Académie de Montpellier
**Université Montpellier II**
**S**ciences et **T**echniques du **L**anguedoc

# Thèse

pour obtenir le grade de docteur

| | | |
|---|---|---|
| *Spécialité* | : | **Informatique** |
| *Formation Doctorale* | : | **Informatique** |
| *École Doctorale* | : | **Information, Structures, Systèmes** |

présenté et soutenue publiquement par

**Mohamad Fady Hamoui**

## Un Système Multi-Agents à Base de Composants pour l'Adaptation Autonomique au Contexte Application à la Domotique

Soutenue le 13 décembre 2010, devant le jury composé de :

**Rapporteurs**
Philippe Aniorté (Prof.) ..... LIUPPA, Université de Pau et des Pays de l'Adour
Lionel Seinturier (Prof.) ..... LIFL, Université de Lille 1

**Examinateurs**
Olivier Barais (MCF) ..... IRISA, Université de Rennes 1
Stefano A. Cerri (Prof.) ..... LIRMM, Université de Montpellier II
Didier Donsez (Prof.) ..... LIG, Université de Grenoble 1

**Directrice de thèse**
Marianne Huchard (Prof.) ..... LIRMM, Université de Montpellier II

**Co-Directeurs de thèse**
Christelle Urtado (MCF) ..... LGI2P, École des Mines d'Alès
Sylvain Vauttier (MCF) ..... LGI2P, École des Mines d'Alès

*À ma mère et mon père bien-aimés...*

# Remerciements

Je remercie Michel Vasquez directeur du LGI2P et la direction de l'École des Mines d'Alès pour m'avoir permis de préparer cette thèse dans leur établissement.

Tout d'abord, je tiens à remercier et exprimer toute ma reconnaissance à ma directrice de thèse, Marianne Huchard professeur au LIRMM de l'Université Montpellier II et à mes encadrants Christelle Urtado et Sylvain Vauttier, maîtres de conférences à l'École des Mines d'Alès. Ils m'ont initié à la recherche, m'ont apporté toute leur expérience afin de mener à terme ce travail de recherche. Ils m'ont toujours soutenu dans mes idées et ont toujours été patients dans des situations parfois délicates. Ce fut un plaisir de travailler avec eux.

Je tiens à remercier l'ensemble des membres du jury de cette thèse. Je remercie Philippe Aniorté, professeur au LIUPPA à l'Université de Pau et des Pays de l'Adour, et Lionel Seinturier, professeur au LIFL de l'université de Lille, pour avoir accepté de rapporter cette thèse et avoir contribué à son amélioration. Je remercie également Olivier Barais, maître de conférence à l'IRISA de l'Université de Rennes 1, Stefano Cerri, professeur au LIRMM de l'Université Montpellier II et Didier Donsez, professeur au LIG de l'Université de Grenoble 1 de m'avoir fait l'honneur de participer à ce jury.

Je remercie également les enseignants chercheurs, les secrétaires et l'ensemble du personnel du LGI2P pour l'ambiance et l'esprit d'amitié. Je remercie notamment Gérard Dray maître de conférences à l'École des Mines d'Alès et Pascal Poncelet professeur au LIRMM de l'Université Montpellier II, pour leur soutien, leurs encouragements et leur optimisme.

Merci aussi à mes collègues et amis Abdelhak, Sami, Ouael, Jorge, Nicolas, Jean, Yulin, Nicolas, Sihem, Mohameth, Benjamin ainsi qu'à tous les autres qui se reconnaîtront ici. Je remercie notamment Guillaume pour son aide très précieuse.

J'aurais tellement aimé que les membres de ma famille soient tous présents le jour de ma soutenance. En tous cas, je ne les remercierai jamais assez.

Je remercie mes chers parents, ma mère et mon père, les être les plus chers au monde qui ont tout le temps cru en moi et m'ont soutenu dans les périodes les plus difficiles.

Je remercie mes grands-mères, mes beaux-parents et mes frères Wael, Danny, et Fares, pour leur soutien et leurs encouragements. Une grande pensée à mon beau-père...

Enfin, je remercie ma chère épouse Zeina qui a été à mes côtés et qui a su m'encourager et me soutenir pendant les longues nuits de travail.

# Table des matières

**1 Introduction** **1**
1.1 Contexte de la thèse . . . 1
1.2 Environnements domotiques . . . 2
1.2.1 Équipements . . . 2
1.2.2 Utilisateurs . . . 3
1.2.3 Caractéristiques des contextes . . . 3
1.2.3.1 Équipements . . . 3
1.2.3.2 Utilisateurs . . . 4
1.3 Systèmes domotiques . . . 4
1.3.1 Définition . . . 4
1.3.2 Problématique . . . 5
1.3.2.1 Équipements . . . 5
1.3.2.2 Utilisateurs . . . 6
1.3.2.3 Système domotique . . . 7
1.4 Objectif de la thèse . . . 7
1.5 Plan du manuscrit . . . 8

**2 État de l'art** **11**
2.1 Introduction . . . 11
2.2 Systèmes domotiques existants . . . 12
2.2.1 Systèmes domotiques centralisés . . . 12
2.2.2 Systèmes domotiques distribués . . . 13
2.2.3 Systèmes domotiques fédérés . . . 13
2.3 Analyse : Architectures des systèmes domotiques . . . 14
2.4 Analyse : Gestion du contexte . . . 15
2.4.1 Gestion des équipements . . . 15
2.4.2 Gestion des utilisateurs . . . 16
2.4.3 Gestion du système . . . 16
2.5 Fonctionnalités des systèmes domotiques . . . 17
2.5.1 Fonctionnalités explicites des systèmes domotiques . . . 17
2.5.1.1 Gestion des services . . . 17
2.5.1.2 Gestion des scénarios . . . 17
2.5.1.3 Gestion des profils des utilisateurs . . . 17
2.5.1.4 Gestion du paramétrage du système . . . 18

2.5.1.5 Fonctionnalités explicites dans les systèmes domotiques étudiés . 18
2.5.2 Fonctionnalités implicites des systèmes domotiques . . . 19
2.5.2.1 Gestion de l'environnement . . . 19
2.5.2.2 Gestion du système . . . 19
2.5.2.3 Fonctionnalités implicites dans les systèmes domotiques étudiés . 19
2.6 Qualités requises des systèmes domotiques . . . 20
2.6.1 Configurabilité . . . 20
2.6.2 Adaptation dynamique . . . 20
2.6.3 Adaptation autonomique . . . 21
2.6.4 Sensibilité au contexte . . . 21
2.6.5 Discussion . . . 21
2.7 Conclusion . . . 22

**3 SAASHA : Architecture et fonctionnalités du SMA 23**
3.1 Introduction . . . 23
3.2 Combinaison des paradigmes *Agent* et *Composant* . . . 24
3.3 Architecture globale de SAASHA . . . 25
3.3.1 Agents de SAASHA . . . 25
3.3.2 Agents d'interface . . . 26
3.3.3 Agents de contrôle . . . 28
3.3.3.1 Agents de contrôle d'équipements . . . 28
3.3.3.2 Agents de contrôle manageurs . . . 28
3.3.3.3 Agents de contrôle médiateurs . . . 29
3.3.4 Analyse de l'architecture de SAASHA . . . 29
3.4 Mode de fonctionnement de SAASHA . . . 30
3.5 Phase de configuration . . . 31
3.5.1 Initialisation . . . 31
3.5.2 Paramétrage du système . . . 32
3.5.2.1 Paramétrage automatique . . . 32
3.5.2.2 Paramétrage manuel . . . 32
3.6 Phase opérationnelle . . . 34
3.6.1 Gestion des profils des utilisateurs . . . 35
3.6.1.1 Définition d'un profil . . . 35
3.6.1.2 Modification d'un profil . . . 36
3.6.1.3 Suppression d'un profil . . . 36
3.6.2 Gestion des services . . . 36
3.6.3 Gestion des scénarios . . . 37
3.6.3.1 Définition d'un scénario . . . 37
3.6.3.2 Modification d'un scénario . . . 40
3.6.3.3 Suppression ou la suspension d'un scénario . . . 40
3.6.4 Gestion de l'environnement . . . 40
3.6.4.1 Ajout d'un nouvel équipement . . . 41
3.6.4.2 Suppression d'un équipement . . . 41
3.6.4.3 Gestion des conflits . . . 41
3.6.5 Gestion du système . . . 42
3.6.5.1 Ajout d'un agent de contrôle . . . 43
3.6.5.2 Gestion des DCA . . . 43
3.6.5.3 Gestion des MgrCA . . . 44

3.7 Conclusion . . . 45
3.8 Limites et perspectives . . . 46

**4 SAASHA : Architectures internes et mécanismes d'adaptation des agents 49**
4.1 Introduction . . . 49
4.2 Méta-modèle de SAASHA . . . 50
4.3 Agents d'interface . . . 52
4.3.1 Architecture interne . . . 52
4.3.2 Cycle de vie . . . 54
4.3.3 Adaptation de la présentation . . . 56
4.3.3.1 Identification des utilisateurs . . . 58
4.3.3.2 Synchronisation des données . . . 59
4.4 Agents de contrôle . . . 59
4.4.1 Architecture interne . . . 60
4.4.2 Cycle de vie . . . 61
4.4.3 Adaptation de l'organisation . . . 65
4.4.3.1 Processus de synchronisation des données . . . 67
4.4.3.2 Processus de déploiement des scénarios . . . 68
4.4.4 Adaptation du comportement . . . 71
4.4.4.1 Adaptation du comportement de contrôle . . . 72
4.4.4.2 Adaptation de comportement de coordination . . . 75
4.5 Conclusion . . . 80
4.6 Limites et perspectives . . . 81

**5 Implémentation de SAASHA 83**
5.1 Introduction . . . 83
5.2 SAASHA et UPnP . . . 84
5.2.1 UPnP . . . 84
5.2.2 Mécanisme d'identification des agents et des équipements . . . 87
5.2.3 Protocole de communication . . . 87
5.3 SAASHA et OSGi . . . 89
5.3.1 OSGi . . . 90
5.3.2 Mécanisme de génération des composants de contrôle . . . 90
5.3.3 Mécanisme de génération des composants de coordination . . . 92
5.4 Expérimentations . . . 95
5.5 Simulateur : Domus . . . 96
5.6 Conclusion . . . 97
5.7 Limites et perspectives . . . 98

**6 Conclusion 99**
6.1 Contributions . . . 99
6.2 Perspectives . . . 101
6.2.1 Gestion des utilisateurs . . . 101
6.2.2 Gestion du système . . . 102
6.2.3 Gestion des équipements . . . 104

**Bibliographie 104**

**A 113**

# Table des figures


3.1 Les différentes interactions de *SAASHA* . . . . . 27
3.2 Le fonctionnement global du système . . . . . 30
3.3 L'onglet de paramétrage du champ de contrôle des DCA . . . . . 33
3.4 L'interface d'exécution des services . . . . . 36
3.5 L'interface de définition des scénarios . . . . . 38

4.1 Une vue du méta-modèle de *SAASHA* : Agents et Composants . . . . . 50
4.2 L'architecture interne des GUIA : Méta-niveau . . . . . 52
4.3 Une vue du méta-modèle de *SAASHA* : Annuaires des GUIA . . . . . 53
4.4 Le diagramme d'état d'un GUIA . . . . . 55
4.5 L'état *UserInteraction* du cycle de vie d'un GUIA . . . . . 56
4.6 L'état *Listening* du cycle de vie d'un GUIA . . . . . 57
4.7 L'architecture interne des agents de contrôle : Méta-niveau . . . . . 60
4.8 Une vue du méta-modèle de *SAASHA* : Annuaire des agents . . . . . 62
4.9 Le diagramme d'état du cycle de vie d'un agent de contrôle . . . . . 63
4.10 L' état *Active* du cycle de vie d'un agent de contrôle . . . . . 64
4.11 Le rôle des composants de contrôle et de coordination . . . . . 71
4.12 Une vue du méta-modèle de *SAASHA* : Annuaire des composants et des services 72
4.13 L'architecture interne d'un DCA après l'adaptation du contrôle . . . . . 74
4.14 Assemblage de composants implémentant le scénario "*température agréable le soir*" dans l'architecture interne d'un DCA . . . . . 77
4.15 Des parties des architectures internes des DCA qui implémentent le scénario *température agréable le soir* . . . . . 79

5.1 Un extrait du descripteur standard d'un chauffage, ventilateur et climatiseur (HVAC) 85
5.2 Un extrait du descripteur standard du service *FanSpeed* de l'équipement *HVAC* . 86
5.3 Un extrait du descripteur UPnP d'un GUIA . . . . . 86
5.4 Un extrait du descripteur UPnP d'un agent de contrôle . . . . . 87
5.5 Le protocole de communication dans SAASHA . . . . . 88
5.6 Une partie du code généré qui représente les interfaces fournies par le composant de contrôle de l'équipement HVAC . . . . . 92
5.7 Un extrait du code généré qui représente l'implémentation des services fournis par le composant de contrôle de l'équipement HVAC . . . . . 93
5.8 Un extrait du code généré pour le composant de coodination qui implémente le scénario *température agréable le soir* . . . . . 94

5.9 Domus : simulateur d'une maison intelligente . . . . . 97

A.1 Méta-modèle de SAASHA . . . . . 114

# Liste des tableaux

2.1 Analyse des systèmes domotiques existants par rapport à la gestion du contexte . 15
2.2 Analyse des systèmes domotiques existants par rapport aux fonctionnalités explicites 18
2.3 Analyse des systèmes domotiques existants par rapport aux fonctionnalités implicites 20
2.4 Analyse des systèmes domotiques existants par rapport aux qualités requises . . . 21

4.1 Processus de déploiement des scénarios . . . . . 70

5.1 Données sur les opérations de services standards . . . . . 91

Chapitre 1

# Introduction

**Sommaire**


**1.1 Contexte de la thèse . . . . . 1**
**1.2 Environnements domotiques . . . . . 2**
1.2.1 Équipements . . . . . 2
1.2.2 Utilisateurs . . . . . 3
1.2.3 Caractéristiques des contextes . . . . . 3
**1.3 Systèmes domotiques . . . . . 4**
1.3.1 Définition . . . . . 4
1.3.2 Problématique . . . . . 5
**1.4 Objectif de la thèse . . . . . 7**
**1.5 Plan du manuscrit . . . . . 8**


## 1.1 Contexte de la thèse

L'informatique ubiquitaire [84, 73, 83] est un paradigme dans lequel des systèmes numériques, intégrés dans les objets de la vie quotidienne, sont mis en réseau pour coopérer et fournir un environnement intelligent qui facilite l'utilisation et automatise l'exécution de services pour les utilisateurs. Une exigence première dans la conception d'un système ubiquitaire contrôlant un environnement intelligent est que les utilisateurs n'aient pas à adapter leurs comportements pour l'utiliser. Au contraire, c'est au système de gérer cette adaptation pour satisfaire au mieux les besoins et les préférences de ses utilisateurs.

Ces systèmes correspondent ainsi à une catégorie de systèmes identifiée par Lehman et Belady [25], celle des systèmes intégrés à un environnement et qui sont donc contraints à co-évoluer avec cet environnement. Cette contrainte a été formulée comme l'une des lois d'évolution de Lehman [57, 58] : "*systems must be continually adapted or they become progressively less satisfactory*" [56].

Cette problématique est encore plus critique dans le cas des systèmes ubiquitaires, qui sont

intégrés à des environnements étendus, ouverts, prenant en compte la variété et le changement des besoins des utilisateurs, donc des environnements qui varient rapidement et constamment. La gestion de l'évolution [60, 5, 24, 23], et plus spécifiquement de l'adaptation à son environnement, est ainsi un élément crucial de la conception d'un système ubiquitaire.

L'informatique ubiquitaire comprend une variété de domaines d'application dans la vie quotidienne de l'utilisateur. Dans cette thèse, nous nous concentrons sur les environnements domotiques. Aussi, nous proposons pour ces systèmes des mécanismes d'adaptation autonomique [52], basés sur l'évolution d'architectures à bases de composants [36, 37, 78, 90] en fonction des informations qu'ils recueillent sur les changements détectés dans leurs environnements (sensibilité au contexte [59, 29]).

Dans le reste de ce chapitre, nous décrivons dans un premier temps les différents éléments des environnements domotiques et leurs caractéristiques. Cette description nous permettra d'introduire dans un deuxième temps les systèmes domotiques et les objectifs attendus de ces systèmes dans les environnements domotiques.

## 1.2 Environnements domotiques

Un *environnement domotique* est défini par un ensemble d'équipements domestiques contrôlables, connectés à un réseau et un ensemble d'utilisateurs. Il est découpé en un ensemble de contextes qui définissent son organisation.
Chaque *contexte* délimite un lieu dans lequel se trouvent des équipements et des utilisateurs. Un contexte peut être composé d'autres contextes.

Prenons l'exemple d'un environnement domotique, représenté par un appartement composé d'un salon, un bureau, trois chambres et une cuisine. Y habitent quatre personnes : un père, une mère et deux enfants. Le salon, par exemple, contient une télévision, deux radiateurs, un climatiseur, deux volets, une horloge, trois lampes, deux fenêtres et une porte verrouillables, deux caméras, un détecteur de présence et de fumée, une alarme. Tout au long de ce manuscrit, nous nous servirons de cet environnement comme cas d'étude pour illustrer des exemples explicatifs.

Dans ce qui suit, nous définissons les termes *équipement* et *utilisateur* et nous décrivons les caractéristiques liées à ces deux éléments.

### 1.2.1 Équipements

Un *équipement* est un appareil qui fournit un ensemble de services aux utilisateurs.
Un *service* permet d'interagir avec l'équipement. Il offre un ensemble d'opérations et émet un ensemble d'événements.
Une *opération* permet d'agir sur l'équipement soit pour exécuter une action, soit pour obtenir une information. Pour exécuter certaines opérations, il peut être nécessaire de fournir des valeurs de paramètre d'entrée.
Un *événement* reflète un changement d'état de l'équipement.

Par exemple, un radiateur fournit un service de contrôle qui permet de gérer le fonctionnement du radiateur.
Le service de contrôle offre trois opérations : une première opération pour allumer le radiateur qui prend en entrée la puissance de chauffage souhaitée, une opération pour éteindre le radiateur et une opération pour obtenir la température ambiante.
Le service de contrôle émet un événement lorsque la température ambiante d'une pièce change. L'événement contient la nouvelle valeur de la température.

### 1.2.2 Utilisateurs

Un *utilisateur* est une personne physique qui interagit avec les équipements disponibles dans l'environnement, en fonction de ses droits d'accès, pour réaliser ses besoins.
Les *droits d'accès* d'un utilisateur spécifient l'ensemble des équipements avec lesquels il a le droit d'interagir. Par exemple, les enfants n'ont pas le droit de manipuler la télévision qui se trouve dans la chambre de leurs parents.
Les *besoins* d'un utilisateur se traduisent soit sous la forme d'exécution directe de services isolés (par exemple, l'utilisateur allume la lumière de la chambre le soir), soit sous la forme de scénarios (par exemple, l'utilisateur ferme les volets, déclenche le système d'alarme et, enfin, éteint les lumières avant de dormir).
Un *scénario* est l'exécution combinée d'un ensemble de services à un moment donné pour satisfaire un besoin complexe. Le scénario précédent a été exécuté par l'utilisateur pour réaliser, au moment de dormir, le besoin de sécurité.

### 1.2.3 Caractéristiques des contextes

Les environnements domotiques peuvent être composés de contextes de natures différentes. Les contextes sont caractérisés par deux aspects : la *variabilité* et le *changement*. La variabilité correspond aux différences entre les éléments de différents contextes, tandis que le changement correspond aux modifications des éléments qui composent un contexte donné. Un changement est distinct d'un événement émis par un équipement qui reflète le changement de son état (section 1.2.1).

Nous présentons dans cette section les caractéristiques des équipements et des utilisateurs des environnements liées à la variabilité et au changement.

#### 1.2.3.1 Équipements

Les équipements se caractérisent par leur type, les services qu'ils fournissent, le contexte dans lequel ils se situent et leur mobilité.

- **Variabilité.** Les contextes contiennent différents nombres et types d'équipements. Les équipements de types différents fournissent des services différents, en nombres et en types.

Par exemple, la cuisine et le salon d'un appartement ou les salons de deux appartements ne contiennent pas les mêmes nombre et types d'équipements. De même, le radiateur du salon n'a pas le même fonctionnement que le radiateur de la cuisine.

- **Changement.** Les équipements sont des appareils électriques qui peuvent être ajoutés ou supprimés/éteints ou qui peuvent tomber en panne à tout moment.

#### 1.2.3.2 Utilisateurs

Les utilisateurs ont des droits d'accès et expriment leurs besoins. Les droits et les besoins évoluent au fil du temps.

- **Variabilité.** Les contextes contiennent différents nombres d'utilisateurs. Les utilisateurs n'ont pas tous les mêmes droits d'accès et les mêmes besoins (scénarios). Un même besoin peut être exprimé différemment par les utilisateurs. Par exemple, un utilisateur peut uniquement déclencher le système de sécurité (sans fermer les volets et éteindre les lumières) pour réaliser le besoin de sécurité (section 1.2.2), s'il juge que cela est suffisant.
- **Changement.** Les droits d'accès et les besoins des utilisateurs peuvent aussi changer ou évoluer au cours du temps. Ces changements peuvent être liés aux changements des équipements. Par exemple, après l'ajout d'un système de détection de fumée, l'utilisateur peut modifier le scénario de sécurité en y ajoutant l'allumage de ce nouveau détecteur pour réaliser le besoin de sécurité. De nouveaux utilisateurs peuvent être intégrés à un environnement domotique ou supprimés à tout moment. Les nouveaux utilisateurs peuvent avoir des besoins différents.

## 1.3 Systèmes domotiques

Nous présentons dans cette section une définition des systèmes domotiques et les problèmes relatifs aux caractéristiques des environnements domotiques auxquels doit faire face un système domotique.

### 1.3.1 Définition

Un *système domotique* se connecte au réseau d'un environnement domotique et est capable de contrôler les équipements pour réaliser les besoins des utilisateurs.
Les utilisateurs peuvent interagir avec un système pour réaliser leurs besoins selon trois rôles d'utilisateur :

- *Utilisateur Final* (UF). Ce rôle permet à l'utilisateur d'exécuter des services et des scénarios en fonction de ses droits d'accès et des équipements disponibles. Par exemple, les enfants qui n'ont accès qu'aux équipements de leur chambre peuvent ouvrir les volets ou allumer

la lumière. Le père déclenche l'exécution du scénario de mise en sécurité (section 1.2.2) en appuyant sur un bouton et le système se charge de l'exécution des services.

- *Utilisateur Privilégié* (UP). Ce rôle permet à l'utilisateur de gérer les droits d'accès des utilisateurs. Par exemple, le père et la mère peuvent avoir ce rôle pour gérer les droits d'accès de leurs enfants aux services fournis par les équipements. Ils peuvent interdire aux enfants d'avoir accès au système d'alarme.

- *Administrateur* (ADM). Ce rôle est attribué à certains utilisateurs pour la gestion du paramétrage du système. Ils peuvent donc paramétrer l'architecture du système et l'accès du système aux équipements. Les ADM ne sont pas des développeurs.

Un système identifie un équipement à l'aide de descripteurs fournis par l'équipement. Chaque équipement fournit deux types de descripteurs ; un *descripteur d'équipement* et des *descripteurs de service*. Le descripteur d'équipement contient des informations sur l'équipement comme par exemple son identifiant, son type, son contexte et la liste des services qu'il fournit. L'équipement fournit aussi un descripteur de service pour chaque service qu'il fournit. Le descripteur de service fournit la liste des opérations offertes par le service et la liste des événements qu'il émet. Ces concepts sont conformes au standard *UPnP* [10, 9], défini par un consortium industriel pour faciliter le développement d'environnements domotiques. Des exemples de ces descripteurs sont fournis dans la section 5.2.
Un système exécute les scénarios à partir de services fournis par des équipements distribués dans l'environnement. Pour exécuter un scénario, un système coordonne ces services. Les systèmes permettent aussi l'automatisation de l'exécution des scénarios. Dans ce cas, le déclenchement de l'exécution d'un scénario est géré par un événement. Par exemple, le scénario sécurité peut être programmé de manière à être exécuté à 22H00.

### 1.3.2 Problématique

Nous remarquons que les environnements domotiques sont composés de trois éléments principaux : les utilisateurs, les équipements et le système domotique. Ces trois éléments interagissent ensemble pour la satisfaction des besoins des utilisateurs. Pour répondre aux besoins des utilisateurs, un système domotique doit être en mesure de gérer les caractéristiques des environnements domotiques (la variabilité et les changements des contextes, voir la section 1.2.3). Nous verrons dans les sections suivantes que de nouvelles caractéristiques liées au système doivent aussi être prises en compte.

#### 1.3.2.1 Équipements

Un système doit être capable d'identifier et de contrôler tous les équipements d'un environnement et de gérer l'apparition et la disparition des équipements.

- **Variabilité.** Un système doit être capable de contrôler tous les types d'équipements standards qui fournissent des services standards et tous les types d'équipements non standards

disponibles dans un environnement. Par exemple, une lampe binaire peut être considérée comme un équipement standard. Par contre, une lampe à luminosité variable qui fournit un service différent qui permet aussi de modifier l'intensité de la lumière n'est pas un équipement standard.
Un système doit être capable d'identifier le contexte dans lequel se situent les équipements lors de l'exécution des services et des scénarios. Le contexte dans lequel se trouve un équipement peut jouer un rôle important. Par exemple, la télévision du salon est différente de celle de la chambre et elles n'ont pas le même rôle. Par contre, l'horloge du salon et celle de la chambre peuvent avoir le même rôle qui est de donner la même heure.

- **Changement.** Un système doit être capable de détecter la panne ou la suppression d'un équipement. Ainsi, un utilisateur ne doit plus être en mesure de demander l'exécution de services fournis par des équipements qui ne sont plus disponibles ou de scénarios qui dépendent de ces équipements.
  Le système doit aussi être capable de contrôler les nouveaux équipements ajoutés à l'environnement. Les utilisateurs doivent être en mesure d'exécuter des services fournis par ces nouveaux équipements ou des scénarios précédemment suspendus à cause de la disparition d'équipements.

#### 1.3.2.2 Utilisateurs

Un système doit être capable de répondre à tous les besoins des utilisateurs en fonction de leurs droits d'accès, de gérer les changements de leurs besoins et de leurs droits d'accès et de gérer l'ajout ou la suppression des utilisateurs.

- **Variabilité.** Un système doit gérer les droits d'accès des utilisateurs et permettre aux utilisateurs d'exécuter uniquement les services des équipements dont ils ont le contrôle et les scénarios compatibles avec leurs droits d'accès. Par exemple, un enfant ne doit pas être en mesure d'exécuter le scénario sécurité, car il n'a pas accès au système d'alarme.
  Un système doit être capable de répondre aux différents besoins des utilisateurs. Il est possible d'intégrer dans le système des scénarios prédéfinis compatibles avec les équipements de l'environnement. Il est cependant impossible de couvrir tous les besoins des utilisateurs et tous les types d'équipements par des scénarios prédéfinis. Il est donc important que le système permette l'expression et l'ajout de nouveaux scénarios.
  Un système doit gérer les conflits d'exécution de services. Un *conflit* a lieu lorsque deux scénarios déclenchent, sur un même équipement et au même moment, des actions incohérentes ou opposées. Par exemple, le scénario *incendie* qui consiste à déverrouiller les portes et ouvrir les volets peut être exécuté lors d'un incendie et entrerait en conflit avec le scénario *sécurité* si celui-ci était exécuté peu après, car ce dernier implique le verrouillage des portes. Le système doit être capable de détecter et gérer ce type de conflit.

- **Changement.** Un système doit pouvoir prendre en compte les changements de scénarios et de droits d'accès des utilisateurs. Un utilisateur peut ne plus être en mesure d'invoquer des services sur des équipements et de déclencher des scénarios à cause de la modification de ses droits d'accès.
  Un système doit pouvoir prendre en compte de nouveaux utilisateurs. Il doit leur permettre

d'invoquer des services et des scénarios en fonction de leurs droits d'accès.
Un système doit suspendre les scénarios liés aux utilisateurs supprimés.

#### 1.3.2.3 Système domotique

Un système doit permettre aux utilisateurs de paramétrer son architecture et de modifier son architecture sans perturber ou interrompre son fonctionnement.

- **Variabilité.** Un système domotique n'a pas nécessairement une architecture centralisée, constituée d'un équipement de contrôle unique. Au contraire, pour répondre à des problématiques de fiabilité et de performance, un système domotique doit plutôt être conçu comme un système distribué, constitué d'un ensemble d'équipements de contrôles, répartis dans l'environnement domotique. Les différents équipements de contrôle constituant le système domotiques peuvent être paramétrés pour jouer des rôles spécifiques dans la gestion du système domotique (manager, médiateur, contrôleur, ...). Il doit être possible de composer et de paramétrer l'architecture d'un système domotique pour s'adapter aux exigences et aux contraintes de contrôle spécifiques à un environnement.

- **Changement.** Le système doit permettre aux utilisateurs de modifier le paramétrage du système à tout moment.
  Un système doit être capable de gérer les pannes liées à ses composants ou l'ajout de nouveaux composants sans intervention humaine. Une situation classique est la gestion de la mobilité de certains équipements de contrôle, tels que les PDA ou les smartphones. Un utilisateur entre dans une pièce. Son PDA se connecte à l'environnement domotique et vient s'ajouter à l'ensemble des composants du système domotique pour exécuter des scénarios. Le système domotique se comporte alors comme un système ouvert dans lequel des composants apparaissent et disparaissent dynamiquement, provoquant des changements dans l'architecture du système.
  La gestion du changement ne doit pas interrompre ou perturber les scénarios en cours d'exécution.

## 1.4 Objectif de la thèse

Comme nous l'avons vu dans les sections précédentes, la variabilité des environnements et les changements pouvant survenir dans ces environnements sont de nature diverse. La standardisation des équipements et des besoins n'est pas suffisante pour répondre à cette problématique. La variabilité des environnements et des changements qui peuvent se produire reste imprévisible. Il est irréaliste d'avoir des configurations de système pour tous les environnements domotiques possibles.

L'objectif de la thèse est de concevoir un système capable de s'adapter à la variabilité du contexte des environnements domotiques et aux changements de ce contexte. Pour pouvoir s'adapter, un système doit être capable de gérer le contexte de la manière suivante :

- **Gestion des équipements.** Le système doit être capable de détecter et d'identifier les équipements, les services qu'ils fournissent et le contexte dans lequel ils se situent. Il doit pouvoir contrôler tous les équipements détectés. Le système doit être capable de coordonner les services fournis par les équipements pour l'exécution des scénarios. Enfin, il doit détecter l'apparition et la disparition des équipements.
- **Gestion des utilisateurs.** Le système doit permettre aux utilisateurs d'exécuter des services fournis par les équipements et de définir eux-mêmes leurs propres scénarios à partir d'interfaces graphiques. Le contenu de ces interfaces doit être modifié en fonction des services fournis par les équipements et des droits d'accès des utilisateurs. Il doit aussi gérer les modifications des scénarios et des droits d'accès des utilisateurs. Enfin, il doit gérer les conflits d'exécution des scénarios.
- **Gestion du système.** Le système doit permettre aux utilisateurs de définir et de paramétrer son architecture en fonction de l'environnement dans lequel il est intégré. Le système doit être fiable et tolérant aux pannes. Il doit gérer les changements dynamiques de son architecture (ajouts et pannes d'éléments).

La gestion de la variabilité des contextes et des changements qui peuvent s'y produire doit être réalisée par l'*adaptation autonomique et dynamique* du système qui lui permet de s'adapter sans connaissance *a priori* du contexte dans lequel il va être intégré, sans intervention humaine et sans perturber son fonctionnement.

## 1.5 Plan du manuscrit

La suite du manuscrit est organisée de la manière suivante.

**Chapitre 2.** Ce chapitre présente un panorama de travaux menés dans le domaine des environnements domotiques. Nous décrivons les types d'architectures des systèmes étudiés. Puis, nous analysons ces systèmes en fonction de leur gestion du contexte. Cette analyse nous permet d'établir une typologie de fonctionnalités que doivent offrir les systèmes domotiques. Premièrement, un ensemble de fonctionnalités explicites correspondent à la gestion du système par les utilisateurs. Elles leur permettent de personnaliser le fonctionnement du système. Deuxièmement, un ensemble de fonctionnalités implicites correspondent à la gestion de l'environnement et de l'architecture du système par le système lui-même. Elles permettent au système de gérer automatiquement son adaptation pour faire face aux changements qui se produisent dans le système et l'environnement. Enfin, nous terminons le chapitre par un ensemble de qualités que les systèmes domotiques doivent posséder pour réaliser ces fonctionnalités explicites et implicites.

**Chapitre 3.** Nous présentons dans ce chapitre l'architecture globale de notre système, *SAASHA*, un système multi-agents à base de composants pour la domotique. Nous présentons les types et rôles de nos agents, leurs interactions et leur mode de fonctionnement. Nous décrivons comment *SAASHA* réalise les fonctionnalités explicites offertes aux utilisateurs et les fonctionnalités implicites gérées par le système.

**Chapitre 4.** Les architectures internes, les cycles de vie et les mécanismes d'adaptation des

agents sont présentés dans ce chapitre. Les architectures internes de nos agents sont basées sur les composants. Le comportement initial de nos agents est réalisé par un assemblage de composants qui représente leur méta-niveau (comportements de contrôle des agents). Nos agents acquièrent de nouveaux comportements par la génération, le déploiement et l'assemblage de nouveaux composants. Nous présentons aussi trois niveaux d'adaptation : adaptation de la présentation (interfaces graphiques), adaptation de l'organisation (architecture du système de contrôle multi-agents) et adaptation du comportement (architecture interne des agents).

**Chapitre 5.** Dans ce chapitre, nous présentons une implémentation de *SAASHA* basée sur les technologies *OSGi* et *UPnP*. Nous présentons le protocole de découverte et de communication entre agents. Ensuite, nous présentons le mécanisme de génération des composants. Nous décrivons les expérimentations qui ont été réalisées et Domus, le simulateur d'environnement domotique que nous avons développé.

Chapitre 2

# État de l'art

**Sommaire**


**2.1 Introduction . . . . . 11**
**2.2 Systèmes domotiques existants . . . . . 12**
2.2.1 Systèmes domotiques centralisés . . . . . 12
2.2.2 Systèmes domotiques distribués . . . . . 13
2.2.3 Systèmes domotiques fédérés . . . . . 13
**2.3 Analyse : Architectures des systèmes domotiques . . . . . 14**
**2.4 Analyse : Gestion du contexte . . . . . 15**
2.4.1 Gestion des équipements . . . . . 15
2.4.2 Gestion des utilisateurs . . . . . 16
2.4.3 Gestion du système . . . . . 16
**2.5 Fonctionnalités des systèmes domotiques . . . . . 17**
2.5.1 Fonctionnalités explicites des systèmes domotiques . . . . . 17
2.5.2 Fonctionnalités implicites des systèmes domotiques . . . . . 19
**2.6 Qualités requises des systèmes domotiques . . . . . 20**
2.6.1 Configurabilité . . . . . 20
2.6.2 Adaptation dynamique . . . . . 20
2.6.3 Adaptation autonomique . . . . . 21
2.6.4 Sensibilité au contexte . . . . . 21
2.6.5 Discussion . . . . . 21
**2.7 Conclusion . . . . . 22**


## 2.1 Introduction

Dans le chapitre 1, nous avons défini les différents éléments des environnements et des systèmes domotiques. Puis, nous avons décrit les caractéristiques de ces environnements et le rôle des systèmes domotiques dans la gestion de ces caractéristiques.

Dans ce chapitre, nous présentons, dans un premier temps, un état de l'art sur les systèmes domotiques existants. Nous décrivons le fonctionnement des différents systèmes étudiés en fonction de leur architecture. Nous présentons, ensuite, les avantages et les inconvénients de ces systèmes en fonction de leur architecture et de leur gestion du contexte. Cela nous permettra, dans un second temps, d'établir un ensemble de fonctionnalités que les systèmes domotiques doivent offrir aux utilisateurs. Enfin, nous terminons ce chapitre avec un ensemble de qualités requises que doivent satisfaire les systèmes pour réaliser les fonctionnalités.

## 2.2 Systèmes domotiques existants

Dans cette section, nous présentons un ensemble de travaux menés dans le domaine des environnements domotiques. Nous commençons par une description rapide de ces systèmes, puis nous les analysons en fonction de leur architecture et de leur gestion du contexte. Cette analyse nous permet de vérifier si les systèmes étudiés possèdent les caractéristiques que nous avons établies dans la section 1.3.2.

En fonction de leur architecture, nous classons les systèmes domotiques existants en trois catégories : les systèmes centralisés, les systèmes distribués et les systèmes fédérés.

### 2.2.1 Systèmes domotiques centralisés

Pour un certain nombre de systèmes, l'ensemble des équipements de l'environnement est piloté par un système de contrôle unique, ce qui facilite la gestion des fonctions d'administration et de contrôle de l'environnement mais peut générer des problèmes de fiabilité, de performance et d'extensibilité. Ces systèmes sont appelés systèmes domotiques centralisés.

MADCAR [39] propose un système à base de composants permettant l'assemblage dynamique et automatique d'architectures. L'assemblage nécessite quatre éléments : la description d'une application, un ensemble de composants, les données du contexte fournies par des capteurs et la politique qui dirige les décisions d'assemblage. La description d'une application consiste en un ensemble de configurations prédéfinies, décrites par des assemblages de composants abstraits. Le système est responsable de sélectionner une configuration convenable et de choisir les composants correspondant aux composants abstraits.

SLCA [47, 79] propose un outil qui permet aux programmeurs de définir des scénarios. Les scénarios sont décrits comme des aspects pouvant être tissés à l'assemblage de composants définissant l'architecture courante du système pour y ajouter des comportements contrôlant l'exécution des scénarios. L'application d'un aspect peut être déclenchée par un utilisateur ou par la détection d'un équipement, permettant d'adapter le système aux changements de contexte.

ENTIMID [62, 61] propose un système à base de composants qui supporte l'hétérogénéité des équipements. Chaque fabricant peut utiliser ce framework pour encapsuler ses technologies propriétaires dans des composants standardisés qui décrivent les fonctionnalités des équipements en termes de capteurs et d'actionneurs. Les capteurs et les actionneurs peuvent être combinés en

scénarios consistant en une succession ordonnée d'actions élémentaires (l'écoute d'un capteur ou le déclenchement d'un actionneur).

PHS [63, 48] propose un système qui détecte automatiquement les équipements dans son environnement et génère des interfaces interactives qui listent les services fournis par les équipements détectés. Les utilisateurs interagissent avec le système à travers des interfaces pour exécuter des services.

### 2.2.2 Systèmes domotiques distribués

D'autres systèmes font l'hypothèse que tous les équipements possèdent des ressources suffisantes pour être programmés et interagir de manière autonome et directe avec les autres équipements (architecture pair à pair). Ces systèmes sont appelés systèmes domotiques distribués.

SODAPOP [32, 45] propose de documenter formellement les opérations des équipements à l'aide de règles décrivant leurs préconditions (état initial de l'environnement) et leur effets (changement d'états de l'environnement). Les besoins des utilisateurs sont exprimés comme les effets qu'ils souhaitent obtenir. Le système calcule automatiquement, par inférence sur les règles, des enchaînements d'opérations produisant les effets demandés par les utilisateurs. Le calcul par inférence des scénarios est réalisé de manière centralisée (composant de planification). Le contrôle de l'exécution des scénarios est distribué sur l'ensemble des équipements.

HNS [65, 66] propose d'exprimer des scénarios sous forme de séquences d'exécutions de services. L'ensemble des scénarios est agrégé en un scénario global unique puis découpé en fragments déployés sur les équipements sous forme de descripteurs. Chaque descripteur définit sous forme de pré/post traitements, les comportements qui sont exécutées lors de l'invocation d'un service de l'équipement. Ces pré/post traitements consistent en des invocations d'autres services. Il est ainsi possible de définir des enchaînements d'exécutions de services requérant éventuellement des collaborations entre les équipements. Le calcul de la répartition des scénarios est centralisé et l'intégration d'un nouveau scénario entraîne la redéfinition complète du comportement de certains équipements. L'exécution des scénarios est, par contre, entièrement distribuée.

### 2.2.3 Systèmes domotiques fédérés

Les systèmes fédérés, quant à eux, proposent une combinaison de ces deux types d'architectures. Le contrôle du système est distribué sur un ensemble de contrôleurs d'équipements. Chaque contrôleur d'équipements contrôle un ensemble d'équipements d'une manière centralisée. Ces systèmes de contrôle fédérés sont implémentés en général comme des systèmes multi-agents.

MASML [88] propose un système qui réalise les scénarios des utilisateurs par génération d'agents mobiles. Chaque agent mobile correspond au scénario d'un utilisateur. Pour réaliser un scénario, un agent mobile doit se déplacer d'un équipement à un autre en fonction de la localisation du service qu'il doit exécuter à chaque étape du scénario. Le scénario est défini comme une séquence de déplacements, d'invocations de services et d'exécutions de scripts (permettant de gérer des structures de contrôle). Chaque déplacement d'un agent est géré par la transmission d'un des-

cripteur de l'agent à l'équipement qui doit instancier l'agent. L'agent interprète le contenu du descripteur (définition d'un scénario) pour exécuter son comportement.

MASBM [75, 76] propose un système composé de systèmes multi-agents. Chaque système est associé à une pièce de l'environnement et contrôle les équipements de la pièce. Les événements émis par les équipements de l'environnement sont envoyés à tous les systèmes. Chaque système possède un entrepôt de composants (drivers) pour le contrôle des différents types d'équipement et un entrepôt de règles pouvant être associées à chaque composant de contrôle. Lorsque le système détecte un équipement, il déploie le composant de contrôle approprié et recherche les règles associées. Le système possède un composant responsable de la vérification des règles. Lorsqu'il reçoit un événement, il vérifie la satisfaction des règles, puis demande l'exécution des services mentionnés dans les règles satisfaites. Un utilisateur peut se connecter à un système pour contrôler les équipements par l'invocation d'un service, mais il ne peut pas définir de scénarios personnalisés.

FCSDS [80, 81] est un système multi-agents basé sur des mécanismes de composition dynamique de services basiques. Trois types d'agents sont définis. Des agents d'assistance qui sont responsables du choix des scénarios à exécuter. Le choix est guidé par une commande (exécution d'un service) exprimée par l'utilisateur, ou par des règles prédéfinies basées sur des informations provenant du contexte. Des agents de composition sont responsables du choix des services qui participent à un scénario et déterminent des relations entre les services. Enfin, des agents de services fournissent la liste des services disponibles dans l'environnement par le biais de descripteurs de services publiés par les équipements pour sélectionner les services qui peuvent exécuter une tâche du scénario.

## 2.3 Analyse : Architectures des systèmes domotiques

Dans cette section, nous analysons plus en détail les architectures des systèmes étudiés.

Les systèmes centralisés sont peu fiables. En cas de panne du système, l'accès aux équipements pour exécuter un service et l'implémentation des scénarios ne sont plus possibles. La reconfiguration des systèmes centralisés a la qualité d'être facile à gérer mais leur point de contrôle forme un point de défaillance unique pour les pannes et est un goulot d'étranglement.

Les systèmes distribués ont été conçus pour résoudre ces problèmes. Dans ces architectures décentralisées, les équipements possèdent un comportement autonome. Cependant leurs ressources limitées ne permettent pas de les doter d'un méta comportement réflexif gérant leur adaptation dynamique aux changements de contexte. Par exemple, lors de l'ajout d'un nouvel équipement, un développeur doit intervenir pour l'intégrer dans le système. La panne d'un équipement stoppe un ensemble de scénarios, mais aucune solution automatique n'est proposée pour tenter de maintenir la continuité de service.

Dans les systèmes fédérés, les agents sont embarqués dans des équipements de contrôle dédiés, pour y assurer une gestion autonomique de l'exécution des scénarios. Lorsque les scénarios nécessitent l'utilisation d'un ensemble distribué d'équipements, les interactions requises entre les contrôleurs d'équipements sont gérées grâce à la mobilité ou à la collaboration des agents. Les agents mobiles se déplacent d'un contrôleur d'équipements à un autre, en fonction de la locali-

sation du service dont l'exécution est requise. Les agents collaboratifs envoient des messages de requête pour demander l'exécution de services sur les équipements contrôlés par d'autres agents. Ainsi, MASML remplace la distribution d'un scénario et la coordination entre agents par la mobilité d'un agent unique pour exécuter un scénario. Cependant, si l'agent mobile est détruit, tout le scénario est détruit. La panne d'un agent mobile est traitée par la génération d'un autre agent mobile. La génération des agents mobiles est gérée de manière centralisée, par un agent fixe. La panne de cet agent (ou de l'équipement qui l'héberge) empêche toute reconfiguration du système (les agents mobiles n'ont pas la capacité de se reconfigurer eux-mêmes). La performance du mécanisme de mobilité ne semble intéressante que dans des contextes spécifiques où les durées d'exécution des services sont supérieures aux déplacements des agents. À notre connaissance, aucun des modèles d'agents proposés ne gère de manière autonomique la reconfiguration du système de contrôle. Un point de contrôle centralisé dédié est toujours utilisé pour gérer la configuration du système et la répartition des scénarios.

Nous remarquons que les systèmes distribués et fédérés sont plus fiables que les systèmes centralisés. Les systèmes fédérés restent néanmoins plus autonomes dans la gestion des pannes que les systèmes distribués qui nécessitent l'intervention d'un programmeur.

## 2.4 Analyse : Gestion du contexte

Dans cette section, nous analysons les systèmes étudiés en fonction de leur capacité à gérer les caractéristiques des environnements domotiques. Cette analyse est basée sur l'étude de la problématique présentée dans la section 1.3.2. Le tableau 2.1 représente le résultat de l'analyse.

| Système | Gestion des équipements | | Gestion des utilisateurs | | Gestion du système | |
|---|---|---|---|---|---|---|
| | var. | chang. | var. | chang. | var. | chang. |
| PHS[63] | × | ✓ | × | × | × | × |
| MADCAR[39] | × | ✓ | × | × | × | × |
| SLCA[47] | × | ✓ | × | × | × | × |
| ENTIMID[62] | × | ✓ | × | × | × | × |
| SODAPOP[32] | × | ✓ | × | × | × | × |
| HNS[65] | ✓ | ✓ | × | × | × | × |
| MASML[88] | × | ✓ | × | × | × | × |
| MASBM[75] | × | ✓ | × | × | × | × |
| FCSDS[80] | ✓ | ✓ | × | × | × | × |

Table 2.1 – Analyse des systèmes domotiques existants par rapport à la gestion du contexte

### 2.4.1 Gestion des équipements

**Variabilité.** Les systèmes étudiés contrôlent les équipements disponibles selon deux méthodes. Les systèmes ENTIMID, SLCA, MADCAR et MASBM, déploient pour chaque équipement détecté un composant de contrôle prédéfini. Les autres systèmes sont basés sur des requêtes de contrôle stockées dans leur annuaire.

Tous les systèmes, sauf HNS et FCSDS, sont limités aux composants et aux requêtes de contrôle qu'ils possèdent dans leurs configurations initiales. Ces systèmes ne sont pas en mesure de contrôler des équipements pour lesquels ils ne possèdent pas un composant ou une requête de contrôle.

**Changement.** Seuls les systèmes HNS et FCSDS sont capables de mettre à jour leur annuaire lors de la détection d'un nouvel équipement.
Tous les systèmes étudiés, sauf PHS qui ne gère pas les scénarios, possèdent des descriptions de scénarios prédéfinies. Le déploiement d'un scénario dans les systèmes ENTIMID, SLCA, MADCAR, HNS, MASML et MASBM, dépend statiquement des équipements disponibles dans l'environnement. Ainsi, lorsqu'un équipement tombe en panne, le scénario est supprimé. FCSDS et SODAPOP, au contraire, sélectionnent dynamiquement les services exécutés dans chaque scénario.

### 2.4.2 Gestion des utilisateurs

**Variabilité.** Aucun des systèmes étudiés ne gère les droits d'accès des utilisateurs aux équipements. Seul MASBM gère l'identification des utilisateurs de manière à interdire l'accès au système à des utilisateurs non-autorisés.
L'interaction avec les utilisateurs finaux est généralement très réduite. Dans certains systèmes tels ENTIMID, SLCA, MADCAR et HNS, les utilisateurs n'ont aucun moyen d'interagir avec le système. Dans les autres systèmes, les utilisateurs ne peuvent que déclencher des services, comme c'est le cas pour MASBM, MASML et PHS ou des scénarios prédéfinis comme dans FCSDS et SODAPOP. Certains systèmes permettent à des utilisateurs experts de définir des scénarios (HNS et SLCA). Par contre, les utilisateurs finaux n'ont pas la possibilité de définir des scénarios.
Seuls MASML et HNS gèrent les conflits d'exécution des scénarios. MASML gère les conflits qui provoquent des problèmes d'interaction entre entités (agents mobiles) [85] par la modification du comportement des agents mobiles en conflit. Lorsque le conflit ne peut pas être résolu, les utilisateurs sont informés de la situation. HNS propose dans [64] une méthode qui permet de calculer statiquement tous les conflits possibles dans un environnement à l'aide de l'étude des interactions sur les équipements et sur l'environnement.

**Changement.** Dans les systèmes FCSDS et SODAPOP, l'expression du besoin des utilisateurs est limitée par la liste des règles proposées aux utilisateurs pour composer les scénarios. L'ajout de nouvelles règles n'est accessible qu'à des utilisateurs *experts* et non pas aux utilisateurs finaux qui n'ont pas même la possibilité de modifier les paramètres des scénarios prédéfinis.

### 2.4.3 Gestion du système

**Variabilité.** Aucun des systèmes précédents ne permet aux utilisateurs de paramétrer son architecture.

**Changement.** Aucun des systèmes précédents, sauf MASML, ne gère les pannes liées aux éléments du système. MASML gère les pannes des agents mobiles par génération d'un autre agent mobile.

## 2.5 Fonctionnalités des systèmes domotiques

L'analyse réalisée dans la section 2.4 souligne que la gestion du contexte par la plupart des systèmes proposés est très limitée. Dans ce qui suit, nous présentons les fonctionnalités que doit offrir un système domotique pour pallier aux inconvénients perçus dans les systèmes étudiés.

Nous définissons deux types de fonctionnalités : les fonctionnalités explicites qui permettent aux utilisateurs de gérer leurs propres besoins et les fonctionnalités implicites qui permettent aux systèmes de réaliser les besoins des utilisateurs en dépit des changements liés à l'environnement et au système.

### 2.5.1 Fonctionnalités explicites des systèmes domotiques

Nous définissons quatre fonctionnalités explicites que doivent offrir les systèmes domotiques aux utilisateurs. Les fonctionnalités explicites sont perçues par les utilisateurs. Elles sont fournies au travers d'interfaces graphiques. Elles permettent aux utilisateurs de paramétrer le comportement du système en fonction de leurs besoins. Elles permettent de répondre à la variabilité et aux changements liés à la gestion des utilisateurs et au paramétrage du système. Les fonctionnalités explicites sont décrites dans les sections suivantes en fonction des rôles des utilisateurs. Ces rôles sont décrits dans la section 1.3.1.

#### 2.5.1.1 Gestion des services

Les utilisateurs finaux doivent pouvoir invoquer des services fournis par les équipements disponibles dans l'environnement en fonction de leurs droits d'accès. Le système se charge de l'exécution des services.

#### 2.5.1.2 Gestion des scénarios

Les utilisateurs finaux doivent pouvoir gérer (définir, modifier ou supprimer) des scénarios personnalisés en fonction de leurs droits d'accès et des équipements disponibles dans l'environnement. Le système se charge de l'automatisation de l'exécution des scénarios et de la suspension des scénarios supprimés.

#### 2.5.1.3 Gestion des profils des utilisateurs

Le *profil* d'un utilisateur contient ses identifiants, son rôle, ses droits d'accès et la liste des scénarios qu'il a défini. Les différents rôles des utilisateurs sont définis dans la section 1.3.1. Un profil permet au système d'identifier l'utilisateur et de personnaliser l'interaction avec l'utilisateur en fonction de son rôle et de ses droits d'accès aux services fournis par les équipements. Un utilisateur ne pourra contrôler les équipements et ne pourra définir des scénarios qu'en fonction

de ses droits d'accès.
Les utilisateurs privilégiés doivent être capables de gérer (définir, modifier ou supprimer) les profils des utilisateurs.
La définition du profil d'un utilisateur consiste à attribuer à un utilisateur un rôle et des droits d'accès aux équipements disponibles dans l'environnement.
La modification du profil d'un utilisateur consiste à ajouter ou supprimer des droits d'accès. Dans le cas de la suppression de droits d'accès, le système doit vérifier la cohérence des scénarios, en cours d'exécution définis par l'utilisateur, en fonction de ses nouveaux droits d'accès.
Lors de la suppression du profil d'un utilisateur, cet utilisateur n'a plus accès au système et les scénarios qu'il a définis sont suspendus.

#### 2.5.1.4 Gestion du paramétrage du système

Les administrateurs du système doivent pouvoir paramétrer l'architecture du système pour des raisons qui peuvent être liées à la performance ou à la fiabilité du système. La modification de l'architecture du système ne doit pas perturber ni interrompre les scénarios en cours d'exécution.

#### 2.5.1.5 Fonctionnalités explicites dans les systèmes domotiques étudiés

Le tableau 2.2 représente une analyse des systèmes domotiques par rapport aux fonctionnalités explicites offertes aux utilisateurs.

| Système | Gestion des services | Gestion des scénarios personnalisés | Gestion des profils utilisateurs | Gestion du paramétrage du système |
|---|---|---|---|---|
| PHS[63] | exéc. service | × | × | × |
| MADCAR[39] | × | × | × | × |
| SLCA[47] | × | × | × | × |
| ENTIMID[62] | × | × | × | × |
| SODAPOP[32] | exéc. scénario | × | × | × |
| HNS[65] | × | × | × | × |
| MASML[88] | exéc. service | × | × | × |
| MASBM[75] | exéc. service | × | × | × |
| FCSDS[80] | exéc. scénario | × | × | × |

Table 2.2 – Analyse des systèmes domotiques existants par rapport aux fonctionnalités explicites

Cette analyse nous confirme que l'interaction avec les utilisateurs dans les systèmes domotiques étudiés est très limitée. Aucun système ne permet aux utilisateurs de gérer les scénarios, les profils des utilisateurs ou le paramétrage du système. Les utilisateurs sont limités à l'exécution de services comme dans les systèmes PHS [63], MASML [88] et MASBM [75] ou à l'exécution de scénarios prédéfinis par les développeurs comme dans les systèmes SODAPOP [32] et FCSDS [80].

### 2.5.2 Fonctionnalités implicites des systèmes domotiques

En plus des fonctionnalités explicites, nous définissons deux fonctionnalités implicites que doit implémenter un système domotique. Ces fonctionnalités sont transparentes à l'utilisateur. Elles permettent au système de maintenir au mieux sa cohérence, sa performance et sa fiabilité [31]. Elles permettent au système de gérer automatiquement son adaptation à la variabilité et aux changements liés aux équipements ou à sa propre architecture. Ces fonctionnalités permettent au système d'être tolérant aux pannes des équipements et de ses propres éléments. Dans la mesure du possible, les scénarios en cours d'exécution ne doivent pas être interrompus ni perturbés et les utilisateurs doivent être capables à tout moment d'interagir avec le système.

#### 2.5.2.1 Gestion de l'environnement

Le système doit gérer l'ajout et la suppression/panne des équipements.
Lors de la suppression/panne d'un équipement, les utilisateurs ne doivent plus être en mesure d'y accéder. Les services fournis par l'équipement doivent être supprimés des interfaces graphiques.
L'exécution d'un scénario implique le contrôle d'un ensemble d'équipements. La panne de l'un de ces équipements peut empêcher l'exécution du scénario. Le système doit être capable de gérer les scénarios en cours d'exécution de manière à préserver au mieux leur exécution. Les scénarios qui ne peuvent plus être exécutés sont suspendus.
Lors de l'ajout d'un équipement, le système doit être en mesure de contrôler cet équipement et doit vérifier si des scénarios suspendus à cause de la panne d'un équipement peuvent être réactivés.
Par ailleurs, un système domotique doit gérer les conflits d'exécution des services.

#### 2.5.2.2 Gestion du système

Ainsi, lors de l'ajout ou de la suppression/panne d'un élément du système, le système doit être capable de reconfigurer son architecture. Cela signifie être capable d'adapter dynamiquement la répartition du contrôle des équipements et de l'exécution des scénarios pour exploiter au mieux les ressources du système.
La suppression/panne d'un élément ne doit pas perturber ou interrompre le fonctionnement du système, dans la mesure du possible. Les scénarios qui ne peuvent plus être exécutés sont suspendus.
Lors de l'ajout d'un élément au système, le système doit vérifier si des scénarios suspendus à cause de la panne d'un élément du système peuvent être réactivés.

#### 2.5.2.3 Fonctionnalités implicites dans les systèmes domotiques étudiés

Le tableau 2.3 présente une analyse des systèmes domotiques par rapport aux fonctionnalités implicites.

| Système | Gestion de l'environnement | Gestion du système |
|---|---|---|
| PHS[63] | ✓ | × |
| MADCAR[39] | ✓ | × |
| SLCA[47] | ✓ | × |
| ENTIMID[62] | ✓ | × |
| SODAPOP[32] | × | × |
| HNS[65] | conflit | × |
| MASML[88] | ✓ | ✓ |
| MASBM[75] | ✓ | × |
| FCSDS[80] | ✓ | × |

Table 2.3 – Analyse des systèmes domotiques existants par rapport aux fonctionnalités implicites

Les systèmes centralisés et fédérés gèrent la panne des équipements par la désinstallation des composants qui implémentent les scénarios. Aucun des systèmes étudiés, à l'exception de MASML [88], ne gère les pannes liées au système. Enfin, aucun des systèmes étudiés, à l'exception de MASML [88] et HNS [65], ne gère les conflits d'exécution de services.

## 2.6 Qualités requises des systèmes domotiques

Dans la section 2.5, nous avons proposé deux types de fonctionnalités, explicites et implicites, qui permettent à un système domotique de gérer son contexte.

Dans cette section, nous présentons quatre qualités indispensables que doivent posséder les systèmes pour réaliser ces fonctionnalités.

### 2.6.1 Configurabilité

Les systèmes domotiques doivent offrir les fonctionnalités explicites aux utilisateurs au travers d'interfaces graphiques. Les interfaces et leurs contenus doivent être adaptés en fonction du profil des utilisateurs, de l'environnement et du système à un moment donné. Les utilisateurs doivent être capables de configurer l'architecture du système (gestion du paramétrage du système) et de personnaliser son comportement envers les utilisateurs (gestion des profils des utilisateurs) et envers les équipements (gestion des services et des scénarios).

### 2.6.2 Adaptation dynamique

Le système doit permettre d'adapter dynamiquement son architecture pour le contrôle des équipements, l'implémentation des scénarios et l'application du paramétrage. L'adaptation dynamique [30] consiste à effectuer des modifications dans l'architecture du système pendant son fonctionnement sans perturber son fonctionnement.

### 2.6.3 Adaptation autonomique

Le système doit être capable de s'adapter automatiquement aux équipements disponibles et aux scénarios définis par les utilisateurs. Il doit aussi gérer automatiquement les fonctionnalités implicites (adaptation aux changements qui se produisent dans l'environnement et le système). L'adaptation autonomique [68] consiste à intégrer au système des mécanismes qui lui permettent d'adapter automatiquement son architecture sans intervention humaine. Cela permet au système de gérer l'environnement et le système sans connaissance *a priori* les équipements disponibles et les besoins des utilisateurs.

### 2.6.4 Sensibilité au contexte

Afin de pouvoir posséder les qualités précédentes, un système domotique doit être capable d'identifier les utilisateurs, les équipements disponibles dans l'environnement et les différents éléments (composants, équipements, agents...) qui composent le système. Il doit être également capable de détecter les changements qui se produisent tant au niveau de l'environnement qu'au niveau du système. La sensibilité au contexte permet au système de détecter les changements et d'y réagir en adaptant son comportement.

### 2.6.5 Discussion

Le tableau 2.4 représente une analyse des systèmes domotiques existants en fonction des qualités requises présentées.

| Système | Configurabilité | Sensibilité au contexte | | Adaptation autonomique | | Adaptation dynamique | |
|---|---|---|---|---|---|---|---|
| | | envir. | syst. | envir. | syst. | envir. | syst. |
| PHS[63] | × | ✓ | × | ✓ | × | × | × |
| MADCAR[39] | × | ✓ | × | ✓ | × | ✓ | × |
| SLCA[47] | × | ✓ | × | ✓ | × | ✓ | × |
| ENTIMID[62] | × | ✓ | × | ✓ | × | ✓ | × |
| SODAPOP[32] | × | × | × | × | × | × | × |
| HNS[65] | × | × | × | × | × | × | × |
| MASML[88] | × | ✓ | ✓ | ✓ | × | ✓ | × |
| MASBM[75] | × | ✓ | × | ✓ | × | ✓ | × |
| FCSDS[80] | × | ✓ | × | ✓ | × | × | × |

Table 2.4 – Analyse des systèmes domotiques existants par rapport aux qualités requises

Cette analyse nous confirme que l'interaction avec les utilisateurs dans les systèmes domotiques étudiés est très limitée. Aucun système ne permet aux utilisateurs de gérer les scénarios, les profils des utilisateurs ou le paramétrage du système. Les utilisateurs sont limités à l'exécution de services comme dans les systèmes PHS, MASML et MASBM, ou à l'exécution de scénarios prédéfinis par les développeurs comme dans les systèmes SODAPOP et FCSDS.

Tous les systèmes centralisés et fédérés sont sensibles aux équipements. En fonction des équipe-

ments détectés, ils exécutent un ensemble de scénarios prédéfinis. Seul MASML, par contre, est sensible à la panne des agents.

Les systèmes distribués sont des systèmes statiques. Les adaptations, pour gérer des changements non-prévus liés aux équipements, doivent être réalisées par les développeurs. Dans les autres systèmes (MADCAR, SLCA et MASML), l'adaptation consiste à assembler dynamiquement des composants existants. Ainsi, l'adaptation autonomique est limitée aux composants disponibles. Aucun des systèmes étudiés, sauf MASML, ne s'adapte aux changements liés au système.

## 2.7 Conclusion

Les systèmes domotiques sont conçus dans le but d'automatiser le contrôle des équipements et l'implémentation de scénarios. La gestion de la variabilité et du changement dans les systèmes est limitée à cause de l'aspect prédéfini du contrôle des équipements et de l'implémentation des scénarios. Elle est basée sur des adaptations standardisées et prédéfinies. L'adaptation standardisée consiste à intégrer au système des mécanismes préétablis dédiés à des équipements et des scénarios standards. En fonction des équipements disponibles dans l'environnement, un ensemble de scénarios est déployé. Cependant, ce type d'adaptation ne peut pas faire face à des équipements non-standardisés ou à tous les scénarios envisageables. Dans ce cas, l'intervention d'un développeur est nécessaire.
Dans les systèmes étudiés, les utilisateurs doivent s'adapter aux systèmes domotiques. Ils sont contraints par les scénarios prédéfinis proposés par les systèmes. Dans certains cas, les utilisateurs n'ont même pas la possibilité de modifier les paramètres des scénarios. Les systèmes domotiques étudiés ne répondent pas à tous les besoins des utilisateurs.

Pour répondre aux besoins des utilisateurs, un système domotique doit offrir aux utilisateurs les fonctionnalités explicites (gestion des services, gestion des scénarios, gestion des profils des utilisateurs et gestion du paramétrage du système) et doit gérer automatiquement les fonctionnalités implicites (gestion de l'environnement et du système).
Pour réaliser les fonctionnalités précédentes, un système domotique doit donc répondre aux qualités : *configurabilité*, *adaptation autonomique*, *adaptation dynamique* et *sensibilité au contexte*.

Dans le reste de ce mémoire, nous présentons notre contribution dans le domaine des systèmes domotiques. Nous présentons dans le chapitre 3 l'architecture globale de notre système et la réalisation des fonctionnalités. Dans le chapitre 4, nous présentons l'architecture interne de notre système qui lui permet de réaliser l'adaptation dynamique et autonomique. Enfin, dans le chapitre 5, nous présentons l'implémentation de notre système.

Chapitre 3

# SAASHA : Architecture et fonctionnalités du SMA

**Sommaire**


**3.1 Introduction . . . . . 23**
**3.2 Combinaison des paradigmes *Agent* et *Composant* . . . . . 24**
**3.3 Architecture globale de SAASHA . . . . . 25**
3.3.1 Agents de SAASHA . . . . . 25
3.3.2 Agents d'interface . . . . . 26
3.3.3 Agents de contrôle . . . . . 28
3.3.4 Analyse de l'architecture de SAASHA . . . . . 29
**3.4 Mode de fonctionnement de SAASHA . . . . . 30**
**3.5 Phase de configuration . . . . . 31**
3.5.1 Initialisation . . . . . 31
3.5.2 Paramétrage du système . . . . . 32
**3.6 Phase opérationnelle . . . . . 34**
3.6.1 Gestion des profils des utilisateurs . . . . . 35
3.6.2 Gestion des services . . . . . 36
3.6.3 Gestion des scénarios . . . . . 37
3.6.4 Gestion de l'environnement . . . . . 40
3.6.5 Gestion du système . . . . . 42
**3.7 Conclusion . . . . . 45**
**3.8 Limites et perspectives . . . . . 46**


## 3.1 Introduction

Le chapitre 2 nous a permis de montrer les fonctionnalités attendues des systèmes domotiques et les qualités que doivent posséder ces systèmes pour être capables de satisfaire ces fonctionnalités. Il nous a aussi permis de nous rendre compte que les systèmes et solutions étudiés ne proposaient

pas d'implémentations satisfaisant à la fois les fonctionnalités et les qualités.

Pour répondre à cette problématique, ce chapitre et les deux suivants présentent la contribution de cette thèse ; *SAASHA* "a Self-Adaptable Agent System for Home Automation". SAASHA est un système multi-agents à base de composants dédié aux environnements domotiques qui offre les fonctionnalités explicites et implicites décrites dans la section 2.5 et qui répond aux qualités requises des systèmes domotiques décrites dans la section 2.6.

Dans ce chapitre, nous présentons plus précisément l'architecture globale de *SAASHA* et la gestion des fonctionnalités explicites et implicites. Nous justifions, dans un premier temps, la combinaison des deux paradigmes *Agent* et *Composant*. Nous décrivons, ensuite, les différents types et rôles des agents de notre système. Puis, nous spécifions la contribution de chaque type d'agents dans le fonctionnement du système pour la réalisation des fonctionnalités explicites et implicites.

Tout au long de ce chapitre, nous nous servirons du cas d'étude présenté dans la section 1.2 pour illustrer nos propos. Certains termes utilisés dans ce chapitre ont été définis dans le chapitre 1. Nous ferons référence à la définition lors de la première utilisation du terme.

## 3.2 Combinaison des paradigmes *Agent* et *Composant*

Dans cette section, nous présentons les atouts de la combinaison des paradigmes *Agent* et *Composant* pour la réalisation des fonctionnalités que doivent offrir les systèmes domotiques (section 2.5) et pour la satisfaction des qualités requises de ces systèmes (section 2.6).

Les *agents* [19, 86, 33, 87] sont des entités autonomes distribuées dans un environnement.
Les agents sont capables de percevoir l'environnement dans lequel ils sont intégrés et les autres agents du système. La perception correspond à la détection et l'identification des éléments présents dans l'environnement et des agents du système et à la détection des changements qui se produisent dans l'environnement et le système. La connaissance par les agents de leur environnement et du système est basée sur leur perception.
Les agents sont capables d'interagir entre eux et avec leur environnement et de réagir aux événements/changements qui se produisent dans cet environnement [33].
Les agents possèdent un comportement qui décrit leur manière de percevoir et d'interagir avec l'environnement et les autres agents du système [33]. Les comportements des agents peuvent être implantés sous la forme d'assemblages de composants [21].

Un *composant logiciel* [77, 26, 55, 22, 13] est une unité de programmation, autonome et découplée, qui peut être assemblée à d'autres composants au travers d'interfaces décrivant ses fonctionnalités et ses dépendances. Les assemblages de composants peuvent être dynamiquement modifiés par ajout, retrait ou substitution de composants ou par modification des connexions [11, 46, 28, 90, 70].

Nous avons choisi de modéliser notre système comme un système multi-agents à base de composants. Cette combinaison permet à notre système de répondre aux qualités requises de la manière suivante.

Les agents sont des entités autonomes sensibles au contexte. La perception et la connaissance des agents de l'environnement permettra à notre système de répondre à la qualité requise *sensibilité au contexte*.

L'interaction des agents avec les utilisateurs permet l'expression des besoins par les utilisateurs. Cela répond à la qualité requise *configurabilité*.
L'interaction des agents avec les équipements permet au système de les contrôler pour exécuter des services qu'ils fournissent (*gestion des services*).
L'interaction des agents entre eux permet aux agents de collaborer ensemble pour l'implémentation des scénarios (*gestion des scénarios*) et pour la gestion des changements liés à l'environnement et au système (*gestion de l'environnement et du système*).

Les composants sont utilisés pour gérer dynamiquement la définition et la modification du comportement des agents (sans avoir besoin de redémarrer et sans perturber le fonctionnement des autres agents). Cela permettra à notre système de répondre à la qualité requise d'*adaptation dynamique*.
Les agents peuvent gérer eux-mêmes leurs composants pour adapter automatiquement leur comportement (sans intervention humaine). Cela permet à notre système de répondre à la qualité requise *adaptation autonomique*.

La combinaison de ces deux paradigmes a été étudiée dans d'autres travaux, pour des objectifs variés [38, 54, 82, 20, 17, 16]. La componification [53, 54] consiste à introduire l'utilisation de composants dans la définition des agents. De nombreux travaux ont proposé une telle combinaison [40, 41, 74, 21, 82]. Ils proposent ainsi des agents adaptables dynamiquement. Cependant, ces agents ne sont pas capables de s'adapter d'une manière autonome à des contextes imprévus.

Dans le reste de ce chapitre, nous présentons l'architecture globale de SAASHA et son mode de fonctionnement.

## 3.3 Architecture globale de SAASHA

Dans cette section, nous présentons l'architecture globale de *SAASHA* qui lui permet de répondre à certaines des qualités requises établies dans la section 2.6 et les différentes raisons qui nous ont poussés à choisir cette architecture.
Dans la section 3.3.1, nous présentons les types et rôles des agents de *SAASHA*. Dans les sections 3.3.2 et 3.3.3, nous décrivons les comportements de chaque type et rôle. Enfin, nous présentons, dans la section 3.3.4, une analyse de cette architecture.

### 3.3.1 Agents de SAASHA

Avant de présenter les types et les rôles des agents de *SAASHA*, nous présentons les types d'interactions et d'adaptations du système.

Nous définissons trois types d'interactions qui représentent le modèle d'interaction des agents

de *SAASHA*.

- *L'interaction visuelle.* Les agents interagissent avec les utilisateurs par le biais d'interfaces graphiques qui permettent aux utilisateurs de définir leurs besoins (section 1.2.2).
- *L'interaction de contrôle et de coordination.* Les agents interagissent avec les équipements de l'environnement pour exécuter les services qu'ils fournissent et recevoir les événements qu'ils émettent (section 1.2.1). Ils interagissent entre eux pour coordonner les services fournis par les équipements lors de l'exécution de scénarios distribués.
- *L'interaction de coopération.* Les agents interagissent entre eux pour la gestion des fonctionnalités de configuration du système, de déploiement des scénarios et d'adaptation dynamique aux changements de contexte.

En fonction des interactions, notre système s'adapte sur les trois niveaux suivants :

- *Adaptation de la présentation.* Le système adapte ses interfaces et leur contenu en fonction du profil de l'utilisateur (section 2.5.1.3) et des informations sur le contexte (environnement et sa propre architecture).
- *Adaptation du comportement.* Le système adapte son comportement à deux niveaux :
  - Contrôle : Le système adapte son comportement en fonction des équipements qu'il doit contrôler.
  - Coordination : Le système adapte son comportement en fonction des scénarios qu'il doit coordonner.
- *Adaptation de l'organisation.* Le système adapte son architecture (types et rôles) en fonction des besoins des utilisateurs (fonctionnalités explicites) et des changements qui se produisent dans l'environnement et le système (fonctionnalités implicites).

Les adaptations précédentes et les mécanismes sur lesquels elles sont basées sont décrits en détail dans le chapitre 4.

Les agents sont caractérisés par leurs interactions [33]. La classification des interactions nous a conduits à la définition de deux types d'agents : les *agents d'interface* et les *agents de contrôle*. Les agents de contrôle peuvent jouer trois rôles : *médiateur* (MedCA), *manageur* (MgrCA) et *contrôleur d'équipements* (DCA). La figure 3.1 représente les différents types et rôles des agents de notre système en fonction des adaptations et des interactions qu'ils gèrent.
Dans les sections suivantes, nous présentons les types et les rôles des agents.

### 3.3.2 Agents d'interface

Les agents d'interface appelés GUIA pour *Graphical User Interface Agent* sont responsables de l'interaction visuelle avec les utilisateurs et de l'adaptation de la présentation. Ils sont installés dans des équipements qui possèdent des périphériques de saisie comme des ordinateurs ou des

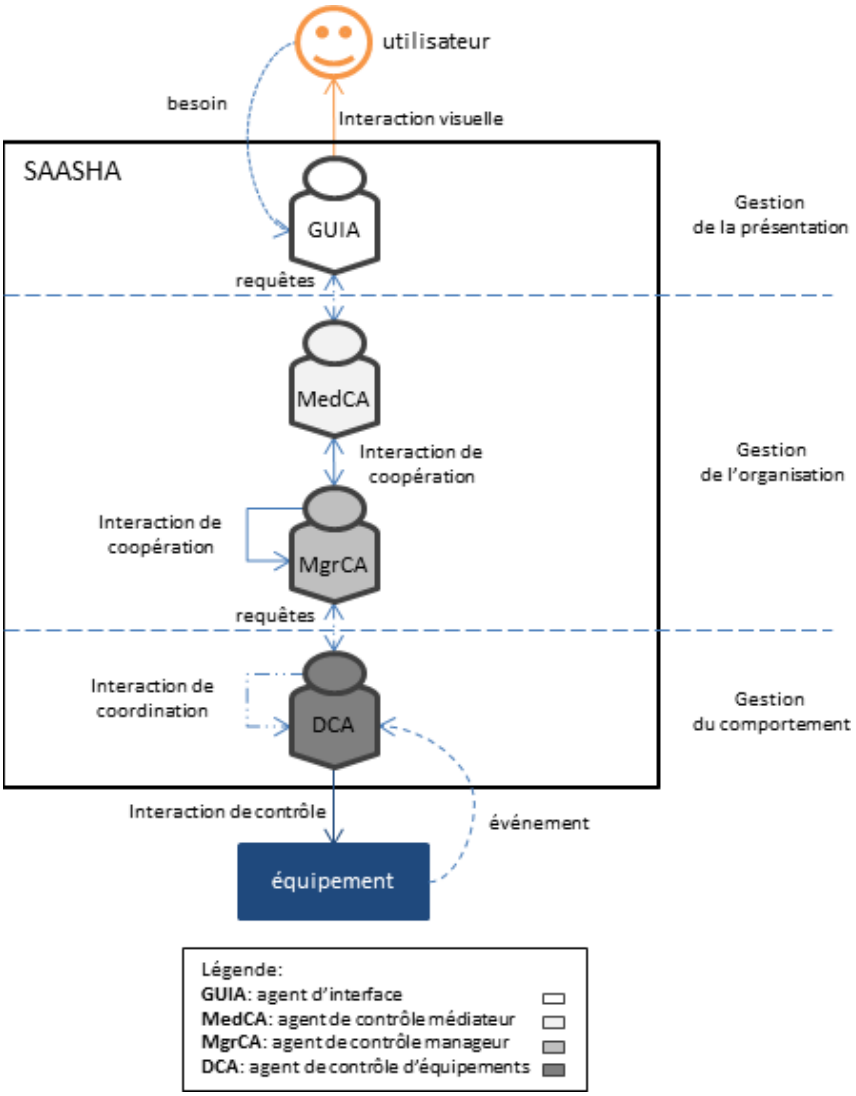



Figure 3.1 – Les différentes interactions de *SAASHA*

PDA. Ils interagissent avec les utilisateurs (section 1.3.1) d'une part et les MedCA d'autre part.

Les GUIA permettent aux utilisateurs de gérer leurs besoins par le biais d'interfaces graphiques. Les GUIA offrent une interface graphique pour chaque fonctionnalité explicite. La gestion de la présentation permet donc aux utilisateurs de se connecter au système à partir de n'importe quel GUIA et de disposer d'interfaces adaptées à leur profil. Plusieurs utilisateurs peuvent se connecter au système au même moment par le biais de différents GUIA. Plusieurs utilisateurs peuvent se connecter au système à des moments différents par le biais d'un même GUIA.

L'interaction des GUIA avec les MedCA consiste à leur transmettre les besoins des utilisateurs sous la forme de requêtes.

### 3.3.3 Agents de contrôle

Les agents de contrôle sont responsables de la réalisation des fonctionnalités explicites et implicites. Ils sont installés dans des boîtiers connectés aux réseaux de l'environnement. Ils sont responsables des interactions de contrôle, de coordination et de coopération. Les agents de contrôle peuvent jouer trois rôles, éventuellement simultanément : agent de contrôle d'équipements (DCA), agent de contrôle manageur (MgrCA) et agent de contrôle médiateur (MedCA).

#### 3.3.3.1 Agents de contrôle d'équipements

Les agents de contrôle d'équipements appelés DCA pour *Device Control Agent* sont responsables du contrôle des équipements et de la coordination de l'exécution des scénarios. Les DCA interagissent avec les équipements d'une part et avec les MgrCA d'autre part. Chaque DCA possède des comportements de contrôle et de coordination gérés sous la forme d'un ensemble de composants adapté à son contexte d'exécution.

Les comportements de contrôle et de coordination de chaque DCA sont caractérisés par deux paramètres : le *champ de contrôle* et *la charge des scénarios*. Ces paramètres sont utilisés lors de la gestion des scénarios (section 3.6.3).

Le champ de contrôle correspond à l'ensemble des équipements contrôlés par l'agent. Les administrateurs peuvent paramétrer le champ de contrôle des DCA pour réduire le contrôle à un sous-ensemble d'équipements. Les DCA adaptent leur comportement de contrôle en fonction de leur champ de contrôle.

La charge des scénarios d'un agent de contrôle représente le nombre de scénarios à l'implémentation desquels l'agent participe. Les administrateurs peuvent spécifier une charge maximale pour chaque DCA au-dessus de laquelle le DCA ne peut plus implémenter de scénarios supplémentaires. Si le DCA atteint cette limite, on dit qu'il est surchargé. Les DCA adaptent leur comportement de coordination pour implémenter les scénarios. Ils collaborent ensemble en fonction de ce comportement.

Les DCA gèrent les conflits d'exécution des scénarios et les changements qui se produisent dans l'environnement. Ils informent les MgrCA des changements liés à l'implémentation des scénarios : en cours d'implémentation ou suspendu. L'implémentation d'un scénario peut être suspendue à cause de la panne d'un équipement ou d'un agent et réactivée grâce à l'apparition d'un équipement ou d'un agent.

#### 3.3.3.2 Agents de contrôle manageurs

Les agents de contrôle manageurs appelés MgrCA pour *Manager Control Agent* sont responsables de la coopération entre eux et de l'adaptation de l'organisation pour la gestion des fonctionnalités implicites (gestion de l'environnement global (ce qui hors de portée du contrôle local géré par les DCA) et du système). Les MgrCA interagissent entre-eux, avec les DCA d'une part et les

MedCA d'autre part.

L'adaptation de l'organisation est caractérisée par le paramètre *champ de gestion*. Le champ de gestion correspond à la liste des DCA qu'un agent de contrôle gère. Un MgrCA peut gérer tous les DCA du système et plusieurs MgrCA peuvent gérer le même DCA. Les administrateurs peuvent paramétrer le champ de gestion des MgrCA pour réduire la gestion à un sous-ensemble de DCA. Les DCA ne communiquent qu'avec leurs MgrCA. Le champ de contrôle d'un MgrCA correspond à l'union des champs de contrôle des DCA qu'il gère. En cas de nécessité, un MgrCA peut lui même remplacer un des DCA qu'il gère si ce DCA tombe en panne ou est surchargé.

Les MgrCA jouent un rôle essentiel dans l'adaptation de l'organisation du système. Ils gèrent la connaissance du système sur son environnement global (utilisateurs et équipements) et sur lui-même (agents).

#### 3.3.3.3 Agents de contrôle médiateurs

Les agents de contrôle médiateurs appelés MedCA pour *Mediator Control Agent* sont responsables de la coopération avec les MgrCA et de l'adaptation de l'organisation pour la gestion des fonctionnalités explicites. Les MedCA interagissent avec les GUIA d'une part et les MgrCA d'autre part.

Un GUIA interagit avec le système par le biais d'un seul agent de contrôle. C'est l'un des MgrCA qui gère cette interaction car les MgrCA possèdent une connaissance globale du système. Le GUIA se base sur cette connaissance pour l'adaptation de la présentation. Les MedCA sont responsables de la transmission de la connaissance aux GUIA. Nous appelons cette opération *la synchronisation des données*. Elle est décrite dans la section 4.3.3.2. Les GUIA transmettent les besoins définis par les utilisateurs à leur MedCA. Un MedCA est responsable ainsi de la gestion des fonctionnalités explicites.

L'adaptation de l'organisation est caractérisée par le paramètre *charge de médiation*. Cette charge correspond au nombre de GUIA géré par un agent de contrôle. C'est le GUIA qui sélectionne son médiateur en fonction de la charge. Le GUIA sélectionne le MgrCA qui a la charge la plus faible. Le MgrCA sélectionné joue alors en plus le rôle de MedCA. Les MedCA peuvent être connectés à plusieurs GUIA au même moment.

### 3.3.4 Analyse de l'architecture de SAASHA

L'architecture de SAASHA est basée sur deux types d'agents. Nous avons défini ces deux types d'agents pour les raisons suivantes.

L'interaction avec les utilisateurs est une interaction qui est réalisée au travers d'équipements avec des périphériques de saisie qui fournissent des interfaces graphiques. Les agents d'interface sont installés dans ces équipements. Ils transmettent les requêtes des utilisateurs au deuxième type d'agents. Ainsi, les agents d'interfaces ne sont pas directement utilisés dans le fonctionnement du système mais uniquement dans son paramétrage par les utilisateurs. Les équipements

dans lesquels les agents d'interfaces sont embarqués peuvent ainsi être éteints, redémarrés ou déplacés au gré des utilisateurs sans perturber le comportement du système.

Au contraire, les agents de contrôle sont responsables des interactions de coopération, de contrôle et de coordination ne peuvent pas être déconnectés de l'environnement ou du reste du système sans impliquer une adaptation du comportement ou de l'architecture du système.
Les agents de contrôle correspondent donc au noyau du système, et en l'absence des agents d'interface, le système reste opérationnel. La connaissance par le système de son environnement et du système lui-même est gérée par les agents de contrôle.

Les agents de contrôle s'organisent selon des rôles et non pas des types. Ils possèdent quatre paramètres qui permettent de définir leur organisation. Tous les agents de contrôle possèdent la capacité de jouer les trois rôles. Ainsi, si un agent de contrôle tombe en panne ou est surchargé, un autre agent de contrôle peut le remplacer. Les rôles permettent aux agents de contrôle de mieux distribuer les tâches entre eux.

## 3.4 Mode de fonctionnement de SAASHA

Dans cette section, nous présentons le mode de fonctionnement de *SAASHA*. La figure 3.2 représente le fonctionnement global du système.

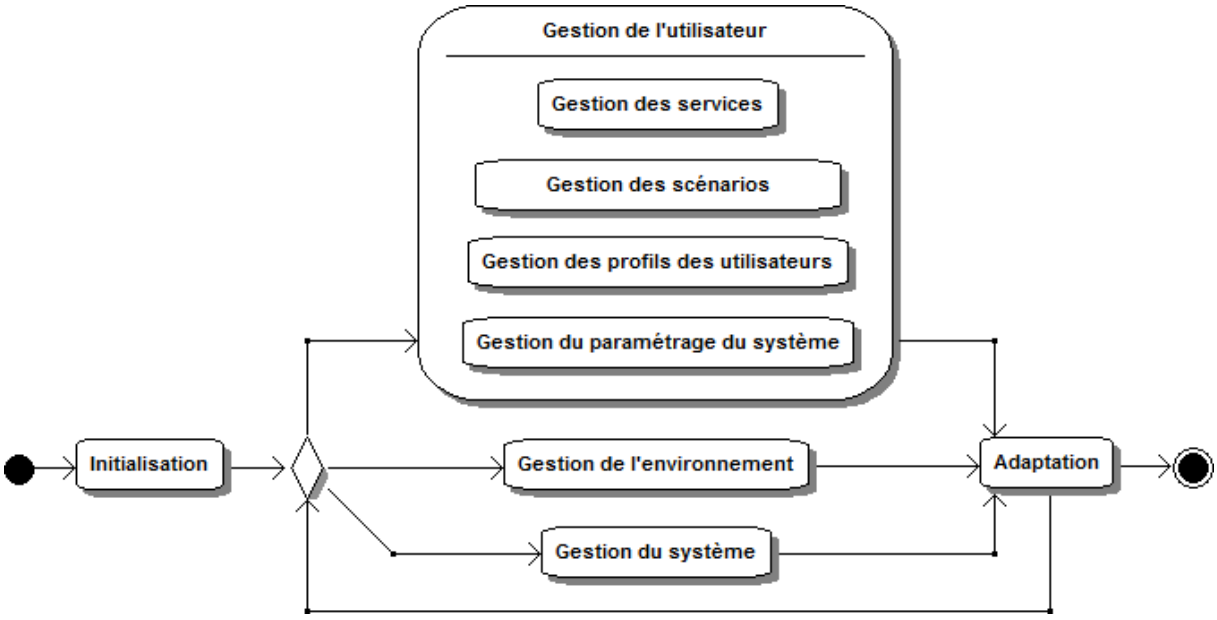



Figure 3.2 – Le fonctionnement global du système

Le fonctionnement global de notre système qui est découpé en deux phases : la phase de configuration et la phase opérationnelle. La phase de configuration permet au système de s'initialiser et de se paramétrer en fonction de l'environnement. La phase opérationnelle du système permet aux utilisateurs de gérer leurs besoins grâce aux fonctionnalités explicites offertes par le système. Le système gère automatiquement les fonctionnalités implicites. Les différents rôles des utilisateurs

qui interagissent avec notre système sont définis dans la section 1.3.1.

Lors de l'intégration des agents dans l'environnement, la phase de configuration est nécessaire pour qu'il soit opérationnel. Elle permet au système de s'adapter aux équipements disponibles dans l'environnement. Elle est composée de deux étapes principales.
La première étape est *l'initialisation*, pendant laquelle les agents se découvrent mutuellement et découvrent leur environnement.
La deuxième étape est *la gestion du paramétrage du système*, pendant laquelle les administrateurs peuvent paramétrer les agents de contrôle en fonction de la composition du système lui-même et celle de l'environnement. L'étape de paramétrage peut être aussi réalisée automatiquement par les agents de contrôle.
Une fois cette phase réalisée, le système est opérationnel.

Dans la phase opérationnelle, les utilisateurs peuvent définir leurs besoins en fonction de leur profil grâce aux fonctionnalités explicites proposées par le biais des interfaces graphiques. Les fonctionnalités explicites sont : la gestion des services, la gestion des scénarios, la gestion des utilisateurs et la gestion du paramétrage du système (les administrateurs peuvent paramétrer à nouveau le système). Le système gère automatiquement les fonctionnalités implicites (gestion de l'environnement et du système) déclenchées par les changements qui peuvent se produire dans l'environnement et dans le système.

Pour l'instant, notre système est destiné à des environnements domotiques composés d'un petit nombre d'équipements comme l'environnement décrit dans la section 1.2. Le nombre d'agents de contrôle qui constituent le système est aussi réduit. L'ordre de grandeur est la dizaine d'agents de contrôle.

## 3.5 Phase de configuration

Pour que le système soit opérationnel, il doit être initialisé et doit ensuite subir une étape de paramétrage. L'étape d'initialisation permet aux agents de contrôle de prendre leurs repères grâce à la perception de l'environnement. L'étape de paramétrage permet de spécifier le rôle des agents de contrôle, les champs de gestion des MgrCA et les champs de contrôle des DCA.

### 3.5.1 Initialisation

Dans un premier temps, lorsque les agents sont déployés (dans des équipements pour les GUIA et dans les boîtiers pour les agents de contrôle), ils initialisent leur architecture interne qui leur permet d'exécuter leur comportement initial. Plus de détails sur l'initialisation de l'architecture interne sont fournis dans le chapitre 4.

Lors de leur intégration dans l'environnement, les agents de contrôle perçoivent les agents et les équipements disponibles grâce aux mécanismes d'identification des agents et des équipements (section 5.2.2). Les GUIA perçoivent uniquement les agents.

Après cette étape, le système est initialisé. Les agents de contrôle ont identifié leur environnement. Tous les agents du système possèdent les profils des administrateurs et des utilisateurs privilégiés. Cependant, pendant la phase de configuration, seuls les administrateurs peuvent se connecter au système par le biais d'un GUIA pour lancer le paramétrage des agents de contrôle.

### 3.5.2 Paramétrage du système

L'étape de paramétrage consiste à spécifier le rôle de chaque agent de contrôle (MgrCA ou DCA) et à paramétrer ensuite le champ de gestion des MgrCA et le champ de contrôle des DCA. Cette étape peut être réalisée soit automatiquement par les agents de contrôle eux-mêmes, soit manuellement par les administrateurs.

#### 3.5.2.1 Paramétrage automatique

C'est l'administrateur qui lance le paramétrage automatique par le biais de son GUIA. Ce paramétrage minimal du système effectué par les agents de contrôle consiste à choisir aléatoirement parmi les agents de contrôle deux agents qui auront le rôle MgrCA. Le choix est aléatoire parce que les agents ont tous initialement les mêmes capacités et la même connaissance de l'environnement et du système. Donc, il n'y a aucun paramètre qui peut les différencier. Nous avons décidé que notre système doit être composé d'au moins deux MgrCA pour que le système puisse être tolérant aux pannes des MgrCA. Si un MgrCA tombe en panne un autre MgrCA doit être présent pour ne pas perdre la connaissance du système sur son environnement ou lui-même. La gestion des pannes des MgrCA fait partie des fonctionnalités implicites décrites dans la section 3.6.5.3. Les autres agents de contrôle ont alors le rôle de DCA.

Par défaut, le champ de gestion des MgrCA et le champ de contrôle des DCA ne sont pas restreints. Les deux MgrCA gèrent tous les DCA du système et les DCA ont le contrôle de tous les équipements disponibles dans l'environnement.

Une fois le paramétrage automatique terminé, le GUIA se connecte par le biais d'un MgrCA qui obtient le rôle MedCA. Le MedCA synchronise les données du GUIA. À ce moment, le paramétrage automatique est terminé.

Dans la section 3.8, nous proposons des perspectives pour améliorer ce paramétrage afin que la distribution des rôles et le paramétrage des champs de gestion et de contrôle soient reliés à la composition de l'environnement et du système.

#### 3.5.2.2 Paramétrage manuel

Cette étape est réalisée par l'administrateur à l'aide d'une interface composée de trois onglets : un onglet pour la spécification des rôles des agents de contrôle, un onglet pour le paramétrage du champ de gestion des MgrCA et un onglet pour le paramétrage du champ de contrôle des DCA. Le GUIA adapte le contenu des deux premiers onglets en fonction des agents de contrôle

qu'il a détecté.

Dans un premier temps, l'administrateur spécifie le rôle de chaque agent de contrôle : soit MgrCA ou soit DCA. Comme nous l'avons dit dans la section 3.5.2.1, il doit y avoir au moins deux MgrCA dans le système. Le GUIA envoie à chaque agent de contrôle son nouveau rôle. Ensuite, l'administrateur paramètre les champs de gestion des MgrCA puis les champs de contrôle des DCA.

Le paramétrage optionnel du champ de gestion des MgrCA consiste à attribuer à chaque MgrCA la gestion de tout ou partie des DCA. Dans l'onglet de paramétrage du champ de gestion des MgrCA, l'administrateur sélectionne pour chaque MgrCA la liste des DCA qu'il gère. Si l'administrateur ne spécifie aucun manageur pour un DCA, ce DCA est géré par tous les MgrCA. De même, si l'administrateur n'attribue aucun DCA à un MgrCA, ce MgrCA gère tous les DCA du système. Chaque MgrCA informe les DCA qui lui ont été attribués qu'il est leur manageur. En retour, chaque DCA lui envoie la valeur actuelle de son champ de contrôle. À cette étape, les DCA ne sont pas encore paramétrés.

Le GUIA se connecte ensuite par le biais d'un MgrCA qui obtient le rôle MedCA. Le MedCA synchronise les données du GUIA. À cette étape, la synchronisation consiste à envoyer au GUIA le valeur de paramétrage des DCA et la liste de tous les équipements détectés dans l'environnement. Le GUIA adapte alors le contenu de l'onglet du paramétrage du champ de contrôle des DCA présenté dans la figure 3.3.

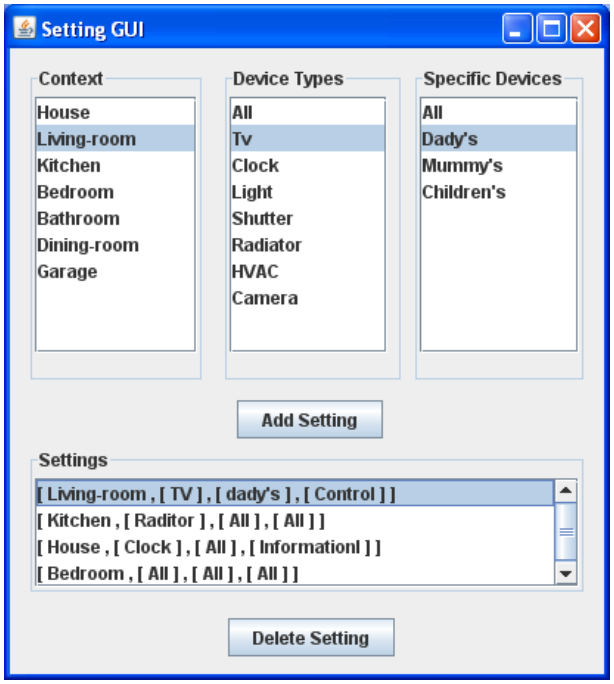



Figure 3.3 – L'onglet de paramétrage du champ de contrôle des DCA

Le paramétrage optionnel du champ de contrôle consiste à spécifier les équipements dont chaque DCA a le contrôle. Lors de l'intégration dans un environnement, les DCA ont, par défaut, le contrôle de tous les équipements de l'environnement. Les administrateurs peuvent définir des règles de paramétrage du champ de contrôle des DCA. Elles ont le format suivant :

⟦ contexte , type d'équipement , ⟦ équipement(s) spécifique(s) ⟧ ⟧ .

Ces règles permettent aux administrateurs de désigner le mode de sélection des équipements :

- Tous les équipements situés dans un contexte.
- Tous les équipements d'un certain type situés dans un contexte.
- Des équipements spécifiques d'un type d'équipement spécifique situés dans un contexte.

Pour chacune des situations précédentes, nous proposons un exemple :

- ⟦ Chambre , tous , ⟦ tous ⟧ ⟧. Cette règle signifie que le DCA a le contrôle de tous les équipements de la chambre. L'appartement est un contexte qui contient les équipements de tous les contextes qui le constituent. Ainsi, chaque équipement peut appartenir à plusieurs contextes, comme par exemple la télévision du salon qui appartient au salon et à l'appartement en même temps.
- ⟦ Cuisine , Radiateur , ⟦ tous ⟧ ⟧. Cette règle signifie que le DCA a le contrôle de tous les équipements de type "Radiateur" situés dans la "Cuisine".
- ⟦ Salon , Télévision , ⟦ TVpapa ⟧ ⟧. Cette règle signifie que le DCA a le contrôle de l'équipement de type "Télévision" appelé "TVpapa" et situé dans le "Salon".

Le GUIA transmet les règles de paramétrage par le biais du MedCA qui les transmet aux MgrCA qui gèrent le DCA paramétré. Les MgrCA enregistrent les règles de paramétrage puis ils les transmettent au DCA. Les règles de paramétrage spécifient les équipements contrôlés par le DCA.

Une fois le paramétrage terminé, les DCA adaptent leur comportement de contrôle pour contrôler les équipements en fonction de leur nouveau paramétrage. Les MgrCA n'adaptent leur comportement de contrôle que si l'un des DCA qu'ils gèrent tombe en panne ou est surchargé.

Un équipement peut être contrôlé par plusieurs DCA. Des DCA peuvent avoir la même valeur de champ de contrôle. Si aucun paramétrage n'est effectué, le DCA est paramétré par défaut. Il contrôle alors tous les équipements qu'il a détectés.

À cette étape, la phase de configuration du système est terminée et le système est opérationnel. Les agents de contrôle sont initialisés et paramétrés. Pour l'instant, seuls les utilisateurs privilégiés possèdent un profil et peuvent interagir avec le système.

## 3.6 Phase opérationnelle

Dans cette phase, le système est opérationnel :

- un utilisateur final peut exécuter des services et gérer des scénarios,

- un utilisateur privilégié peut gérer les profils des utilisateurs,
- un administrateur peut paramétrer le système,
- le système exécute automatiquement la gestion de l'environnement et du système.

Les utilisateurs interagissent avec le système par le biais des GUIA. Les GUIA adaptent la présentation une fois leurs données synchronisées. Les interfaces et leur contenu sont adaptés en fonction du profil de l'utilisateur. Dans les sections suivantes, nous présentons la gestion des fonctionnalités explicites et implicites par notre système. Pour chacune des fonctionnalités, nous précisons les agents qui interviennent dans sa réalisation.

### 3.6.1 Gestion des profils des utilisateurs

Chaque utilisateur du système est lié à un profil utilisateur. Le profil d'un utilisateur contient ses identifiants, son rôle, ses droits d'accès aux équipements de l'environnement et la liste des scénarios qu'il a défini. Lorsqu'un utilisateur se connecte au système, les GUIA utilisent le profil de l'utilisateur pour gérer l'adaptation de la présentation. Les données des profils des utilisateurs sont gérées par les MgrCA.

Un utilisateur privilégié peut définir, modifier et supprimer les profils des utilisateurs à tout moment. Les droits d'accès sont des règles de paramétrage équivalentes aux règles de paramétrage du champ de contrôle des DCA. L'interface graphique qui permet aux utilisateurs privilégiés la gestion des profils des utilisateurs est similaire à l'interface qui permet le paramétrage du champ de contrôle des DCA présentée dans la figure 3.3.

Après la phase de configuration, seuls les utilisateurs privilégiés peuvent se connecter. Le système possède leurs profils. Dans notre cas d'étude, seuls les parents peuvent se connecter. Pour que les enfants puissent se connecter, les parents doivent définir un profil pour chaque enfant. Les parents peuvent à tout moment modifier ou supprimer un profil.

#### 3.6.1.1 Définition d'un profil

La définition du profil utilisateur consiste à choisir des identifiants pour l'utilisateur et à paramétrer ses droits d'accès aux équipements. Les droits d'accès des utilisateurs aux équipements peuvent être restreints par des règles de paramétrage. Ces règles sont similaires à celles utilisées pour définir le paramétrage du champ de contrôle des DCA.

Par exemple, les parents définissent un profil utilisateur pour chaque enfant leur permettant d'avoir accès uniquement aux équipements de leur chambre. Ainsi, les enfants ne pourront pas contrôler les équipements des autres pièces.

Lorsqu'un profil est défini, le GUIA envoie au MedCA la description du profil. Le MedCA enregistre le profil et le diffuse à tous les MgrCA qui l'enregistrent aussi. Les profils des utilisateurs sont répliqués sur tous les MgrCA pour augmenter la tolérance aux pannes des MgrCA.

#### 3.6.1.2 Modification d'un profil

Un utilisateur privilégié peut ajouter, modifier ou supprimer des règles de paramétrage des droits d'accès. Par exemple, les parents peuvent modifier les profils de leurs enfants pour les empêcher d'avoir le contrôle des volets et des lumières de leur chambre.

Le GUIA transmet aux MgrCA, par le biais du MedCA, les changements du paramétrage. Chaque MgrCA met à jour le profil de l'utilisateur.

Dans le cas de la suppression d'une règle de paramétrage (suppression d'un droit d'accès à un équipement), le médiateur lance une opération de vérification de la validité des scénarios définis par l'utilisateur. Cette opération consiste à vérifier si les scénarios sont compatibles avec le nouveau paramétrage de l'utilisateur. Le médiateur diffuse aux MgrCA la liste des scénarios incompatibles. L'implémentation des scénarios incompatibles est stoppée. La suppression de l'implémentation d'un scénario est décrite dans la section 3.6.3.3.

#### 3.6.1.3 Suppression d'un profil

Lorsqu'un profil est supprimé, le GUIA informe les MgrCA par le biais du médiateur. Les MgrCA suppriment le descripteur contenant le profil de l'utilisateur. Les scénarios définis par l'utilisateur sont supprimés. L'utilisateur dont le profil a été supprimé ne pourra plus se connecter au système.

### 3.6.2 Gestion des services

Chaque utilisateur peut exécuter une opération offerte par un service (section 1.2.1) en fonction de son profil. Nous définissons deux types d'opérations : *actionneur* et *capteur*. Une opération de type actionneur permet d'agir sur l'équipement, comme par exemple allumer la lumière. Une opération de type capteur est une opération qui rend une valeur reflétant l'état de l'équipement, comme par exemple allumé ou éteint, ou de l'environnement, comme par exemple la température. Les GUIA offrent une interface graphique pour l'exécution des services. Elle contient tous les équipements que l'utilisateur peut contrôler. La figure 3.4 représente l'interface qui permet à l'utilisateur d'exécuter une opération offerte par un service.

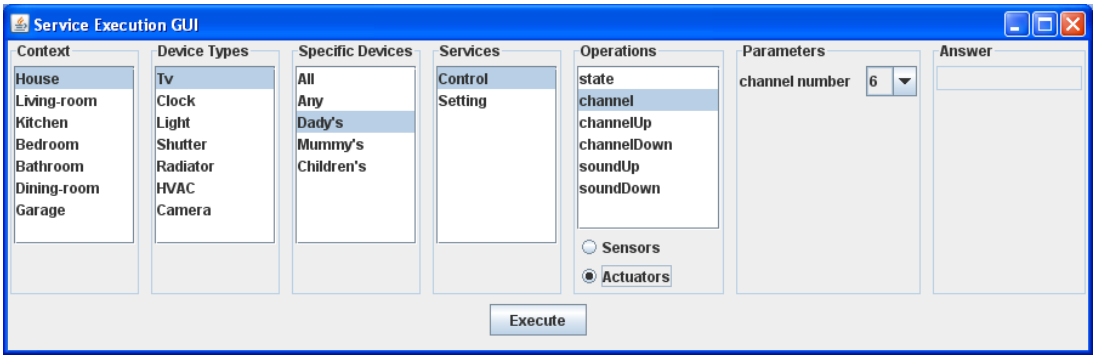



Figure 3.4 – L'interface d'exécution des services

Les utilisateurs peuvent choisir le ou les équipements qui exécutent l'opération selon trois modes de sélection :

- Mode *tous les équipements* du même type dans un contexte
- Mode *n'importe quel équipement* d'un type et contexte précis.
- Mode *un équipement particulier* d'un contexte précis.

Les utilisateurs peuvent aussi choisir le type de l'opération (actionneur ou capteur) pour les trier dans l'interface.
Pour exécuter une opération, ils sélectionnent le mode, le service puis l'opération. Ils doivent aussi fournir les valeurs des paramètres en entrée de l'opération à exécuter.

Le GUIA transmet la requête au médiateur. La requête contient le mode de sélection de l'équipement, l'opération et les valeurs des paramètres de l'opération. L'exécution de l'opération est réalisée de la manière suivante. Le MedCA envoie aux MgrCA une requête pour être informé des DCA qui contrôlent le ou les équipements sur lesquels l'opération doit être exécutée. Chaque MgrCA lui renvoie la liste de DCA qui contrôlent l'équipement. Le MedCA choisit alors le ou les DCA qui vont exécuter l'opération. Il transmet la requête à leurs MgrCA.
Une fois l'opération exécutée par les DCA, les MgrCA informent le MedCA qui informe le GUIA. Si les opérations retournent une valeur, les valeurs sont retournées au GUIA.

### 3.6.3 Gestion des scénarios

Les utilisateurs peuvent gérer eux-mêmes les scénarios en fonction de leur profil. Les scénarios permettent aux utilisateurs de définir des exécutions programmées de services lancées explicitement par l'utilisateur ou automatiquement sous certaines conditions.

Nous avons choisi de définir les scénarios sous la forme de règles ECA (Événement - Condition - Action) [50, 12]. La sémantique d'une règle ECA est "lorsqu'un événement E a lieu, si la condition C est satisfaite, l'action A doit être exécutée". Les règles ECA permettent aux systèmes d'acquérir un comportement réactif qui est compatible avec les environnements domotiques basés sur les événements. La clause événement n'est composée que d'un seul événement qui déclenche soit la vérification de la clause condition soit directement l'exécution de la clause action. La clause condition, optionnelle, est une expression booléenne (composée de plusieurs conditions) et la clause action peut être composée de plusieurs actions exécutées en séquence.

#### 3.6.3.1 Définition d'un scénario

Les utilisateurs finaux peuvent définir des scénarios à tout moment via l'interface de définition des scénarios. Cette interface contient tous les services disponibles dans l'environnement compatibles avec le profil de l'utilisateur. La figure 3.5 présente l'interface graphique.

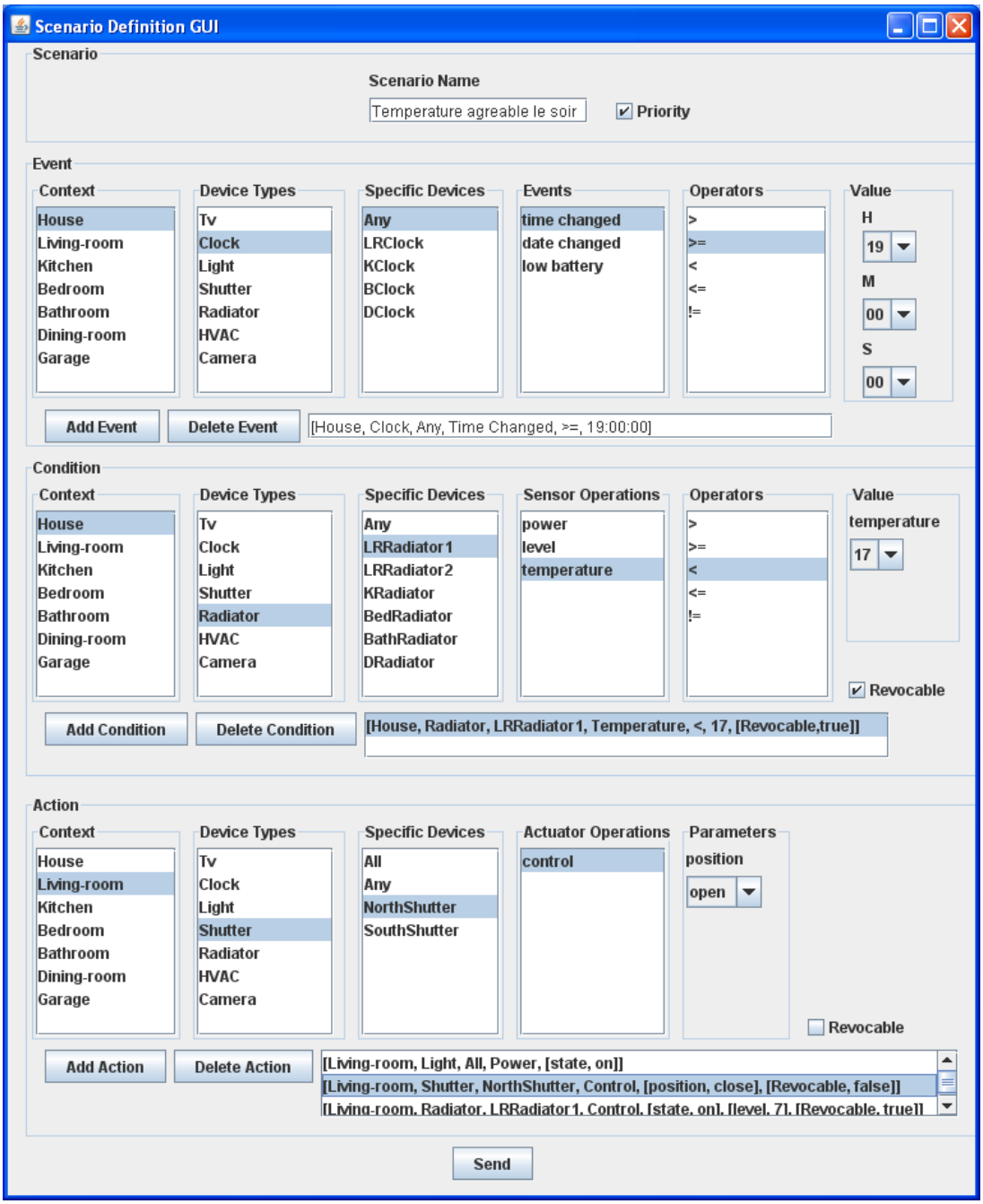



Figure 3.5 – L'interface de définition des scénarios

L'interface graphique est découpée en trois sections qui correspondent aux trois parties des règles ECA : *événement*, *condition* et *action*.
La section *événement* contient tous les événements fournis par les équipements de l'environnement. Elle permet aux utilisateurs de définir l'événement qui déclenche le scénario.
La section *condition* contient toutes les opérations de type capteur offertes par les équipements de l'environnement. Elle permet aux utilisateurs de définir une expression booléenne composée d'expressions booléennes simples.
La section *action* contient toutes les opérations de type actionneur offertes par les équipements

de l'environnement.
Les opérations de type capteur et actionneur sont définies dans la section 3.6.2.
L'interface offre aussi aux utilisateurs trois options :

- **Mode de sélection des équipements.** Au lieu d'imposer aux utilisateurs le choix des équipements dans la définition des scénarios, le système propose aux utilisateurs trois modes de sélection des équipements de l'environnement qui fournissent l'événement et les opérations. Les modes de sélection sont décrits dans la section 3.6.2. Le mode *tous les équipements* ne peut être attribué qu'à la sélection des équipements de la clause *action*. Cette option guide le système dans la gestion automatique des changements liés à l'environnement. Plus de détails sont fournis dans la section 3.6.4.
- **Scénario prioritaire.** Les opérations définies dans la clause *action* d'un scénario prioritaire doivent être exécutées immédiatement. Leur exécution ne peut ni être bloquée ni être repoussée à un autre moment. Cette option guide le système dans la gestion des conflits (section 3.6.4.3).
- **Condition ou action Révocable.** Lors de la panne d'un équipement qui fournit une opération de type capteur ou actionneur, cette option autorise le système à remplacer automatiquement l'équipement par autre équipement. Cette option guide le système dans la gestion des pannes des équipements (section 3.6.4).

L'interface permet aux utilisateurs de trier les événements et opérations par rapport au contexte et au type de l'équipement qui les fournit. Un utilisateur définit un scénario selon les étapes suivantes :

- **Clause scénario.** Il attribue un identifiant au scénario et spécifie s'il est prioritaire.
- **Clause événement.** Il sélectionne un événement, spécifie un opérateur et une valeur de comparaison.
- **Clause Condition.** Pour définir une condition, il sélectionne une opération de type capteur, un opérateur et une valeur de comparaison et spécifie si cette condition est révocable. Il peut ajouter autant de conditions que souhaité.
- **Clause Action.** Pour définir une action, il sélectionne une opération de type actionneur, spécifie les valeurs des paramètres en entrée et si cette action est révocable. Il peut ajouter autant d'actions que souhaité.

Une fois un scénario défini, le GUIA envoie la description du scénario au MedCA. Le médiateur est le responsable du choix des DCA qui vont implémenter le scénario. Nous appelons cela le déploiement d'un scénario. Le médiateur déploie un scénario par le biais d'un processus de déploiement de scénarios contraint par les champs de contrôle des DCA et leur charge. Le principal objectif du mécanisme est le déploiement des scénarios tout en minimisant les écarts de charges entre les DCA. Une fois les DCA sélectionnés, le MedCA diffuse la description du scénario à tous les MgrCA. Cette description contient les DCA qui implémentent le scénario. Les MgrCA transmettent ensuite la description aux DCA impliqués. Les DCA adaptent alors leur comportement de coordination pour l'implémentation du scénario.

Une extension possible de ce modèle pourrait être de proposer aux utilisateurs la définition de *scénarios de contrôle de cycle de vie des scénarios* qu'ils ont définis eux-mêmes ou qui ont été définis par les autres utilisateurs. Lors de la survenue de l'événement, ce scénario vérifie des conditions et puis active ou désactive des scénarios définis dans la clause action. Ils sont définis de la même manière que les autres scénarios, sauf que les actions sont exclusivement des actions d'activation ou de désactivation de scénarios. Ces scénarios permettraient de réutiliser des scénarios existants pour composer de nouveaux scénarios.

#### 3.6.3.2 Modification d'un scénario

Un utilisateur peut modifier un scénario à tout moment. La modification d'un scénario peut être la modification d'une valeur de comparaison ou d'un paramètre d'une opération, le remplacement de l'événement ou l'ajout ou la suppression d'une condition ou d'une action. Le GUIA transmet au MedCA la nouvelle description du scénario.
Dans le cas du remplacement de l'événement et de l'ajout d'une condition, le scénario est redéployé. Le MedCA se charge du redéploiement du scénario. Le redéploiement d'un scénario consiste à supprimer l'ancien scénario et à le déployer à nouveau. Cela est réalisé par l'adaptation du comportement de coordination des DCA. Le déploiement du scénario est réalisé comme dans la section précédente.
Dans les autres cas de modification, le scénario n'est pas redéployé. Les DCA prennent en compte les modifications en modifiant des valeurs de paramètres correspondants.

#### 3.6.3.3 Suppression ou la suspension d'un scénario

Un utilisateur peut supprimer ou suspendre un scénario à tout moment.
Dans le cas de la suppression, les MgrCA suppriment la description du scénario et informent les DCA qui implémentent le scénario. Les DCA suppriment le scénario par l'adaptation de leur comportement de coordination.
Dans le cas de la suspension, l'adaptation du comportement de coordination des DCA n'est pas nécessaire. Les DCA suspendent simplement l'exécution de ce scénario. Le scénario pourra être repris à tout moment.

### 3.6.4 Gestion de l'environnement

Les équipements de l'environnement peuvent être ajoutés ou supprimés à tout moment. Les DCA sont responsables de la gestion des changements qui se produisent dans l'environnement et de la gestion des conflits d'exécution des scénarios.
Les changements sont l'ajout ou la suppression d'un équipement. Ces changements peuvent avoir un impact sur le comportement de contrôle et de coordination. Des scénarios peuvent être suspendus ou modifiés. Les DCA informent les MgrCA des changements et de leurs impacts sur les scénarios.

#### 3.6.4.1 Ajout d'un nouvel équipement

**Gestion du comportement de contrôle.** Lorsque les DCA détectent l'ajout d'un nouvel équipement, ils adaptent leur comportement de contrôle en fonction de leur champ de contrôle. Les MgrCA enregistrent l'équipement ainsi que les nouveaux services et événements qui lui sont associés.

**Gestion du comportement de coordination.** Les DCA vérifient si des scénarios suspendus à cause de la panne d'un équipement du même type et/ou dans le même contexte peuvent être réactivés. Si c'est le cas, ils informent les MgrCA.

#### 3.6.4.2 Suppression d'un équipement

**Gestion du comportement de contrôle.** Lorsque les DCA détectent la suppression d'un équipement, ils adaptent leur comportement de contrôle. Les MgrCA désenregistrent l'équipement ainsi que les services et événements qu'il fournissait.

**Gestion du comportement de coordination.** La suppression d'un équipement peut être effectuée par l'utilisateur en éteignant l'équipement ou à cause de la panne de l'équipement. Voici comment le système se reconfigure pour essayer de préserver au mieux le fonctionnement des scénarios en fonction des modes de sélection établis lors de la définition des scénarios :

- Mode *tous les équipements* : le scénario n'est pas interrompu. L'action est exécutée par tous les équipements.
- Mode *n'importe quel équipement* : le système recherche dans le même contexte, un équipement du même type. Si aucun équipement n'est trouvé, le scénario est suspendu.
- Mode *un équipement en particulier* : le système ne peut pas remplacer cet équipement spécifique. Cependant, le système propose à l'utilisateur l'option *révocable*, qui permet au système de remplacer automatiquement l'équipement par un autre équipement équivalent dans le même contexte.

Lorsque les DCA détectent la suppression d'un équipement, ils adaptent leur comportement de coordination pour les scénarios impactés qui sont définis dans le mode *n'importe quel équipement* et dans le mode *un équipement en particulier* si l'option *révocable* a été choisie.

#### 3.6.4.3 Gestion des conflits

Lors de l'exécution des actions des scénarios, des conflits peuvent avoir lieu. Un *conflit d'exécution* est provoqué par l'envoi de demandes contradictoires de changements d'états à un équipement (d'opérations de type actionneur sur un équipement). Cette situation se produit lorsque le changement d'état réalisé par une opération exécutée dans un premier scénario est annulé par l'exécution d'une opération opposée dans un autre scénario, avant qu'une certaine durée,

appelée *période de rémanence*, ne soit écoulée. La période de rémanence définit la durée durant laquelle les effets d'une opération doivent être normalement maintenus (en dehors de situations d'urgence). Pour détecter ce conflit, le système dispose d'une liste d'antonymes qui lui permet de déterminer les opérations opposées. Voici les règles qui permettent aux agents de contrôle de détecter l'opposition :

- Deux opérations sont considérées comme opposées, si leurs noms sont composés d'une partie commune concaténée à des antonymes. Par exemple, les opératios *stepUp* et *stepDown* ou *increaseBrightness* et *decreaseBrightness* sont considérées comme opposées.
- Une opération est aussi considérée comme opposée à elle-même, si son nom commence par le préfixe *set* et a un seul paramètre d'entrée de type *integer*, *boolean* ou *chaîne de caractères*. De telles opérations affectent l'état d'un équipement et nous pensons que les changements d'état ne devraient normalement pas se succéder dans un laps de temps trop court. Par exmple, chacune des opérations *setFanSpeed(int fanSpeed)*, *setPower(boolean power)* et *setFanStatus(String fanStatus)* sont opposées à elles-mêmes.

Les DCA sont responsables de la gestion des conflits. Les règles précédentes leur permettent de détecter automatiquement certains conflits. Chaque DCA possède une liste d'antonymes qui lui permet de comparer les noms des opérations. Cette liste contient par exemple les antonymes suivants : *start/stop*, *open/close* et *up/down*. Lorsqu'une opération est exécutée sur un équipement, les DCA enregistrent le moment de l'exécution. Avant d'exécuter une nouvelle opération, les DCA vérifient si une opération opposée n'a pas été trop récemment exécutée dans la période de rémanence sur le même équipement. Si c'est le cas, les DCA bloquent l'exécution de la nouvelle opération. Cependant, il y a deux exceptions pour lesquelles les opérations opposées ne sont pas bloquées :

- L'exécution d'une opération sous la demande d'un utilisateur par le biais d'un GUIA. Cette opération est exécutée directement. Par exemple, si un scénario ferme un volet, l'utilisateur doit avoir la possibilité de l'ouvrir immédiatement.
- L'exécution d'une opération par un *scénario prioritaire* (section 3.6.3). Les scénarios prioritaires permettent de gérer des situations d'urgences. Ces scénarios sont toujours exécutés. Par exemple, si un scénario verrouille les portes et un scénario prioritaire doit déverrouiller les portes tout de suite à cause de la détection d'un incendie, le second scénario doit être exécuté.

### 3.6.5 Gestion du système

Le système doit être capable de gérer les changements qui se produisent dans le système. Les changements sont les suivants :

- Un agent de contrôle peut apparaître ou disparaître (désactivé ou tombé en panne) à tout moment.

- L'administrateur peut modifier le rôle ou le paramétrage d'un agent de contrôle à tout moment.

Le système gère automatiquement et dynamiquement les changements précédents par des adaptations de l'architecture globale du système et des architectures internes des agents. Cette gestion ne doit pas perturber le fonctionnement des autres agents ni interrompre l'exécution des scénarios et doit préserver l'exécution des scénarios affectés par les changements.

#### 3.6.5.1 Ajout d'un agent de contrôle

Lorsqu'un agent de contrôle est intégré, il effectue automatiquement une opération de recherche des agents et des équipements disponibles. Les administrateurs peuvent ensuite le paramétrer en précisant tout d'abord son rôle.

**L'agent a le rôle de DCA.** L'administrateur doit d'abord spécifier l'affectation de l'agent à un ou plusieurs MgrCA. Cette affectation est transmise aux MgrCA par le biais du MedCA. L'administrateur doit ensuite paramétrer son champ de contrôle qui est transmis à ses MgrCA. Un de ces MgrCA lui transmet ensuite les règles de paramétrage. Le DCA adapte son architecture interne conformément à ces règles. Si le DCA n'est pas affecté à un MgrCA, il est géré par tous les MgrCA. Si le champ de contrôle du DCA n'est pas paramétré, il a le contrôle de tous les équipements de l'environnement.

**L'agent a le rôle de MgrCA.** L'administrateur doit spécifier les DCA qui lui sont attribués. Ensuite, le médiateur gère la synchronisation des données avec le nouveau MgrCA (section 4.3.3.2). Puis, tous les MgrCA sont informés par le MedCA du nouveau MgrCA. Les DCA qu'il gère lui transmettent la valeur de leur champ de contrôle. Si le champ de gestion du MgrCA n'est pas paramétré, il gère tous les DCA du système.

Enfin, si les administrateurs ne paramètrent pas le nouvel agent de contrôle, le système paramètre cet agent en tant que DCA. Il est géré par tous les MgrCA et il a le contrôle de tous les équipements.

#### 3.6.5.2 Gestion des DCA

Nous traitons dans cette section la modification du champ de contrôle ou du rôle d'un DCA et la disparition d'un DCA.

**Modification du champ de contrôle d'un DCA.** La modification de la valeur du champ de contrôle d'un DCA est effectuée par la suppression ou l'ajout d'une ou plusieurs règles de paramétrage.
Le MedCa transmet aux MgrCA qui gèrent ce DCA les modifications du paramétrage du DCA. L'ajout ou la suppression de règles de paramétrage nécessite une adaptation du comportement du DCA.
La suppression de règles de paramétrage implique que le DCA contrôle moins d'équipements.

Ainsi, le DCA doit vérifier la validité des scénarios qu'il implémente. Si son nouveau champ de contrôle n'est plus compatible avec certains scénarios, il informe ses MgrCA et adapte son comportement de coordination. Les MgrCA sont responsables du redéploiement des scénarios.

**Modification du rôle d'un DCA.** Un DCA peut devenir un MgrCA. L'administrateur peut lui attribuer la gestion d'un ensemble de DCA qui lui transmettent la valeur de leurs champs de contrôle. Si l'administrateur ne paramètre pas son champ de gestion, il gère tous les DCA du système. Puis, le MedCA connecté au GUIA de l'administrateur synchronise les données de cet agent.
En cas de surcharge (prise en charge du nouveau rôle), le MgrCA essaie de redéployer ses scénarios par le biais du processus de déploiement des scénarios. Par contre, il continue à exécuter les scénarios qu'il n'arrive pas à redéployer afin d'assurer la continuité de leur fonctionnement.

**Suppression ou la panne d'un DCA.** La suppression ou la panne d'un DCA impliquent le redéploiement des scénarios implémentés par cet agent et peut impliquer la modification du comportement de coordination des MgrCA qui lui sont associés. Ce redéploiement est géré par l'un de ces MgrCA. Tous les MgrCA du système disposent des descriptions des scénarios. Cet aspect nous permet d'augmenter la tolérance aux pannes des agents de contrôle.
Le MgrCA redéploie chacun des scénarios implémentés par le DCA. Si des scénarios ne peuvent plus être déployés (aucun DCA du système ne contrôle certains équipements contrôlés par le DCA tombé en panne ou les DCA sont surchargés), le MgrCA adapte son comportement de contrôle et de coordination pour remplacer le DCA tombé en panne.

#### 3.6.5.3 Gestion des MgrCA

Nous traitons dans cette section la modification du champ de gestion ou du rôle d'un MgrCA et la disparition d'un MgrCA.

**Modification du champ de gestion d'un MgrCA.** La modification du paramétrage d'un MgrCA consiste soit à lui attribuer la gestion de nouveaux DCA soit à lui retirer la gestion d'un ensemble de DCA.
Dans le cas de nouvelles attributions, les nouveaux DCA lui transmettent la valeur de leur champ de contrôle.
Dans le cas de retrait, le MgrCA supprime la valeur des champs de contrôle des DCA correspondants.

**Modification du rôle d'un MgrCA.** Un MgrCA peut devenir un simple DCA. L'administrateur peut l'attribuer à un ou plusieurs MgrCA et paramétrer son champ de contrôle. Si le DCA n'est attribué à aucun MgrCA, tous les MgrCA le gèrent. Si son champ de contrôle n'est pas paramétré, il contrôle tous les équipements de l'environnement. Enfin, le DCA informe ses nouveaux manageurs de la valeur de son champ de contrôle.
Si les DCA qu'il gérait ne sont pas gérés par un autre MgrCA ou bien si l'administrateur ne les a pas attribués à un autre MgrCA, ces DCA sont gérés par tous les MgrCA du système.
Si après la modification, il ne reste plus qu'un seul MgrCA et que l'utilisateur n'a pas spécifié un autre MgrCA, le MgrCA restant sélectionne un DCA pour devenir MgrCA. Ce choix est basé sur la charge des DCA. Le MgrCA choisit le DCA qui a le moins de charge. Les adaptations que

le nouveau DCA doit accomplir sont expliquées dans la section 3.6.5.2.
Un MgrCA devient MedCA automatiquement lorsqu'un GUIA le sélectionne. Ainsi, l'administrateur ne modifie pas le rôle de cet agent vers un MedCA explicitement.

**Suppression ou la panne d'un MgrCA.** La suppression ou la panne d'un MgrCA implique la réaffectation des DCA qui ne sont plus gérés par aucun MgrCA. Les DCA qui ne sont plus gérés par aucun MgrCA sont gérés par tous les MgrCA du système. Les DCA transmettent aux MgrCA le paramétrage de leur champ de contrôle.
Lorsqu'un MgrCA qui est aussi MedCA tombe en panne, le GUIA le remplace directement par un autre MgrCA.
Nous justifions la réplique des descriptions des scénarios sur tous les MgrCA pour pallier cette situation. Si les descripteurs n'étaient pas répliqués, c'est-à-dire si chaque MgrCA ne possédait que les descriptions des scénarios implémentés par les DCA qu'il gère, la panne du MgrCA entraînerait la perte des descriptions des scénarios.
S'il ne reste plus qu'un seul MgrCA, le MgrCA restant se comporte comme décrit dans le cas précédent.

## 3.7 Conclusion

*SAASHA* combine deux paradigmes, les agents et les composants.

Dans ce chapitre, nous avons présenté l'architecture globale de *SAASHA* qui est basée sur des agents à base de composants. Notre système est composé de deux types d'agents : les *agents d'interface* et les *agents de contrôle*.

Les agents d'interface interagissent avec les utilisateurs par le biais d'interfaces graphiques adaptées en fonction de leur profil et de la composition de l'environnement et du système. Les utilisateurs peuvent gérer leurs besoins grâce aux fonctionnalités explicites. Les agents d'interface transmettent les requêtes aux agents de contrôle. Cela permet au système de répondre à la qualité requise *configurabilité*.

Les agents de contrôle peuvent posséder trois rôles :

- Les *agents de contrôle d'équipements* qui adaptent leurs comportements pour le contrôle des équipements et l'implémentation des scénarios. L'adaptation du comportement est réalisée dynamiquement par la génération, le déploiement et l'assemblage de composants ou la désinstallation de composants. Cela permet au système de répondre à la qualité requise *adaptation dynamique*.

- Les *agents de contrôle manageur* pour la gestion automatique du système et de l'environnement. Cela permet au système de répondre à la qualité requise *adaptation autonomique*.

- Les *agents de contrôle médiateur* pour la gestion automatique des requêtes des utilisateurs. Cela permet au système de répondre à la qualité requise *adaptation autonomique*.

Enfin, nos agents sont sensibles à leur contexte. Ils détectent l'apparition et la disparition des agents et des équipements. Cela permet au système de répondre à la qualité requise *sensibilité au contexte*. Cette qualité permet au système de réagir aux changements qui se produisent dans l'environnement et le système.

Dans le chapitre suivant, nous présentons l'architecture interne des agents basée sur les composants et les mécanismes d'adaptation qui permettent aux agents d'adapter la présentation du système, l'organisation du système et enfin le comportement du système pour réaliser les fonctionnalités explicites et implicites.

## 3.8 Limites et perspectives

Nous présentons dans cette section les limites et perspectives liées à l'architecture globale du système et aux fonctionnalités offertes.

**Profils prédéfinis.** Nous souhaitons proposer des profils prédéfinis pour les utilisateurs qui contiennent des rôles, des droits d'accès et des scénarios également prédéfinis. La définition d'un utilisateur consisterait à lui attribuer un profil prédéfini.

**Paramétrage automatique.** Pour le moment, le paramétrage automatique du système est très simple. Nous souhaitons définir des stratégies de paramétrage du système. Les administrateurs pourront sélectionner une stratégie et le système se paramétrera automatiquement en fonction de la stratégie sélectionnée. Nous envisageons de spécifier les trois stratégies suivantes :

- *Un paramétrage plus fiable.* Dans ce type de paramétrage, les champs de gestion des MgrCA et les champs de contrôle des DCA ne sont pas restreints. Les MgrCA gèrent tous les DCA et les DCA contrôlent tous les équipements. Ce paramétrage est plus fiable, car si un MgrCA tombe en panne, aucune reconfiguration du système n'est nécessaire. Une simple notification doit être envoyée aux utilisateurs. Si un DCA tombe en panne, les scénarios qu'il implémentait sont directement déployés sur les autres DCA. Ce paramétrage serait préféré dans les systèmes constitués d'un petit nombre d'agents de contrôle et/ou d'équipements. Dans les environnements constitués d'un grand nombre d'agents ou d'équipements, ce paramétrage pourrait provoquer une surcharge des MgrCA ou des DCA, en contraignant les MgrCA à gérer un nombre important d'agents et les DCA à contrôler un nombre important d'équipements.

- *Un paramétrage plus performant.* Dans ce type de paramétrage, les champs de gestion des MgrCA et les champs de contrôle des DCA sont restreints à un ensemble de DCA et d'équipements, respectivement. Ce paramétrage serait préféré dans les systèmes constitués d'un nombre élevé d'agents de contrôle et/ou d'équipements. Ce paramétrage est plus performant car les agents ont une charge moins importante, mais il pourrait être moins fiable. Lorsqu'un DCA tombe en panne, si aucun autre DCA du système ne peut le remplacer pour l'implémentation des scénarios, c'est son MgrCA qui le remplace. Ainsi, ce manageur peut être rapidement surchargé et ne plus être capable de remplacer un autre DCA tombant en panne.

- *Un paramétrage hybride.* Dans ce type de paramétrage, certains DCA sont gérés par tous les MgrCA et/ou contrôlent tous les équipements de l'environnement. Ces DCA permettraient de résoudre efficacement les problèmes de pannes des DCA. Ils pourraient remplacer directement les DCA qui tomberaient en panne. Ce paramétrage est préféré dans les systèmes constitués d'un nombre moyen d'agents de contrôle et/ou d'équipements.

Nous devons réaliser des expérimentations pour déterminer les nombres d'équipements, d'agents et d'utilisateurs qui nous permettront de choisir une stratégie.

Chapitre 4

# SAASHA : Architectures internes et mécanismes d'adaptation des agents

**Sommaire**


**4.1 Introduction . . . . . 49**
**4.2 Méta-modèle de SAASHA . . . . . 50**
**4.3 Agents d'interface . . . . . 52**
4.3.1 Architecture interne . . . . . 52
4.3.2 Cycle de vie . . . . . 54
4.3.3 Adaptation de la présentation . . . . . 56
**4.4 Agents de contrôle . . . . . 59**
4.4.1 Architecture interne . . . . . 60
4.4.2 Cycle de vie . . . . . 61
4.4.3 Adaptation de l'organisation . . . . . 65
4.4.4 Adaptation du comportement . . . . . 71
**4.5 Conclusion . . . . . 80**
**4.6 Limites et perspectives . . . . . 81**


## 4.1 Introduction

Dans le chapitre 3, nous avons présenté l'architecture de *SAASHA* et les fonctionnalités explicites et implicites qu'il offre.

Nous proposons dans ce chapitre une description détaillée de *SAASHA* concentrée sur les architectures internes des agents et les mécanismes d'adaptation mis en place par les agents. L'architecture interne des agents est basée sur les composants. Les agents gèrent trois niveaux d'adaptation : l'adaptation de la présentation, l'adaptation de l'organisation et l'adaptation du comportement (section 3.3.1).
Nous présentons, dans un premier temps, le méta-modèle de *SAASHA* [6, 15]. Puis, pour chaque

type d'agents, nous présentons son architecture interne, son cycle de vie et les mécanismes d'adaptation qu'il gère.

## 4.2 Méta-modèle de SAASHA

*SAASHA* est un système multi-agents à base de composants. Le méta-modèle complet de *SAASHA* est présenté dans la figure A.1 de l'annexe A. De manière à ce que la présentation de ce méta-modèle soit plus claire, nous présentons différentes parties de ce méta-modèle tout au long de ce chapitre.

La figure 4.1 présente une vue du méta-modèle de *SAASHA* qui représente les deux types d'agents de *SAASHA* et leur architecture interne basée sur les composants.

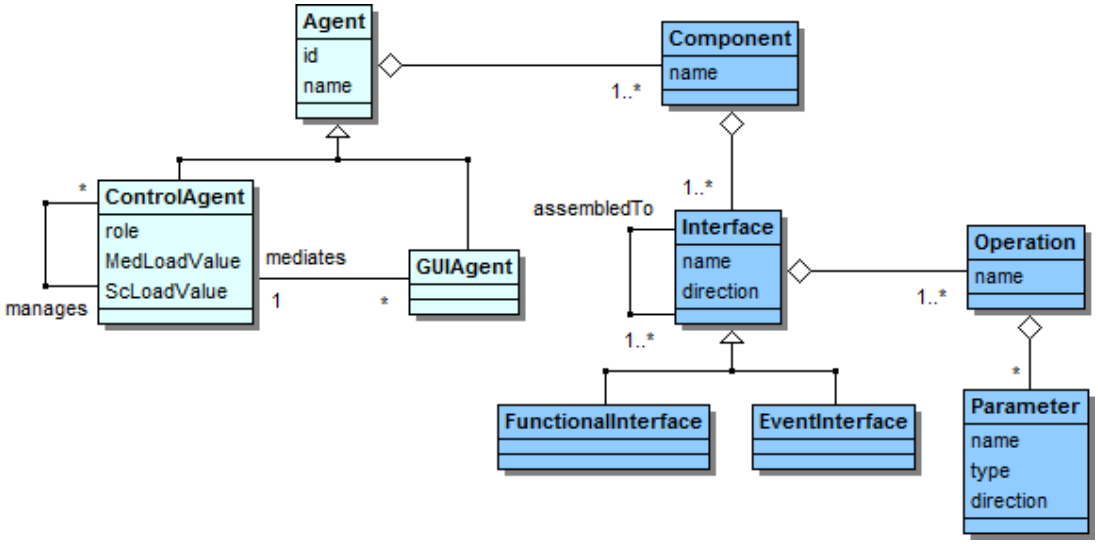



Figure 4.1 – Une vue du méta-modèle de *SAASHA* : Agents et Composants

Un composant est représenté par la classe *Component* et ses interfaces sont représentées par la classe *Interface* [28, 89].
Deux composants interagissent par la connexion de deux interfaces, une fournie par le premier composant et l'autre requise par le second composant. La direction d'une interface est spécifiée par l'attribut *direction* de la classe *Interface*. Plusieurs interfaces requises peuvent être connectées à une interface fournie [28, 89].
Les composants exposent des interfaces fonctionnelles, représentées par la classe *FunctionnalInterface*, et événementielles, représentées par la classe *EventInterface*. Ces deux dernières classes sont modélisées comme spécialisations de la classe *Interface*.
Ces interfaces regroupent des déclarations d'opérations modélisées par la classe *Operation*. Chacune de ces opérations est définie par un ensemble de types de paramètres d'entrée et de sortie représentés par la classe *Parameter*.
Une interface événementielle est un canal par lequel les événements sont émis.

Les deux types d'agents que nous avons identifiés dans *SAASHA*, les agents d'interface, repré-

sentés par la classe *GUIAgent*, et les agents de contrôle, représentés par la classe *ControlAgent*, spécialisent une super classe d'agents représentée par la classe abstraite *Agent*. Les agents de contrôle se distinguent au travers de trois rôles (attribut *role*) : les agents de contrôle médiateurs (MedCA), les agents de contrôle manageurs (MgrCA) et les agents de contrôle d'équipements (DCA). Les charges de travail qu'a un agent de contrôle sont représentées par les attributs : la charge de médiation *MedLoadValue* pour le nombre de GUIA gérés (section 3.3.3.3) et la charge de scénarios *ScLoadValue* pour le nombre de scénarios implémentés (section 3.3.3.1).

Nous définissons deux types de comportements pour les agents : les *comportements initiaux* et les *comportements acquis*. Dès leur intégration dans un environnement, les agents sont fournis avec un assemblage de composants que nous appelons *les composants initiaux*. Ces composants permettent aux agents de réaliser leurs comportements initiaux :

- La perception active de leur environnement et des autres agents du système. Cette capacité de perception est utilisée pour la détection et l'identification des utilisateurs, des équipements et des agents. Ce comportement permet aux agents de construire leur connaissance.
- La communication entre les agents. Elle est utilisée entre les GUIA et les MedCA pour l'envoi des requêtes des utilisateurs et pour la synchronisation des données des annuaires. Elle est utilisée entre les MgrCA eux-mêmes pour la gestion du système et des changements qui se produisent dans l'environnement et le système. Elle est aussi utilisée entre les MgrCA et les DCA pour la gestion de leur comportement. Enfin, elle est utilisée entre les DCA pour la gestion des scénarios et des changements dans l'environnement.
- La gestion de leur connaissance de l'environnement et du système. Nous définissons pour chaque agent un ensemble d'annuaires qui contiennent des données qui représentent la connaissance que l'agent a de son environnement et des autres agents du système. Chaque annuaire est géré par un composant. Le contenu des annuaires reflète la connaissance qu'à l'agent de l'état actuel de l'environnement (les utilisateurs et les équipements) et du système (les autres agents).
- La gestion des fonctionnalités explicites. Elle permet aux MedCA de gérer les requêtes des utilisateurs.
- La gestion des fonctionnalités implicites. Elle permet aux GUIA d'adapter la présentation et aux MgrCA d'adapter l'organisation en fonction des changements qui se produisent dans leur environnement et le système.
- Le méta-contrôle de leur architecture interne. Il est utilisé par les agents de contrôle pour adapter leur architecture interne afin d'adapter leur comportement de contrôle et de coordination.

Seuls les agents de contrôle peuvent acquérir de nouveaux comportements. Ces comportements correspondent aux comportements de contrôle d'équipements et de coordination pour l'implémentation des scénarios (section 3.3.1).

Dans le reste de ce chapitre, nous présentons pour chaque type d'agent son architecture interne, son cycle de vie et les mécanismes d'adaptation qu'il met en œuvre.

## 4.3 Agents d'interface

Les agents d'interface (GUIA) sont responsables de l'interaction avec les utilisateurs (section 3.3.2). Ils offrent des interfaces graphiques qui représentent les fonctionnalités explicites offertes par le système. À partir de ces interfaces graphiques, les utilisateurs peuvent définir leurs besoins. Les interfaces graphiques sont présentées dans le chapitre 3. Les agents d'interface sont responsables de l'adaptation de la présentation du système.

Dans le reste de cette section, nous présentons l'architecture interne des agents d'interface, leur cycle de vie et l'adaptation de la présentation qu'ils effectuent.

### 4.3.1 Architecture interne

La figure 4.2 représente, sous la forme d'un assemblage de composants, l'architecture interne d'un GUIA.

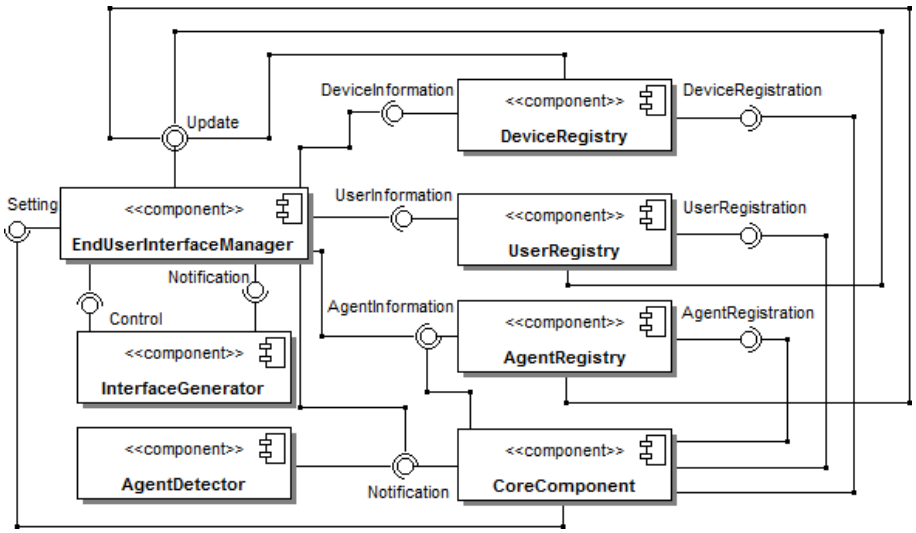



Figure 4.2 – L'architecture interne des GUIA : Méta-niveau

Les GUIA possèdent un composant principal *CoreComponent* qui est responsable de la gestion de la communication avec les MgrCA.
Les GUIA possèdent trois types d'annuaires : l'annuaire des équipements, l'annuaire des utilisateurs et l'annuaire des agents. Chacun de ces annuaires est géré par un composant : *DeviceRegistry*, *AgentRegistry* et *UserRegistry*, respectivement. Ces composants sont responsables de la gestion de la connaissance de l'agent de son environnement et du système. Les annuaires jouent un rôle essentiel dans l'adaptation de la présentation (adaptation des interfaces et de leur contenu en fonction de l'utilisateur) (section 4.3.3).
Les agents de contrôle sont automatiquement détectés par le composant *AgentDetector* qui est responsable de la perception active. Ce composant informe le composant *CoreComponent* qui enregistre leur identifiant dans l'annuaire des agents.
Les interfaces graphiques sont générées par le composant *InterfaceGenerator* et leur contenu est géré par le composant *EndUserInterfaceManager*. Le composant *EndUserInterfaceManager* est

responsable de l'adaptation de présentation. Le contenu des interfaces est extrait des annuaires de la manière suivante :

- l'interface de paramétrage du système est liée aux annuaires des agents et des équipements,
- les interfaces de gestion des profils des utilisateurs et des scénarios sont liées aux annuaires des utilisateurs et des équipements.

Une vue du méta-modèle de *SAASHA* est présentée dans la figure 4.3. Elle représente le méta-modèle des annuaires des agents (la classe *Agent*), des utilisateurs (la classe *User*) et des équipements (la classe *Device*).

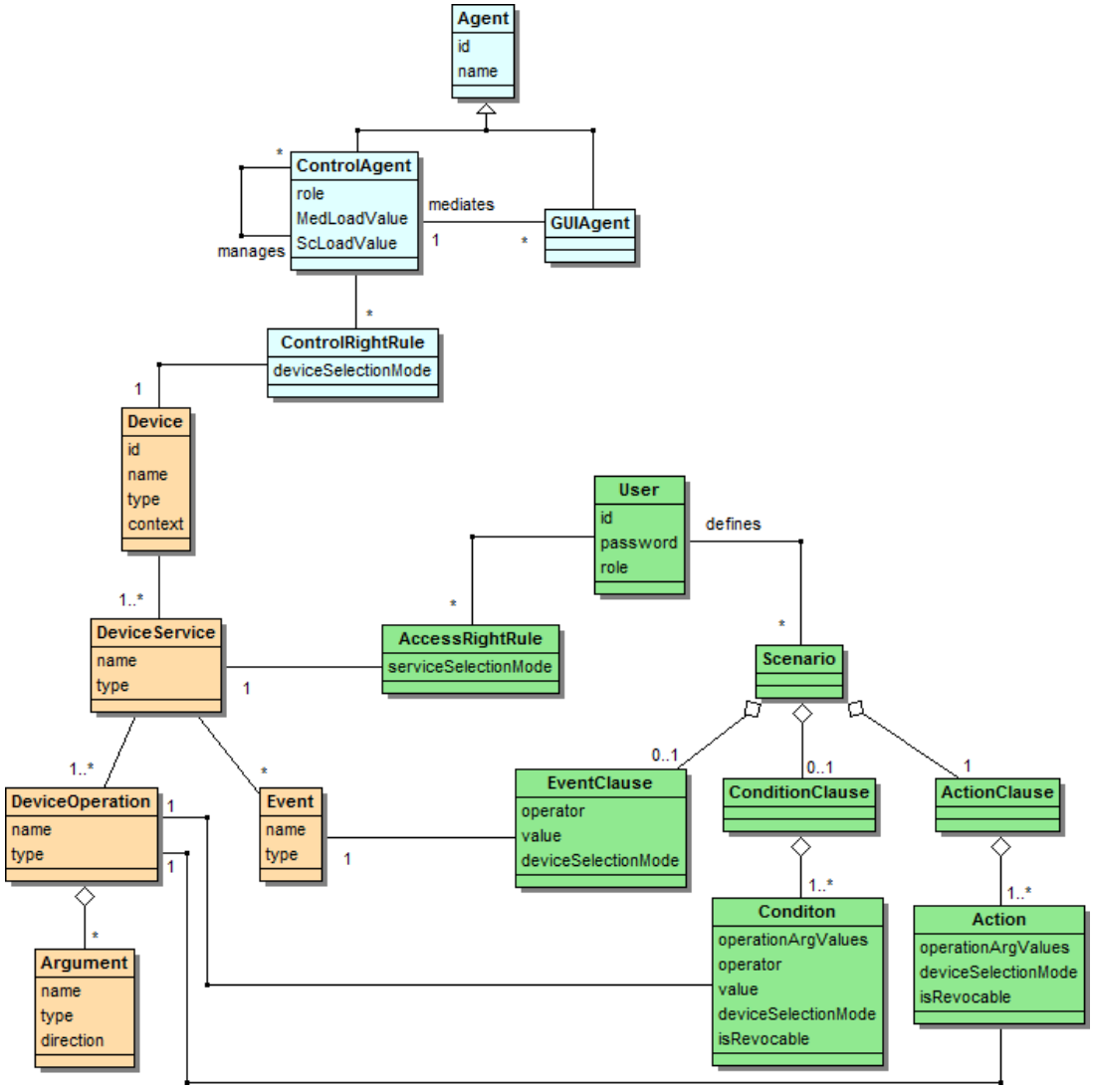



Figure 4.3 – Une vue du méta-modèle de *SAASHA* : Annuaires des GUIA

L'annuaire des agents (la classe *Agent*) contient pour chaque agent de contrôle (la classe *Control-Agent*) son identifiant, son rôle, les valeurs des charges, la liste des règles de paramétrage de son champ de contrôle (la classe *ControlRightRule*), la liste des DCA et des GUIA qu'il gère. Les

données des agents de contrôle sont fournies par le MedCA lors de la synchronisation des données (section 4.3.3.2).
L'annuaire des utilisateurs (la classe *User*) contient pour chaque utilisateur son profil représenté par ses identifiants, ses droits d'accès aux services fournis par les équipements (la classe *AccessRightRule*) et la liste des scénarios (la classe *Scenario*) qu'il a définis.
L'annuaire des équipements (la classe *Device*) contient les descriptions des équipements disponibles dans l'environnement. Chaque équipement est décrit par son type, son contexte et la liste des services (la classe *DeviceService*) qu'il fournit. Chaque service est décrit par les opérations (la classe *DeviceOperation*) qu'il fournit et les événements (la classe *Event*) qu'il émet.

### 4.3.2 Cycle de vie

Nous présentons dans cette section les différents états du cycle de vie des GUIA. Pour chaque état, nous précisons les composants et les annuaires impliqués. La figure 4.4 représente le cycle de vie des GUIA.

L'initialisation, état *Initializing*, consiste à déployer les composants initiaux de l'agent. Le composant *CoreComponent* est le responsable de l'activation des autres composants. Une fois lancé par l'utilisateur, état *Starting*, la connexion de l'agent au réseau de l'environnement est réalisée par le biais des composants *CoreComponent* et *AgentDetector*. Cela permet à l'agent de détecter les agents disponibles, d'être détecté par les autres agents et de communiquer avec les agents. Enfin, le composant *InterfaceGenerator* génère les interfaces graphiques. Le contenu des interfaces est vide. Les annuaires ne possèdent pas encore de données, sauf l'annuaire des utilisateurs qui contient les profils des administrateurs (ADM) et des utilisateurs privilégiés (UP). Cela permet aux ADM de se connecter pour lancer le paramétrage (automatique ou manuel comme décrit dans la section 3.5.2) et activer le système et cela permet aux UP de se connecter, une fois que le système est opérationnel. Les différents rôles des utilisateurs qui interagissent avec notre système sont définis dans la section 1.3.1.

L'état *Active* est composé de deux états principaux : l'état *UserInteraction* pour l'envoi des requêtes des utilisateurs et l'état *Listening* pour la reception des messages envoyés par les MgrCA.

La figure 4.5 représente l'état *UserInteraction* du cycle de vie des GUIA.
L'état *UserInteraction* représente l'interaction du GUIA avec un utilisateur. Elle commence par l'identification de l'utilisateur. L'utilisateur peut ensuite définir ses besoins au travers des fonctionnalités explicites offertes par le système. Cette interaction aboutit à l'envoi des besoins sous la forme de requêtes (messages) aux agents de contrôle pendant la phase de configuration et au MedCA pendant la phase opérationnelle. Dans ce qui suit, nous décrivons chaque état interne à cet état.

- L'état *UserConnection* représente la connexion de l'utilisateur au système qui correspond au choix du médiateur et à la vérification des identifiants de l'utilisateur (section 4.3.3.1).
- Les états *ScenarioDef*, *ScenarioMod*, *ScenarioDel*, *UserProfileDef*, *UserProfileMod* et *UserProfileDel* représentent la gestion des scénarios (section 3.6.3) et des profils des utilisateurs (section 3.6.1), respectivement.

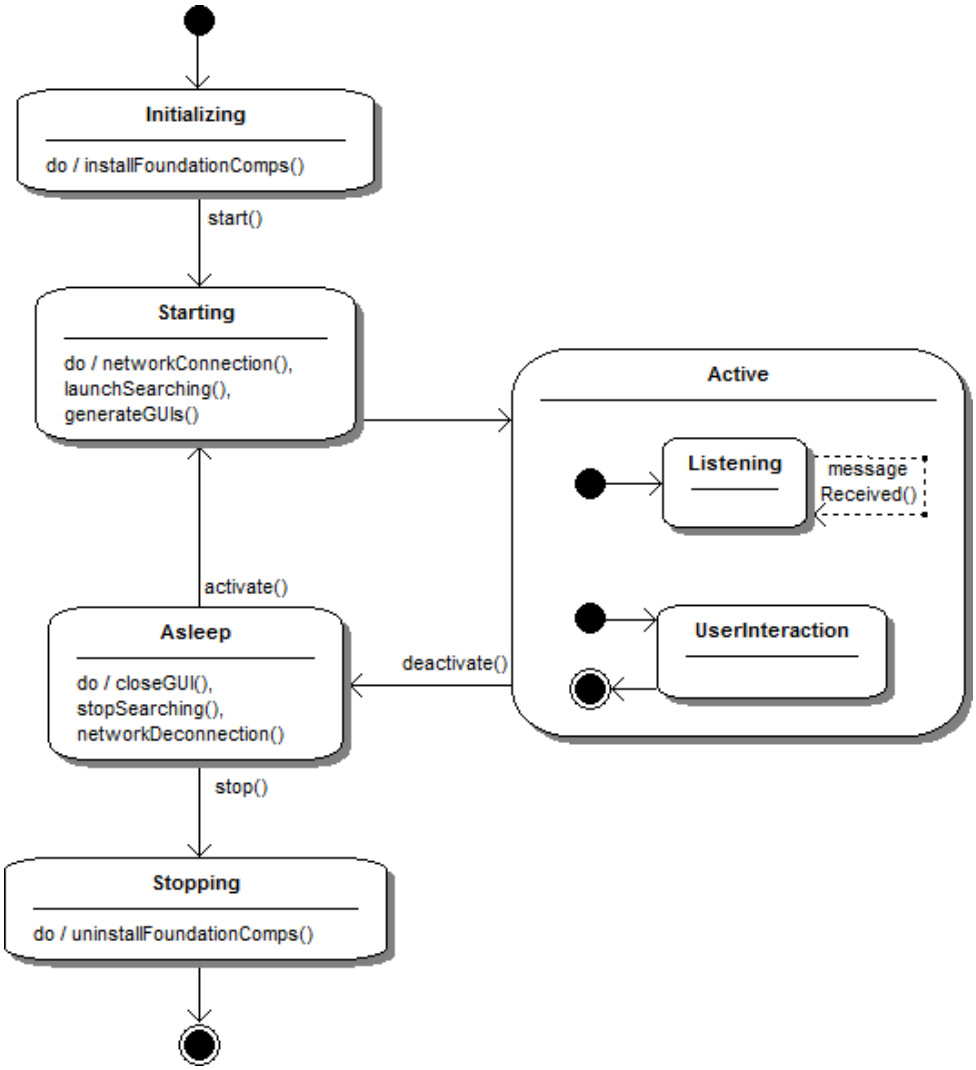



Figure 4.4 – Le diagramme d'état d'un GUIA

- L'état *SystemSettingMgt* représente la gestion du paramétrage des agents de contrôle (section 3.5.2.2).

- L'état *ServiceExecution* représente la demande d'exécution d'un service (section 3.6.2).

La figure 4.6 représente l'état *Listening* du cycle de vie des GUIA.
L'état *Listening* représente l'interaction du GUIA avec un agent de contrôle pendant la phase de configuration (section 3.5) et avec un MedCA pendant la phase opérationnelle (section 3.6). Cette interaction met à jour le contenu des annuaires et peut aboutir à la mise à jour du contenu des interfaces graphiques. Dans ce qui suit, nous décrivons chaque état composant cet état.

- Les états *AgentRegistration* et *AgentUnregistration* représentent la détection de l'apparition et de la disparition d'un agent de contrôle, respectivement. Dans cet état, le GUIA met à jour l'annuaire des agents et le contenu de l'interface de paramétrage du système.

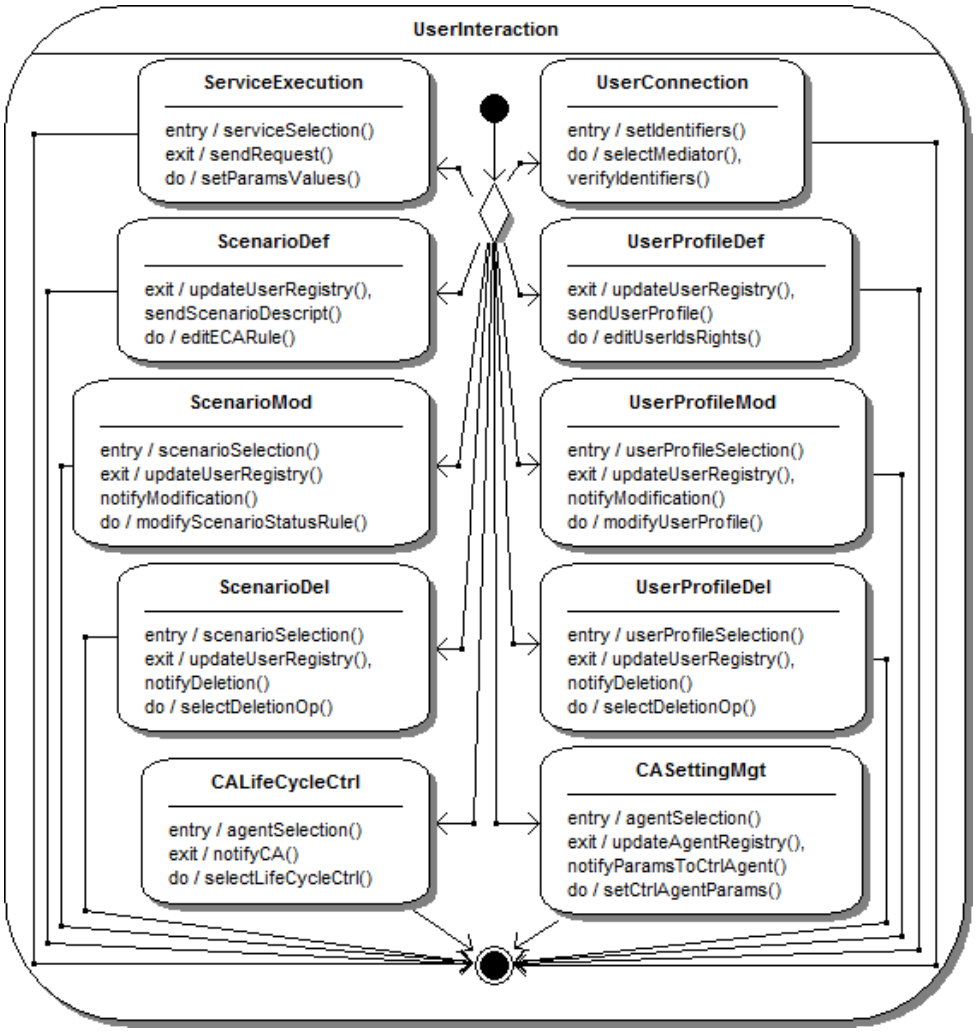



Figure 4.5 – L'état *UserInteraction* du cycle de vie d'un GUIA

- Les états *UserRegistration*, *DeviceRegistration*, *UserUnregistration* et *DeviceUnregistration* représentent les différents états du GUIA pendant la synchronisation des données. Ils correspondent à la réception des données d'un profil utilisateur ou d'un équipement, ou à la suppression du profil d'un utilisateur ou d'un équipement, respectivement. Le GUIA enregistre les données dans les annuaires.

- L'état *ServiceExecutionResult* représente la réception du résultat de l'exécution d'un service.

### 4.3.3 Adaptation de la présentation

Les GUIA adaptent les interfaces graphiques et leur contenu en fonction des profils des utilisateurs. Dans cette section, nous présentons le mécanisme d'adaptation de la présentation géré par les GUIA.

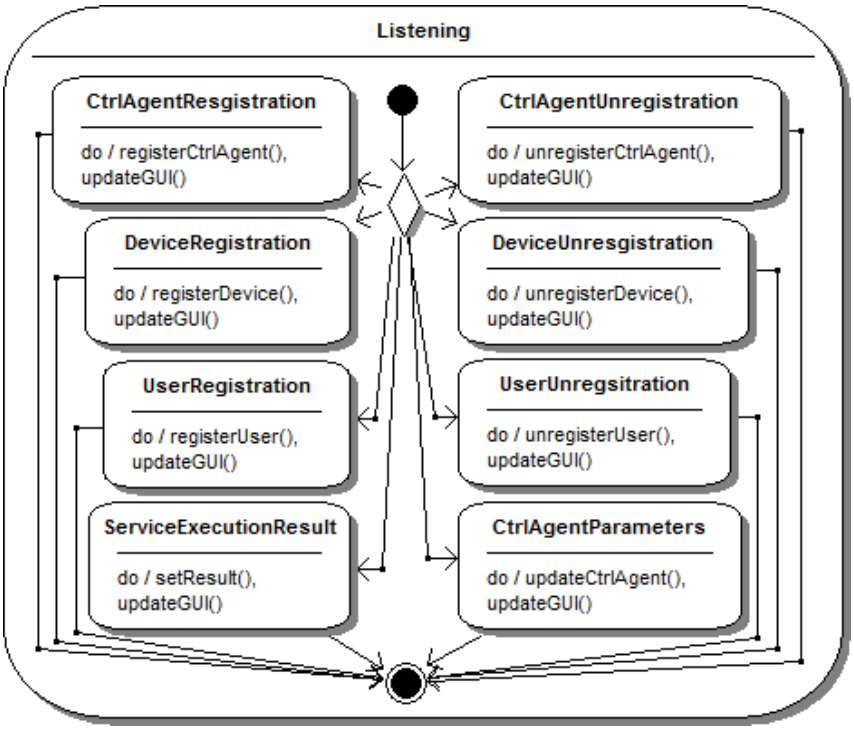



Figure 4.6 – L'état *Listening* du cycle de vie d'un GUIA

Les utilisateurs se connectent au système par le biais d'un GUIA. Ils définissent leurs besoins par le biais des interfaces graphiques. Cependant, ces interfaces doivent être adaptées, ainsi que leur contenu pour les raisons suivantes :

- Les utilisateurs peuvent se connecter au système par le biais de différents GUIA. Un GUIA doit adapter les interfaces graphiques et leur contenu en fonction du profil de l'utilisateur.
- Les utilisateurs possèdent des rôles différents. Par exemple, seuls les ADM doivent avoir accès à l'interface de paramétrage du système et seuls les UP doivent avoir accès à l'interface de gestion des profils des utilisateurs.
- Les utilisateurs finaux (UF) ne doivent avoir accès qu'aux équipements pour lesquels ils possèdent les droits d'accès et qu'aux scénarios qu'ils ont définis.
- Les changements se produisent à tout moment. Le contenu des interfaces doit refléter l'état actuel de l'environnement et du système.

Le GUIA communique avec le système par le biais d'un seul MgrCA qui joue alors le rôle de MedCA. Le GUIA sélectionne un médiateur parmi les MgrCA disponibles. La priorité est donnée au MgrCA le moins chargé. Pour l'instant, le GUIA se base sur la charge de médiation représentée par le paramètre MedLoadValue (section 3.3.3.3) qui contient le nombre de GUIA connectés au MgrCA.

L'adaptation du contenu des interfaces est basée sur le profil de l'utilisateur et la connaissance du GUIA de l'environnement et du système. Les annuaires du GUIA contiennent les informations

sur son contexte. Cette adaptation est réalisée en deux étapes : l'*identification des utilisateurs* et la *synchronisation des données des annuaires.*

#### 4.3.3.1 Identification des utilisateurs

Chaque GUIA doit donc être capable d'identifier l'utilisateur et ainsi accéder à son profil. Deux situations sont possibles :

- Le GUIA possède le profil de l'utilisateur. Cet utilisateur s'est déjà connecté par le biais du GUIA.
- Le GUIA ne possède pas le profil de l'utilisateur. C'est la première fois que l'utilisateur se connecte par le biais du GUIA.

Dans ce qui suit nous présentons le processus d'identification d'un utilisateur.

Le processus d'identification des utilisateurs est basé sur les composants *CoreComponent*, *EndUserInterfaceManager* et *InterfaceGenerator* et sur l'annuaire des utilisateurs. Les utilisateurs entrent leurs identifiants au travers d'une interface générique pour l'identification. Le composant *GUIGenerator* transmet les identifiants au composant *EndUserInterfaceManager* qui vérifie la validité des identifiants dans l'annuaire des utilisateurs. Deux situations sont possibles :

- les identifiants sont valides : le GUIA possède le profil de l'utilisateur. Le composant *EndUserInterfaceManager* informe le composant *CoreComponent*. Le composant *CoreComponent* envoie une requête pour la synchronisation des données. Cette requête contient les identifiants de l'utilisateur. Le MedCA effectue la synchronisation des données en fonction du profil de l'utilisateur.
- les identifiants ne sont pas valides : le composant *EndUserInterfaceManager* transmet une requête d'identification au composant *CoreComponent* qui la transmet au MedCA. Le MedCA vérifie la validité des identifiants. Dans ce cas, deux situations sont possibles :
  - les identifiants sont valides : ceci signifie que c'est la première connexion de l'utilisateur par le biais du GUIA. Le MedCA informe le GUIA et lance la synchronisation des données.
  - les identifiants ne sont pas valides : les identifiants entrés ne sont pas reconnus. Le MedCA informe le GUIA qui informe l'utilisateur que la demande de connexion est refusée.

Cette étape a permis au GUIA d'identifier le profil de l'utilisateur et de déclencher la synchronisation des données.

#### 4.3.3.2 Synchronisation des données

La synchronisation des données permet de mettre à jour les données des annuaires des GUIA. Les annuaires des GUIA doivent être synchronisés pour les raisons suivantes :

- Le GUIA se connecte pour la première fois au système. Il n'a aucune connaissance de l'environnement et du système. Les annuaires ne contiennent aucune donnée.
- Le GUIA se reconnecte au système. Un GUIA peut être installé sur un équipement mobile, les données de ces annuaires doivent être mises à jour en fonction des changements qui se sont produits.
- Le GUIA ne possède pas le profil de l'utilisateur qui lui permet d'identifier l'utilisateur.

La synchronisation des données est effectuée par le MedCA qui possède les données relatives à l'environnement (équipements et utilisateurs) et au système (agents) (section 3.3.3.3). Le processus de synchronisation des données, expliqué en détail dans la section 4.4.3.1, consiste à synchroniser le contenu des annuaires des agents, des équipements et des utilisateurs.

Une fois la synchronisation terminée, les annuaires contiennent les données qui correspondent au profil de l'utilisateur et à l'état actuel de l'environnement et du système. Au fur et à mesure de la synchronisation, les composants (*UserRegistry*, *DeviceRegistry* et *AgentRegistry*) qui gèrent les annuaires informent le composant *EndUserInterfaceManager* de la mise à jour de leurs données. En fonction du profil de l'utilisateur, le composant *EndUserInterfaceManager* transmet au composant InterfaceGenerator la liste des interfaces à générer et les données à y intégrer.

Le processus de synchronisation des données est réalisé de manière continue par le MedCA tant que le GUIA est connecté au système. Le MedCA transmet au GUIA les mises à jour des données des annuaires en fonction des changements qui se produisent dans l'environnement et dans le système.

## 4.4 Agents de contrôle

Les agents de contrôle sont responsables de l'interaction avec les équipements de l'environnement et les GUIA. Ils peuvent avoir trois rôles :

- agent de contrôle d'équipements (DCA) responsable du contrôle des équipements et de l'implémentation des scénarios (section 3.3.3.1),
- agent de contrôle manageurs (MgrCA) responsable de la gestion des fonctionnalités implicites (section 3.3.3.2),
- agent de contrôle médiateurs (MedCA) responsable de la gestion des fonctionnalités explicites (section 3.3.3.3).

Les agents de contrôle sont responsables de l'adaptation de l'organisation et du comportement. L'adaptation de l'organisation permet aux agents de répondre aux changements qui se produisent dans l'environnement et le système. L'adaptation du comportement permet aux agents d'acquérir de nouveaux comportements pour le contrôle des équipements et l'implémentation des scénarios. Les agents de contrôle acquièrent un nouveau comportement par adaptation de leur architecture interne. Cette adaptation consiste à générer, déployer et assembler de nouveaux composants, ainsi qu'à modifier l'assemblage des composants et à désinstaller des composants.

Dans le reste de cette section, nous présentons l'architecture interne des agents de contrôle, leur cycle de vie et l'adaptation de l'organisation et du comportement qu'ils exécutent.

### 4.4.1 Architecture interne

Initialement, les agents de contrôle possèdent tous le même assemblage de composants. La figure 4.7 représente, sous la forme d'un assemblage de composants, le méta-niveau de l'architecture interne d'un agent de contrôle.

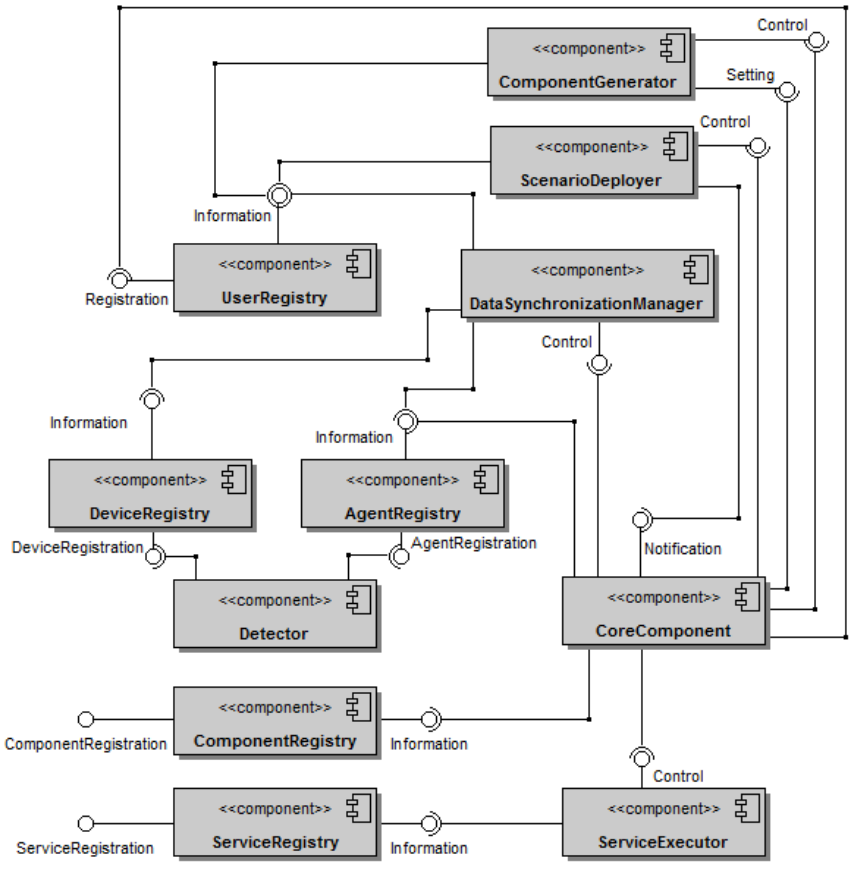



Figure 4.7 – L'architecture interne des agents de contrôle : Méta-niveau

Un agent de contrôle possède un composant principal, *Core Component*, qui est responsable de la gestion de la communication entre les agents et de la gestion de l'adaptation de l'organisation. La détection des agents du système et des équipements présents dans l'environnement est

effectuée par le composant *Detector* qui est responsable de la perception active de l'agent. Il enregistre les agents dans l'annuaire des agents géré par le composant *AgentRegistry* et les équipements dans l'annuaire des équipements géré par le composant *DeviceRegistry*. Les profils des utilisateurs sont enregistrés dans l'annuaire des utilisateurs géré par le composant *UserRegistry*. Initialement, cet annuaire contient les profils des ADM et UP. Ces trois derniers composants sont responsables de la gestion de la connaissance de l'agent de son environnement et du système.

Un agent de contrôle exécute le processus de synchronisation des données des annuaires d'un autre agent de contrôle ou d'un GUIA. Ce processus est exécuté par le biais du composant *DataSynchronizationManager*. Un agent de contrôle est aussi responsable de l'exécution du processus de déploiement des scénarios sur les DCA. Le processus est réalisé par le biais du composant *ScenarioDeployer*. Ces composants participent avec le composant *Core Component* dans l'adaptation de l'organisation.

Un agent de contrôle contrôle des équipements par le biais des *composants de contrôle*. Le composant *ComponentGenerator* génère ces composants en fonction du champ de contrôle de l'agent de contrôle. Un agent de contrôle déclenche l'exécution de services par le biais du composant *ServiceExecutor*. Les requêtes lui sont transmises par le biais du composant *CoreComponent*.

Un agent de contrôle implémente des scénarios par le biais des *composants de coordination*. Les composants de coordination sont générés par le biais du composant *ComponentGenerator*. Ce composant est responsable du méta-contrôle de l'architecture interne des agents de contrôle.

À cette architecture viendront s'ajouter des composants de contrôle en fonction du champ de contrôle et des composants de coordination pour les scénarios implémentés par l'agent. Après leur génération, les composants de contrôle et de coordination s'enregistrent dans l'annuaire des composants géré par le composant *ComponentRegistry*. Ils enregistrent aussi les services qu'ils fournissent dans l'annuaire des services gérés par le composant *ServiceRegistry*. Les méta-modèles des annuaires des composants et des services sont représentés dans la figure 4.12.

Les annuaires *AgentRegistry*, *DeviceRegistry* et *UserRegistry* sont identiques aux annuaires décrits dans la section 4.3.1. La figure 4.8 présente une autre vue du méta-modèle des annuaires présentés dans la figure 4.3. Dans la figure 4.8, les associations entre les annuaires des agents et des utilisateurs apparaissent. Elles n'ont pas été ajoutées dans la figure 4.3 pour des raisons de lisibilité. Les deux associations entre l'annuaire des agents et les utilisateurs représentent les GUIA avec lesquels un utilisateur s'est connecté et les scénarios implémentés par un DCA.

### 4.4.2 Cycle de vie

Nous présentons dans cette section les différents états du cycle de vie d'un agent de contrôle. Pour chaque état, nous précisons les composants et les annuaires impliqués. La figure 4.9 représente le cycle de vie d'un agent de contrôle. L'état *Active* est détaillé dans la figure 4.10.

Pendant l'initialisation (état *Initializing*), l'agent de contrôle n'a pas encore obtenu de rôle. Il installe les composants initiaux (méta-niveau) puis se connecte au réseau de l'environnement. Une fois connecté, le GUIA de l'ADM peut le détecter. L'ADM peut alors l'activer. L'activation

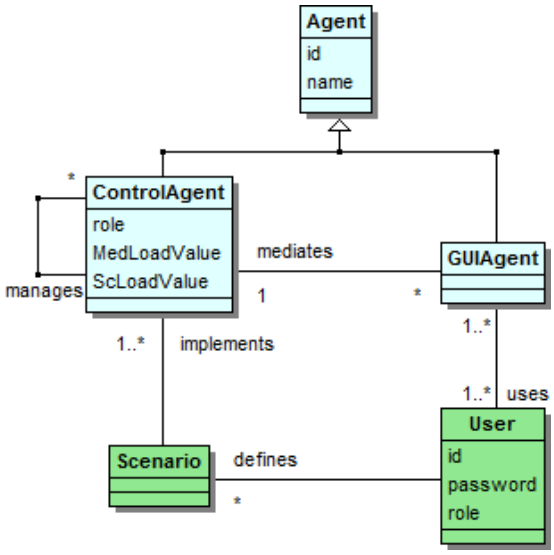



Figure 4.8 – Une vue du méta-modèle de *SAASHA* : Annuaire des agents

(état *Starting*) consiste à lancer le mécanisme de détection de l'agent, en activant le composant *Detector*.

Pendant la configuration (état *Configuration*), l'agent de contrôle enregistre les agents (état *AgentDetected*) et les équipements (état *DeviceDetected*) qu'il détecte dans l'annuaire des agents et des équipements, respectivement. Le composant *Detector* récupère les informations des agents et des équipements, les analyse et enregistre les informations nécessaires par le biais des composants *AgentRegistry* et *DeviceRegistry*. Plus de détails sur l'extraction des informations sont fournis dans le chapitre 5.
Dans l'état *Configuration*, l'ADM peut choisir la configuration automatique du système ou configurer manuellement les agents de contrôle (section 3.5.2). L'agent de contrôle obtient soit le rôle DCA, soit le rôle MgrCA. Un MgrCA peut ensuite obtenir le rôle MedCA lorsqu'il est connecté à un GUIA. Puis, en fonction du rôle qui est attribué à l'agent de contrôle soit :

- son champ de contrôle est paramétré (état *CtrlSetting*), s'il est un DCA. Le composant *ComponentGenerator* génère, en fonction du paramétrage, les composants de contrôle.
- son champ de gestion est paramétré (état *MgtSetting*), s'il est un MgrCA. Il informe les DCA qui lui sont attribués qu'il est leur manageur.

Une fois la configuration terminée, l'agent de contrôle est actif (état *Active*). La figure 4.10 représente l'état *Active* d'un agent de contrôle, quel que soit son rôle[1]. Dans ce qui suit, nous présentons, en fonction du rôle de l'agent de contrôle, les sous-états de l'état *Active*.

Lorsqu'un agent de contrôle détecte un agent de contrôle ou un GUIA (état *AgentDetection*), il l'enregistre dans l'annuaire des agents. L'état précédent est commun à tous les agents de contrôle.

[1]Pour des raisons de lisibilité, nous n'avons pas fait figurer sur les transitions du diagramme d'états les gardes qui correspondent au filtrage des différents types de messages reçus.

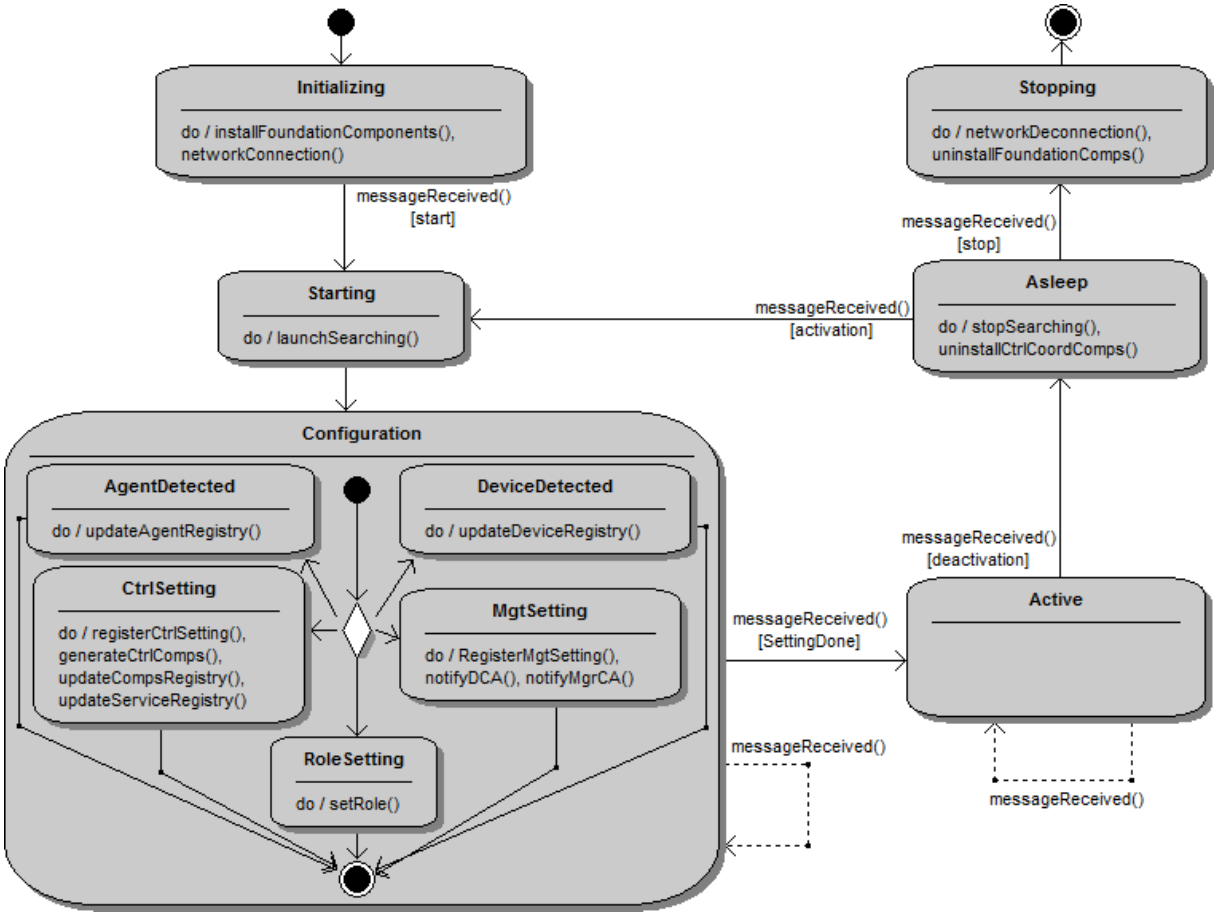



Figure 4.9 – Le diagramme d'état du cycle de vie d'un agent de contrôle

- **Le rôle de l'agent est MedCA.** Le MedCA est responsable de la gestion des fonctionnalités explicites :
  - Gestion des scénarios (état *ScenarioMgt*). Il met à jour l'annuaire des utilisateurs et déploie un scénario défini par l'utilisateur.
  - Gestion des profils des utilisateurs (état *UserProfileMgt*). Il met à jour l'annuaire des utilisateurs et gère les profils des utilisateurs.
  - Gestion des services (état *ServiceExecRequest*). Il sélectionne les DCA pour l'exécution d'un service, puis leur envoie une requête.
  - Gestion du paramétrage du système (état *SettingMgt*). Il met à jour l'annuaire des agents et gère la synchronisation des données des nouveaux MgrCA.
- **Le rôle de l'agent est MgrCA.** Le MgrCA est responsable de la gestion de l'organisation du système (fonctionnalités implicites) en fonction des changements qui se produisent dans l'environnement et le système.
  - Modification de son champ de gestion (état *MgtSetting*). Il informe les DCA de la modification de son champ de gestion.
  - Modification du champ de contrôle d'un DCA qu'il gère (état *DCACtrlSetting*). Il gère le redéploiement des scénarios.

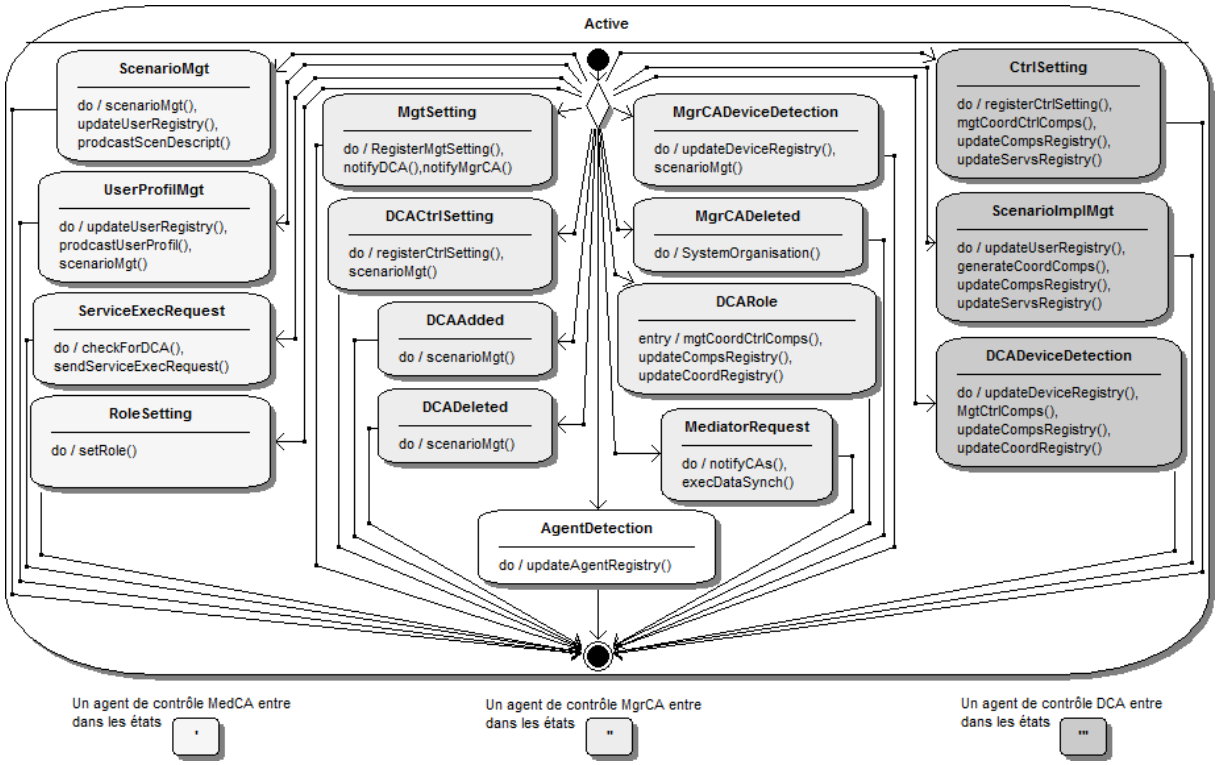



Figure 4.10 – L' état *Active* du cycle de vie d'un agent de contrôle

- Ajout d'un nouveau DCA (état *DCAAdded*). Si le MgrCA gère un nouveau DCA, il gère le redéploiement des scénarios.
- Suppression d'un DCA qu'il gère (état *DCADeleted*). Il gère le redéploiement des scénarios.
- Ajout ou suppression d'un équipement (état *MgrCADeviceDetection*). Il gère le redéploiement des scénarios.
- Suppression d'un MgrCA (état *MgrCADeleted*). Cet état représente la situation dans laquelle, il ne reste plus qu'un seul MgrCA. Le MgrCA restant est alors responsable du choix d'un autre MgrCA parmi les DCA. Le MgrCA effectue aussi la synchronisation des données du nouveau MgrCA.
- Rôle DCA (état *DCARole*). Le MgrCA joue le rôle DCA lors de la surcharge ou de la panne d'un DCA qu'il gère. L'administrateur peut aussi lui attribuer ce rôle. Dans les deux cas, il adapte son comportement de contrôle et/ou de coordination.
- Rôle MedCA (état *MedCARole*). Un GUIA se connecte au système par le biais d'un seul MgrCA qui joue alors le rôle MedCA.

- **Le rôle de l'agent est DCA.** Le DCA est responsable du contrôle des équipements et de l'implémentation des scénarios.
  - Modification de son champ de contrôle (état *CtrlSetting*). Il adapte son comportement de contrôle et gère les scénarios qu'il implémente.
  - Attribution de l'implémentation d'un scénario (état *ScenarioImpl*). Il adapte son comportement de coordination.

- Détection de l'ajout ou de la suppression d'un équipement (état *DCADeviceDetection*). Il gère les scénarios qu'il implémente.
- Exécution d'un service (état *ServiceExec*). Il exécute un service suite à une requête envoyée par un utilisateur.

À tout moment, un ADM peut désactiver un agent de contrôle. La désactivation cause la désinstallation des composants de contrôle et de coordination (état *Asleep*). Après cela, les utilisateurs peuvent soit détruire l'agent de contrôle qui désinstalle ses composants du méta-niveau et se déconnecte du réseau de l'environnement (état *Stopping*) ou réactive l'agent de contrôle (l'état *Starting*).

### 4.4.3 Adaptation de l'organisation

Le principal objectif de notre système est de contrôler les équipements d'un environnement domotique et d'implémenter les scénarios définis par les utilisateurs. Le système doit faire son possible pour que cet objectif soit réalisable et ce malgré les changements qui se produisent dans l'environnement et le système. Enfin, la réalisation de cet objectif doit être faite avec de bonnes performances.

Les aspects précédents sont réalisés par les agents de contrôle par le biais de l'adaptation de l'organisation. Cette adaptation comprend l'adaptation de l'architecture du système et le processus de déploiement des scénarios (section 4.4.3.2).
L'adaptation de l'architecture du système consiste à modifier les rôles des agents de contrôle et à synchroniser leurs données (section 4.4.3.1). Cette adaptation contribue à la gestion des pannes des équipements et des agents et augmente la tolérance à ces pannes.

L'organisation du système est la suivante. Lors du paramétrage, automatique ou manuel (section 3.5.2), du système, un rôle est attribué à chaque agent de contrôle. Une minorité d'entre eux ont le rôle de MgrCA et les autres le rôle de DCA. Les modifications de rôle possibles sont les suivantes :

- Un MgrCA peut jouer le rôle MedCA directement,
- Un MgrCA peut jour le rôle DCA mais il doit adapter son comportement de contrôle et/ou de coordination (section 4.4.4),
- Un DCA peut jouer le rôle MgrCA mais ses annuaires doivent être synchronisés avec un MgrCA (section 4.4.3.1).

Dans ce qui suit, nous présentons les situations pour lesquelles l'adaptation de l'organisation est réalisée et, pour chaque situation, ce qui se produit :

- **panne d'un MgrCA.** Dans la section 3.5.2.1, nous avions fixé le nombre minimum de MgrCA à deux.
  Après la panne du MgrCA, s'il ne reste qu'un seul MgrCA, un DCA doit le remplacer. Le

choix du DCA est réalisé par le MgrCA restant. Le MgrCA sélectionne le DCA avec le moins de charge de scénario (section 3.3.3.1). Le DCA sélectionné gère alors les DCA qui étaient gérés par le MgrCA tombé en panne. Il récupère la valeur du champ de contrôle de chaque DCA.
Dans le cas où le nombre de MgrCA restants est suffisant, la gestion des DCA, gérés par le MgrCA tombé en panne, est attribuée à tous les MgrCA restants. Ces MgrCA récupèrent la valeur du champ de contrôle de chaque DCA.

- **panne d'un MedCA.** Le GUIA sélectionne un autre MgrCA pour le remplacer. Pour l'instant, nous ne gérons pas la panne d'un MedCA pendant la synchronisation des données ou l'envoi des requêtes.

- **panne d'un DCA.** Le MgrCA qui gère ce DCA vérifie si les équipements contrôlés par le DCA sont contrôlés par un autre DCA du système. Si ce n'est pas le cas, le MgrCA joue également le rôle d'un DCA. Il contrôle les équipements en adaptant son architecture (section 4.4.4). Le champ de contrôle d'un MgrCA correspond aux champs de contrôle des DCA qu'il gère (section 3.3.3.2). Puis, il lance le processus de déploiement des scénarios implémentés par le DCA tombé en panne. Des scénarios peuvent ne pas être redéployés à cause de la surcharge des DCA. Cette situation est décrite dans le point suivant.

- **surcharge des DCA.** Dans cette situation, des scénarios ne peuvent plus être déployés. Les MgrCA des DCA surchargés jouent le rôle de DCA. Si cela n'est pas possible, les MgrCA tentent de diminuer la charge des DCA surchargés en redéployant un ensemble de scénarios. Pour l'instant, le choix des scénarios à redéployer est réalisé aléatoirement.

- **panne d'un équipement.** Comme nous l'avons dit dans la section 3.6.3.1, nous proposons aux utilisateurs trois modes de sélection d'équipements et l'option *révocable*. Dans la section 4.4.3.2, nous présentons le processus de déploiement d'un scénario. La panne d'un équipement peut affecter l'exécution d'un ensemble de scénarios. Les scénarios pour lesquels le mode de sélection de l'équipement a été spécifié en mode *n'importe quel équipement* ou pour lesquels l'option *révocable* a été sélectionnée peuvent être redéployés à condition de trouver un équipement qui peut remplacer l'équipement tombé en panne.

- **détection d'un GUIA.** Un MgrCA joue le rôle de MedCA et synchronise les données du GUIA. Il est le responsable de la gestion des requêtes du GUIA.

L'adaptation de l'organisation permet au système d'adapter son architecture et le déploiement des scénarios pour gérer au mieux les changements qui se produisent dans le système (section 1.2.3). Le principal but de cette adaptation est que le système soit en mesure, si possible, de conserver le contrôle des équipements et de ne pas interrompre l'implémentation des scénarios en cours d'exécution. Dans le cas où le système est surchargé, la réorganisation n'est pas possible.

Dans ce qui suit, nous présentons les processus de synchronisation des données et de déploiement des scénarios.

#### 4.4.3.1 Processus de synchronisation des données

Les MgrCA possèdent toutes les données du système (section 3.3.3.2). Le processus de synchronisation des données consiste à synchroniser tout ou partie des données des annuaires entre deux MgrCA ou entre un MedCA et un GUIA. Nous définissons deux types de données :

- Les données publiques qui sont les données relatives aux équipements et aux agents. Elles sont contenues dans les annuaires des équipements et des agents, respectivement. Les données publiques sont des données qui ne sont pas répliquées car les agents de contrôle peuvent les obtenir par le biais du mécanisme d'identification des agents et des équipements (section 5.2.2). L'*annuaire des agents* d'un MgrCA contient les informations sur les DCA qu'il gère. Ainsi, les *annuaires des agents* des MgrCA ne contiennent pas les mêmes données. Par contre, les annuaires des équipements contiennent les mêmes données du fait de la même perception de l'environnement.
- Les données privées qui représentent les profils des utilisateurs. Elles sont contenues dans l'annuaire des utilisateurs. Afin d'augmenter la tolérance aux pannes des MgrCA, les données privées sont répliquées sur tous les MgrCA.

Le processus de synchronisation des données comprend deux fonctions, la transmission des données et la transmission des mises à jour des données.
La transmission des données est réalisée lors de la première connexion du GUIA ou du MgrCA. Les annuaires à synchroniser ne contiennent pas de données.
Les mises à jour des données sont transmises au GUIA ou au MgrCA connecté ou au GUIA qui se connecte et dont les annuaires ont déjà été synchronisés. Les GUIA conservent les données lors de leur déconnexion. Les MgrCA journalisent les mises à jour des données pour les GUIA déconnectés. Chaque MgrCA possède un journal. Pour augmenter la tolérance aux pannes des MgrCA, les données des journaux sont aussi synchronisées entre les MgrCA. Les mises à jour sont effectuées dans les situations suivantes :

- lors de l'ajout ou de la suppression (pannes) des équipements ou des agents de l'environnement. Dans cette situation, les données publiques doivent être mises à jour. Les GUIA doivent être informés des équipements et des services supplémentaires ou supprimés et/ou des scénarios suspendus, supprimés ou réactivés à cause de la panne des équipements ou des agents.
- lorsqu'un UP effectue une modification sur les identifiants ou sur les droits d'accès d'un UF ou supprime un utilisateur. Dans cette situation, les données privées des GUIA des autres UP et de l'UF doivent être mises à jour. Ils doivent être informés des modifications apportées et/ou des scénarios suspendus, supprimés ou réactivés à cause de cette modification.
- lorsqu'un utilisateur effectue une modification dans les scénarios depuis un GUIA. Dans cette situation, les données privées des autres GUIA de cet utilisateur doivent être mises à jour.

Ainsi, les journaux contiennent les mises à jour des données publiques en fonction des GUIA qui ne sont pas connectés. Ils contiennent aussi les mises à jour des données privées pour chaque

couple utilisateur-GUIA. Les MgrCA enregistrent, pour chaque utilisateur, les GUIA avec lesquels ils se sont connectés. Par exemple, si un utilisateur définit un scénario par le biais d'un GUIA, il doit être en mesure de modifier ce scénario par le biais d'un autre GUIA.

La synchronisation des données entre deux MgrCA consiste à synchroniser uniquement les données privées et les données des journaux, étant donné que le MgrCA peut obtenir les données publiques. Cette opération ne nécessite l'intervention que d'un seul MgrCA car ces données sont répliquées.

La synchronisation des données entre un MedCA et un GUIA consiste à synchroniser les données publiques et les données privées. Ces données sont utilisées dans l'adaptation de la présentation (section 4.3.3). La synchronisation est réalisée en fonction du profil de l'utilisateur. Pour un ADM, seules les données publiques sont synchronisées. Pour un UP, seules les données des équipements et des utilisateurs sont synchronisées. Pour un UF, seules les données des équipements et de son profil sont synchronisées. Le MedCA transmet, seul, les données des équipements (communes à tous les MgrCA), les données privées (répliquées) et les mises à jour (répliquées). Par contre, tous les MgrCA participent à la transmission des données des agents (non-communes). C'est le MedCA qui organise la transmission. Les MgrCA, à tour de rôle, transmettent les données des agents. Ces données peuvent être présentes sur plusieurs MgrCA du fait que les DCA peuvent être gérés par plusieurs MgrCA. Ainsi, plusieurs MgrCA peuvent posséder des données provenant des mêmes DCA. Lors de la transmission, les MgrCA gèrent la duplication des données et ne transmettent pas plusieurs fois les mêmes données des agents.

Les MedCA et les MrgCA exécutent ce mécanisme par le biais du composant *DataSynchronizationManager*. Une fois un utilisateur connecté, le composant *CoreComponent* l'informe de la connexion d'un utilisateur et lui spécifie l'identité du GUIA. Lors de la détection d'un MgrCA, il lui spécifie son identité. Le composant *DataSynchronizationManager* consulte le profil de l'utilisateur dans l'annuaire des utilisateurs par le biais du composant *UserRegistry*, puis il commence la synchronisation des données. Les données sont envoyées par le biais du composant *CoreComponent*. Dans le cas de la transmission des données à un GUIA, le MedCA gère la synchronisation des données des MgrCA qui synchronisent les données à tour de rôle. La synchronisation des données est réalisée par le biais d'échanges de messages.

#### 4.4.3.2 Processus de déploiement des scénarios

Les DCA sont responsables de l'implémentation des scénarios. Pour l'instant, nous considérons que la charge de scénario d'un DCA correspond au nombre de scénarios implémentés par le DCA. Un scénario peut être implémenté selon deux modes : centralisé ou décentralisé.

- Un mode centralisé : Dans ce mode, le scénario est géré par un seul DCA. Ce DCA contrôle les équipements nécessaires pour l'implémentation du scénario. L'agent n'a besoin d'aucun échange de messages avec les autres DCA du système pour la réalisation du scénario.
- Un mode décentralisé : Dans ce mode, le scénario est géré par un ensemble de DCA qui collaborent par échange de messages pour la réalisation du scénario. Chaque DCA gère un ou plusieurs éléments du scénario.

Cependant, que cela soit en mode centralisé ou décentralisé, il est possible que plusieurs DCA aient la capacité d'implémenter un scénario. Dans ce cas, le déploiement consiste à choisir les DCA qui implémentent le scénario. Ce choix est basé sur les champs de contrôle des DCA et de leur charge de coordination. Nous avons choisi de déployer un scénario de manière à minimiser les écarts de charge entre les DCA.

Nous avons choisi de modéliser le déploiement des scénarios comme un programme linéaire en nombres entiers (PLNE) [69, 72]. Nous définissons les variables de décision, les contraintes et la fonction de coût comme suivant :

- Les variables de décision permettent de spécifier les DCA qui implémentent le scénario.
- Les contraintes sont que l'événement et chaque condition ne doivent être gérés que par un seul DCA et que la nouvelle charge des DCA choisis soit inférieure à la charge maximale autorisée aux DCA.
- L'objectif consiste à minimiser l'écart de la charge des DCA choisis. La nouvelle valeur de charge d'un DCA est la valeur de charge actuelle plus un si le DCA est choisi pour implémenter le scénario.

Pour clarifier les concepts précédents, nous présentons l'exemple de déploiement d'un scénario composé d'un événement (E) et de cinq conditions (C1-C5). Le déploiement du scénario peut être réalisé sur cinq DCA (DCA1-DCA5).
L'état des DCA avant déploiement est le suivant :

- Le champ de contrôle du DCA1 lui permet de gérer E, C2 et C4. Sa charge est de 2.
- Le champ de contrôle du DCA2 lui permet de gérer C1, C3 et C5. Sa charge est de 3.
- Le champ de contrôle du DCA3 lui permet de gérer C1, C2, C3, C4 et C5. Sa charge est de 2.
- Le champ de contrôle du DCA4 lui permet de gérer E. Sa charge est de 3.
- Le champ de contrôle du DCA5 lui permet de gérer E, C1, C2, C3, C4 et C5. Sa charge est de 6.

En observant les champs de contrôle des DCA, nous constatons que le DCA5 peut implémenter le scénario en mode centralisé. Le scénario peut être aussi implémenté en mode décentralisée selon différentes configurations, comme par exemple DCA1+DCA2, DCA3+DCA4, DCA1+DCA3... Le paramètre qui permet de trancher pour le choix de la configuration est la charge des DCA. Le tableau 4.1 représente le résultat de l'exécution du PLNE en considérant que la charge maximale est égale à 10. Les DCA1 et DCA3 ont été sélectionnés.

Ce problème de répartition de tâches sur l'ensemble des DCA est analogue au problème d'optimisation combinatoire du sac à dos, et plus précisément du sac à dos multiple [51]. Il s'agit d'un problème NP-complet dont la résolution exacte est coûteuse en temps de calcul et en ressources mémoire (complexité temporelle et spatiale). Il peut être résolu de manière approchée par des

| | E | C1 | C2 | C3 | C4 | C5 | charge | nouvelle charge |
|---|---|---|---|---|---|---|---|---|
| DCA1 | 1 | 0 | 0 | 0 | 0 | 0 | 2 | 3 |
| DCA2 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 |
| DCA3 | 0 | 1 | 1 | 1 | 1 | 1 | 3 | 4 |
| DCA4 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 |
| DCA5 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 6 |

Table 4.1 – Processus de déploiement des scénarios

méthodes heuristiques (algorithmes génétiques, recherche locale, ...)[35, 18] et des approches hybrides [71]. Bien que réduisant considérablement la complexité du problème, ces méthodes restent néanmoins trop sophistiquées et coûteuses pour des systèmes embarqués à ressources limitées.

Un meilleur compromis doit être recherché entre la qualité de la solution fournie par l'algorithme de résolution du problème et sa consommation des ressources. Dans le contexte de notre système, l'objectif n'est ni de trouver une solution optimale ni même une excellente approximation de cette solution optimale. L'objectif est de trouver une solution raisonnable, qui évite la surcharge inutile d'un DCA en répartissant convenablement la charge sur l'ensemble des agents disponibles. Ainsi, il est possible d'utiliser l'algorithme de résolution le plus simple, dit *algorithme glouton*, qui construit une solution de manière itérative. Les tâches à répartir sont préalablement triées par ordre de priorité (par exemple les tâches les plus critiques et les plus coûteuses en premier). A chaque itération, la tâche est affectée à l'agent le moins chargé. Cet algorithme peut échouer à construire une solution respectant la charge *maximale* autorisée pour chaque agent. On peut envisager de relâcher ces contraintes (ce n'est en général par une contrainte dure, les systèmes étant multitâches et capables de gérer une mémoire virtuelle) et de construire en suivant le même principe, une solution qui génère un minimum de surcharge.
Une amélioration de cette solution serait l'adoption d'une approche hybride, utilisant un algorithme de recherche locale pour optimiser une première solution fournie par l'algorithme glouton. Cependant, cette approche hybride ne pourrait garantir l'obtention d'une solution, si elle existe, en un temps contraint. Elle ne pourrait pas non plus démontrer l'absence d'une solution en un temps contraint. La résolution de problème par un simple algorithme glouton semble donc l'approche la plus raisonnable.

Pour mettre concrètement en œuvre ce mécanisme de répartition, le MedCA ou le MgrCA responsable du déploiement d'un scénario doit, dans un premier temps, récupérer la liste des DCA qui contrôlent les équipements décrits dans le scénario et leur charge. Il envoie une requête aux MgrCA qui lui renvoient pour chaque équipement les DCA qui le contrôlent. Ce mécanisme est implémenté par le composant *ScenarioDeployer*. Le composant *CoreComponent* lui transmet l'identifiant du scénario à déployer et la liste des DCA qui peuvent participer dans l'implémentation du scénario. Le composant récupère la description du scénario à partir du composant *UserRegistry*. Une fois le choix des DCA effectué, il modifie la description du scénario dans l'annuaire des utilisateurs et informe le composant *CoreComponent* qui transmet la description du scénario à tous les MgrCA. Ces DCA adaptent leur architecture interne en fonction de leur rôle dans la réalisation du scénario.

Le déploiement de certains scénarios peut être impossible à cause de la surcharge d'un ou plusieurs DCA. Dans cette situation, le MedCA essaie de diminuer la charge de ces DCA par le redéploiement de certains scénarios qu'ils implémentent. Cependant, si aucun redéploiement n'est possible, le scénario en cours de déploiement est suspendu. Le MedCA informe les MgrCA

et le GUIA. Les MgrCA mettent à jour le statut du scénario. Le message contient la cause de la suspension *système surchargé*.

### 4.4.4 Adaptation du comportement

Les DCA sont responsables du contrôle des équipements disponibles dans l'environnement pour l'exécution des services fournis par les équipements et pour l'implémentation des scénarios. Les DCA contrôlent les équipements par le biais des composants de contrôle et implémentent les scénarios par le biais des composants de coordination qui s'assemblent aux composants de contrôle.

Le mécanisme d'adaptation du comportement permet aux agents de contrôle d'adapter leur architecture en fonction de leur champ de contrôle et des scénarios qui leur sont attribués. L'adaptation consiste en la génération, le déploiement et l'assemblage des composants de contrôle et de coordination ou en la modification d'assemblage ou en la suppression de composants de contrôle et de coordination. La figure 4.11 représente le rôle des composants de contrôle et de coordination.

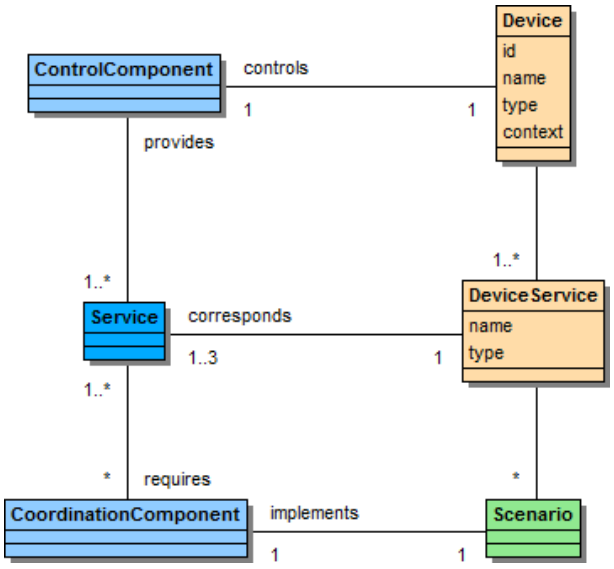



Figure 4.11 – Le rôle des composants de contrôle et de coordination

Chaque équipement est contrôlé par un composant de contrôle qui transforme les services fournis par l'équipement en services qu'il fournit lui-même aux composants de coordination. Un composant de coordination implémente un seul scénario à l'aide des services fournis par les composants de contrôle. Plus de détails techniques sur la génération des composants de contrôle et de coordination sont présentés dans le chapitre 5.

Le mécanisme d'adaptation du comportement est basé sur le composant *ComponentGenerator* et les annuaires des services et des composants. Les annuaires sont représentés dans la figure 4.12.

Le mécanisme d'adaptation du comportement permet aux agents de contrôle d'adapter leur comportement de contrôle et de coordination. Ce mécanisme est implémenté de la manière suivante. Le composant *ComponentGenerator* est responsable de la génération des composants de

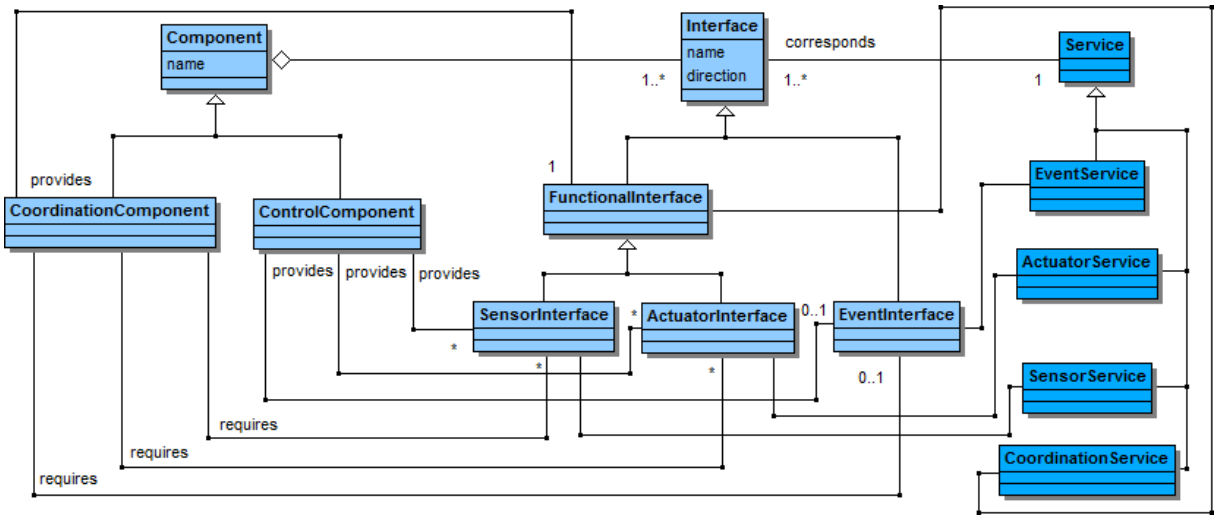



Figure 4.12 – Une vue du méta-modèle de *SAASHA* : Annuaire des composants et des services

contrôle (la classe *ControlComponent*) à l'aide du composant *DeviceRegistry* qui lui fournit des informations sur les équipements. Il génère les composants de coordination à l'aide du composant *UserRegistry* qui lui fournit des informations sur les scénarios. Il est paramétré par le composant *CoreComponent* en fonction du champ de contrôle du DCA.
Une fois générés, les composants de contrôle s'enregistrent dans l'annuaire des composants (classe *Component*) gérés par le composant *ComponentRegistry* et enregistrent les services qu'ils fournissent dans l'annuaire des services géré par le composant *ServiceRegistry*.
Une fois générés, les composants de coordination s'enregistrent dans l'annuaire des composants, enregistrent leur service dans l'annuaire des services et recherchent les services fournis par les composants de contrôle dont ils ont besoin pour l'implémentation des scénarios dans l'annuaire des services. Les composants de coordination peuvent modifier automatiquement leur assemblage.

Le composant *CoreComponent* est responsable de la désactivation et de la suppression des composants de contrôle et de coordination.

Dans les sections suivantes, nous décrivons en détail l'adaptation du comportement de contrôle et de coordination des agents de contrôle.

#### 4.4.4.1 Adaptation du comportement de contrôle

Le mécanisme d'adaptation du comportement de contrôle permet aux agents de contrôle d'adapter leur comportement de contrôle lors de la détection de l'apparition ou de la disparition d'un équipement, de la modification de leur champ de contrôle ou de la panne d'un DCA.

Le composant *ComponentGenerator* génère les composants de contrôle à l'aide des informations fournies par l'annuaire des équipements (Figure 4.3). L'annuaire des équipements fournit le contexte et le type des équipements et la liste de leurs services qu'ils fournissent. Pour chaque service, il fournit son type, la liste des opérations qu'il offre et la liste des événements qu'il émet. Une fois générés, les composants de contrôle s'enregistrent dans l'annuaire des composants

(figure 4.12) en fonction de l'identifiant, du type et du contexte de l'équipement. Cela permet au composant *CoreComponent* de rechercher un composant de contrôle pour contrôler son cycle de vie. Un composant de contrôle est un composant indépendant qui fournit des interfaces qui représentent des services et événements fournis par un seul équipement. Puisque les équipements sont des entités indépendantes qui peuvent apparaître et disparaître (panne) à tout moment, nous avons choisi de contrôler chaque équipement par un seul composant (et non pas plusieurs équipements par un seul composant), pour une meilleure gestion (génération ou désinstallation) des composants de contrôle lors de l'apparition et la disparition des équipements.

Les composants de contrôle peuvent fournir trois types d'interfaces : événement (la classe *EventInterface*), capteur (la classe *SensorInterface*) et actionneur (la classe *ActuatorInterface*). Chacune des classes des interfaces spécialise la classe *Interface*. Chaque interface (classe *Interface*) fournie correspond à un service (classe *Service*). Nous établissons une typologie de services (interfaces) fournis par les composants de contrôle et de coordination qui correspond à quatre types de services :

- Un service de type événement (la classe *EventService*) qui fournit les événements émis par les services d'un équipement. Il est utilisé pour constituer la clause *événement* des règles ECA représentant les scénarios. Ce service correspond aux interfaces événementielles.
- Un service de type capteur (la classe *SensorService*) qui offre les opérations de type capteur offertes par un service d'un équipement. Il est utilisé pour fournir des mesures dans la clause *condition* des règles ECA représentant les scénarios. Ce service correspond aux interfaces de type capteur.
- Un service de type actionneur (la classe *ActuatorService*) qui offre les opérations de type actionneur offertes par un service d'un équipement. Il est utilisé dans la clause *action* des règles ECA représentant les scénarios. Ce service correspond aux interfaces de type actionneur.
- Un service de type coordination (la classe *CoordinationService*) fourni par un composant de coordination. Il permet de modifier les paramètres du scénario.

Chacune des classes de service précédentes spécialise la classe *Service*. Les composants de contrôle enregistrent leurs services fournis dans l'annuaire des services. Chaque service est enregistré en fonction du type de service, de l'identifiant, du type et du contexte de l'équipement. Le composant *ServiceExecutor* recherche dans l'annuaire les services à exécuter. Les composants de coordination recherchent aussi les services qu'ils requièrent pour implémenter un scénario.

À présent, prenons l'exemple d'un DCA paramétré de telle sorte qu'il ne contrôle qu'une horloge, un volet, une lampe et un radiateur du salon. La figure 4.13 représente l'architecture interne de ce DCA après l'adaptation du contrôle. Nous remarquons que quatre composants de contrôle (*ClockCtrlComp*, *ShutterCtrlComp*, *LightCtrlComp* et *RadiatorCtrlComp*) ont été générés et déployés et puis ajoutés à l'architecture interne du DCA représentée dans la figure 4.13. Les composants de contrôle sont représentés dans la partie basse de la figure.
Le composant *RadiatorCtrlComp*, par exemple, est le composant de contrôle du radiateur. Le radiateur fournit deux services :

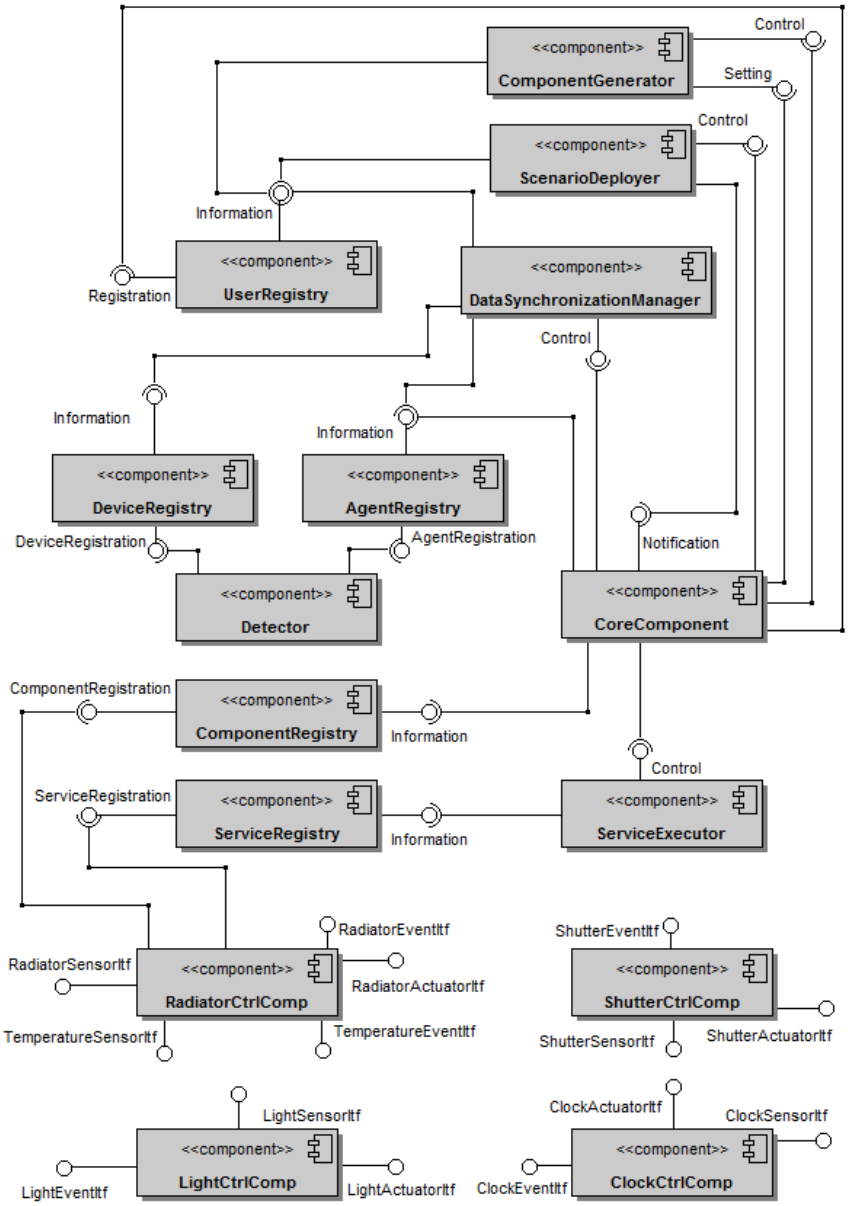



Figure 4.13 – L'architecture interne d'un DCA après l'adaptation du contrôle

- Le service *Radiator* qui permet d'allumer le radiateur à une puissance donnée, d'éteindre le radiateur, d'obtenir le statut du radiateur et d'obtenir la puissance actuelle du radiateur. Ce service émet des événements lors du changement du statut ou de la puissance du radiateur.

- Le service *Temperature* qui permet d'obtenir la température ambiante. Ce service émet des événements lors du changement de la température ambiante.

Le composant *RadiatorCtrlComp* fournit cinq interfaces : *RadiatorEventItf*, *RadiatorSensorItf*, *RadiatorActuatorItf*, *TemperatureSensorItf* et *TemperatureEventItf*. Les trois premières représentent le service *Radiator* fourni par l'équipement pour allumer et éteindre le radiateur. Ce service émet un événement lors du changement de statut du radiateur (allumé ou éteint). Les

deux dernières interfaces représentent le service *Temperature* fourni par l'équipement pour obtenir la température. Ce service émet un événement lors du changement de la température.

Ce composant possède deux interfaces requises : *ComponentRegistration* et *ServiceRegistration* fournies par les composants *ComponentRegistry* et *ServiceRegistry*, respectivement. La première interface lui permet de s'inscrire dans l'annuaire des composants en tant que le contrôleur du radiateur *R* de type *Radiator* et du contexte *salon*. La seconde interface lui permet d'enregistrer les services qu'il fournit dans l'annuaire des services en étant les services qui correspondent aux services, *Radiator* et *Temperature*, fournis le radiateur *R* de type *Radiator* et du contexte *salon*. Les autres composants de contrôle possèdent aussi les mêmes interfaces requises. Pour que la figure soit plus lisible, nous ne les avons pas représentées.

Lorsque le composant *Detector* détecte un nouvel équipement, le composant *CoreComponent* informe le composant *ComponentGenerator* de l'identité de l'équipement détecté. Il décide en fonction de son paramétrage si un composant de contrôle doit être généré.

Lorsque le composant *Detector* détecte la disparition d'un équipement, le composant *CoreComponent* recherche dans l'annuaire des composants si un composant contrôle cet équipement. Si c'est le cas, il désinstalle le composant de contrôle.

#### 4.4.4.2 Adaptation de comportement de coordination

Le mécanisme d'adaptation du comportement de coordination permet aux agents de contrôle d'adapter leur comportement de coordination pour implémenter des scénarios lors de la définition de scénarios ou lors du redéploiement des scénarios (modification des scénarios ou panne d'un équipement ou d'un agent).

L'annuaire des utilisateurs (Figure 4.3) contient les descriptions des scénarios implémentés par le DCA. Chaque description de scénario (classe *Scenario*) correspond à une règle *ECA* qui est composée d'un événement (classe *EventClause*), d'une ou de plusieurs conditions (classe *Condition*) et d'une ou de plusieurs actions (classe *Action*). Chaque description de scénario est liée aux DCA qui implémentent le scénario. Ceci permet au DCA de connaître les DCA avec lesquels il doit se coordonner pour l'implémentation des scénarios.

Le composant *ComponentGenerator* génère un composant de coordination *CoordinationComponent* (Figure 4.12) pour chaque scénario à l'aide des informations fournies par l'annuaire des utilisateurs. Les composants de coordination s'enregistrent dans l'annuaire des composants (Figure 4.12) par le biais de l'identifiant du scénario. Cela permet au composant *CoreComponent* de rechercher un composant de coordination pour contrôler son cycle de vie. Les composants de coordination exposent une interface fonctionnelle fournie (classe *FunctionalInterface*), une ou plusieurs interfaces requises, de type capteur et actionneur, et une interface événementielle requise. Le composant de coordination enregistre son service de coordination fourni (*CoordinationService*) dans l'annuaire des services en fonction de l'identifiant du scénario. Le composant *CoreComponent* recherche le service de coordination fourni par un composant de coordination pour modifier les paramètres du scénario.

Comme nous l'avons dit dans la section 4.4.3.2, un scénario peut être déployé selon deux modes ; centralisé ou décentralisé. Nous allons, pour chacun de ces deux modes, décrire le déploiement du composant de coordination. Nous allons en profiter aussi pour évoquer des exemples de changements et illustrer l'adaptation comportementale des agents à ces changements.

Pour cela, nous revenons au cas d'étude présenté dans le chapitre 3. Le salon contient les équipements suivants : deux radiateurs (*R1* et *R2*), un volet (*V1*), trois lampes (*L1*, *L2* et *L3*) et une horloge (*H1*). Dans cet environnement, nous souhaitons définir le scénario *température agréable le soir dans le salon*. Le scénario est décrit de la manière suivante : "À partir de 19H00, si la température du salon est inférieure à 17 °C, le radiateur *R1* du salon doit s'allumer à la puissance 6, le volet *V1* du salon doit se fermer et les lumières du salon doivent s'allumer." Il est défini par le biais de l'interface de gestion des scénarios (section 3.6.3.1) et il possède les particularités suivantes :

- événement : n'importe quelle horloge du contexte *House* peut fournir l'événement. Les MgrCA sont responsables du choix de l'horloge.
- condition : n'importe quel équipement du type *Radiator* et du contexte *Living-room* peut fournir la température. Les MgrCA ont le choix entre les radiateurs *R1* et *R2* du salon.
- action :
  - le radiateur *R1* du type *Radiator* et du contexte *Living-room* doit s'allumer et l'option *révocable* est sélectionnée. C'est le radiateur *R1* qui doit s'allumer, les MgrCA n'ont pas le choix. Cependant, s'il tombe en panne, il peut être remplacé automatiquement par un autre radiateur du salon (section 3.6.3.1).
  - le volet *V1* de type *Shutter* et du contexte *Living-room* doit se fermer.
  - toutes les lumières de type *Light* et du contexte *Living-room* doivent s'allumer.

**Réalisation du scénario par un seul DCA.** Cette situation peut se produire si un DCA est paramétré de manière à avoir le contrôle des équipements nécessaires pour l'implémentation d'un scénario. Dans notre exemple, cela est possible car le choix de l'horloge qui fournit l'événement est laissé au MedCA, il y a une horloge dans le salon et le DCA a le contrôle de tous les équipements du salon.

Le MgrCA qui gère le DCA lui envoie la description du scénario. Le DCA génère et déploie dynamiquement un composant de coordination qui se connecte aux composants de contrôle qu'il requiert pour fonctionner (connexion des interfaces événement, capteur et actionneur). Le composant de coordination s'enregistre et enregistre le service de contrôle qu'il fournit dans les annuaires des composants et des services, respectivement. Il recherche les services qu'il requiert à partir de l'annuaire des services en précisant le type du service fourni par l'équipement, l'identifiant, le type et le contexte de l'équipement. La figure 4.14 correspond à l'assemblage de composants qui réalise le scénario "*température agréable le soir dans le salon*" au sein d'un DCA.

Une fois l'assemblage réalisé, le composant de coordination attend la notification de l'événement. Il vérifie la satisfaction de la condition et demande alors éventuellement l'exécution des actions. Nous pouvons remarquer que le composant de coordination s'est connecté au composant

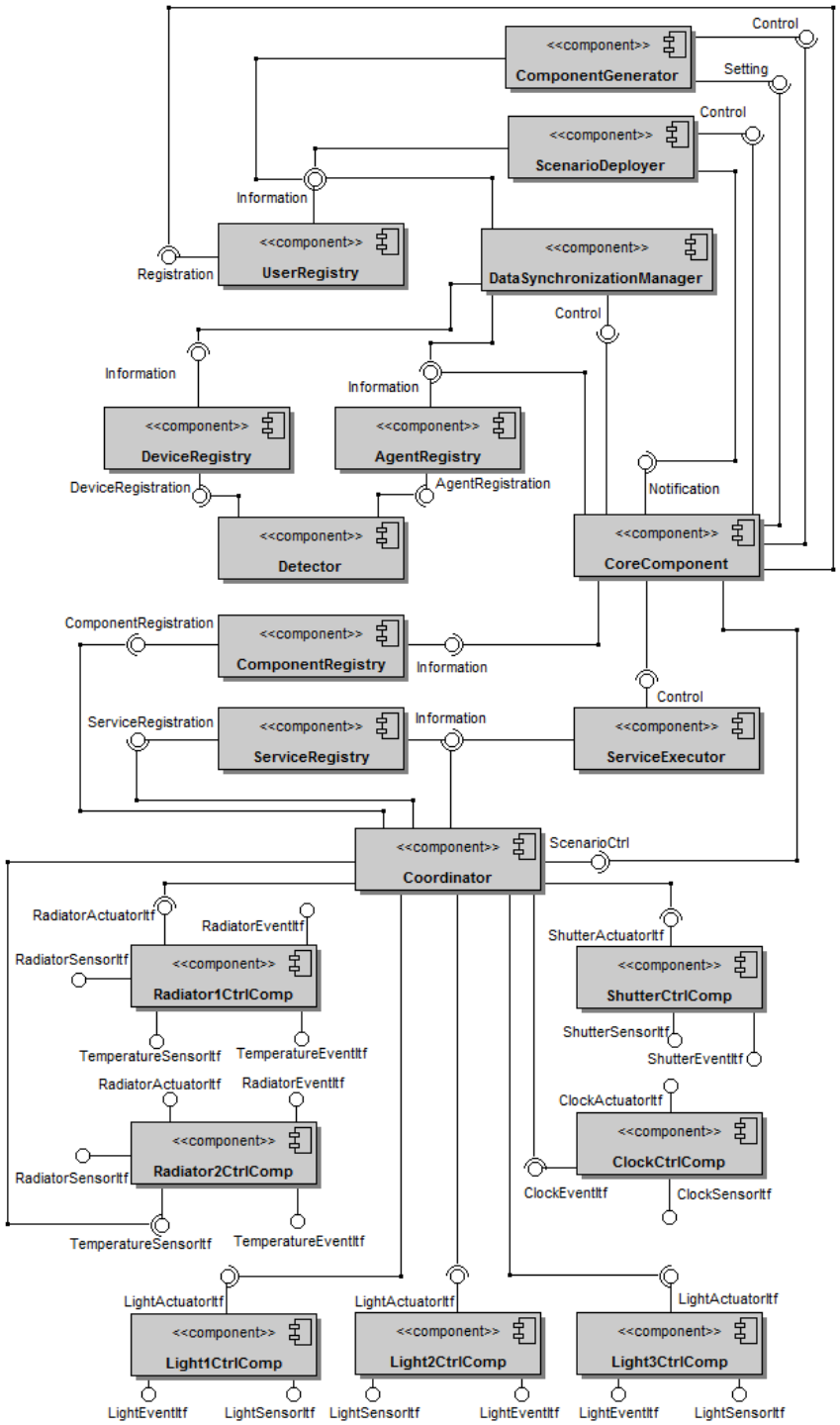



Figure 4.14 – Assemblage de composants implémentant le scénario "*température agréable le soir*" dans l'architecture interne d'un DCA

*Radiator2CtrlComp* pour fournir la valeur de la température et non pas au composant *Radiator1CtrlComp* qui doit exécuter l'allumage du radiateur *R1*. Ce choix aléatoire a été réalisé par le MgrCA puisque l'utilisateur a spécifié que n'importe quel radiateur du salon peut fournir la température. Le composant *Radiator1CtrlComp*, donc le radiateur *R1*, aurait pu également four-

nir la valeur de la température.

**Tolérance aux pannes des équipements.** Lorsque le DCA détecte la panne d'un équipement, il informe son MgrCA des identifiants des scénarios affectés. Prenons comme exemple les situations suivantes :

- Panne du radiateur *R2* : Cela empêche la vérification de la condition. Cependant, lors de la définition du scénario, le choix du radiateur qui fournit la température a été attribué au système et puisqu'il y a dans le salon un autre radiateur (*R1*), le remplacement est possible et le scénario n'est pas suspendu. Dans cette situation, une adaptation de l'assemblage du composant de coordination est suffisante. Le composant de coordination recherche le service *TemperatureEventItf* fourni par le composant *Radiator1CtrlComp*.
- Panne du radiateur *R2* : La solution appliquée dans la situation précédente est possible parce que l'option 'révocable' a été sélectionnée par l'utilisateur. Ainsi, le composant de coordination se connecte au composant *Radiator2CtrlComp* à la place du composant *Radiator2CtrlComp*.
- Panne des deux radiateurs R1 et R2 : Cette situation entraîne la suspension du scénario car il n'y a plus de radiateur dans le salon et la condition ne peut plus être vérifiée.
- Panne du volet ou d'une ou plusieurs lampes : Cela n'entraîne pas la suspension du scénario. Les autres actions sont exécutées. Le composant de contrôle de l'équipement tombé en panne est supprimé. Si l'équipement apparaît à nouveau, le composant de contrôle est généré à nouveau et il enregistre ses services dans l'annuaire des services. Le composant de coordination est averti par l'annuaire des services. Cela permet au composant de coordination de se connecter à nouveau au composant de contrôle.
- Panne de l'horloge : Dans cette situation, le DCA ne peut plus implémenter le scénario tout seul, car il ne contrôle pas les autres horloges de la maison. Un redéploiement distribué du scénario est nécessaire.

**Réalisation du scénario par plusieurs DCA.** Un scénario peut être implémenté par plusieurs DCA. Ces DCA se coordonnent pour l'implémentation du scénario grâce à des échanges de messages. Cette situation peut avoir lieu si les équipements nécessaires à l'implémentation d'un scénario sont contrôlés par différents DCA ou si des DCA sont surchargés (ils ne peuvent plus implémenter les nouveaux scénarios).

Nous distinguons trois fonctions pour les DCA qui implémentent un scénario dans le mode distribué :

- Le DCA qui gère au moins l'événement du scénario. Nous avons choisi de lui attribuer le rôle de coordinateur entre les DCA impliqués car le scénario n'est composé que d'un seul événement et que la clause condition est optionnelle. Nous lui donnons le nom de *DCAcoord*. Pour l'implémentation d'un scénario, il n'y a qu'un seul *DCAcoord*.
- Le DCA qui gère au moins une condition du scénario, mais ne gère pas l'événement. Nous lui donnons le nom de *DCAcond*. Pour l'implémentation d'un scénario, il peut y avoir plusieurs *DCAcond*.

- Le DCA qui est uniquement responsable de l'exécution d'une ou plusieurs actions du scénario, nous lui donnons le nom de *DCAact*.

Seuls les *DCAcoord* et les *DCAcond* génèrent des composants de coordination. Dans le mode décentralisé, les composants de coordination se connectent au composant *CoreComponent* pour l'échange des messages entre les DCA. Les *DCAact* exécutent les actions par le biais du composant *ServiceExecutor*.

La figure 4.15 représente la réalisation du scénario "*température agréable le soir*" par plusieurs DCA. Dans cette configuration, nous considérons que les DCA sont paramétrés de manière à ce que chaque DCA ne contrôle qu'un seul type d'équipement. Par exemple, le *DCAcoord* a généré tous les composants de contrôle nécessaires pour le contrôle de toutes les horloges de la maison.

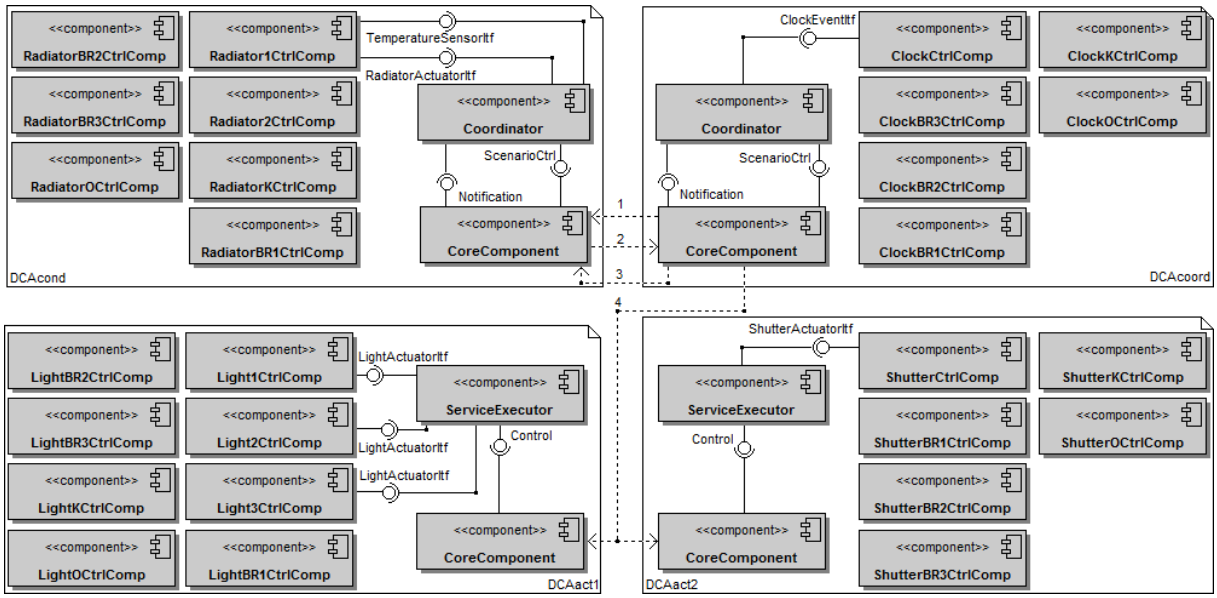



Figure 4.15 – Des parties des architectures internes des DCA qui implémentent le scénario *température agréable le soir*

Le *DCAcoord*, responsable de la vérification de l'événement, génère et déploie un composant de coordination qui se connecte au composant *Clock1CtrlComp* qui lui notifie le changement de l'heure. Le composant de coordination aurait pu se connecter à un autre composant de contrôle d'horloge. Le choix a été réalisé aléatoirement par le médiateur.

Le *DCAcond*, responsable de la vérification de la condition, génère et déploie un composant de coordination qui se connecte au composant *Radiator1CtrlComp* qui lui fournit la température. Comme pour le mode centralisé, le composant de coordination aurait pu se connecter au composant *Radiator2CtrlComp*.

La communication entre les DCA est effectuée par le biais des composants *CoreComponent*. Lorsque l'événement est satisfait, le *DCAcoord* envoie une notification (message 1 de la figure 4.15) au *DCAcond* qui vérifie la condition. Puis, lorsque la condition est satisfaite, le *DCAcond* envoie une notification (message 2 de la figure 4.15) au *DCAcoord* l'informant de la satisfaction de la condition. Le *DCAcoord* vérifie alors la satisfaction de la clause condition qui est composée

dans notre exemple d'une seule condition. Il envoie ensuite une notification (message 3 de la figure 4.15) au *DCAcond* pour l'exécution des actions et des requêtes d'exécution de services (message 4 de la figure 4.15) aux *DCAact1* et *DCAact2*. C'est le composant de coordination du *DCAcond* qui déclenche l'allumage du radiateur et les composants *ServiceExecutor* des *DCAact1* et *DCAact2* qui déclenchent l'allumage des lumières et la fermeture du volet, respectivement. Cependant, si entre temps, l'événement n'est plus satisfait, donc avant la satisfaction de la condition, le *DCAcoord* envoie une notification au *DCAcond*. Le *DCAcond* doit attendre à nouveau la satisfaction de l'événement.

**Tolérance aux pannes des équipements.** Dans cette situation, si l'horloge tombe en panne, le composant de coordination modifie son assemblage dynamiquement et le scénario ne subit aucun changement de déploiement. Par contre, dans cette situation, si le *DCAcoord* tombe en panne, son MgrCA doit prendre sa place directement par la génération du composant de contrôle *Clock1CtrlComp* et du composant de coordination. L'agent *DCAcond* est averti du changement. Lorsque la condition est satisfaite, le *DCAcond* envoie alors le message (2) au MgrCA. Cette modification ne nécessite pas de redéploiement du scénario.

La clause condition d'autres scénarios peut être composée de plusieurs conditions. Ces conditions peuvent être gérées par un *DCAcond* ou par plusieurs. Dans cette situation, le *DCAcoord* et les *DCAcond* se coordonnent jusqu'à la satisfaction de l'événement et de la clause condition. Une fois l'événement et les conditions satisfaits, les actions sont exécutées par les *DCAact* et éventuellement par le *DCAcoord* et les *DCAcond*.

## 4.5 Conclusion

Dans ce chapitre, nous avons présenté les architectures internes et les cycles de vie des différents types d'agents de *SAASHA*. Nous avons aussi présenté les mécanismes d'adaptation au contexte mis en place par le système : l'adaptation de la présentation, l'adaptation de l'organisation et l'adaptation du comportement. Les trois niveaux d'adaptation permettent à notre système de répondre aux qualités requises, adaptation autonomique et dynamique, que nous avons établies dans la section 2.6.

L'adaptation de la présentation permet au système d'adapter les interfaces graphiques et leur contenu en fonction du contexte (du profil de l'utilisateur, des équipements de l'environnement et des agents du système).

Le système adapte son organisation pour gérer avec de bonnes performances la médiation des GUIA et la gestion des DCA, le déploiement des scénarios, les pannes des équipements et des agents. Cette adaptation permet au système d'augmenter sa tolérance aux pannes.

Le système adapte son architecture interne en fonction du contexte et de son paramétrage pour contrôler les équipements de l'environnement et implémenter les scénarios.

Les adaptations ont été réalisées grâce aux annuaires et aux composants. Les annuaires contiennent des données qui reflètent l'état actuel du contexte. Le mécanisme de génération, de déploiement et de désinstallation, d'assemblage et de modification de l'assemblage des composants permet

aux agents d'adapter automatiquement et dynamiquement leur comportement en fonction du contexte.

## 4.6 Limites et perspectives

Nous relevons les limites suivantes :

- **calcul de la charge d'un agent.** Pour l'instant, le calcul de la charge d'un agent est très simple. Nous devons mener des expérimentations qui nous permettent de déduire des méthodes de calcul de charge.
- **redéploiement des scénarios.** Des scénarios sont redéployés lors de la surcharge des DCA. Ce redéploiement permet de diminuer la charge de ces DCA surchargés. Pour l'instant, ce redéploiement est modélisé pour être exécuté aléatoirement.
- **panne pendant la synchronisation.** Si le MedCA tombe en panne pendant la synchronisation des données ou pendant la réception des requêtes, un processus de gestion de communication doit être mis en place pour continuer la synchronisation des données ou l'envoi des requêtes par un autre MedCA à partir du moment où la communication a été coupée. Cela évite de redémarrer la synchronisation des données ou l'envoi des requêtes.

Chapitre 5

# Implémentation de SAASHA

**Sommaire**


**5.1 Introduction . . . . . 83**
**5.2 SAASHA et UPnP . . . . . 84**
5.2.1 UPnP . . . . . 84
5.2.2 Mécanisme d'identification des agents et des équipements . . . . . 87
5.2.3 Protocole de communication . . . . . 87
**5.3 SAASHA et OSGi . . . . . 89**
5.3.1 OSGi . . . . . 90
5.3.2 Mécanisme de génération des composants de contrôle . . . . . 90
5.3.3 Mécanisme de génération des composants de coordination . . . . . 92
**5.4 Expérimentations . . . . . 95**
**5.5 Simulateur : Domus . . . . . 96**
**5.6 Conclusion . . . . . 97**
**5.7 Limites et perspectives . . . . . 98**


## 5.1 Introduction

Nous avons présenté dans le chapitre 3 l'architecture globale de SAASHA composée de deux types d'agents : les agents d'interface et les agents de contrôle. Nous avons présenté également les fonctionnalités explicites qui permettent aux utilisateurs de réaliser leurs besoins et les fonctionnalités implicites gérées par le système par la gestion des changements qui se produisent dans l'environnement et dans le système.

Dans le chapitre 4, nous avons présenté les architectures internes de nos agents qui leur permettent de réaliser les fonctionnalités précédentes. Notre système est basé sur trois adaptations : adaptation de la présentation, de l'organisation et du comportement.

Dans ce chapitre, nous présentons l'implémentation de notre système qui lui permet de gérer son

comportement initial (détection, communication et acquisition de nouveaux comportements). Un prototype de SAASHA, basé sur les standards industriels *UPnP* [10, 9] et *OSGi* [7], a été développé en Java pour évaluer la faisabilité de notre proposition. Dans la suite de ce chapitre, nous présentons comment ces technologies nous ont permis de réaliser un système multi-agents à base de composants s'adaptant automatiquement et dynamiquement en fonction de son environnement, des utilisateurs et des équipements.

## 5.2 SAASHA et UPnP

Dans cette section, nous présentons la technologie UPnP [10, 9] et son utilisation dans le développement de notre système multi-agents. Elle nous a permis de mettre en œuvre un mécanisme d'identification des équipements et des agents et un protocole de communication entre les agents.

### 5.2.1 UPnP

La technologie *Universal Plug and Play* (UPnP) [10, 9] est un protocole réseau qui permet de créer un réseau d'équipements (serveurs) détectables et contrôlables par des *Points de Contrôle* (PC) (clients). UPnP spécifie un ensemble de protocoles basés sur les protocoles standards d'internet pour mettre en place une architecture à service dynamique spécialisé dans la domotique. Elle offre un mécanisme d'envoi de messages qui permet aux PC de détecter la présence, l'insertion, la suppression et la disparition d'un équipement du réseau, de contrôler les équipements et de recevoir des événements émis par les équipements.

L'identification des équipements par les PC, pour les contrôler et recevoir leurs événements, est basée sur des descripteurs XML dont le format est standardisé.
Chaque équipement possède un descripteur qui contient les caractéristiques de l'équipement (marque, modèle, type, nom, ...), les caractéristiques des services qu'il offre (type, nom, ...), les URLs auxquelles les PC doivent envoyer les commandes de contrôle des services et les demandes d'abonnement aux événements.
Les équipements possèdent pour chaque service un descripteur qui contient l'ensemble des opérations, appelées *actions*, qu'il offre et un ensemble de variables d'état, appelées *state variables*. Le changement de la valeur d'une variable d'état est signalé aux PC abonnés par l'envoi d'un événement qui contient la nouvelle valeur de la variable.
La technologie UPnP propose des descripteurs standardisés pour un ensemble de types d'équipements et de services [8] usuels tels qu'un lampe simple, une caméra de sécurité,...
Prenons l'exemple d'un équipement standard HVAC [4] qui fait chauffage, ventilateur et climatiseur à la fois. La figure 5.1 représente le descripteur de cet équipement.

L'équipement fournit cinq services, dont le service *FanSpeed* qui offre six actions qui permettent de spécifier et d'obtenir la direction et la vitesse de ventilation. Ce service émet un événement lorsque la vitesse (la variable d'état *FanSpeedStatus*) et la direction (la variable d'état *DirectionStatus*) de ventilation actuelle change. Le descripteur de ce service est présenté dans la figure 5.2.

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
 ....
<device>
    <deviceType>urn:schemas-upnp-org:device:HVAC_System:1</deviceType>
    <friendlyName>short user-friendly title</friendlyName>
    <manufacturer>manufacturer name</manufacturer>
    <manufacturerURL>URL to manufacturer site</manufacturerURL>
    <modelDescription>long user-friendly title</modelDescription>
    <modelName>model name</modelName>
    <modelNumber>model number</modelNumber>
    <modelURL>URL to model site</modelURL>
    <serialNumber>manufacturer's serial number</serialNumber>
    <UDN>uuid:UUID</UDN>
    <UPC>Universal Product Code</UPC>
    ....
    <serviceList>
        <service>
            <serviceType>urn:schemas-upnp-org:service:HVAC_UserOperatingMode:1</serviceType>
            <serviceId>urn:upnp-org:serviceId:SystemUserMode</serviceId>
            <SCPDURL>URL to service description</SCPDURL>
            <controlURL>URL for control</controlURL>
            <eventSubURL>URL for eventing</eventSubURL>
        </service>
        <service>
            <serviceType>urn:schemas-upnp-org:service:HVAC_FanOperatingMode:1</serviceType>
            ....
        <service>
            <serviceType>urn:schemas-upnp-org:service:FanSpeed:1</serviceType>
            ....
        <service>
            <serviceType>urn:schemas-upnp-org:service:TemperatureSensor:1</serviceType>
            ....
        <service>
            <serviceType>urn:schemas-upnp-org:service:HVAC_SetpointSchedule:1</serviceType>
            ....
     ....
    </serviceList>
 ....
<presentationURL>URL for presentation</presentationURL>
</device>
</root>
```

FIGURE 5.1 – Un extrait du descripteur standard d'un chauffage, ventilateur et climatiseur (HVAC)

Notre système d'agents a été développé à l'aide de cette technologie. Le composant principal de nos agents *CoreCompoment* est implanté comme un équipement UPnP et le composant *Detector* comme un PC. Ce choix d'implantation original fait que les mécanismes de détection et de communication par messages ne sont pas uniquement utilisés entre agents et équipements mais aussi entre les agents eux-mêmes. UPnP facilite la gestion du système multi-agents en fournissant un mécanisme de configuration automatique des ressources de communication des agents et de détection automatique de la présence et de la disparition d'autres agents.

Chaque agent fournit un descripteur qui contient essentiellement son type (agent d'interface ou agent de contrôle), son identifiant et la liste des services qu'il fournit. La figure 5.3 représente le descripteur des GUIA. Chaque GUIA fournit deux services : *DataSynchronization* et *Notification*. Le service *DataSynchronization* permet au MedCa de synchroniser les annuaires du GUIA avec les informations du contexte (section 4.3.3.2). Le service *Notification* permet au MedCA de renvoyer au GUIA le résultat d'exécution d'un service.

La figure 5.4 représente le descripteur des agents de contrôle. Chaque agent de contrôle four-

```
.....
<actionList>
        <action>
                <name>SetFanSpeed</name>
.....
        <action>
                <name>GetFanSpeed</name>
.....
        <action>
                <name>GetFanSpeedTarget</name>
.....
        <action>
                <name>SetFanDirection</name>
.....
        <action>
                <name>GetFanDirection</name>
.....
        <action>
                <name>GetFanDirectionTarget</name>
.....
</actionList>
<serviceStateTable>
        <stateVariable sendEvents="no">
                <name>FanSpeedTarget</name>
.....
        <stateVariable sendEvents="yes">
                <name>FanSpeedStatus</name>
.....
        <stateVariable sendEvents="no">
                <name>DirectionTarget</name>
.....
        <stateVariable sendEvents="yes">
                <name>DirectionStatus</name>
.....
</serviceStateTable>
```

Figure 5.2 – Un extrait du descripteur standard du service *FanSpeed* de l'équipement *HVAC*

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  ....
<device>
    <deviceType>urn:schemas-upnp-org:device:GUIAgent:1</deviceType>
    <friendlyName>GUIA1</friendlyName>
....
    <serviceList>
        <service>
            <serviceType>urn:schemas-upnp-org:service:DataSynchronization:1</serviceType>
....
        </service>
        <service>
            <serviceType>urn:schemas-upnp-org:service:Notification:1</serviceType>
....
        </service>
....
    </serviceList>
....
</device>
</root>
```

Figure 5.3 – Un extrait du descripteur UPnP d'un GUIA

nit huit services. Le service *LifeCycleControl* permet aux administrateurs de contrôler le cycle de vie des agents de contrôle. Les services *ScenarioMgt*, *SystemSettingMgt*, *UserProfilMgt* et *ServiceMgt* permettent au GUIA de transmettre les requêtes des utilisateurs au MedCA. Le service *ScenarioImpl* permet aux DCA de coordonner ensemble pour l'implémentation des scénarios distribués. Le serivce *DataSynchronization* permet à un MgrCA de synchroniser les annuaires d'un MgrCA. Enfin, le service *Notification* permet aux agents de contrôle de signaler les changements qui se produisent dans l'environnement.

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
....
<device>
    <deviceType>urn:schemas-upnp-org:device:ControlAgent:1</deviceType>
    <friendlyName>CA1</friendlyName>
....
    <serviceList>
        <service>
            <serviceType>urn:schemas-upnp-org:service:LifeCycleControl:1</serviceType>
....
        </service>
        <service>
            <serviceType>urn:schemas-upnp-org:service:ScenarioMgt:1</serviceType>
....
        </service>
        <service>
            <serviceType>urn:schemas-upnp-org:service:SystemSettingMgt:1</serviceType>
....
        </service>
        <service>
            <serviceType>urn:schemas-upnp-org:service:UserProfilMgt:1</serviceType>
....
        </service>
        <service>
            <serviceType>urn:schemas-upnp-org:service:ServiceMgt:1</serviceType>
....
        </service>
        <service>
            <serviceType>urn:schemas-upnp-org:service:ScenarioImpl:1</serviceType>
....
        </service>
        <service>
            <serviceType>urn:schemas-upnp-org:service:DataSynchronization:1</serviceType>
....
        </service>
        <service>
            <serviceType>urn:schemas-upnp-org:service:Notification:1</serviceType>
....
        </service>
....
    </serviceList>
....
</device>
</root>
```

FIGURE 5.4 – Un extrait du descripteur UPnP d'un agent de contrôle

### 5.2.2 Mécanisme d'identification des agents et des équipements

Le mécanisme d'identification permet aux agents du système, lors de leur intégration dans un environnement, de détecter la présence des agents et des équipements disponibles et puis d'être informer de l'apparition et de la disparition d'un agent ou d'un équipement. Il permet aussi aux agents d'identifier les équipements (contexte, type et identifiant) et les services qu'ils fournissent (types, opérations et événements) et d'identifier les agents (type et rôle). L'identification est réalisée grâce aux descripteurs des équipements et des agents.

### 5.2.3 Protocole de communication

En tant qu'équipements UPnP, les agents possèdent un ensemble de descripteurs décrivant les services qu'ils fournissent aux autres équipements UPnP. Ces services permettent aux agents de communiquer ensemble par l'envoi de messages. Les agents communiquent (contrôlent) avec les équipements par le biais du même mécanisme. Les services fournis par les agents sont im-

plantés par le composant *CoreComponent* de chaque agent. Les composants *CoreComponent* communiquent ensemble en mode asynchrone tandis que les composants au sein des agents communiquent ensemble en mode synchrone par appel de méthodes à travers les interfaces fournies et requises. Sur les figures qui suivent, nous modélisons conceptuellement les communications entre agents à l'aide de connexions reliant les interfaces requises et fournies de leurs composants *CoreComponent*. Ces connexions correspondent aux envois de messages réalisés par les agents à l'aide d'UPnP. La figure 5.5 représente les connexions d'interfaces entre les composants *CoreComponent*. Cette figure ne contient pas toutes les connexions.

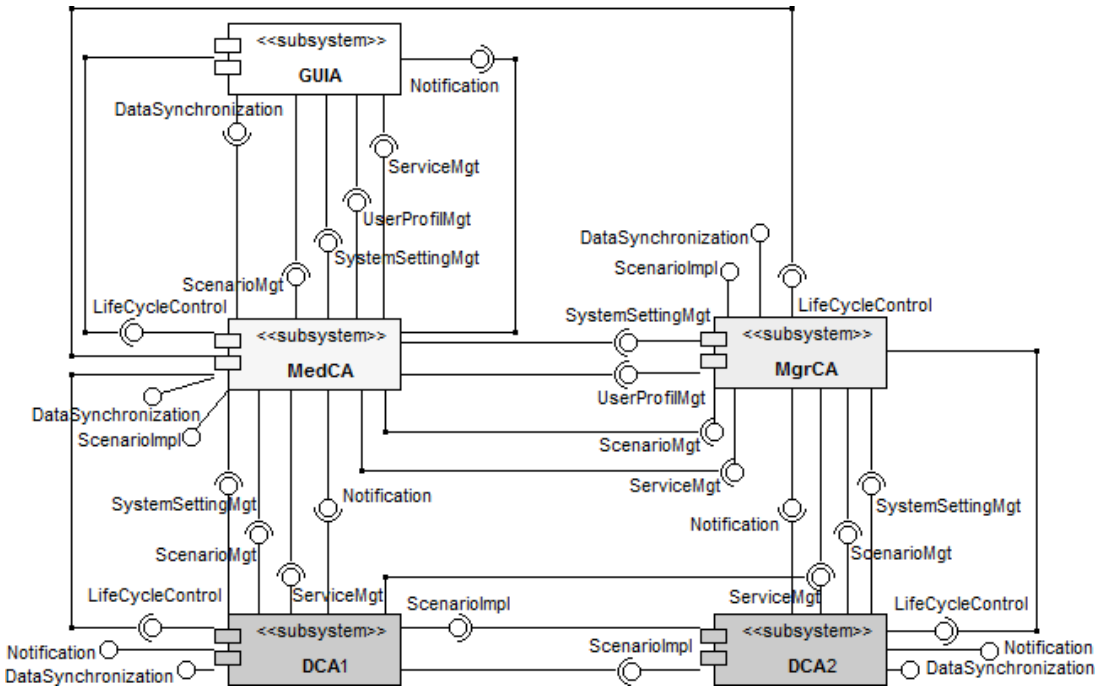



Figure 5.5 – Le protocole de communication dans SAASHA

Les GUIA fournissent un service *DataSynchronization* qui permet aux médiateurs de synchroniser les données de leurs annuaires. Les agents de contrôle fournissent les services suivants :

- un service *LifeCycleControl* pour le contrôle de leur cycle de vie (voir le chapitre 4).
- un service *SystemSettingMgt* pour la transmission du paramétrage des agents de contrôle. Il permet au GUIA de transmettre aux agents de contrôle leur rôle, les paramètres du champ de gestion des MgrCA, les paramètres du champ de contrôle des DCA. Ce service permet au MedCA de diffuser ces informations aux MgrCA et il permet à chaque MgrCA de déclarer aux DCA qu'il gère qu'il est leur manageur et puis de leur transmettre le paramétrage de leur champ de contrôle. Enfin, il permet au MedCA d'envoyer une requête pour connaître les DCA qui contrôlent une liste d'équipements.
- un service *UserProfilMgt* pour la transmission du paramétrage des profils utilisateurs. Il permet au GUIA de vérifier la validité des identifiants entrés par un utilisateur et de transmettre au médiateur la définition, la modification ou la suppression du profil d'un utilisateur. Il permet au MedCA de diffuser ces informations aux MgrCA.

- un service *ServiceMgt* pour la transmission des requêtes d'exécution de services. Il permet au GUIA de transmettre au MedCA les requêtes d'exécution de services. Il permet au MedCA de diffuser ces informations aux MgrCA, permet à chaque MgrCA de transmettre les requêtes d'exécution des services aux DCA qu'il gère et enfin, permet aux DCA responsables de la gestion de l'implémentation des scénarios d'envoyer aux DCA responsables de l'exécution des actions les requêtes d'exécution des services.

- un service *ScenarioMgt* pour la transmission du paramétrage des scénarios. Il permet au GUIA de transmettre au MedCA la définition, la modification, la suppression ou la suspension des scénarios. Il permet au MedCA de diffuser ces informations aux MgrCA et puis aux MgrCA de les transmettre aux DCA qu'ils gèrent.

- un service *ScenarioImpl* pour l'échange des messages de notification entre les DCA pour l'implémentation des scénarios.

- un service *DataSynchronization* qui permet à un MgrCA de synchroniser les données des annuaires des autres MgrCA.

- un service *Notification* pour la transmission des réponses et des changements qui se produisent dans l'environnement et le système. Le MedCA renvoie au GUIA le résultat de l'identification d'un utilisateur. Les MgrCA renvoient au MedCA, pour une liste d'équipements, une liste de DCA qui permettent de les contrôler. Il permet aux agents de s'échanger le(s) résultat(s) d'exécution d'un service, le statut des scénarios (suspendus ou réactivés) et les changements qui se produisent dans l'environnement et dans le système.

Certaines communications nécessitent l'envoi d'un ensemble de messages organisés en *conversations* [3]. Par exemple, la définition d'un scénario nécessite l'envoi : d'un message pour l'événement et d'un message pour chaque condition et action. Une conversation est donc réalisée entre deux agents A et B. Elle commence par une demande d'ouverture de conversation envoyée par l'agent A et elle se termine par une déclaration de fermeture de conversation envoyée par l'agent A. Cependant, une conversation ne peut commencer sans l'accord de l'agent B. Nous définissons quatre types de conversations pour la gestion des profils utilisateurs, pour la gestion des scénarios, pour la gestion des scénarios simples et pour le paramétrage des DCA. Pour l'instant, les conversations servent à gérer les protocoles d'échange de messages. Ces conversations pourraient également permettre aux agents de gérer des conversations prioritaires en suspendant d'autres conversations.

## 5.3 SAASHA et OSGi

Dans cette section, nous présentons la technologie OSGi [7] et son utilisation dans la gestion des architectures internes de nos agents et du mécanisme de génération des composants de contrôle et de coordination.

### 5.3.1 OSGi

L'Open Services Gateway initiative (OSGi) [7, 27] est un groupe industriel organisé en société à but non lucratif fondé en mai 1999. Il vise à définir et promouvoir des spécifications ouvertes d'une plateforme à service. La plateforme OSGi fournit tous les mécanismes pour créer des applications à service dynamique. OSGi est un environnement standardisé utilisé par l'industrie pour administrer dynamiquement l'installation de modules Java dans un environnement d'exécution.

Ce standard propose un modèle de déploiement de *bundles* gérant leur interdépendance éventuelle grâce à des descripteurs définissant les services qu'ils requièrent et fournissent. Cette caractéristique nous permet d'utiliser les *bundles* OSGi comme une implantation des composants logiciels qui composent l'architecture interne des agents. Les composants logiciels peuvent être installés, mis à jour ou supprimés à l'exécution indépendamment des autres composants. Les composants peuvent dynamiquement découvrir et utiliser d'autres composants grâce à un annuaire de services intégré à toute plateforme OSGi. L'annuaire permet aux composants de publier les services qu'ils fournissent ou de rechercher des services fournis par d'autres composants, et de s'abonner pour être informés de l'insertion, de la suppression ou de la mise à jour d'un service. Cet annuaire correspond à l'annuaire des services présents dans l'architecture interne des DCA.

Les mécanismes d'OSGi sont utilisés pour construire automatiquement les assemblages de composants formant l'architecture interne des agents. L'annuaire est utilisé pour gérer les connexions entre les composants qui fournissent (interface fournie) et les composants qui utilisent (interface requise) un service. Les classes des composants de contrôle et de coordination sont générées, compilées et encapsulées, à la volée, dans des bundles OSGi. Les mécanismes d'OSGi sont alors utilisés pour ajouter le composant à l'architecture interne de l'agent.

Nous présentons, dans les sections suivantes, les mécanismes qui permettent aux agents de contrôle d'adapter leur comportement de contrôle et de coordination.

### 5.3.2 Mécanisme de génération des composants de contrôle

Ce mécanisme permet aux DCA de générer et de déployer des composants de contrôle. Les DCA génèrent, en fonction de leur champ de contrôle, des composants de contrôle pour contrôler les équipements.
La génération des composants de contrôle est basée sur les informations extraites des descripteurs des équipements et des services. Ces informations sont enregistrées dans l'annuaire des équipements. Lorsque le composant *Detector* détecte un équipement, il analyse les descripteurs des équipements et des services pour extraire les informations qu'il enregistre dans l'annuaire des équipements. Lors de cette extraction, il doit classifier les opérations dans les catégories actionneur ou capteur. Cette classification est réalisée automatiquement selon les six règles suivantes :

- Les actions dont le nom commence par le préfixe *set* sont considérées de type actionneur.
- Les actions dont le nom commence par un des préfixes *get*, *browse*, *list* ou *read* sont considérées de type capteur.

- Les actions qui ont des paramètres en entrée et en sortie sont considérées de type actionneur.
- Les actions qui n'ont pas de paramètres en sortie sont considérées de type actionneur.
- Les actions qui n'ont pas de paramètres en entrée sont considérées de type capteur.
- Les actions qui n'ont pas de paramètres sont considérées de type actionneur.

La classification précédente est basée sur une analyse des 496 opérations offertes par les 47 services fournis par les 16 équipements standards proposés par le consortium UPnP [8]. Le tableau 5.1 résume cette analyse.

| **opérations de type actionneur** | | | | **opérations de type capteur** | |
|---|---|---|---|---|---|
| préfixe *set* | paramètres en entrée et sortie | paramètres en entrée | sans paramètres | préfixe *get* | paramètres en sortie |
| 116 | 24 | 37 | 3 | 256 | 60 |

TABLE 5.1 – Données sur les opérations de services standards

Le service FanSpeed fourni par l'équipement standard *HVAC* et décrit dans la section 5.2.1, est utilisé pour expliquer la génération automatique du code d'un composant de contrôle. La partie gauche de la figure 5.6 contient une partie du descripteur du service FanSpeed.
À partir de ce descripteur, nous générons trois interfaces : une interface de type capteur, une interface de type actionneur et une interface événementielle. Ces interfaces sont représentées dans la partie droite de la figure 5.6.

La génération de la signature d'une opération est réalisée de la manière suivante. Dans le descripteur, chaque opération est marquée par une balise *action* qui contient le nom de l'opération et la liste des paramètres (*argumentList*). Chaque paramètre est marqué par la balise *argument* qui contient le nom du paramètre, sa direction (entrée *in* ou sortie *out*) et une balise *relatedStateVariable* qui pointe vers la variable d'état associée au paramètre. La variable d'état est représentée par la balise *stateVariable* qui contient principalement le type du paramètre et qui indique si le changement de la valeur du paramètre déclenchera l'envoi d'un événement. L'envoi d'un événement est indiqué par l'attribut *sendEvents*. L'interface événementielle est générée à l'aide des variables d'état. Elle contient une seule opération qui permet aux composants de coordination de s'abonner aux événements émis par le service. Le composant de coordination doit spécifier le nom de l'événement (variable d'état).

Un extrait du code qui implémente les interfaces du composant de contrôle est présenté dans la figure 5.7.
Nous distinguons les parties suivantes :

- **Enregistrement des services fournis par le composant de contrôle.** Tous les composants de contrôle enregistrent les services qu'ils fournissent dans l'annuaire des services du DCA. Cela permet aux composants de coordination de rechercher dynamiquement les services qu'ils requièrent.
- **Instanciation du point de contrôle.** Cela permet au composant de contrôle d'accéder à l'équipement et aux services qu'il fournit et de s'abonner aux événements qu'il émet.

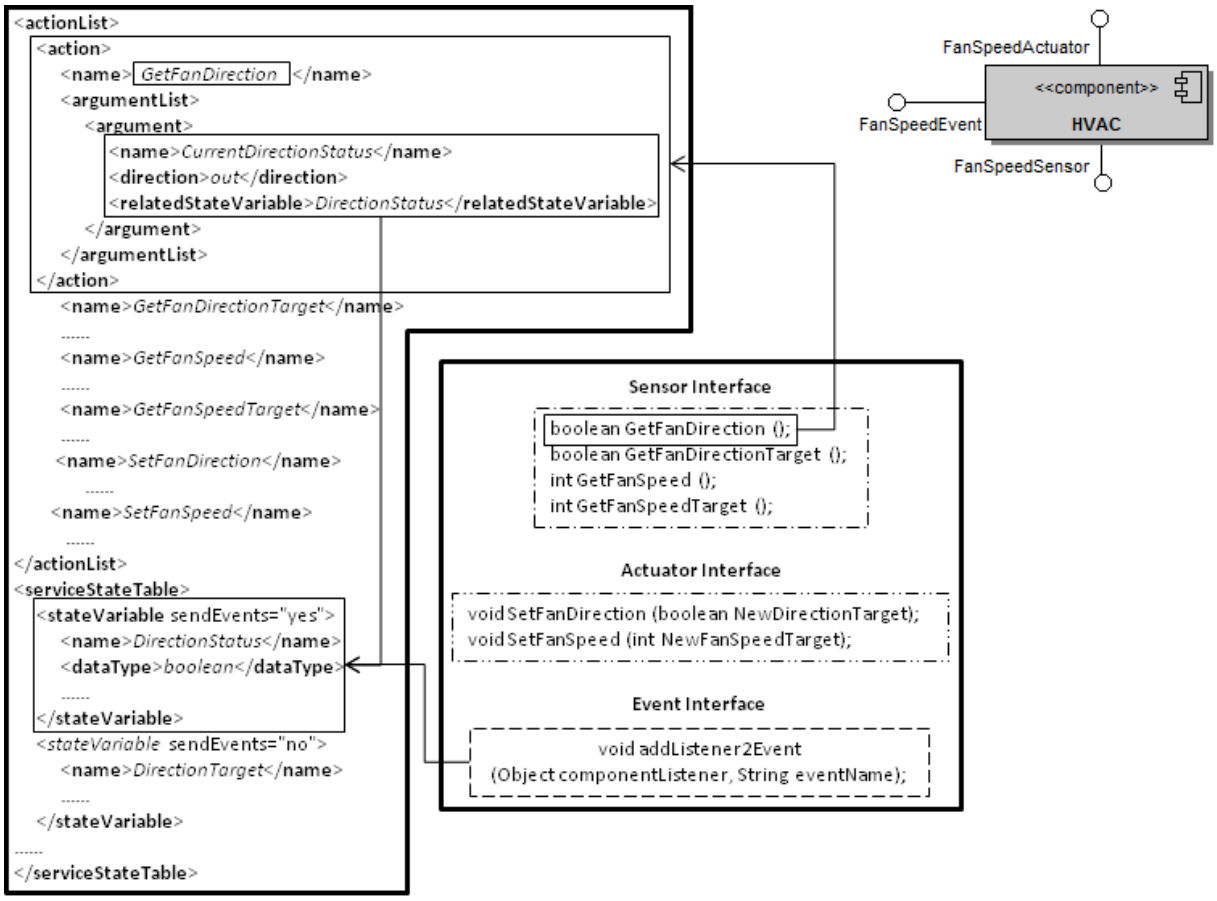



Figure 5.6 – Une partie du code généré qui représente les interfaces fournies par le composant de contrôle de l'équipement HVAC

- **Implémentation des événements.** Cette partie permet aux composants de coordination de s'abonner aux événements émis par l'équipement. Le composant de contrôle transmet les événements aux composants de coordination abonnés.
- **Implémentation des services.** Cette partie correspond aux invocations des services fournis par l'équipement et représente l'implémentation des services fournis par le composant de contrôle.

### 5.3.3 Mécanisme de génération des composants de coordination

Ce mécanisme permet aux agents de contrôle de générer et de déployer des composants de coordination pour l'implémentation des scénarios. Le processus de génération des composants de coordination est réalisé en deux étapes :

- la génération de la méthode d'activation qui recherche les services requis dans l'annuaire des services et réalise les assemblages nécessaires avec les composants qui fournissent les services et la souscription à l'événement nécessaire.

```
public void start(BundleContext context){
        ....
        //enregistrement des services fournis par le composant dans l'annuaire des services
        Properties props = new Properties();
        props.put("BundleName", "HVACCtrlComp");
        props.put("DeviceType", "HVAC_System");
        props.put("Context", "LivingRoom");
        ....
        context.registerService(FanSpeedSensorService.class.getName(), this, props);
        context.registerService(FanSpeedActuatorService.class.getName(), this, props);
        ....
        //accès à l'équipement appelé "DeviceFriendlyName"
        //et aux services fournis par l'équipement et à leurs actions
        //et abonnement aux événements
        ControlPoint ctrlPoint = new ControlPoint();
        ctrlPoint.addEventListener(this);

        Device HVACDevice = ctrlPoint.getDevice(DeviceFriendlyName);

        Service service = HVACDevice.getService("FanSpeed");
        ctrlPoint.subscribe(service);
        Action GetFanDirectionAction = HVACDevice.getAction("GetFanDirection");
        GetFanDirectionAction.setActionListener(this);
        Action SetFanDirectionAction = HVACDevice.getAction("SetFanDirection");
        SetFanDirectionAction.setActionListener(this);
        ....
}

public void addListener2Event (Object componentListener, String eventName){
        //ajout d'un composant de coordination à la liste des abonnés à l'événement
}

public void eventNotifyReceived (String eventName, String value){
        //envoie de l'événement aux abonnés
}

//liste des opérations pour l'invocation des actions offertes par les services de l'équipement

public boolean GetFanDirection (){
        if(GetFanDirectionAction.postControlAction() == true){
                Argument argument = GetFanDirectionAction.getOutputArgumentList().
                        getArgument("CurrentDirectionStatus");
                return new Boolean (argument.getValue()).booleanValue();

        }
}

public void SetFanDirection (boolean NewDirectionTarget){
        String value = new Boolean(NewDirectionTarget).toString();
        SetFanDirectionAction.setArgumentValue("NewDirectionTarget", value);
        SetFanDirectionAction.postControlAction();
}
....
```

FIGURE 5.7 – Un extrait du code généré qui représente l'implémentation des services fournis par le composant de contrôle de l'équipement HVAC

- l'implémentation du scénario qui utilise les interfaces requises du composant de coordination connectées aux interfaces fournies par les composants de contrôle.

Dans le cas d'un scénario distribué, les composants de coordination doivent se connecter aux composants *CoreComponent* pour s'échanger les notifications de satisfaction de l'événement et

des conditions et pour l'envoi des requêtes pour l'exécution des actions. Un extrait du code de la méthode d'activation du composant de coordination qui implémente le scénario *température agréable le soir* est présenté dans la figure 5.8. Le scénario est implémenté en mode centralisé.

```
Public void start(BundleContext context){
        ....
        //enregistrement du service fourni par le composant dans l'annuaire des services
        Properties props = new Properties();
        props.put("ScenarioId", "TemperatureAgreableSoir");
        context.registerService(ScenarioCtrl.class.getName(), this, props);
        ....
        //opération pour l'abonnement à l'événement
        Method addListener;
        //opération de type capteur pour la condition
        Method getTemperature;
        //opération de type actionneur pour les actions
        Method turnOnRadiator;
        Method turnOnLight1;
        Method turnOnLight2;
        Method turnOnLight3;
        Method closeShutter;
        ....
        //recherche des services fournis par les composants de contrôle
        //et par le composant CoreComponent dans l'annuaire des services
        //pour l'abonnement à l'événement et invocation de services
        ServiceReference[] ref = context.getServiceReferences("clockctrlcomp.ClockEventItf",
                "(BundleName=ClockCtrlComp)", "(DeviceType=Clock)", "(Context=Salon)");
        Object clockService = context.getService(ref[0]);
        Method[] methods = clockService.getClass().getMethods();
        for(int i = 0; i < methods.length; i++){
                if(methods[i].getName().equals("addListener2Event"))
                        addListener = methods[i];
        }
        ....
        //abonnement à l'événement
        addListener.invoke(clockService, this, "Time");
        ....
}//end start

//déclenchement par l'événement de la vérification du scénario
public void eventNotification(String value){
        ....
        if(value=="19:00:00"){
                int temperature =
                new Interger(getTemperature.invoke(radiatorService).toString()).intValue;
                //vérification de la condition
                if(temperature < 17){
                //exécution des actions
                turnOnLight1.invoke(light1service);
                ....
                }
        }
}//end eventNotification
```

Figure 5.8 – Un extrait du code généré pour le composant de coodination qui implémente le scénario *température agréable le soir*

Nous distinguons les principales parties suivantes :

- **Enregistrement du service fourni par le composant de coordination.** Les composants de coordination fournissent un service qui permet au composant *CoreComponent* de contrôler le cycle de vie du composant de coordination. Dans l'implémentation distribuée

des scénarios, ce service permet au composant *CoreComponent* de transmettre au composant de coordination les messages de notification envoyés par les autre composants de coordination qui implémentent le même scénario.

- **Recherche des services fournis par les composants de contrôle et par le composant *CoreComponent*.** Ce processus permet au composant de coordination de se connecter aux composants de contrôle nécessaires pour l'implémentation du scénario. Il lui permet aussi, dans l'implémentation distribuée d'un scénario, de se connecter au composant *CoreComponent* pour envoyer des messages de notification aux autres composants de coordination qui implémentent le même scénario. Le composant de coordination recherche les services fournis par les composants de contrôle en fonction du contexte, du type et de l'identifiant de l'équipement.
- **Abonnement à l'événement fourni par un composant de contrôle.** Le composant de coordination s'abonne auprès du composant de contrôle qui fournit l'événement décrit dans le scénario.
- **Implémentation du scénario.** Lorsque le composant de coordination reçoit l'événement, il déclenche la vérification de l'événement.
  - Dans le mode centralisé, il vérifie ensuite la condition si l'événement est satisfait et enfin, si la condition est satisfaite, il déclenche l'exécution des actions définies dans le scénario.
  - Dans le mode décentralisé, il envoie, par le biais des composants *CoreComponent*, aux composants de coordination une notification les informant que l'événement est satisfait. Lorsque les conditions sont satisfaites, les composants de coordination lui renvoient un message de notification l'informant que les conditions sont satisfaites. À ce moment, il déclenche l'exécution des actions localement et/ou en envoyant des messages de notification aux composants *CoreComponent* des DCA responsables de l'exécution des actions.

## 5.4 Expérimentations

Un prototype de SAASHA a été réalisé et déployé sur un ensemble de conteneurs OSGi Felix [1]. Chacun de ces conteneurs contient un seul agent. Le prototype de SAASHA met en œuvre un GUIA et un ensemble de DCA contrôlant différents équipements virtuels (fournis par le projet CyberGarage [2]. Une variété de tests correspondant à une variété de compositions de DCA a été réalisée avec succès pour l'implémentation du scénario *température agréable le soir* décrit dans le chapitre 4 :

1. 1 GUIA and 1 DCA. Dans ce cas, le DCA contrôle tous les équipements et il implémente tout seul les scénarios.
2. 1 GUIA and 2 DCA. Dans ce cas, nous avons testé deux situations. (1) le DCAcoord informe le DCAcond de l'occurrence de l'événement. Le DCAcond est responsable de la vérification de la condition et de l'exécution des actions. (2) L'événement et la condition sont gérés par le DCAcoord qui envoie des requêtes au DCAact pour exécuter les actions.

3. 1 GUIA and 4 DCAs. Dans ce cas, nous avons testé trois situations. (1) Le DCAcoord gère uniquement l'événement et la vérification de la condition est gérée par un DCAcond. Les deux DCA coordonnent ensemble jusqu'à la satisfaction de l'événement et de la condition. Puis, le DCAcoord envoie une notification au DCAcond et des requêtes aux deux autres DCA pour l'exécution des actions. (2) L'événement et l'exécution sont gérés par le DCAcoord qui envoient des requêtes aux autres DCA pour l'exécution des actions. (3) Nous rajoutons une condition. L'événement est géré par un DCAcoord et les vérifications de chacune des conditions sont gérées par un DCAcond. Lorsque l'événement est satisfait, le DCAcoord informe les DCAcond qui l'informent en retour lors de la satisfaction des conditions. Une fois les deux conditions satisfaites, le DCAcoord envoie une notification aux DCAcond et une requête au dernier DCA pour l'exécution des actions.

Les situations testées confirment le bon fonctionnement de l'architecture choisie et nous ont permis de vérifier les aspects suivants :

- La détection et l'identification des équipements et des agents disponibles et l'enregistrement dans les annuaires des équipements et des agents, respectivement.
- La communication entre les agents du système pour la synchronisation des données du GUIA, la transmission des scénarios et l'implémentation des scénarios.
- Le contrôle des équipements de l'environnement pour l'exécution des services décrits dans les scénarios.
- L'adaptation du contenu des interfaces graphiques en fonction des agents et des équipements disponibles.
- L'adaptation du comportement de contrôle des DCA (génération et déploiement des composants de contrôle) en fonction de leur champ de contrôle.
- L'adaptation du comportement de coordination des DCA (en mode centralisé ou distribué) (génération et déploiement des composants de coordination et l'assemblage de ces composants avec les composants de contrôle).
- La détection de la disparition des agents et des équipements et la mise à jour du contenu des interface graphiques et des annuaires des agents et des équipements, respectivement.

Les autres formes d'adaptation n'ont pas encore pu être testées avec le prototype développé. La gestion des rôles MgrCA et MedCA reste également à intégrer à l'architecture des agents de contrôle.

## 5.5 Simulateur : Domus

La figure 5.9 représente Domus un simulateur que nous avons développé pour expérimenter les interactions entre un environnement domotique virtuel et SAASHA. Domus est une interface graphique interactive développée en JavaFX qui fournit une vue réaliste d'une maison et d'équipements UPnP virtuels installés (dans cette figure, une lampe, un système d'air conditionné, machine à laver et volets) et permet d'agir sur les équipements par le biais de leurs vues.

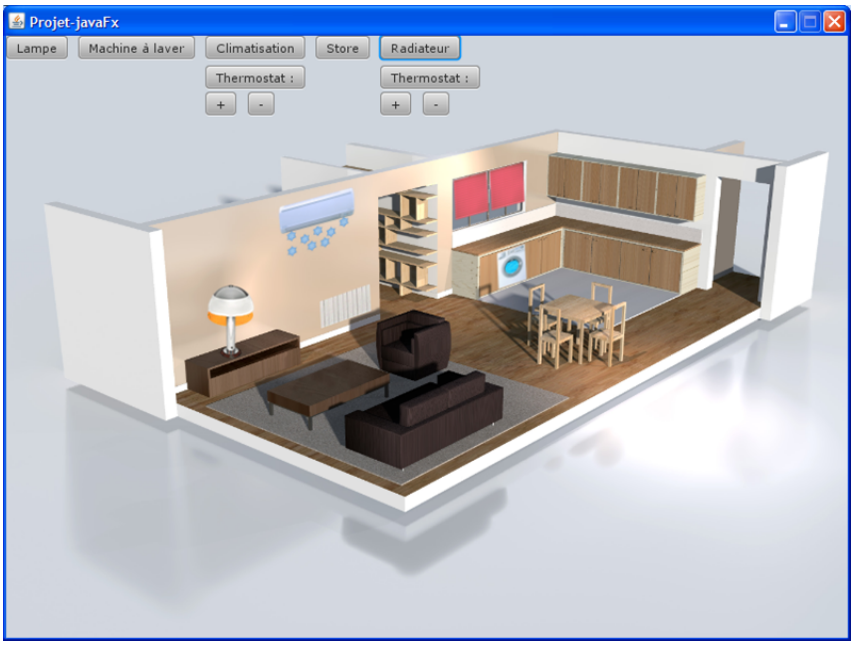



Figure 5.9 – Domus : simulateur d'une maison intelligente

## 5.6 Conclusion

Nous avons présenté dans ce chapitre les technologies UPnP et OSGi sur lesquelles l'architecture globale de notre système et les architectures internes de nos agents sont basées. La combinaison des deux technologies UPnP et OSGi nous a permis de développer SAASHA, un système d'agents à base de composants adaptable automatiquement et dynamiquement. Cela a permis à SAASHA de répondre aux qualités requises que nous avons établies dans la section 2.6 et aux fonctionnalités explicites et implicites décrites dans la section 2.5.

Grâce à UPnP, les agents détectent les autres agents du système et les équipements disponibles dans l'environnement. Ils peuvent aussi détecter la panne des agents et des équipements. Les agents communiquent ensemble par l'échange de messages. L'utilisation de UPnP a permis a SAASHA de répondre à la qualité requise *sensibilité au contexte*.

Grâce à OSGi, les agents adaptent dynamiquement leur comportement par la génération, le déploiement et l'assemblage de composants. Ils génèrent les composants de contrôle d'équipements à l'aide des descripteurs des équipements et les composants de coordination à l'aide des descriptions des scénarios. Cette adaptation automatique du comportement permet au système d'avoir une architecture flexible. En effet, les agents de contrôle peuvent changer de rôle ou être paramétrer à tout moment. L'adaptation des agents n'est pas basée sur des composants *sur étagères*. Les agents génèrent les composants de contrôle en fonction du contexte. L'utilisation de OSGi a permis à SAASHA de répondre aux qualités requises *adaptation autonomique* et *adaptation dynamique*.

## 5.7 Limites et perspectives

Certaines actions fournies par les équipements ne sont pas destinées aux utilisateurs, mais elles sont utilisées pour des raisons d'administration de l'équipement ou des raisons techniques liées à l'équipement. En plus, les noms des actions ne sont pas tous compréhensibles par les utilisateurs (abréviation, autre langue, etc). Nous envisageons d'intégrer au système des descripteurs orientés utilisateurs qui fournissent des informations supplémentaires sur les services des équipements comme les noms des actions et des paramètres orientés utilisateurs et le type des actions (capteur ou actionneur) qui empêche de commettre des erreurs de classification. Ces informations sont fournies en plusieurs langues.

Les tests ont été réalisés sur un petit nombre d'équipements et d'agents. Il n'a pas été possible de définir une grande variété de scénarios. Nous envisageons de réaliser des tests sur un plus grand nombre d'équipements et de scénarios pour valider le passage à l'échelle du système. Dans un premier temps, ces tests seront réalisés par le biais de Domus. À terme, nous envisageons de réaliser des tests dans un environnement domotique réel. Pour l'instant, nous n'avons pas développé de MedCA et de MgrCA indépendants. Ce sont les GUIA qui gèrent leur rôle.

Chapitre 6

# Conclusion

L'objectif d'un système domotique est la réalisation des besoins des utilisateurs. Pour cela, le système contrôle les équipements et coordonne entre les équipements pour l'exécution de services et la réalisation de scénarios. Cependant, les environnements domotiques sont des environnements caractérisés par la variabilité de leurs contextes et par les changements qui s'y produisent.

L'objectif de la thèse est de modéliser un système auto-adaptable en fonction de son contexte.

## 6.1 Contributions

Les contributions de cette thèse sont les suivantes :

**Chapitre 2.** Ce chapitre présente un état de l'art centré sur les systèmes domotiques existants afin d'en identifier les avantages et les limites par rapport à nos objectifs. Cet état de l'art étudie ainsi deux facettes : les fonctionnalités offertes par les systèmes aux utilisateurs et les différents styles d'architecture de ces systèmes. Nous terminons ce chapitre en proposant une synthèse des qualités nécessaires aux systèmes domotiques.

**Originalité de notre contribution.** La modélisation d'un *système multi-agents à base de composants* [42, 43, 44] dédié aux environnements domotiques nous a permis d'obtenir un système *auto-adaptable* en fonction du *contexte* dans lequel il est intégré et des *changements* qui se produisent dans ce contexte. Cette adaptation est réalisée *dynamiquement* et sans connaissance a priori du contexte. Les agents interagissent avec les *utilisateurs* et contrôlent les *équipements* de l'environnement. Les utilisateurs peuvent définir des *scénarios personnalisés* en fonction de leur *profil* et du contexte. Le *comportement* des agents est basé sur un *assemblage initial de composants* et *l'adaptation du comportement* des agents est basée sur la *génération, le déploiement et l'assemblage des composants, la modification des assemblages et la désinstallation des composants*. Les agents adaptent leur comportement pour le contrôle des équipements et l'implémentation des scénarios.

**Chapitre 3.** Ce chapitre présente *l'architecture globale* de notre système multi-agents à base

de composants et les *fonctionnalités* offertes par le système. Notre système fait intervenir deux types d'agents : les *agents d'interface* qui interagissent avec les utilisateurs et les *agents de contrôle* responsables de la découverte et du contrôle des équipements de l'environnement. Le système gère trois types d'*adaptation* :

- l'adaptation de la *présentation* qui permet à l'interface de refléter en permanence l'état actuel de l'environnement et du système
- l'adaptation de l'*organisation* qui permet au système de gérer son architecture en fonction des changements qui se produisent dans l'environnement et le système
- l'adaptation du *comportement* qui permet aux agents de modifier leur architecture interne pour contrôler les équipements et pour implémenter les scénarios

Les agents d'interface sont responsables de la gestion de l'adaptation de la présentation. Ils adaptent les interfaces et leur contenu en fonction de l'environnement (utilisateurs et équipements) et du système (agents).
Nous définissons trois rôles pour les agents de contrôle :

- Les *agents de contrôle d'équipements* qui sont responsables de la réalisation de l'adaptation du comportement du système. Ils adaptent leur architecture interne pour le contrôle des équipements et l'implémentation des scénarios.
- Les *agents de contrôle manageurs* qui sont responsables de la réalisation de l'adaptation de l'organisation du système. Ils adaptent l'architecture du système en fonction de la disponibilité des équipements et des agents.
- Les *agents de contrôle médiateurs* qui sont responsables de la réalisation de l'adaptation de l'organisation du système. Ils adaptent l'architecture du système en fonction des besoins définis par les utilisateurs.

Le système propose aux utilisateurs quatre fonctionnalités pour gérer eux-mêmes leurs besoins :

- *l'exécution de services* fournis par les équipements (comportement),
- *la gestion des scénarios* (comportement),
- *la gestion des profils des utilisateurs* (présentation),
- *la gestion du paramétrage du système* (organisation et comportement).

Le système gère automatiquement deux fonctionnalités :

- la *gestion de l'environnement* (ajout ou retrait/panne des équipements)
- La *gestion du système* (ajout ou retrait/panne des agents)

**Chapitre 4.** Ce chapitre décrit les *architectures internes* de nos agents et la réalisation des trois niveaux d'adaptation qui lui permettent de gérer automatiquement l'environnement et le système et les changements qui s'y produisent. Les architectures internes de nos agents sont basées sur les *composants*. Le comportement initial de chaque type d'agent est réalisé par le biais d'un assemblage initial de composants. Les agents de contrôle d'équipements contrôlent les équipements par le biais des *composants de contrôle* et exécutent les scénarios par le biais des *composants de coordination*.
L'adaptation de la présentation est réalisée à l'aide des *annuaires* des agents d'interface qui adaptent le contenu des interfaces en fonction des données des annuaires.
L'adaptation du comportement est réalisée par le biais d'un mécanisme d'adaptation qui est basé sur la génération, le déploiement et l'assemblage des composants de contrôle et de coordination.
L'adaptation de l'organisation est réalisée, en fonction des besoins définis par les utilisateurs, à l'aide des annuaires et du mécanisme d'adaptation.

**Chapitre 5.** Ce chapitre décrit notre implémentation du système multi-agents basée sur *UPnP* et du mécanisme d'adaptation basé sur *OSGi*.
Les agents sont développés en tant qu'*équipements UPnP*. La *perception* qu'ont les agents des équipements disponibles dans l'environnement et des autres agents du système est basée sur le *mécanisme de découverte* mis en place par le protocole UPnP. La *communication* entre les agents est réalisée par le biais des *services UPnP* fournis par les agents.
Chaque agent est déployé sur une *plateforme OSGi*. Les composants qui constituent les architectures internes des agents sont des *bundles OSGi*. La plateforme permet l'installation, la mise à jour ou la suppression des composants à l'exécution sans perturber le fonctionnement des autres composants. Le mécanisme d'adaptation est basé sur cet aspect.
Des *expérimentations* ont été réalisées pour vérifier la faisabilité et l'efficacité de notre modèle d'agents. Ces expérimentations mettent en œuvre un petit ensemble d'équipements virtuels et d'agents déployés sur un réseau. Les agents détectent et identifient les équipements puis ils génèrent des interfaces graphiques mises à jour. Les agents de contrôle d'équipements sont paramétrés et des scénarios sont définis. La génération, le déploiement et l'assemblage des composants de contrôle et de coordination sont mis en œuvre.
Un *simulateur* d'environnement domotique a été développé pour permettre d'avoir une interprétation visuelle de nos expérimentations.

## 6.2 Perspectives

Les perspectives ouvertes pour ce travail sont nombreuses. En effet, cette thèse a permis d'obtenir un premier système qui a ouvert des portes vers de nombreuses études. Dans cette section, nous proposons les perspectives liées à chacun des éléments des environnements domotiques : utilisateur, système et équipements.

### 6.2.1 Gestion des utilisateurs

**Définition de descripteurs orientés utilisateur.** Le contenu des interfaces graphiques est extrait des descripteurs des équipements et des services. Ces descripteurs sont orientés dévelop-

peur. Les descripteurs orientés utilisateur contiendraient des descriptions d'équipements et de services destinées aux utilisateurs non-experts. Cette méta-information permettant le découplage entre le service tel qu'il est proposé par l'équipement et le service tel qu'il est vu par l'utilisateur pourrait être également utile à l'internationalisation des applications (descriptions des services dans différentes langues).

**Définition de scénarios standards.** Les utilisateurs peuvent définir des scénarios en fonction de leur profil et des équipements disponibles dans l'environnement. Nous pensons que le système doit intégrer des descriptions de scénarios standards. Le système sera en mesure de proposer des scénarios adaptés au contexte.

**Définition de profils standards.** Le choix des fonctionnalités offertes à un utilisateur et du contenu des interfaces graphiques est guidé par son profil. Les profils des utilisateurs sont définis par les utilisateurs privilégiés. Nous pensons que le système doit intégrer des profils standards qui pourront être attribués directement à des utilisateurs. Ces profils contiennent des scénarios standards proposés à l'utilisateur en fonction des droits d'accès et de l'environnement.

**Intégration autonomique de scénarios.** La définition des scénarios pourrait être basée sur l'apprentissage du système des activités quotidienne des utilisateurs de l'environnement. Le système proposerait des scénarios personnalisés aux utilisateurs.

### 6.2.2 Gestion du système

**Développement du système.** Les expérimentations réalisées jusqu'à présent ont été menées :

- avec un petit nombre et une petite variété d'équipements,
- avec un seul utilisateur par le biais d'un seul agent d'interface,
- avec un nombre très limité d'agents de contrôle d'équipements.

À court terme, nous envisageons de réaliser des expérimentations à plus grande échelle (un plus grand nombre d'utilisateurs, d'équipements et d'agents) afin de tester les performances de notre système. Puis, nous poursuivrons dans le développement du système pour y intégrer les agents de contrôle médiateurs et manageurs et les mécanismes de gestion de l'organisation.
À long terme, nous souhaitons valider notre système en l'intégrant dans un environnement domotique réel où de vrais utilisateurs définiraient de vrais scénarios.

**Calcul de la charge de coordination.** Pour l'instant, cette charge est calculée en fonction du nombre de scénarios implémentés par l'agent de contrôle. Elle est calculée en fonction du nombre de composants de coordination générés par l'agent. Ce calcul est simple pour une première approche. Nous devons intégrer dans le calcul le coût de la vérification de l'événement et de chaque condition et le coût d'envoi des messages échangés entre les agents coordinateurs et des requêtes d'exécution des actions. Ce calcul plus sophistiqué améliorera le choix des agents de contrôle qui implémenteront un scénario.

**Optimisation de la répartition de charge.** Nous envisageons de réaliser des tests nous permettant de sélectionner un algorithme combinatoire efficace pour le déploiement des scénarios sur les DCA. Il est donc très important de trouver une méthode pour le calcul de la charge d'un agent de contrôle.

**Optimisation du redéploiement des scénarios.** Le redéploiement des scénarios est réalisé par la désinstallation de composants de coordination et la génération de nouveaux composants. Nous devons intégrer des méthodes pour adapter les composants de coordination sans avoir besoin de les désinstaller.

**Définition de stratégies de déploiement des scénarios.** Nous avons choisi de déployer les scénarios en fonction de la charge des agents de contrôle d'équipements. Nous pensons qu'il serait intéressant de proposer aux administrateurs des stratégies de déploiement de scénarios. Les stratégies favoriseraient la performance ou la fiabilité.

**Amélioration de l'adaptation du contrôle.** Les agents de contrôle adaptent leur architecture interne en fonction de leur champ de contrôle. Les DCA peuvent avoir les mêmes champs de contrôle ou au moins le contrôle d'un ensemble d'équipements en commun. Pour l'instant, tous les DCA génèrent des composants de contrôle. Nous pensons que les agents de contrôle devraient générer les composants de contrôle de manière à avoir au moins un agent qui contrôle l'équipement. Puis, les autres agents génèrent le composant de contrôle pour l'implémentation des scénarios.

**Définition agrégée de la charge des agents de contrôle.** Nous calculons la charge des agents de contrôle de manière séparée pour chaque rôle. Nous envisageons de définir un paramètre de charge unique pour chaque agent de contrôle. Ce paramètre désignerait la charge globale de travail sur un agent en fonction de la charge de gestion des agents, de contrôle des équipements et de coordination pour l'implémentation des scénarios.

**Paramétrage automatique du système.** Pour l'instant, le système peut être soit paramétré automatiquement soit par les utilisateurs. Le paramétrage automatique est réalisé aléatoirement d'une manière trop simple. Nous voulons proposer aux administrateurs deux solutions pour faciliter le paramétrage et pour qu'il soit plus efficace. La première solution consiste à proposer des stratégies de paramétrage automatique prédéfinies. Les stratégies seraient basées sur la performance ou la fiabilité. En fonction du nombre d'équipements et d'utilisateurs, les administrateurs sélectionnent la stratégie qui leur convient. La deuxième solution est basée sur une analyse effectuée automatiquement par le système. L'analyse dépend de la nature de l'environnement (nombre d'équipements et de services) et du système (nombre des agents de contrôle). L'administrateur fournit un nombre approximatif d'utilisateurs. Enfin, le système propose le paramétrage le plus adapté.

**Gestion des messages échangés.** Nous envisageons de développer pour chaque agent un composant *boîte aux lettres* pour la gestion des messages reçus. Cela nous permettra de définir des types de messages prioritaires. Ce composant diminuera la charge de travail attribuée au composant *CoreComponent* pour la gestion des messages.

**Gestion des conflits.** La gestion des conflits d'exécution des services est réalisée au moment de l'exécution des scénarios. Nous pensons que le système doit être capable de prévoir lors de la

définition d'un scénario par un calcul statique, certains des risques de conflits potentiels avec les scénarios en cours d'exécution.

### 6.2.3 Gestion des équipements

**Gestion de la mobilité.** Lors de la suppression/panne d'un équipement, les DCA adaptent leur architecture par la suppression du composant de contrôle de l'équipement. Cette suppression peut causer l'invalidité d'un ensemble de scénarios. Cependant, la suppression d'un équipement peut être liée à sa mobilité. Nous souhaitons définir un type d'équipement dit *mobile* pour lequel les composants de contrôle ne sont pas supprimés. Les scénarios qui utilisent ce type d'équipements doivent être adapté si possible, sinon ils sont suspendus en attendant le retour de l'équipement. Cela permet au système d'être plus performant en évitant de générer un composant de contrôle à chaque fois.

**Immotique.** La réalisation de la coordination entre plusieurs environnements domotiques autoriserait le partage de certains services ou ressources pour la réalisation des scénarios et diminuer l'impact de la panne d'un équipement.

**Environnement à grande échelle.** Nous souhaitons intégrer notre système dans un environnement à plus grande échelle comme les grilles [67]. Nous introduirons au système la gestion des services web. Nous nous inspirerons notamment des travaux qui traitent de la gestion de la composition de services à l'aide d'agents [49, 14, 34].

# Bibliographie


[1] Apache felix, http ://felix.apache.org/site/index.html.

[2] CyberGarage, http ://www.cybergarage.org/.

[3] FIPA ACL Message Structure Specification, http ://www.fipa.org/specs/fipa00061/index.html.

[4] HVAC v 1.0, http ://upnp.org/specs/ha/hvac/.

[5] ISO. Standard 14764 on software engineering - software maintenance. iso/iec, 1999.

[6] OMG Unified Modeling LanguageTM (OMG UML), Superstructure v2.2, http ://www.omg.org/spec/uml/2.2/superstructure/pdf/.

[7] OSGi alliance, http ://www.osgi.org.

[8] Standards : Device control protocols, http ://upnp.org/sdcps-and-certification/standards/sdcps/.

[9] UPnP Device Architecture v1.0, http ://upnp.org/specs/arch/upnp-arch-devicearchitecture-v1.0.pdf.

[10] Upnp forum, http ://www.upnp.org.

[11] Projet ACCORD. Assemblage de composants par contrats. Livrable 1.1-1, 2002.

[12] J. J. Alferes, F. Banti, and A. Brogi. An event-condition-action logic programming language. In M. Fisher, W. van der Hoek, B. Konev, and A. Lisitsa, editors, *in JELIA'06 : Proceedings of the 10th European Conference on Logics in Artificial Intelligence*, volume 4160 of *Lecture Notes in Computer Science*, pages 29–42. Springer, September 2006.

[13] P. Aniorté and J. Lacouture. CompAA : A self-adaptable component model for open systems. In *15th IEEE International Conference and Workshop on Engineering of Computer Based Systems (ECBS 2008)*, pages 19–25. IEEE Computer Society, 2008.

[14] M. Baker, R. Buyya, and D. Laforenza. Grids and grid technologies for wide-area distributed computing. *Software : Practice and Experience*, 32 :1437–1466, December 2002.

[15] C. Bock. UML 2 composition model. *Journal of Object Technology*, 3(10) :47–74, November 2004.

[16] O. Boissier, N. Bouraqadi, J. C. Royer, D. Seriai, C. Urtado, S. Vauttier, and L. Vercouter, editors. *Actes de la première Journée Multi-Agents et Composants (JMAC 2004)*, Paris, France, Novembre 2004.

[17] O. Boissier, N. Bouraqadi, J. C. Royer, D. Seriai, C. Urtado, S. Vauttier, and L. Vercouter, editors. *Actes de la deuxième Journée Multi-Agents et Composants (JMAC 2006)*, Nîmes, France, Mars 2006.

[18] S. Boussier, M. Vasquez, Y. Vimont, S. Hanafi, and P. Michelon. A multi-level search strategy for the 0-1 multidimensional knapsack problem. *Discrete Applied Mathematics*, 158(2) :97–109, January 2010.

[19] J. P. Briot and Y. Demazeau. *Principes et Architecture des Systèmes Multi-Agents*. Hermes, 2001.

[20] J. P. Briot, T. Meurisse, and F. Peschanski. Architectural design of component-based agents : A behavior-based approach. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah-Seghrouchni, editors, *PROMAS'06 : 4th International Workshop on Programming Multi-Agent Systems*, volume 4411 of *Lecture Notes in Computer Science*, pages 71–90. Springer, May 2006.

[21] J. P. Briot, T. Meurisse, and F. Peschanski. Une expérience de conception et de composition de comportements d'agents à l'aide de composants. *L'OBJET, Revue des Sciences et Technologies de l'Information*, 12(4) :11–41, 2006.

[22] A. W. Brown and K. C. Wallnau. The current state of CBSE. *IEEE Software*, 15(5) :37–46, September 1998.

[23] J. Buckley, T. Mens, M. Zenger, A. Rashid, and G. Kniesel. Towards a taxonomy of software change : Research articles. *Journal of Software Maintenance and Evolution*, 17(5) :309–332, September 2005.

[24] N. Chapin, J. E. Hale, K. Md. Kham, J. F. Ramil, and W. G. Tan. Types of software evolution and software maintenance. *Journal of Software Maintenance and Evolution*, 13(1) :3–30, January 2001.

[25] S. Cook, R. Harrison, M. M. Lehman, and P. Wernick. Evolution in software systems : foundations of the SPE classification scheme : Research articles. *Journal of Software Maintenance and Evolution : Research and Practice*, 18(1) :1–35, January 2006.

[26] I. Crnkovic, S. Sentilles, A. Vulgarakis, and M. Chaudron. A classification framework for software component models. *IEEE Transactions on Software Engineering*, 99 :1–25, October 2010.

[27] M. Desertot, H. Cervantes, and D. Donsez. FROGi : fractal components deployment over OSGi. In W. Löwe and M. Südholt, editors, *5th International Symposium on Software Composition*, volume 4089 of *Lecture Notes in Computer Science*, pages 275–290. Springer, March 2006.

[28] N. Desnos, M. Huchard, G. Tremblay, C. Urtado, and S. Vauttier. Search-based many-to-one component substitution. *Journal of Software Maintenance and Evolution : Research and Practice, Special issue on Search-Based Software Engineering*, 20(5) :321–344, September / October 2008.

[29] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1) :4–7, January 2001.

[30] D. Donsez, K. Gama, and W. Rudametkin. Developing adaptable components using dynamic languages. In *SEAA'09 : Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications*, pages 396–403, Washington, DC, USA, August 2009.

[31] C. Dony, C. Tibermacine, C. Urtado, and S. Vauttier. Specification of an exception handling system for a replicated agent environment. In *Proceedings of WEH'08, the 4th international workshop on Exception handling*, pages 24–31, November 2008.

[32] J. L. Encarnação and T. Kirste. Ambient intelligence : Towards smart appliance ensembles. In M. Hemmje, C. Niederée, and T. Risse, editors, *From Integrated Publication and Information Systems to Virtual Information and Knowledge Environments*, volume 3379 of *Lecture Notes in Computer Science*, pages 261–270. Springer, 2005.

[33] J. Ferber. *Les Systèmes multi-agents : Vers une intelligence collective.* Dunod, January 2007.

[34] I. T. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. Grid services for distributed system integration. *IEEE Computer*, 35(6) :37–46, June 2002.

[35] P. Galinier and J. K. Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4) :379–397, 1999.

[36] D. Garlan. Software architecture : a roadmap. In *ICSE'00 : Proceedings of the Conference on The Future of Software Engineering*, pages 91–101, New York, NY, USA, ACM Press, 2000.

[37] J. C. Georgas, E. M. Dashofy, and R. N. Taylor. Architecture-centric development : a different approach to software engineering. *ACM Crossroads*, 12(4) :6, August 2006.

[38] G. Grondin. *MaDcAr-Agent : un modèle d'agents auto-adaptables à base de composants.* PhD thesis, École Nationale Supérieure des Mines de Saint-Étienne, 2008.

[39] G. Grondin, N. Bouraqadi, and L. Vercouter. MaDcAr : An abstract model for dynamic and automatic (re-)assembling of component-based applications. In I. Gorton, G. T. Heineman, I. Crnkovic, H. W. Schmidt, J. A. Stafford, C. A. Szyperski, and K. C. Wallnau, editors, *CBSE'09 : 9th International Symposium on Component-Based Software Engineering*, volume 4063 of *Lecture Notes in Computer Science*, pages 360–367. Springer, July 2006.

[40] G. Grondin, N. Bouraqadi, and L. Vercouter. Madcar-agent : un modèle pour le développement d'agents auto-adaptables. In *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'09)*, Lyon, France, Octobre 2009.

[41] Z. Guessoum. *Un environnement opérationnel de conception et de réalisation de systèmes multi-agents.* PhD thesis, Université Paris 6, LAFORIA, 1996.

[42] F. Hamoui, M. Huchard, C. Urtado, and S. Vauttier. Specification of a component-based domotic system to support user-defined scenarios. In D. Zhang G. Wirtz, J. Gao, editor, *SEKE'09 : 21st International Conference on Software Engineering and Knowledge Engineering*, pages 597–602. Knowledge Systems Institute Graduate School, July 2009.

[43] F. Hamoui, M. Huchard, C. Urtado, and S. Vauttier. Un système d'agents à base de composants pour les environnements domotiques. In E. Cariou and J. C. Royer, editors, *Actes de la 16ème conférence francophone sur les Langages et Modèles à Objets (LMO 2010)*, pages 35–49, Pau, France, Mars 2010.

[44] F. Hamoui, C. Urtado, S. Vauttier, and M. Huchard. SAASHA : A self-adaptable agent system for home automation. In *Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, pages 227–230. IEEE Computer Society, September 2010.

[45] T. Heider and T. Kirste. Multimodal appliance cooperation based on explicit goals : concepts & potentials. In *sOc-EUSAI'05 : Proceedings of the 2005 joint conference on Smart objects and ambient intelligence*, pages 271–276. ACM, October 2005.

[46] G. Heineman and H. Ohlenbusch. An evaluation of component adaptation techniques. In *Proceedings of the 2nd Annual Workshop on Component-Based Software Engineering*, Los Angeles, CA, May 1999.

[47] V. Hourdin, J. Y. Tigli, S. Lavirotte, G. Rey, and M. Riveill. SLCA, composite services for ubiquitous computing. In *Mobility'08 : Proceedings of the International Conference on Mobile Technology, Applications, and Systems*, pages 1–8. ACM, September 2008.

[48] H. Ishikawa, Y. Ogata, K. Adachi, and T. Nakajima. Building smart appliance integration middleware on the osgi framework. In *ISORC'04 : 7th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, pages 139–146. IEEE Computer Society, May 2004.

[49] C. Jonquet, P. Dugenie, and S. A. Cerri. Service-based integration of grid and multi-agent systems models. In R. Kowalczyk, M. N. Huhns, M. Klusch, Z. Maamar, and Q. B. Vo, editors, *AAMAS'08 : International Workshop on Service-Oriented Computing : Agents, Semantics, and Engineering (SOCASE)*, volume 5006 of *Lecture Notes in Computer Science*, pages 56–68. Springer, May 2008.

[50] J. Y. Jung, J. Park, S. K. Han, and K. Lee. An ECA-based framework for decentralized coordination of ubiquitous web services. *Information & Software Technology*, 49(11-12) :1141–1161, 2007.

[51] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Berlin, Germany, 2004.

[52] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1) :41–50, January 2003.

[53] R. Krutisch, P. Meier, and M. Wirsing. The agent component approach, combining agents, and components. In M. Schillo, M. Klusch, J. P. Müller, and H. Tianfield, editors, *MATES'03 : Proceedings of the First German Conference on Multiagent System Technologies*, volume 2831 of *Lecture Notes in Computer Science*, pages 1–12. Springer, September 2003.

[54] J. Lacouture. *Ingénierie logicielle orientée services : une contribution ţa l'adaptationdynamique basée sur une approche mixte composant/agent*. PhD thesis, L'Université de Pau et des Pays de l'Adour, 2008.

[55] K. K. Lau and Z. Wang. Software component models. *IEEE Transactions on Software Engineering*, 33 :709–724, October 2007.

[56] M. M. Lehman. On understanding laws, evolution, and conservation in the large-program life cycle. *Journal of Systems and Software*, 1 :213–221, 1980.

[57] M. M. Lehman. Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68(9) :1060–1076, September 1980.

[58] M. M. Lehman, J. F. Ramil, P. Wernick, D. E. Perry, and W. M. Turski. Metrics and laws of software evolution - the nineties view. In *METRICS'97 : Proceedings of the 4th International Symposium on Software Metrics*, pages 20–32. IEEE Computer Society, November 1997.

[59] H. Lieberman and T. Selker. Out of context : computer systems that adapt to, and learn from, context. *IBM Systems Journal*, 39(3-4) :617–632, July 2000.

[60] B. P. Lientz and B. E. Swanson. *Software Maintenance Management*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1980.

[61] G. Nain, O. Barais, R. Fleurquin, and J. M. Jézéquel. EntiMid : un middleware aux services de la maison. In *3ème Conférence Francophone sur les Architectures Logicielles (CAL'09)*, Nancy, France, Mars 2009.

[62] G. Nain, E. Daubert, O. Barais, and J. M. Jézéquel. Using MDE to build a schizophrenic middleware for home/building automation. In Petri Mähönen, Klaus Pohl, and Thierry Priol, editors, *In ServiceWave'08 : Conference on Networked European Software & Services Initiative (NESSI)*, volume 5377 of *Lecture Notes in Computer Science*, pages 49–61. Springer, December 2008.

[63] T. Nakajima and I. Satoh. A software infrastructure for supporting spontaneous and personalized interaction in home computing environments. *Personal Ubiquitous Computing*, 10(6) :379–391, September 2006.

[64] M. Nakamura, H. Igaki, and K. I. Matsumoto. Feature interactions in integrated services of networked home appliances : An object-oriented approach. In S. Reiff-Marganiec and M. Ryan, editors, *FIW : In Proceedings of International Conference on Feature Interactions in Telecommunication Networks and Distributed Systems (ICFI'05)*, pages 236–251. IOS Press, June 2005.

[65] M. Nakamura, H. Igaki, H. Tamada, and K. I. Matsumoto. Implementing integrated services of networked home appliances using service oriented architecture. In Marco Aiello, Mikio Aoyama, Francisco Curbera, and Mike P. Papazoglou, editors, *ICSOC'04 : Proceedings of the 2nd International Conference on Service Oriented Computing*, pages 269–278. ACM, November 2004.

[66] M. Nakamura, A. Tanaka, H. Igaki, H. Tamada, and K. I. Matsumoto. Constructing home network systems and integrated services using legacy home appliances and web services. *International Journal of Web Services Research (IJWSR)*, 5(1) :81–97, January 2008.

[67] Z. Németh and V. S. Sunderam. Characterizing Grids : Attributes, Definitions, and Formalisms. *Journal of Grid Computing*, 1(1) :9–23, 2003.

[68] R. Nzekwa, R. Rouvoy, and L. Seinturier. A flexible context stabilization approach for self-adaptive application. In *Workshop Proceedings of Eigth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 7–12. IEEE, March 2010.

[69] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization : Algorithms and Complexity*. Dover Publications, July 1998.

[70] N. Pessemier, O. Barais, L. Seinturier, T. Coupaye, and L. Duchien. A three level framework for adapting component-based systems. In *2nd Workshop on Coordination and Adaptation Techniques for Software Entities (WCAT) at the 19th European Conference on Object-Oriented Programming (ECOOP'05)*, July 2005.

[71] G. Plateau and M. Elkihel. A hybrid method for the 0-1 knapsack problem. *Methods of Operations Research*, 49 :277–293, 1985.

[72] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, June 1998.

[73] H. Schulzrinne, X. Wu, S. Sidiroglou, and S. Berger. Ubiquitous Computing in Home Networks. *IEEE Communications Magazine*, 41(11) :128–135, November 2003.

[74] Y. Secq. *RIO : rôles, interactions et organisations, une méthodologie pour les systèmes multi-agents ouverts*. PhD thesis, Université des Sciences et Technologies de Lille, 2003.

[75] M. Son, D. Shin, and D. Shin. Design and implementation of the intelligent multi-agent system based on web services. In *WAIMW '06 : Proceedings of the Seventh International Conference on Web-Age Information Management Workshops*, page 20, Washington, DC, USA, June 2006. IEEE Computer Society.

[76] M. Son, D. Shin, and D. Shin. Research on multi-agent service bundle middleware for smart space. In G. Wang, J. F. Peters, A. Skowron, and Y. Yao, editors, *RSKT'06 : Proceedings of the First International Conference on Rough Sets and Knowledge Technology*, volume 4062 of *Lecture Notes in Computer Science*, pages 618–625. Springer, 2006.

[77] C. Szyperski. *Component Software : Beyond Object-Oriented Programming (2nd Edition)*. Addison-Wesley Professional, 2 edition, November 2002.

[78] R. N. Taylor, N. Medvidovic, and E. M. Dashofy. *Software Architecture : Foundations, Theory, and Practice*. Wiley, January 2009.

[79] J. Y. Tigli, S. Lavirotte, G. Rey, V. Hourdin, D. Cheung-Foo-Wo, E. Callegari, and M. Riveill. WComp Middleware for Ubiquitous Computing : Aspects and Composite Event-based Web Services. *Annals of Telecommunications (AoT)*, 64(3-4) :197–214, April 2009.

[80] M. Vallée, F. Ramparany, and L. Vercouter. Using device services and flexible composition in ambient communication environments. In *1st International Workshop on Requirements and Solutions for Pervasice Software Infrastructure (RSPSI)*, May 2006.

[81] M. Vallée, F. Ramparany, and L. Vercouter. Flexible composition of smart device services. In Laurence Tianruo Yang, Jianhua Ma, Makoto Takizawa, and Timothy K. Shih, editors, *PSC'05 : International Conference on Pervasive Systems and Computing*, pages 165–171. CSREA Press, May 2005.

[82] L. Vercouter. MAST : Un modèle de composants pour la conception de SMA. In *1ère Journée Multi-Agents et Composants (JMAC'04)*, Paris, France, Novembre 2004.

[83] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3) :66–75, September 1991.

[84] M. Weiser. Ubiquitous computing. *Computer*, 26 :71–72, October 1993.

[85] M. Wilson, E. H. Magill, and M. Kolberg. An online approach for the service interaction problem in home automation. In *IEEE Consumer Communications and Networking Conference*, pages 251–256, January 2005.

[86] M. Wooldridge. Agent-based computing. *Interoperable Communication Networks*, 1 :71–97, September 1997.

[87] M. Wooldridge. *An Introduction to MultiAgent Systems - Second Edition.* John Wiley & Sons, May 2009.

[88] C. L. Wu, C. F. Liao, and L. C. Fu. Service-oriented smart-home architecture based on OSGi and mobile-agent technology. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 37(2) :193–205, 2007.

[89] H. Y. Zhang, C. Urtado, and S. Vauttier. Architecture-centric component-based development needs a three-level ADL. In Muhammad Ali Babar and Ian Gorton, editors, *Proceedings of the 4th European Conference on Software Architecture*, volume 6285 of *LNCS*, pages 295–310. Springer, August 2010.

[90] H. Y. Zhang, C. Urtado, and S. Vauttier. Architecture-centric development and evolution processes for component-based software. In *Proceedings of 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE 2010)*, Redwood City, USA, July 2010.

# Annexe A

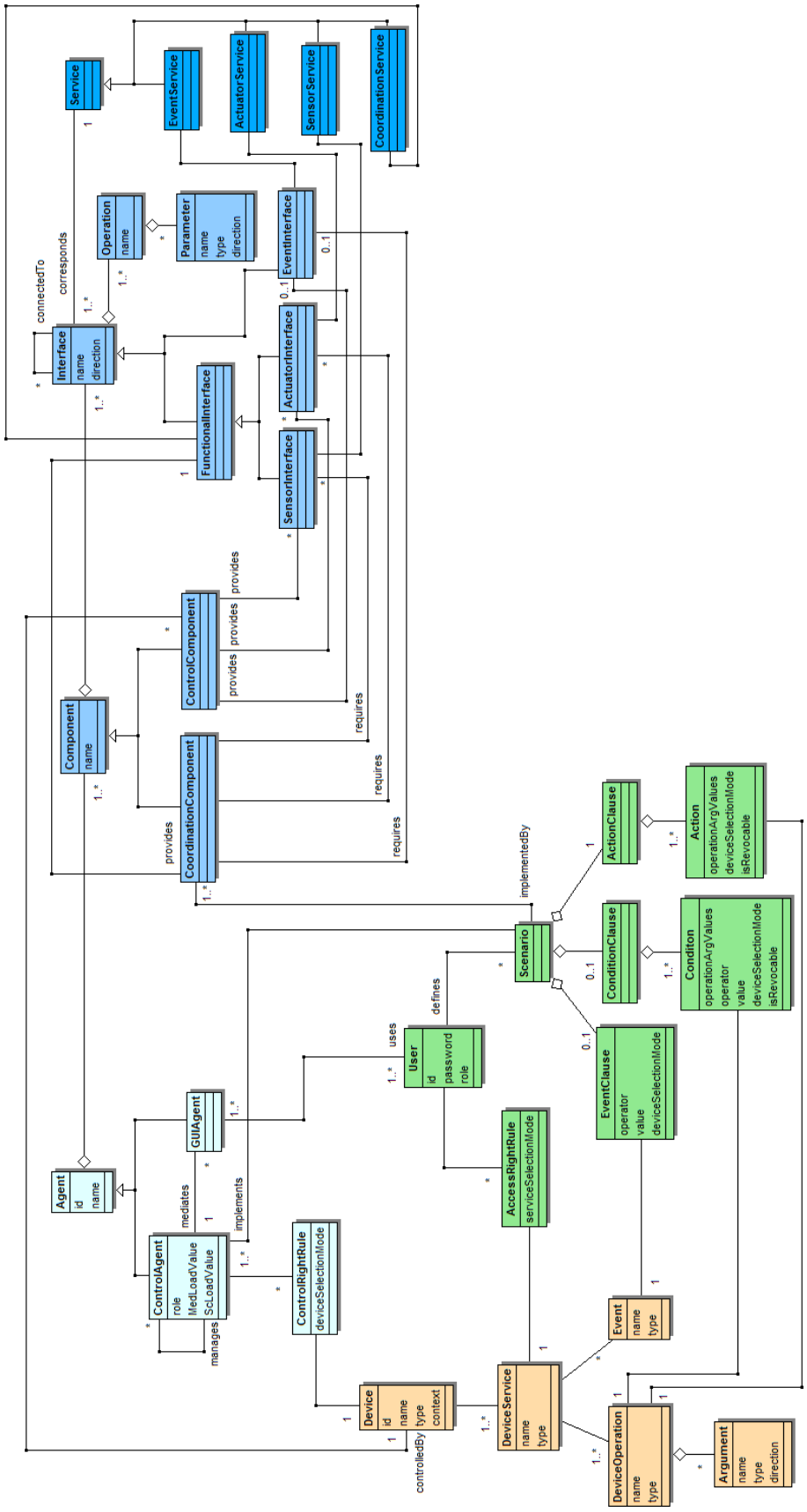



FIGURE A.1 – Méta-modèle de SAASHA

**Résumé**

Les environnements domotiques sont des environnements ubiquitaires dans lesquels des équipements domestiques, disséminés dans une habitation, fournissent des services utilisables à distance au travers d'un réseau. Des systèmes domotiques sont proposés pour permettre aux utilisateurs de contrôler les équipements en fonction de leurs besoins. Idéalement, ces systèmes orchestrent l'exécution des services fournis par les équipements pour réaliser des services complexes. Mieux encore, ces systèmes doivent s'adapter à la variété des environnements en termes d'équipements et des besoins des utilisateurs. Ils doivent également pouvoir s'adapter dynamiquement, si possible de manière autonome, au changement de leur contexte d'exécution (apparition ou disparition d'un équipement, évolution des besoins).
Dans cette thèse, nous apportons une réponse à cette problématique avec *SAASHA*, un système domotique multi-agents à base de composants. La combinaison de ces deux paradigmes permet de gérer l'adaptation à trois niveaux : présentation (interfaces utilisateur), organisation (architecture du système) et comportement (architecture interne des agents). Les agents perçoivent le contexte et ses changements. Les utilisateurs se voient proposer une vue dynamique du contexte leur permettant de définir des scénarios personnalisés sous forme de règles. Les agents se répartissent les rôles pour réaliser les scénarios. Ils modifient dynamiquement leur architecture interne grâce à la génération, au déploiement et à l'assemblage de composants pour se doter de nouveaux comportements de contrôle des équipements et des scénarios. Les agents collaborent ainsi pour exécuter les scénarios. En cas de changement, ces trois niveaux d'adaptation sont mis en oeuvre de manière dynamique et autonome pour maintenir la continuité de service. Un prototype de *SAASHA*, basé sur les standards industriels UPnP et OSGi, a été développé pour évaluer la faisabilité de notre proposition.

---

**Title**

**A component-based multi-agent system for autonomous context-aware adaptation Application to smart homes**

**Abstract**

Smart homes are ubiquitous environments where a whole house distributed set of domestic devices provides services that can be remotely used through a network. Home automation systems enable users to control the devices according to their needs. Ideally, these systems must be able to orchestrate the services provided by the devices to perform complex services. Moreover, these systems must enable to adapt them to the variety of the contexts in which they are used (various devices and user requirements). They must also ba able to adapt themselves dynamically, possibly autonomously, to changes in their execution context (device appearance or disappearance, evolving requirements).
In this thesis, we provide an answer to these issues with Saasha (acronym for Self-Adaptable Agent System for Home Automation), a component-based multi-agent home automation system. The combination of these two paradigms defines three adaptation levels : presentation (user interface), organization (system architecture) and behavior (agents' internal architecture). Agents perceive context and its changes. Users are provided with a dynamic view of the context so they can define custom scenarios as rules. Agents dispatch roles among themselves to implement scenarios. They dynamically modify their own internal architecture by generating, deploying and assembling components to gain new device and scenario control behaviors. This way, agents collaborate to implement scenarios. In case of changes, these three adaptation levels are used dynamically and autonomously to maintain service continuity. In order to evaluate our proposal's feasibility, Saasha has been prototyped using UPnP and OSGi industry standards.

---

**Mots clefs**

développement à base de composants, systèmes multi-agents, services, système domotique, sensibilité au contexte, adaptation autonomique, déploiement et assemblage de composants