent software, dont ses fonctions principales sont :

- signaler la présence du périphérique à la machine principale,
- répondre aux requêtes des utilisateurs de la machine principale en leur offrant un accès aux différents services implémentés dans le périphérique.

D'autre part, implémenter un noyau pour la reconfiguration dynamique au niveau du système d'exploitation de la machine mère, qui aura comme objectif de réceptionner tous les périphériques présents dans son environnement et de présenter leurs services à l'utilisateur.

De cette manière, nous changeons la façon de concevoir des périphériques informatiques et aussi les réactions des utilisateurs envers leurs nouveaux dispositifs (pas de driver à installer).

Annexe A

Liste des acronymes

Sigle	**Signification**
ACMAS	Agent Centered Multi-Agent System
ACC	Agent Communication Channel
ACL	Agent Communication Langage
AMS	Agent Management System
AUML	Agent Unified Modeling Language
API	Application Programming Interface
AA	Auditing Agent
BDI	Belief Desire Intention
CPU	Central Processing Unit
CORBA	Common Object Request Broker Architecture
CLB	Configurable Logical Blocks
CLDC	Connected Limited Device Configuration
CCL	Contraint Choice Language
CPS	Cyber Physical System
DESIRE	DEsign et Specification of Interacting REasoning components
DF	Directory Facilitator
DHCP	Dynamic Host Configuration Protocol

DRS	Dynamic Reconfiguration Services
EAPR	Early Access Partial Reconfiguration
EJB	Enterprise JavaBeans
ELF	Executable and Linkable Format
XML	eXtensible Markup Language
FPGA	Field Programmable Gate Arrays
FIPA	Foundation for Intelligent Physical Agents
HDL	Hardware Description Language
HTTP	Hyper Text Transfer Protocol
IRE	Identification Reference Expression
IOB	Input/Output Blocks
ISC	Interface-Scénarios-Contraintes
ICAP	Internal Configuration Access Port
IP	Internet Protocol
6LoWPAN	IPv6 Low power Wireless Personal Area Networks
J2EE	Java 2 Enterprise Edition
J2SE	Java 2 Platform Standard Edition
JADE	Java Agent DEvelopment Framework
JVM	Java Virtual Machine
JTAG	Joint Test Action Group
KIF	Knowledge Interchange Format
KQML	Knowledge Query and Manipulation Language
LGPL	Lesser General Public License
LGPL	Lesser General Public License

LoRaWAN	Long Range Wide Area Nerwork
LTE	Long Term Evolution
LUT	Look-Up-Table
LPWAN	Low Power Wide Area Network
MHS	Microprocessor Hardware Specification
MSS	Microprocessor Software Specification
Module PR	Module Partiellement Reconfiguration
Partition PR	Partition Partiellement Reconfiguration
P2P	Peer to Peer
PowerPC	Performance Optimization With Enhanced RISC Performance Computing
PA	Proxy Agent
Qr Code	Quick Response Code
RFID	Radio Frequency Identification
RMS	Reconfigurable Manufacturing System
RD	Reconfiguration dynamique
Region PR	Régtion Partiellement Reconfiguration
RMI	Remote Method Invocation
RDF	Resource Description Framework
SL	Semantic Langage
SD	Service Directory
Slice BM	Slice Bus-Macro
SODA	Societies in Open and Distributed Agent spaces
SSND	Standard Service Naming Directory
SMA	Système multi-agent

TILAB	Telecom Italia LABoratories
TCP	Transimission Control Protocol
TuCSoN	Tuple Centre Spread over the Network
VHDL	Very high-speed integrated circuit Hardware Description Language
WIFI	Wireless Fidelity
WSN	Wireless Sensor Network

Reférences

[1] D. Saha and A. Mukherjee, "Pervasive computing: a paradigm for the 21st century," *Computer*, vol. 36, no. 3, pp. 25–31, 2003.

[2] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," *ACM Computing Surveys (csuR)*, vol. 34, no. 2, pp. 171–210, 2002.

[3] M. Weiser, "The computer for the 21 st century," *Scientific american*, vol. 265, no. 3, pp. 94–105, 1991.

[4] K. Li, H. Shen, K. Tajima, and L. Huang, "An effective cache replacement algorithm in transcoding-enabled proxies," *The Journal of Supercomputing*, vol. 35, no. 2, pp. 165–184, 2006.

[5] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker, "Agile application-aware adaptation for mobility," in *ACM SIGOPS Operating Systems Review*, vol. 31, no. 5. ACM, 1997, pp. 276–287.

[6] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced cpu energy," in *Mobile Computing*. Springer, 1994, pp. 449–471.

[7] A. R. Lebeck, X. Fan, H. Zeng, and C. Ellis, "Power aware page allocation," *ACM Sigplan Notices*, vol. 35, no. 11, pp. 105–116, 2000.

[8] D. Grawrock, *Dynamics of a Trusted Platform: A building block approach*. Intel Press, 2009.

[9] J. Polakovic, "Architecture logicielle et outils pour systèmes d'exploitation reconfigurables," Ph.D. dissertation, Institut National Polytechnique De Gronoble, 2011.

[10] A. Ketfi, "Une approche générique pour la reconfiguration dynamique des applications à base de composants logiciels," Ph.D. dissertation, Université Joseph-Fourier-Grenoble I, 2004.

[11] W. T. G. T. Heineman, *Component-based software engineering : putting the pieces togethe*. Addison-Wesley Professional, 2001.

[12] P. Kruchten, "Modeling component systems with the unified modeling language," in *International Workshop on Component-Based Software Engineering*, 1998.

[13] W.-T. Tsai, W. Song, R. Paul, Z. Cao, and H. Huang, "Services-oriented dynamic reconfiguration framework for dependable distributed computing," in *COMPSAC*, vol. 1, 2004, pp. 554–559.

[14] N. S. Voros and K. Masselos, *System level design of reconfigurable systems-on-chip*. Springer, 2005.

[15] R. J. Fong, S. J. Harper, and P. M. Athanas, "A versatile framework for fpga field updates: an application of partial self-reconfiguration," in *Rapid Systems Prototyping, 2003. Proceedings. 14th IEEE International Workshop on*. IEEE, 2003, pp. 117–123.

[16] D. A. Wescott, "Embedding a jtag host controller into an fpga design," Jan. 3 2006, uS Patent 6,983,441.

[17] A. Upegui and E. Sanchez, "Evolving hardware by dynamically reconfiguring xilinx fpgas," in *International Conference on Evolvable Systems*. Springer, 2005, pp. 56–65.

[18] B. Blodget, S. McMillan, and P. Lysaght, "A lightweight approach for embedded reconfiguration of fpgas," in *Design, Automation and Test in Europe Conference and Exhibition, 2003*. IEEE, 2003, pp. 399–400.

[19] Q. Zhu and A. T. Azar, *Complex system modelling and control through intelligent soft computations*. Springer, 2015.

[20] W. Lie and W. Feng-Yan, "Dynamic partial reconfiguration in fpgas," in *Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on*, vol. 2. IEEE, 2009, pp. 445–448.

[21] C. Claus, B. Zhang, M. Hübner, C. Schmutzler, J. Becker, and W. Stechele, "An xdl-based busmacro generator for customizable communication interfaces for dynamically and partially reconfigurable systems," in *Workshop on Reconfigurable Computing Education at ISVLSI*, vol. 7, 2007.

[22] M. K. Chowdary, S. S. Babu, S. S. Babu, and H. Khan, "Fpga implementation of moving object detection in frames by using background subtraction algorithm," in *Communications and Signal Processing (ICCSP), 2013 International Conference on*. IEEE, 2013, pp. 1032–1036.

[23] M. J. S. Smith, *Application-specific integrated circuits*. Addison-Wesley Reading, MA, 1997, vol. 7.

[24] Y. Koren, "Reconfigurable manufacturing system," in *CIRP Encyclopedia of Production Engineering*. Springer, 2014, pp. 1035–1039.

[25] C. Y. Baldwin and K. B. Clark, "Modularity in the design of complex engineering systems," in *Complex engineered systems*. Springer, 2006, pp. 175–205.

[26] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[27] J. Ferber, *Les systèmes multi-agents: vers une intelligence collective*. InterEditions, 1997.

[28] S. D. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziargyriou, F. Ponci, and T. Funabashi, "Multi-agent systems for power engineering applications—part i: Concepts, approaches, and technical challenges," *IEEE Transactions on Power systems*, vol. 22, no. 4, pp. 1743–1752, 2007.

[29] L. Búrdalo, A. Terrasa, V. Julián, and A. García-Fornes, "Trammas: A tracing model for multiagent systems," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 7, pp. 1110–1119, 2011.

[30] T. Finin, R. Fritzson, D. McKay, and R. McEntire, "Kqml as an agent communication language," in *Proceedings of the third international conference on Information and knowledge management*. ACM, 1994, pp. 456–463.

[31] A. Fipa, "Fipa acl message structure specification," *Foundation for Intelligent Physical Agents, http://www. fipa. org/specs/fipa00061/SC00061G. html (30.6. 2004)*, 2002.

[32] J. Ferber, *Multi-agent systems: an introduction to distributed artificial intelligence*. Addison-Wesley Reading, 1999, vol. 1.

[33] B. Burmeister, A. Haddadi, and G. Matylis, "Application of multi-agent systems in traffic and transportation," *IEE Proceedings-Software*, vol. 144, no. 1, pp. 51–60, 1997.

[34] F. I. C. A. Specification, "Foundation for intelligent physical agents, 2000," 2004.

[35] M. R. Genesereth, R. E. Fikes *et al.*, "Knowledge interchange format-version 3.0: reference manual," 1992.

[36] J. J. Odell, H. V. D. Parunak, and B. Bauer, "Representing agent interaction protocols in uml," in *International Workshop on Agent-Oriented Software Engineering*. Springer, 2000, pp. 121–140.

[37] R. S. Cost, Y. Peng, Y. Chen, Y. K. Labrou, T. Finin *et al.*, "Modeling agent conversations with colored petri nets," in *Working notes of the Autonomous Agents' 99 Workshop on Specifying and Implementing Conversation Policies*, 1999.

[38] D. Poutakidis, L. Padgham, and M. Winikoff, "Debugging multi-agent systems using design artifacts: The case of interaction protocols," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*. ACM, 2002, pp. 960–967.

[39] J. Odell and M. Huget, "Fipa modeling: interaction diagrams. working draft, foundation for intelligent physical agents," 2003.

[40] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2013.

[41] G. D. M. Serugendo, M.-P. Gleizes, and A. Karageorgos, *Self-organising software: From natural to artificial adaptation*. Springer Science & Business Media, 2011.

[42] E. Nardini, M. Viroli, and E. Panzavolta, "Coordination in open and dynamic environments with tucson semantic tuple centres," in *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM, 2010, pp. 2037–2044.

[43] A. Omicini, "Soda: Societies and infrastructures in the analysis and design of agent-based systems," in *International Workshop on Agent-Oriented Software Engineering*. Springer, 2000, pp. 185–193.

[44] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*. John Wiley & Sons, 2007, vol. 7.

[45] F. Bellifemine, A. Poggi, and G. Rimassa, "Jade: a fipa2000 compliant agent development environment," in *Proceedings of the fifth international conference on Autonomous agents*. ACM, 2001, pp. 216–217.

[46] M. Calisti, P. Funk, S. Biellman, and T. Bugnon, "A multi-agent system for organ transplant management," in *Applications of Software Agent Technology in the Health Care Domain*. Springer, 2003, pp. 199–212.

[47] A. Pokahr, L. Braubach, and W. Lamersdorf, "Jadex: A bdi reasoning engine," in *Multi-agent programming*. Springer, 2005, pp. 149–174.

[48] T. O. Paulussen, A. Zöller, A. Heinzl, A. Pokahr, L. Braubach, and W. Lamersdorf, "Dynamic patient scheduling in hospitals," *Coordination and Agent Technology in Value Networks. GITO, Berlin*, pp. 149–174, 2004.

[49] F. M. Brazier, B. M. Dunin-Keplicz, N. R. Jennings, and J. Treur, "Desire: Modelling multi-agent systems in a compositional formal framework," *International Journal of Cooperative Information Systems*, vol. 6, no. 01, pp. 67–94, 1997.

[50] J.-P. Bardyn, T. Melly, O. Seller, and N. Sornin, "Iot: The era of lpwan is starting now," in *European Solid-State Circuits Conference, ESSCIRC Conference 2016: 42nd*. IEEE, 2016, pp. 25–30.

[51] C. Goursaud and J.-M. Gorce, "Dedicated networks for iot: Phy/mac state of the art and challenges," *EAI endorsed transactions on Internet of Things*, 2015.

[52] S. Barrachina, B. Bellalta, T. Adame, and A. Bel, "Multi-hop communication in the uplink for lpwans," *arXiv preprint arXiv:1611.08703*, 2016.

[53] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of lorawan," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[54] J. C. Zuniga and B. Ponsard, "Sigfox system description," *LPWAN@ IETF97, Nov. 14th*, 2016.

[55] F. B. Pope Jr, A. A. Giordano, W. A. Biagini, R. G. Eckard, and D. M. Fye, "Wireless area network having flexible backhauls for creating backhaul network," Nov. 25 2003, uS Patent 6,654,616.

[56] Z. Gao, L. Dai, D. Mi, Z. Wang, M. A. Imran, and M. Z. Shakir, "Mmwave massive-mimo-based wireless backhaul for the 5g ultra-dense network," *IEEE Wireless Communications*, vol. 22, no. 5, pp. 13–21, 2015.

[57] Z. Shelby and C. Bormann, *6LoWPAN: The wireless embedded Internet*. John Wiley & Sons, 2011, vol. 43.

[58] G. Abdellaoui and F. T. Bendimerad, "Dynamic reconfiguration of lpwans pervasive system using multi-agent approach," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 2, pp. 300–305, 2018.

[59] E. A. Lee, "Cyber physical systems: Design challenges," in *Object oriented real-time distributed computing (isorc), 2008 11th ieee international symposium on*. IEEE, 2008, pp. 363–369.

الملخص

أصبحت تكنولوجيا الأنظمة السائدة نموذجًا جديدًا للعقود القادمة.حاليا ، نجد هذا النوع من الأنظمة في كل مكان في حياتنا اليومية، مما أدى إلى خلق بيئة معلوماتية في غاية الديناميكية، لذلك يتوجب على التطبيق الرئيسي الذي يدير النظام أن يغير من سلوكياته من أجل التكيف مع الشروط الجديدة. ولحل هذه المشكلة، نقترح منهجًا جديدًا لإعادة الهيكلة، يسمح بإعادة تكوين نظام في الوقت الفعلي. يعتمد نموذجنا على منهجية العملاء، بحيث نفترض أن العملاء هم أصغر الكيانات المستقلة لأي نظام. و بهذا يمكننا تلخيص إعادة الهيكلة الديناميكية في طريقة إضافة أو إزالة عميل من النظام. يتيح التنظيم الجديد للنظام تحقيق أهدافه بفعالية تامة من خلال دمج الخدمات الجديدة التي يقدمها العملاء الجدد وضمان حسن سير النظام بشكل عام في ظل مختلف الظروف التي تفرضها البيئة.من أجل التحقق من صحة منهجيتنا، اعتمدنا عملية إعادة الهيكلة الديناميكية جديدة في مجال LPWANs ، والتي تمثل أنسب منصة اتصالات لربط جميع الأجهزة المشكلة للأنظمة السائدة.

كلمات البحث: إعادة الهيكلة الديناميكية، الأنظمة السائدة، منهجية العملاء، LPWAN.

Abstract

Pervasive systems technology has become a new paradigm for the coming decades. Currently, we nd this type of system everywhere in our daily life, while creating a very dynamic pervasive environment, therefore the application that must manage this kind of system have to change thier behavior to adapt to new situations. To solve this problem, a new approach of recon guration is proposed, allowinga real time recon guration of the pervasive systems. This new technique is based on the agent-oriented approach.Because agents are the smallest elemental and autonomous entities for any system. With this approach, dynamic recon guration is the automation of adding or removing an agent from the system. This con guration change, allows the system to achieve its objectives e ectively by integrating new services o ered by the new agents,it also helps to ensure the proper functioning of the system in di erent situations. In order to validate our approach, LPWANs have been chosen as a communication platform to interconnect all pervasive devices, due to its wide coverage in terms of network.

Keywords: Dynamic Recon guration; Pervasive system; Multi-agent system; LPWAN.

Résumé

La technologie des systèmes pervasifs est devenue un nouveau paradigme pour les prochaines décennies. Actuellement, nous trouvons ce type de système partout dans notre vie quotidienne, tout en créant un environnement pervasif très dynamique, par conséquent l'application qui gère ce genre de systèmes doit changer de comportement a n de s'adapter aux nouvelles situations. Pour résoudre ce problème, une nouvelle approche de recon guration est proposée, permettant la recon guration en temps réel des systèmes pervasifs. Cette nouvelle technique est basée sur l'approche orientée agent, en considérant que les agents sont les plus petites entités élémentaires et autonomes pour n'importe quel système. Par cette approche, la recon guration dynamique se résume à l'automatisation de l'ajout ou de la suppression d'un agent du système. Ce changement de con guration, permet au système d'atteindre ses objectifs e cacement par l'intégration des nouveaux services o erts par les nouveaux agents, il permet aussi d'assurer le bon fonctionnement du système dans les di érentes situations. A n de valider notre approche, les réseaux LPWANs ont été choisis comme plate-forme de communications pour interconnecter tous les périphériques pervasifs, en raison de sa grande couverture en termes de réseau.

Mots clés : Recon guration dynamique; Système pervasif; Système multi-agent; LPWAN.