Adaptation d'une Application de Gestion du Bâtiment Intelligent "Smart Building"

Marwa Chaabane*, Ismael Bouassida Rodriguez*,**

*Université de Sfax, ReDCAD, 3038 Sfax, Tunisie
marwa.mchaabane@gmail.com,
**Centre de Recherche en Numérique de Sfax, 3021 Sfax, Tunisie
bouassida@redcad.org

Résumé. Le domaine des Smart Spaces prend de plus en plus d'importance de nos jours surtout avec l'intelligence des équipements informatiques. Les applications appartenant à ce domaine sont qualifiées "smart" grace à leur capacité à détecter les changements de leur contexte et à s'adapter à ces contextes évolutifs. En revanche, le défi soulevé par ces applications ne se limite pas à cela mais le dépasse à rationaliser la consommation de l'énergie sans affecter le confort des utilisateurs. Dans ce travail, nous proposons un processus d'adaptation d'une application distribuée intitulé "Smart Building" dans le but de concilier les deux contraintes.

1 Introduction

Notre monde est en plein changement, des changements qui ont un impact significatif sur notre environnement, ces changements nous obligent de penser et agir différemment. Les bâtiments sont également concernés. Les études démontrent que les exigences énergétiques dans les bâtiments sont en constante évolution. Toutefois, la performance globale du bâtiment est la priorité et non pas uniquement l'optimisation de la consommation énergétique. Il est nécessaire d'augmenter le confort et la qualité des bâtiments pour le bien être des utilisateurs. Dans ce travail, notre défi consiste ainsi à trouver des solutions qui apportent une adéquation entre efficacité énergétique et confort. Des solutions qui considèrent les bâtiments comme des objets modulables, interconnectés, intelligents et sensibles à leur environnement. En effet, un système intelligent de gestion énergétique contrôle la régulation des flux en fonction des besoins des utilisateurs, des énergies produites et des tarifs des énergies en cours.

L'objet de ce travail est d'adapter une application distribuée intitulée "Smart Building" afin de rationaliser la consommation de l'énergie du bâtiment. L'auto-adaptation du "Smart Building" aux contraintes énergétiques requière le développement d'un gestionnaire d'un bâtiment intelligent. Ce gestionnaire gère le fonctionnement du bâtiment et lui permet de s'adapter aux changements de contexte et de répondre aux besoins et aux exigences des utilisateurs. Il met en oeuvre des scénarios d'adaptation agissant sur les valeurs des caractéristiques et des grandeurs physiques des entités logicielles du "Smart Building" tout en respectant le confort de l'utilisateur de l'application. Ce manuscrit, qui s'articule autour de trois parties, définit la solution

que nous envisageons afin d'adapter une application “Smart Building" aux changements de son contexte. Dans la première partie, nous définissons des notions de bases liées essentiellement à l'“intelligence ambiante" appelée aussi dans la littérature : informatique ubiquitaire. Nous présentons dans la deuxième partie notre cas d'étude “Smart Building", nous expliquons sa manière de fonctionnement, et nous listons les fonctionnalités que nous devons atteindre. Dans la dernière partie, nous nous interessons à la mise en œuvre de notre solution et nous présentons les scénarios d'adaptation de notre application.

2 Contexte général

L'intelligence des systèmes dépend de leur façon d'exploiter les données de contexte acquises. Ces données doivent être interprétées pour exécuter les actions adéquates. Des techniques de contrôle sont nécessaires pour adapter les actions du système aux informations perçues de l'environnement et de l'utilisateur. Pour cela, un bâtiment est qualifié intelligent “Smart" lorsqu'il est capable de détecter les changements de son contexte et d'adapter son comportement à ce contexte évolutif.

2.1 Concepts de base

Nous nous intéressons particulièrement aux trois notions suivantes : l'informatique ubiquitaire, la sensibilité au contexte et l'adaptation.

2.1.1 Informatique ubiquitaire

Le concept d'informatique ubiquitaire a été introduit par Mark Weiser en 1991 dans son article “The computer for the 21st Century" Weiser (1991). Ainsi, Weiser propose une révolution dans les technologies de l'information afin qu'elles développent complètement leur potentiel : les ordinateurs doivent devenir invisibles. La façon de rendre les ordinateurs invisibles consiste à les intégrer avec les objets et les endroits que nous utilisons dans la vie quotidienne. En effet, Weiser affirme : “Les technologies les plus profondes sont celles qui disparaissent. Elles se mêlent dans le tissu de la vie de tous les jours jusqu'au point où elles en deviennent indiscernables." L'informatique ubiquitaire désigne alors le fait que l'informatique est omniprésente et ceci d'une façon tellement invisible qu'on ne remarque pas sa présence. Selon Saha et Mukerjee Saha et Mukherjee (2003), un environnement ubiquitaire comporte essentiellement quatre domaines :

- **Les dispositifs** : tels que les périphériques d'entrées et les périphériques de sorties, les dispositifs mobiles (le cas des assistants personnels).
- **Les réseaux ubiquitaires** : ont pour rôle de supporter l'intégration des dispositifs informatiques ubiquitaires.
- **Les intergiciels ubiqutaires** : permettent l'interaction entre les applications ubiquitaires, les réseaux d'accès et leurs environnements.
- **Les applications ubiquitaires** : s'intéressent principalement à l'environnement et aux informations liées au contexte

Ces environnements sont hautement dynamiques, c'est-à-dire qu'ils se caractérisent par un contexte qui varie fréquemment. Ainsi, les concepts contexte et adaptation jouent un rôle clé dans les systèmes ubiquitaires.

2.1.2 Sensibilité au contexte

Nous commençons d'abord par expliquer le terme contexte. Étant donné que le contexte est une notion complexe et abstraite, il est difficile de lui trouver une définition détaillée. Citant, par example, la position de Charith et al.Perera et al. (2013) qui adoptent la définition de Dey Abowd et al. (1999) : "Le contexte couvre toutes les informations qui peuvent être utilisées pour caractériser la situation d'une entité. Une entité est une personne, un endroit ou un objet que l'on considère pertinent par rapport à l'interaction entre un utilisateur et une application, y compris l'utilisateur et l'application eux mêmes." Après avoir clarifié la notion de contexte, nous nous intéressons à la sensibilité au contexte. Schilit et Theimer étaient les premiers qui ont essayé de définir la sensibilité au contexte : il s'agit de la capacité d'un système à découvrir et à réagir à des changements dans l'environnement où il se trouve Schilit et Theimer. L'objectif principal de la sensibilité au contexte est de minimiser l'interaction de l'utilisateur, en répondant automatiquement aux changements du contexte.

2.1.3 Adaptation

D'une façon générale, nous parlons de la capacité d'un système à s'adapter à des changements. En informatique, et selon le grand dictionnaire québécois de la langue française (2003), l'adaptation est définie comme étant l'opération qui consiste à apporter des modifications à un logiciel ou à un système informatique, à la fin de son développement, dans le but d'améliorer ses performances dans un contexte précis d'utilisation. En revanche, dans le cadre des systèmes sensibles au contexte, l'adaptation est étendue par la notion d'auto-adaptation Oreizy et al. (1999). Raibulet Raibulet (2008) identifie trois concepts importants des systèmes adaptatifs. En premier lieu, l'adaptabilité est demandée comme réponse à des changements qui arrivent soit à l'intérieur du système, soit dans son environnement. Dans ce deuxième cas, le système est sensible au contexte. En deuxième lieu, l'adaptabilité est atteinte avec des changements que le système effectue sur lui même. Finalement, pour considérer qu'un système est pleinement adaptatif, ces changements doivent s'effectuer lors de l'exécution du système. Ainsi, un système auto-adaptatif sensible au contexte doit respecter ces trois concepts afin de pouvoir s'auto-adapter aux changements de son contexte.

3 Etude préalable

3.1 Présentation du cas d'étude : "Smart Building"

Les "Smart Buildings" ou bâtiments intelligents constituent une réponse à plusieurs problématiques actuelles, comme la maîtrise de consommation des bâtiments et leur approvisionnement en énergie. Un "Smart Building" correspond à l'utilisation de solutions techniques communicantes permettant d'influer le fonctionnement du bâtiment. Ceci peut être soit par automatisme soit par une action des utilisateurs ou gestionnaires en fonction des informations qui leur sont communiquées par les équipements intelligents. Cette définition nous mène à la

necessité de tenir compte des aspects dynamiques du “Smart Building" ; la gestion du confort des utilisateurs et la capacité de communiquer avec l'exterieur afin d'adapter le comportement énergétique vis à vis son environnement. Ainsi, ce genre de cas d'étude exige le développement d'une applicaion permettant la réalisation de ces aspects. Pour détailler, cette adaptation du bâtiment nécessite une communication entre différents dispositifs (capteurs, actionneur etc). Les capteurs assurent des observations du contexte. En revanche, les actionneurs répondent aux changements du contexte et envoient leurs états à une entité centrale. Dans ce travail, nous étudions le fonctionnement d'une application existante Abdennadher (2013). Notre tâche est d'améliorer cette application qui comporte précisemment : un gestionnaire du bâtiment “Building Manager" qui centralise les données de l'application. A ce dernier, se connectent les entités suivantes : un climatiseur “Air Conditioner", une lampe “Lamp", un thermomètre “Thermometer" et un compteur d'énergie intelligent “Power Smart Meter" qui permet la gestion de la consommation et la production d'énergie dans le bâtiment. L'accès à notre application est assurée pour l'utilisateur à travers une interface, appelée “User Agent", depuis son PC, sa tablette, son smartphone etc. Ce “User Agent" peut se connecter à l'entité “Building Manager" afin d'assurer différentes fonctionalités à son utilisateur.

3.2 Fonctionnement du cas d'étude : “Smart Building"

Dans notre cas d'étude “Smart Bulding", les entités “Thrmomètre" et “Compteur d'énergie intelligent" envoient périodiquement leurs valeurs (valeur de température et valeur d'énergie consommée et produite à l'entité “Gestionnaire du bâtiment". L'entité “User Agent" envoie des requêtes (soit pour modifier soit pour s'informer) à l'entité “Gestionnaire du bâtiment". Cette dernière, transmet à son tour les requêtes à l'entité concernée. L'entité “Lampe" et l'entité “Climatiseur" agissent soit pour changer des valeurs (d'intensité, d'état ou de température) soit pour envoyer d'une réponse, lors d'une requête reçue par l'entité “Gestionnaire du bâtiment".

3.3 Contribution

La motivation de notre cas d'étude “Smart Building" est née du besoin de réduire la consommation de l'énergie d'un bâtiment. Pour cela, il est nécessaire de développer d'une application qui tient en compte les aspects dynamiques du “Smart Building", en particulier l'aspect de l'adaptation aux changements du contexte. Cepandant, l'application que nous devons améliorer, réalise seulement la communication entre les différentes entités à travers l'échange des requêtes et des réponses.Dans ce cas là, le rôle du gestionnaire du bâtiment se limite à gérer les flux d'information entre l'agent utilisateur, les capteurs et les actionneurs. Ainsi, nos principales contributions sont les suivantes :

— Proposer une amélioration du fonctionnement actuel du “Smart Building".
— Rendre le gestionnaire du bâtiment capable de s'adapter aux changements de son contexte (précisemment les changements des valeurs de consommation et de production de l'énergie dans le bâtiment).

3.4 Les fonctionnalités fournies par l'application "Smart Building"

En se basant sur ce qui précède et en reprenant la spécification de Abdennadher (2013) et Taktak et al. (2016), dans cette section nous présentons les nouvelles fonctionnalités de notre "Smart Building".

3.4.1 Module de gestion de l'entité "Building Manager"

Le changement majeur touche ce module. Afin de rendre l'entité "Building Manager" capable de réaliser l'adaptation du "Smart Building" aux changements de son environnement, en plus des fonctionalités citées par Abdennadher (2013), nous avons ajouté les fonctionnalités suivantes :

- Calculer les changements du contexte.
- Mettre à jour les caractéristiques ou les grandeurs physiques d'une entité.

Ceci nous permet de qualifier l'entité "Building Manager" comme responsable à piloter tout le fonctionnement du bâtiment. Les fonctionnalités offertes par ce module sont présentées dans ce qui suit.

- Se connecter à une entité.
- Reçevoir les réponses envoyées par une entité ("Thermometer", "Power Smart Meter", "Air Conditioner" et "Lamp") ou les requêtes envoyées par l'entité "User Agent".
- Calculer les changements de son contexte (exemple : comparer les valeurs d'énergie produite et consommée dans le bâtiment) et effectuer une mise à jour des valeurs des caractéristiques et des grandeurs physiques des entitées concernées
- Envoyer soit des réponses à l'entité "User Agent", soit des messages au autres entités. Ces messages peuvent être :
  - des requêtes de consultation des valeurs des grandeurs physiques d'une entité (valeur de température dans le cas du climatiseur et valeur d'état ou d'intensité dans le cas de la lampe).
  - des messages de mise à jour des valeurs des grandeurs physiques d'une entité (valeur de température dans le cas du "Air conditioner" et valeur d'état et d'intensité dans le cas de "Lamp") ou des caractéristiques d'une entité (valeurs des périodes de mesure et d'envoi dans le cas du "Power Smart Meter" et du "Thermometer"). Ceci peut être due à une mise à jour effectuée par le gestionnaire du bâtiment suite à un changement de contexte ou à un message reçu de l'entité "User Agent". L'entité "Building Manager" est en état de fonctionnement permanent.

3.4.2 Module de gestion d'une entité :"User Agent", actionneur ou capteur

Toutes les entités doivent fournir principalement ces deux fonctionalités : l'envoi des messages au "Building Manager" et la réception des messages envoyés par ce dernier. Ceci assure la communication des différentes entités dans le "Smart Building". Nous détaillons les fonctionalités de chaque type d'entité dans les sections suivantes.

- **Module de gestion d' une entité actionneur : "Lamp" et "Air Conditioner" :**
  Pour ce module, par rapport à la spécification de Abdennadher Abdennadher (2013), nous avons supprimé les deux cas d'utilisation concernant l'écoute et le calcul des caractéristiques de l'entité (période de mesure et période d'envoi), puisque l'entité

"Lamp" et l'entité "Air Conditioner" sont des actionneurs, alors ils n'effectuent pas des envois périodiques. Une entité actionneur permet d'ajuster les valeurs des grandeurs physiques selon les valeurs requises reçues du "Building Manager". De plus, un actionneur envoie les valeurs de ses grandeurs physiques à la demande du "Building Manager". Dans ce qui suit, nous prenons l'exemple de l'entité "Lamp" étant un actionneur et nous listons les fonctionnalités que fournit le module de gestion de cette entité .

- Se connecter à l'entité "Building Manager".
- Envoyer les valeurs d'état (On/Off) et d'intensité (de la lumière) de la lampe.
- Modifier les valeurs de l'état et de l'intensité de la lampe dans le but de répondre aux requêtes reçues de l'entité "Building Manager"

— **Module de gestion d' une entité capteur : "Thermometer" et "Power Smart Meter" :**
Ce module comme l'indique la spécification de Abdennadher Abdennadher (2013) a principalement pour objectif d'envoyer périodiquement les valeurs des grandeurs physiques de l'entité (valeur de température pour le thermometre et valeurs d'énergie produite et consommées pour le compteur d'énergie intelligent). Nous avons ajouté le cas d'utilisation suivant : "Ecouter les caractéristiques de l'entité" c'est-à-dire que cette entité doit être capable de recevoir les nouvelles valeurs de ses carctéristiques après la modification éffectuée par le "Building Manager" suite à l'adaptation. Dans ce qui suit, nous prenons l'exemple de l'entité "Power Smart Meter" étant un capteur et nous listons les fonctionnalités que fournit le module de gestion de cette entité.

- Se connecter à l'entité "Building Manager".
- Ecouter les périodes d'envoi et de mesure.
- Ecouter la valeur de l'énergie produite et consommée.
- Calculer les périodes d'envoi et de mesure.
- Envoyer (périodiquement) les valeurs de l'énergie produite et consommée à l'entité "BuildingManager". Afin d'améliorer la gestion de consommation d'énergie, nous avons proposé que l'envoi dépend non seulement de la valeur de la période d'envoi mais aussi de la valeur de la grandeur physique envoyée précedemment. C'est-à-dire que l'envoi n'aura lieu que si la valeur de la grandeur physique envoyée précédemment et la nouvelle valeur mesurée sont différentes.

— **Module de gestion de l'entité "User Agent" :**
Ce Module ne subit aucun changement par rapport à la specification de Abdennadher (2013). L'entité "User Agent" représente l'interface de l'utilisateur du "Smart Building". Elle envoie des requêtes de consultation et de modification des caractéristiques et des grandeurs physiques des entités ("Lamp" et "Air Conditioner"). Dans ce qui suit, nous listons les fonctionnalités fournies par le module de gestion de l'entité "User Agent".

- Se connecter à l'entité "Building Manager".
- Envoyer des requêtes de consultation et des messages de mise à jour des valeurs des grandeurs physiques et des caractéristiques des entités "Lamp" et "Air Conditioner").
- Ecouter les valeurs des caractéristiques envoyées périodiquement depuis les entités capteur (température, énergie produite et énergie consommée).

— Calculer les périodes d'envoi et de mesure.
— Envoyer (périodiquement) les valeurs de l'énergie produite et consommée à l'entité "BuildingManager". Afin d'améliorer la gestion de consommation d'énergie, nous avons proposé que l'envoi dépend non seulement de la valeur de la période d'envoi mais aussi de la valeur de la grandeur physique envoyée précedemment. C'est-à-dire que l'envoi n'aura lieu que si la valeur de la grandeur physique envoyée précédemment et la nouvelle valeur mesurée sont différentes.

4 Mise en œuvre

Dans cette partie, nous présentons la mise en œuvre de notre cas d'étude et sa validation. Nous nous interessons à présenter et détailler le diagramme de classes pour chaque module et, puis, expliquer les différentes phases qui permettent à l'application de gestion du "Smart Building" d'être un système auto-adaptatif.

4.1 Diagrammes de classes du cas d'étude "Smart Building"

La conception est une étape primordiale dans le cycle de vie d'une application en ce basant sur différents langages de modélisation tel que les graphes (Bouassida Rodriguez et al. (2008)), les Bi-graphs (Gassara et al. (2017)), l'UML, etc. La conception a pour objectif d'élaborer à partir des besoins obtenus et des solutions proposées suite à l'étape de l'étude préable, des diagrammes détaillés de l'architecture de l'application.

Dans cette section, nous présentons les diagrammes de classes des modules de gestion des différentes entités de notre application "Smart Building" que nous avons implanté en utilisant Java, Eclipse et OSGi. Nous nous basons sur la conception de Abdennadher (2013) avec des modifications et quelques améliorations dans le but de rendre le système auto-adaptatif.

4.1.1 Diagramme de classes du module de gestion de l'entité "Building Manager"

La figure 1 présente le diagramme de classes du module de gestion de l'entité "Building Manager". La classe SmartBuildingDaemon est le point d'entrée de l'entité "Building Manager". Cette dernière crée une instance de la classe SmartBuildingListener qui gère la réception des messages envoyés des autres entités à travers la methode receiveNewMessage(). Dans l'objectif de réaliser l'adaptation, nous avons ajouté une classe SmartBuildingManager. Cette classe est responsable de traiter les messages (requêtes et réponses) à l'aide de la méthode manageMsg() (qui utilise l'attribut msgList composé des clés de type String et des valeurs de type Message)et contient aussi le traitement d'adaptation réalisé par les méthodes (increasePeriod(), reducePeriod, changeCharacteristics() et restoreCharacteristics()). La communication entre le module de gestion de l'entité "Building Manager" et les modules de gestion des autres entités est assurée par un objet de la classe ClientConnection à travers les methodes receiveNewMessage() et la methode sendToEntity(). La classe SmartBuildingListener appelle la méthode manageMsg() en prenant en argument la méthode receiveNewMessage(). Ceci nous a poussé à ajouter une classe abstraite Message qui contient la méthode handle() contenant le traitement convenable de chaque message. Pour améliorer le fonctionnement de la classe

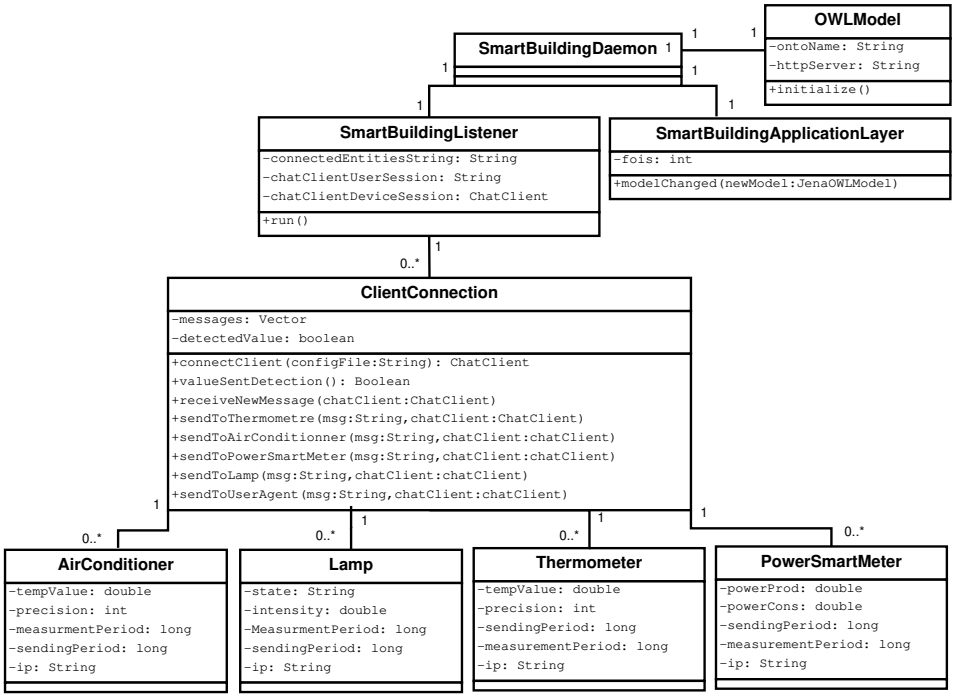

FIG. 1 – *Diagramme de classes du module de gestion de l'entité "Building Manager"*

ClientConnection, nous avons fusionné toutes les méthodes permettant l'envoi en sendToEntity(). Ceci engendre un changement au niveau des entités qui consiste à ajouter une classe Entity contenant un attribut nom et un attribut name. Nous avons effectué d'autres modifications sur le niveau conceptuel par rapport au diagramme de Abdennadher (2013).

4.1.2 Diagramme de classes du module de gestion de l'entité "User Agent"

Ce module ne subit aucun changement au niveau de son fonctionnement. Toutefois, l'organisation de ses classes a été modifiée. La classe UserAgentDaemon est le point d'entrée de l'entité "User Agent". Cette classe crée une instance de la classe UserAgentUI qui permet de gérer l'interface fournie à l'utilisateur du "Smart Building". Cette interface assure la représentation de l'ensemble des entités actives et de leurs fonctionnalités. La classe UserAgentListener détecte les entités qui deviennent actives dans l'applicationet met à jour l'interface du module de gestion de l'entité "User Agent". La classe UserAgent permet de suivre le changement de l'état et des caractéristiques des différentes entités. La figure 2 présente le diagramme de classes du module de gestion de l'entité "User Agent". Nous avons effectué d'autres modifications sur le niveau conceptuel par rapport au diagramme de Abdennadher (2013).

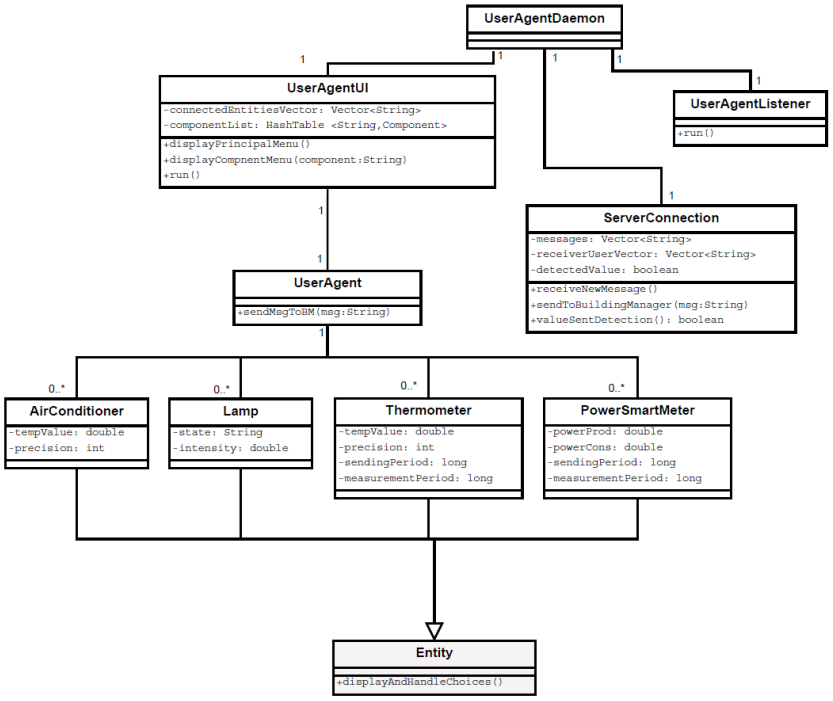

FIG. 2 – *Diagramme de classes du module de gestion de l'entité "User Agent"*

4.1.3 Diagramme de classes du module de gestion de l'entité "Thermometer"

La classe ThermometerDaemon est le point d'entrée de l'entité "Thermometer". Cette classe crée une instance de la classe ThermometerListener. Cette dernière permet l'écoute des caractéristiques du thermomètre (measurementPeriod, sendingPeriod et precision). La classe ThermometerDaemon aussi crée une instance de la classe ThermometerManager qui est responsable de l'envoi périodique les caractéristiques du thermomètre (tempValue) déjà mesurées par le thermomètre.

La classe ServerConnection assure la communication entre les modules de gestion du thermomètre et l'entité "Building Manager" à travers la méthode receiveNewMessage() et la methode sendToBuildingManager(). La figure 3 illustre le diagramme de classes du module de gestion de l'entité "Thermometer". Nous avons effectué d'autres modifications sur le niveau conceptuel par rapport au diagramme de Abdennadher (2013) .

4.1.4 Diagramme de classes du module de gestion de l'entité "Air Conditioner"

Pour ce module nous avons supprimé les attributs des périodes de mesure et d'envoi par rapport à la conception de Abdennadher Abdennadher (2013). La classe AirCoditionerDaemon est le point d'entrée de l'entité "Air Conditioner". Cette classe crée une instance des classes

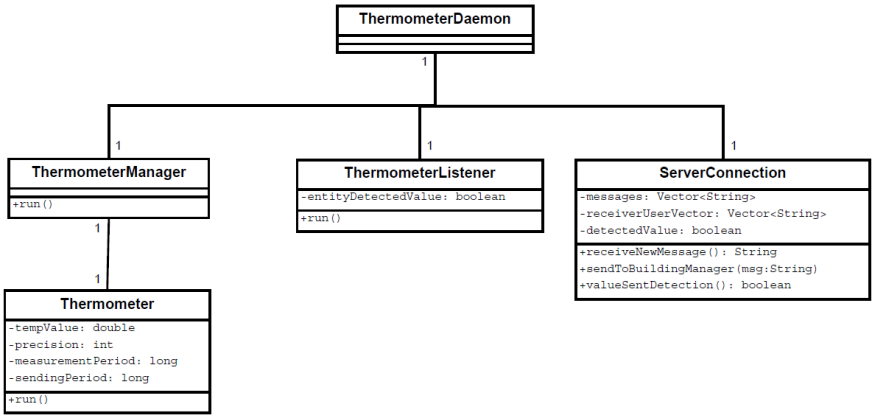

FIG. 3 – *Diagramme de classes du module de gestion de l'entité "Thermometer"*

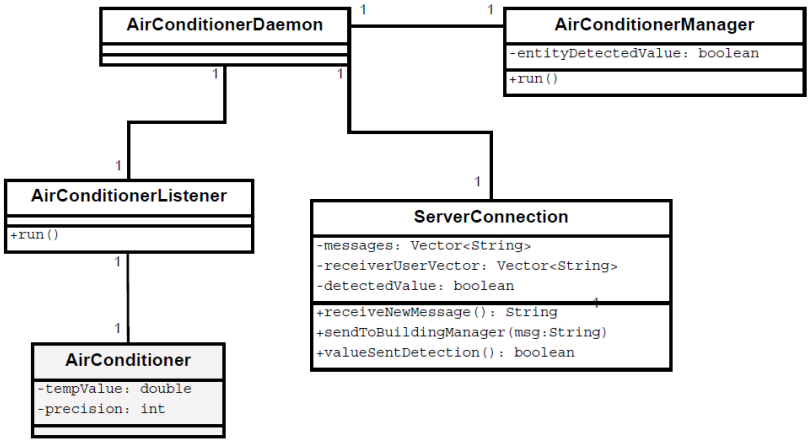

FIG. 4 – *Diagramme de classes du module de gestion de l'entité "Air Conditioner"*

AirConditionerListner qui est responsable d'écoutee des caractéristiques du Climatiseur (tempValue).

La classe ServerConnection assure la communication à travers la méthode receiveNewMessage() et la methode sendToBuildingManager(). En revanche, AirConditionerManager

qui est responsable de gérer les requêtes et les réponses échangées entre l'entité climatiseur et le gestionnaire du bâtiment. La figure 4 illustre le diagramme de classes du module de gestion de l'entité "Air Conditioner". Nous avons effectué d'autres modifications sur le niveau conceptuel par rapport au diagramme de Abdennadher (2013).

4.2 Scénario d'adaptation de gestion du bâtiment intelligent "Smart Building"

En effet, ce scénario d'adaptation, ayant comme objectif d'économiser la consommation de l'énergie dans le bâtiment, s'éxecute en trois phases.

4.2.1 Phase 1 : Fonctionnement sans intervention de l'utilisateur et sans adaptation

Cette phase se fait d'une manière automatique, elle comporte seulement les évènements qui se déroulent systématiquement suite à la connexion des entités sans intervention ni d'utilisateur ni du gestionnaire du bâtiment tel que l'envoi périodique effectué par les entités considérées comme étant des capteurs. Suite à leur connexion à l'entité "Building Manager", l'entité "Thermometer" et l'entité "Power Smart Meter" commencent à mesurer la température et l'énergie produite et celle consommée et à envoyer les valeurs mesurées au gestionnaire du bâtiment. Le processus de mesure se repète d'une façon périodique selon une période de mesure. En revanche, dans le but d'optimisation de la consommation d'énergie, le processus d'envoi dépend non seulement de la période d'envoi mais aussi d'une comparaison entre la valeur envoyée précedemment et la nouvelle valeur mesurée. C'est à dire dans le cas où la valeur précédemment envoyée et la nouvelle valeur mesurée sont égales alors l'entité capteur n'effectue pas un nouveau envoi.

4.2.2 Phase 2 : Fonctionnement avec intervention de l'utilisateur et sans adaptation

Cette phase se déclenche comme conséquence système et provoque un changement de contexte. à une intervention d'utilisateur.

D'abord, l'utilisateur allume la lampe avec une intensité qu'il la désire et demande une température au climatiseur. Les valeurs requises sont envoyées par l'agent utilisateur au gestionnaire du bâtiment. Le gestionnaire du bâtiment transmet ces valeurs aux entités concernées et ces entités s'éxecutent pour effectuer ces demandes. Ceci engendre, alors une augmentation de consommation d'énergie dans le bâtiement.

4.2.3 Phase 3 : Fonctionnement avec intervention de l'utilisateur et avec adaptation

Cette phase est une conséquence d'un changement (augmentation de la consommation d'énergie). A chaque valeur envoyée par le compteur d'énergie intelligent, le gestionnaire du bâtiment compare la valeur d'énergie produite et celle consommée. Ceci déclenche l'adaptation. Dans une première étape, le gestionnaire du bâtiment augmente les périodes de mesure et d'envoi des entités capteurs et leur envoie des messages contenant ces valeurs. Cet opération se répète tant que l'augmentation des valeurs initiales de périodes ne dépasse pas dix pour cent. Lorsque cette condition est réalisée, dans une deuxième étape, le "Building Manager" effectue des calculs de changements des grandeurs physiques des entités actionneurs (augmentation de température et diminution d'intensité), en prenant le confort de l'utilisateur en considération. Ensuite, il envoie les valeurs après changements aux entitées cibles pour une mise à jour. Suite à une diminution de consommation d'énergie (énergie consommée $<$ énergie produite), le "Building Manager" rétablit toutes les valeurs qui ont subi des changements et envoie des messages aux entités concernées pour une mise à jour.

Ces scénarios illustrent l'adaptation du comportement du bâtiment grâce à l'execution du "Smart Building". Suite à la connexion des différentes entités, une intervention de l'utilisateur mène à un changement de contexte. Le système réagit face à ce changement en tenant compte de son confort et l'optimise la consommation de l'énergie dans le bâtiment.

5 Conclusion

Dans notre travail, nous nous sommes focalisés sur l'adaptation du comportement d'un bâtiment intelligent dans l'objectif d'optimiser sa consommation en énergie. En premier lieu, nous avons effectué dans le première partie une explication des notions de base en s'appuyant sur notre étude des traveaux de recherche liés au domaine des Smart Spaces. Dans la deuxième partie nous nous somme intéressés à l'application "Smart Building" développée par Imen Abdennadher. Nous avons ajouté des fonctionnalités permettant l'adaptation. Finalement, Nous avons mis en oeuvre l'entité logicielle gestionnaire du bâtiment. Dans la dernière partie, nous avons présenté les diagrammes de classes nécessaires pour chaque entité et les scénarios expliquant un exemple d'éxecution de notre travail. Pour implanter l'application, nous avons utilisé Java comme language, Eclipse comme IDE et OSGi comme étant un framework.

En perspective, nous envisageons à court terme, à étendre notre travail pour supporter une interface graphique pour l'utilisateur de l'application. Nous visons également à développer une interface d'administration pour la configuration des paramètres de l'entité "BuildingManager". A long terme, nous nous intéressons au déploiment réel de l'application.

Références

Abdennadher, I. (2013). Environnement logiciel pour le déploiement dynamique de l'application smart building. Master's thesis, ISIMS, Sfax University, Tunisia.

Abowd, G. D., A. K. Dey, P. J. Brown, N. Davies, M. Smith, et P. Steggles (1999). Towards a better understanding of context and context-awareness. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, London, UK, UK, pp. 304–307. Springer-Verlag.

Bouassida Rodriguez, I., N. Van Wambeke, K. Drira, C. Chassot, et M. Jmaiel (2008). Multilayer coordinated adaptation based on graph refinement for cooperative activities.

Gassara, A., I. B. Rodriguez, M. Jmaiel, et K. Drira (2017). A bigraphical multi-scale modeling methodology for system of systems. *Computers and Electrical Engineering 58*, 113 – 125.

Oreizy, P., M. Gorlick, R. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Q. D. Rosenblum, et A. Wo (1999). An architecture-based approach to self-adaptive software. *IEEE Intelligent Systems 14*(3), 54–62.

Perera, C., A. B. Zaslavsky, P. Christen, et D. Georgakopoulos (2013). Context aware computing for the internet of things : A survey. *IEEE Communications Surveys and Tutorials Journal*.

québécois de la langue française, O. (2003). Le grand dictionnaire terminologique [en ligne]. `http://www.granddictionnaire.com/ficheOqlf.aspx?Id_Fiche=8360445`. Accessed : 24-05-2013.

Raibulet, C. (2008). Facets of Adaptivity. In K. F. Ron Morrison, Dharini Balasubramaniam (Ed.), *Software Architecture, Second European Conference, ECSA 2008 Paphos, Cyprus, September 29-October 1, 2008 Proceedings*, pp. 342–345. Springer Berlin Heidelberg.

Saha, D. et A. Mukherjee (2003). Pervasive computing : A paradigm for the 21st century. *Computer 36*(3), 25–31.

Schilit, B. N. et M. M. Theimer. Disseminating active map information to mobile hosts. *Netwrk. Mag. of Global Internetwkg. 8*(5), 22–32.

Taktak, E., I. Abdennadher, et I. Bouassida Rodriguez (2016). An adaptation approach for smart buildings. In *2016 IEEE 18th International Conference on High Performance Computing and Communications ; IEEE 14th International Conference on Smart City ; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 1107–1114.

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

Summary

The Smart Spaces field is becoming increasingly important today especially with the smart computer devices. Applications belonging to this domain are called “smart” because of their ability to detect changes in their context and to adapt to these changing contexts. Nevertheless, the challenge for these applications goes beyond rationalizing energy consumption without affecting user comfort. In this work, we propose an adaptation process of a distributed application called “Smart Building” in order to reconcile the two above-mentioned constraints.