Gestion de la Dynamicité de l'Architecture Logicielle d'une "Smart City"

Ameni Hadrich*, Mariam Chaabane*
Ismael Bouassida Rodriguez*,**

*Université de Sfax, ReDCAD, 3038 Sfax, Tunisie
hadrich.ameni1@gmail.com,
**Centre de Recherche en Numérique de Sfax, 3021 Sfax, Tunisie
mariam.chaabane@redcad.org,bouassida@redcad.org

Résumé. Les "Smart Cities" utilisent la technologie pour créer un confort urbain durable à un moment où les ressources énergétiques sont épuisées. Le développement urbain pose la question de la durabilité des ressources, de l'adaptabilité des services, du confort des citoyens. Par conséquent, l'objectif de ce papier est de développer une "Smart City" dynamique, qui utilise des systèmes intelligents, économes en énergie et capables de s'adapter à l'environnement pour améliorer la qualité de vie. Le fonctionnement de ces systèmes est dépendant aux règles de réduction d'énergie.

1 Introduction

La "Smart City" est une ville qui collecte et utilise les données générées par ses habitants et son infrastructure, pour améliorer la qualité de vie. Elle intègre des services basés sur les technologies de l'information et de la communication (TIC), qui offrent une valeur ajoutée au service de la ville, via des équipements (tels que les capteurs, les actionneurs, etc). Ces équipements communiquent entre eux à travers un réseau qui permet d'optimiser la transmission des données. Ainsi, la "Smart City" offre des services urbains qui rendent la ville plus intelligente et plus économe .

Dans une "Smart City", la bonne gestion d'énergie est une chose très importante pour le fonctionnement des équipements déployés dans la ville, parce qui' ils ont une grande influence sur la qualité des services et par conséquent, sur le bien être des citoyens. Lorsque des contraintes énergétique surviennent, plusieurs événements produisent une perte de qualité de services. Comme par exemple la désactivation des lampadaires du au manque d'énergie peut augmenter le risque des accidents, la dégradation du service de qualité de l'air peut augmenter le risque de problèmes de santé pour les personnes qui ont des problèmes respiratoires.

Ainsi, pour remédier à ce problème, nous proposons un éclairage public adaptatif, un service pour surveiller la qualité de l'air, et des algorithmes utilisant des règles pour gérer la consommation d'énergie. Ces algorithmes aident à répondre aux questions suivantes : Comment adapter l'éclairage public ? Comment choisir le bon moment de réduction de l'énergie ? A quels services pouvons-nous affecter la réduction et comment ?

Les "Smart Cities" s'appuient sur l'utilisation de l'informatique et des nouvelles technologies de l'information et de la communication (TIC) pour améliorer l'efficacité de ses services.

Dans la premier section de ce papier, nous présentons notre approche "Smart City". La deuxième section présente l'état de l'art. Nous décrivons dans la troisième section l'approche de conception d'une "Smart City" auto-adaptative en présentant des diagrammes des composants et des diagrammes SCA (Service Component Architecture) de chaque service (Paik et al. (2017)). La dernière section présente la mise en oeuvre de la "Smart City", en utilisant des modèles BPMN (Business Process Model and Notation) pour décrire les fonctionnalités d'auto-adaptation.

2 Smart City

Selon Manon Bril "Les smart cities sont des espaces urbains qui utilisent les données issues de capteurs ainsi que les nouvelles technologies d'informatique et de communication (TIC) pour mieux consommer leurs ressources, faire des économies d'énergie, répondre plus efficacement à nos besoins, renforcer la sécurité et mieux gérer leur territoire à court terme"Bril (2016).

Notre "Smart City" comporte un ensemble des services permettant d'obtenir une ville dynamique. Nous citons :

Éclairage Public Notre système de d'éclairage public permet d'améliorer la qualité de l'éclairage dans la ville, par l'optimisation de l'intensité des lampadaires en fonction de la luminosité du jour et de la présence des piétons et des véhicules.

Qualité de l'air Notre système de qualité de l'air est nécessaire pour surveiller la qualité de l'air, en mesurant les quantités de quelques composés organiques volatils (COV) dans l'air, et donc de distinguer l'état de l'air "Bonne qualité", "Moyenne qualité", ou "Mauvaise qualité".

Gestion de l'énergie Notre système de gestion d'énergie permet de gérer la consommation de l'énergie dans la "Smart City" à partir d'un centre de contrôle en utilisant des algorithmes auto-adaptatifs pour rationaliser la consommation d'énergie.

3 État de l'art

Dans un contexte d'écologie et de bien être des citoyens, les "Smart Cities" s'appuient sur les technologies numériques (comme les capteurs, les réseaux,etc.) dans un but d'optimisation de qualité de la vie. Dans ce contexte, plusieurs travaux ont été proposé pour améliorer la qualité des service de la "Smart City".

Les travaux de Sanseverino et al. (2015) ont pour objectif de donner un aperçu du concept de ville intelligente et de définir quels sont actuellement les principaux domaines d'intervention envisagés au niveau européen. Ces travaux se concentrent sur l'éclairage public qui fait partie des secteurs les plus touchés par les actions intelligentes des administrations publiques. Les travaux se terminent par une proposition de modification du système d'éclairage public dans une petite ville de Sicile (Italie).

Travaux	auto-adaptation	économie en énergie	réduction d'énergie en cas de pic	assurance du confort de l'utilisateur
Sanseverino et al. (2015)	+	+	-	-
Mohandas et al. (2019)	+	+	-	-
Dutta et al. (2017)	-	+	-	-

TAB. 1: *Tableau comparatif des travaux existants*

Le travail de Mohandas et al. (2019) présente un système d'éclairage public intelligent écoénergétique. La conception proposée a été mise en œuvre et exécutée dans une zone résidentielle (Hosur) et les résultats sont réalisés selon différents scénarios. Ce travail présente cinq niveaux de scénarios sont testés et mis en œuvre en temps réel permettant d'éviter l'utilisation inutile de l'éclairage public. Son approche est basée sur les conditions et les contrôleurs flous, qui sont utiles pour contrôler les niveaux de luminance de la lumière dans l'éclairage public.

Le travail de Dutta et al. (2017) présente un système de surveillance de la qualité de l'air AirSense basé sur la détection de foule, visant à collecter et à agréger des données de capteurs afin de surveiller la pollution de l'air dans la ville. Ce travail introduit un dispositif de surveillance de la qualité de l'air (AQMD) léger et à faible consommation d'énergie et à faible coût.

Les travaux que nous avons cité se concentre sur l'intelligence des services offerts par la "Smart City", notamment, en terme de consommation d'énergie. Ces derniers présentent des modèles dynamique de "Smart Cities", ils discutent l'adaptabilité de l'éclairage publique, la collecte dynamique de déchets et le contrôle de qualité de l'air, pour offrir une bonne qualité de vie.

Dans la littérature, les critères d'évaluation d'une "Smart City" sont : L'auto-adaptation des services, la gestion de consommation d'énergie en cas de pic et l'assurance du confort de l'utilisateur.

Les travaux que nous avons étudié (TAB. 1) répondent seulement à l'auto-adaptation et l'économie d'énergie.

C'est très intéressants de trouver dans une "Smart City" des systèmes intelligents et économique en terme de consommation d'énergie. Mais ceci n'est pas suffisant en cas pic de consommation malgré ces systèmes économiques. Dans ce cas, il faut appliquer une réduction d'énergie tout en gardant une bonne qualité de service, qui assure le confort de l'utilisateur.

Notre objectif, est d'avoir une "Smart City" auto-adaptative, qui répond à tous les critères que nous avons cité dans le tableau 1. Donc, notre approche se concentre sur le fonctionnement dynamique des systèmes intelligents dans la "Smart City" et décrire comment garder une bonne qualité de service tout en garantissant le confort de l'utilisateur, lorsque une contrainte d'énergie est survenu.

Pour assurer l'auto-adaptabilité ainsi que les critères mentionnés dans la Table 1 dans notre "Smart City", nous avons choisi de modéliser la "Smart City"" en se basant sur l'architecture

orientée services en s'inspirant du travail de Chaabane et al. (2015).

4 Approche de conception de la "Smart City" auto-adaptative

La phase de conception est la phase préparatoire dans un projet, elle tient une place centrale pour la réalisation d'un projet en ce basant sur différents langages de modélisation tel que les Graphes (Bouassida Rodriguez et al. (2008)), les Bi-Graphes (Gassara et al. (2017)), le SCA (Paik et al. (2017)), etc. Pour atteindre nos objectifs cités dans la premier section, nous avons choisi de modéliser la "Smart City" d'un point de vue structurel et du point de vue opérationnel en s'inspirant du travail de Chaabane et al. (2017). Ainsi, nous avons utilisé le diagramme de composants pour présenter l'architecture logicielle de la "Smart City" et mettre en oeuvre le point de vue structurel. En plus, nous avons utilisé les modèles SCA (Service Component Architecture), qui est conçue pour fournir un modèle qui respecte les principes de l'architecture orientée services, pour présenter les différents équipements, leurs fonctionnement et les interactions entre eux et pour mettre en oeuvre le point de vue opérationnel.

Dans cette section, nous avons deux parties. La première partie donne une présentation générale de notre approche "Smart city". La deuxième partie donne une description détaillée des services de la "Smart City", avec les diagrammes des composants et les modèles SCA.

4.1 Présentation générale de la"Smart City"

Une "Smart City" est une ville qui collecte et utilise les données générées par ses habitants et son infrastructure pour améliorer la qualité de vie et optimiser les ressources (Low (2018)). Dans notre "Smart City", nous avons concentré sur le système d'éclairage public et le système de gestion d'énergie, pour garder une ville économique. Dans ce qui suit, nous avons présenter l'architecture générale du cas d'étude "Smart City" en présentant les différents équipements de la ville.

Dans ce travail, les services Web sont orchestrés sur plusieurs niveaux. Une unité de contrôle centrale qui est un orchestrateur de services Web (CUCity), gère un autre niveau d'unités de contrôle (CUPublicLighting et CUAirQuality) , chacune est spécifique à un seul service de "Smart City". Chacune de ces dernières gère un autre niveau d'unités de contrôle (CUAirQualitySensor et CULampPost) qui sont des orchestrateurs de service Web. Chacune de ces dernières orchestre un seul service Web (AirQualitySensor) ou un groupe de services Web (Lamp, LuminositySensor, PresenceSensor, Radar).

Ainsi, nous avons un client responsable à l'unité de contrôle centrale, qui permet d'envoyer des requêtes au moteur d'exécution (orchestrateur de services Web). Ensuite, ce dernier appelle les services Web suivant l'ordre d'exécution des tâches. L'orchestrateur de services Web retourne la réponse au client une fois qu'il termine l'exécution du processus entier.

4.2 Présentation des services de la "Smart City"

Notre "Smart City" est composé par un système de gestion d'énergie qui gère la consommation de toute la ville. La ville est décomposé en un ensemble de zones qui contiennent des différents systèmes intelligents. Chaque zone comporte un service d'éclairage public, un service de qualité de l'air.

4.2.1 Description du service d'éclairage public

Ce service est composé par des lampadaires et une unité de contrôle centrale d'éclairage public "CUPublicLighting" installé dans un centre d'éclairage.

Chaque lampadaire est équipé par une unité de contrôle "CULampPost" qui permet de gérer le fonctionnement de l'éclairage d'une lampadaire, un capteur de présence "Presence-Sensor" qui permet de détecter le passage des piétons, un radar "Radar" qui permet de détecter le passage des véhicule, un capteur de luminosité "LumnositySensor" qui permet de détecter la valeur de luminosité du jour et une lampe "Lamp".

L'unité de contrôle centrale d'éclairage public "CUPublicLighting" permet de gérer le fonctionnement de tous les lampadaires de la ville.

4.2.2 Description du service de la qualité de l'air

Ce service est composé par des stations de qualité de l'air et une unité de contrôle centrale "CUAirQuality" installé dans un centre de contrôle de qualité de l'air.

Chaque station est équipé par un capteur de qualité de l'air "AirQualitySensor", qui permet de détecter le niveau des quatre composants COV dans l'air (CO, CO2, NO2, O3) et une unité de contrôle "CUAirQualitySensor" qui permet de gérer le fonctionnement du système de qualité de l'air pour un seul capteur.

L'unité de contrôle centrale de qualité de l'air "CUAirQuality" permet de gérer toutes les stations de qualité de l'air.

Ce service, possède aussi une interface client nommé "AirQualityUserAgent" qui permet d'accéder au "CUAirQualitySensor" pour prendre la qualité de l'air.

4.2.3 Description du service de la gestion d'énergie

Ce service permet de gérer la consommation d'énergie depuis l'unité de contrôle centrale de la ville "CUCity" et un Administrateur "CityUserAgent" qui permet de gérer les seuils nécessaire et contrôler la consommation d'énergie. Nous avons besoin de prendre la quantité d'énergie consommées depuis un "Electric Power Utility".

4.3 Présentation des diagrammes de composants

Pour atteindre nos objectifs, nous avons choisi de décrire l'architecture de notre système, du point de vue des structurel avec des diagrammes de composants. Ces diagrammes agréent à mettre en évidence les dépendances entre les différents composants. Chaque composant fournit et requiert des services via des interfaces.

L'utilisation des diagrammes des composants nous a permis de présenter les interactions possibles entre les composants de la "Smart City", ainsi les interactions nécessaire pour avoir un fonctionnement adaptables des services.

Dans les diagrammes de composants, nous présentons les composants dans des nœuds pour les classer, en suivant trois modélisations :

— Les nœuds des actionneurs et des capteurs sont représentés par deux composants : Un "Daemon" qui présente les composants réels et un "Manager" qui présente les composants logiciels.

— Les nœuds des unités de contrôle appartenant au système d'éclairage public sont représentés par deux composants : Un "Listener" qui présente l'écouteur des flux d'entrées et un "Manager" qui présente le contrôleur du fonctionnement.
— Les nœuds des unités de contrôle appartenant au système de qualité de l'air sont représentés par un seul composant, qui est le "Manager".

Nous avons présenté les dépendance entre les composants en utilisant des interfaces offertes, qui sont des services implémentés par le composant et qui peuvent être utilisés par d'autres composants, et des interfaces requises, qui sont des services que le composant a besoin pour son fonctionnement.

4.3.1 Diagramme de composants du service d'éclairage public

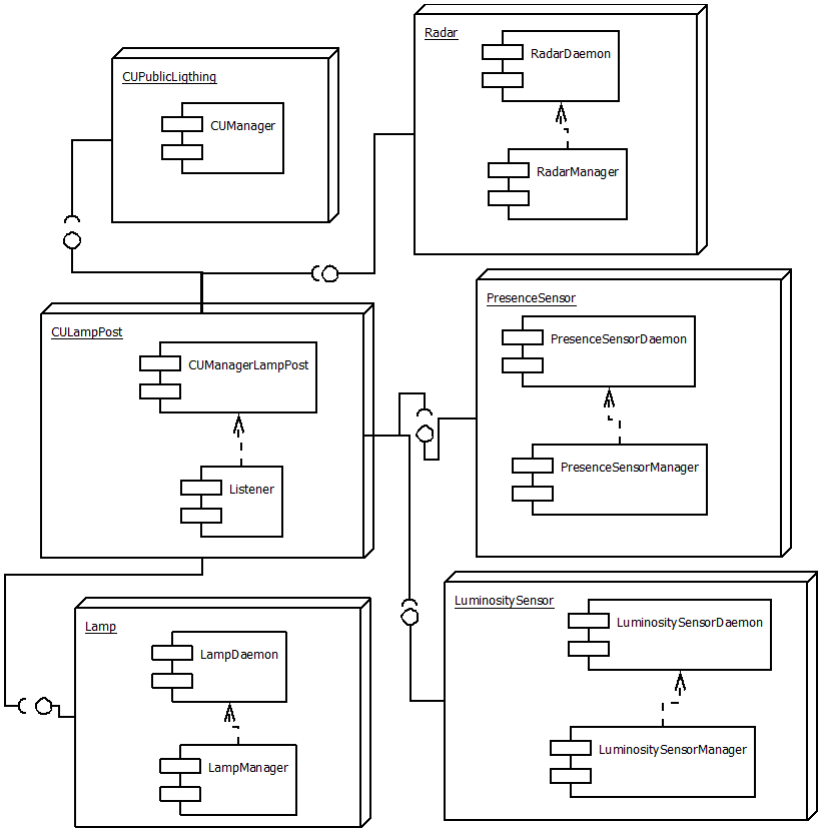

FIG. 1: *Diagramme de composants du système d'éclairage public*

Dans la Figure 1, nous avons présenté tous les composants du système d'éclairage public et les dépendances entre eux. Les composants que nous avons implémenté (Figure 1) sont :

— Le composant "Radar" qui fournit son service au composant "CULampPost".

— Le composant “PresenceSensor” qui fournit son service au composant “CULampPost”.
— Le composant “LuminositySensor” qui fournit son service au composant “CULampPost”.
— Le composant “Lamp” qui fournit son service au composant “CULampPost”.
— Le composant “CULampPost” qui fournit son service au composant “CUPublicLighting”.
— Le composant “CUPublicLighting” qui fournit son service au composant “CUCity”.

4.3.2 Diagramme de composants du système de gestion d’énergie

Dans le diagramme de la Figure 2, nous avons présenté les composants du système de gestion d’énergie et les dépendances entre eux. Les composants que nous avons implémenté (Figure 2) sont :

— Le composant “CUCity” qui fournit son service au composant “CityUserAgent”.
— Le composant “ElectricPowerUtility” qui fournit son service au composant “CUCity”.

Pour appliquer les réductions d’énergie, nous avons ajouté une interface “CityUserAgent” qui présente un Administrateur. Ce dernier permet d’enregistrer les seuils nécessaire pour déclencher la réduction d’énergie.

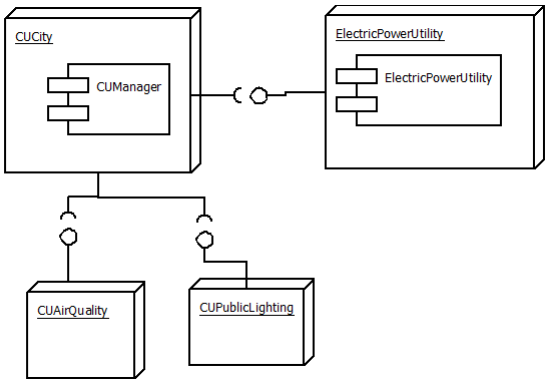

FIG. 2: Diagramme de composants du système de gestion de consommation d’énergie

En conclusion, les diagrammes de composants nous ont permis gérer la complexité, par encapsulation des détails d’implémentation. Ces diagrammes permettent, aussi, d’établir des différentes configurations liant les composants, pour obtenir des systèmes adaptables.

4.4 Présentation des modèles d’assemblage SCA

Le modèle d’assemblage SCA est un modèle qui permet de manipuler les assemblages entre les différents composants, en décrivant la composition de chacun, les services Web fournis et offerts et les interactions entre les composants. Ce modèle d’assemblage permet de fournir l’interopérabilité entre les divers services Web qui fonctionnent sur divers machines. D’où, il nous aide à illustrer le point de vue opérationnel.

Le modèle SCA dispose deux types de liaisons (bindings) :

— Liaisons SCA : nous les avons utilisé entre les composants dont ses "Software" se trouve dans la même machine.

— Liaisons RESTCroës (2011) : nous les avons utilisé entre les composants dont ses "Software" se trouve dans des machines distants.

4.4.1 Modèle d'assemblage SCA du système d'éclairage public

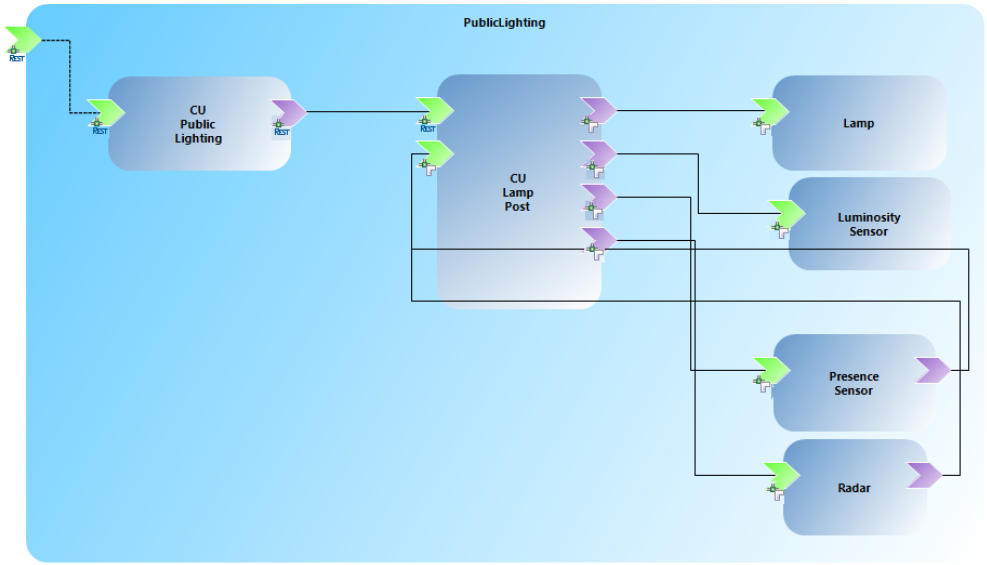

FIG. 3: *Modèle d'assemblage de l'éclairage public*

Le composite "PublicLighting"(Figure 3) est composé par un ensemble des composants qui portent des identifiants (Id).

Le composant "CUPublicLighting" expose un service au composant de l'unité de contrôle de la ville et requiert un autre service du composant "CULampPost". Ainsi, Le composant "CULampPost" expose un service au composant "PresensSensor" et un autre service au composant"Radar" plus que le service exposé au composant "CUPublicLighting". Le "CULampPost" requiert un ensemble de services Web :

— Le premier service est fourni par le composant de la lampe "Lamp".

— Le deuxième service est fourni par le composant du capteur de luminosité "LuminositySensor".

— Le troisième service est fourni par le composant du capteur de présence "PresenseSensor".

— Le quatrième service est fourni par le composant du radar "Radar".

Le modèle SCA de la Figure 3 dispose les deux types de liaisons (bindings) : des liaisons SCA et des liaisons REST.

Dans notre cas, nous avons quatre liaisons SCA qui se trouvent entre :

— Le composant “CULampPost” et le composant “Lamp”.
— Le composant “CULampPost” et le composant “LuminositySensor”.
— Le composant “CULampPost” et le composant “PresenceSensor”.
— Le composant “CULampPost” et le composant “Radar”.

Une liaison REST se trouve entre le composant “CUPublicLighting” et le composant “CU-LampPost”.

4.4.2 Modèle d’assemblage SCA du système de gestion d’énergie

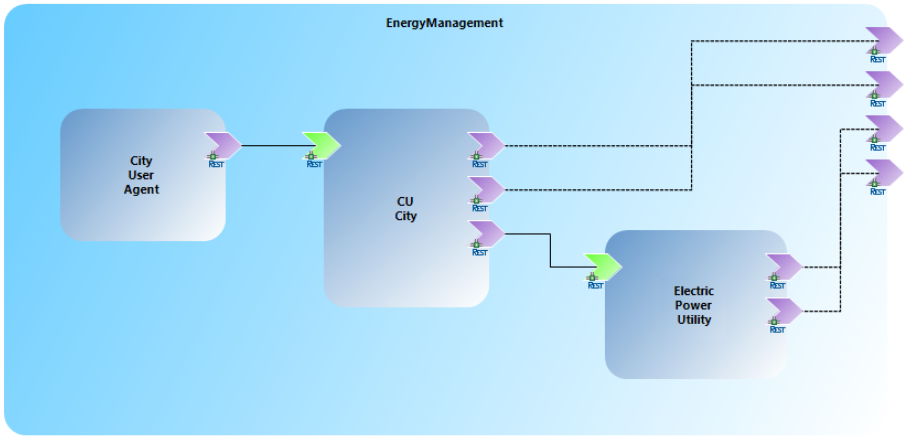

FIG. 4: Modèle d’assemblage SCA de la gestion de consommation d’énergie

Dans la Figure 4, nous présentons un modèle manipulant les assemblages entre les différentes composants nécessaire pour gérer la consommation d’énergie de la ville en décrivant les services fournis et offerts. Le modèle SCA (Figure 4) est composé par un ensemble des composants qui portent des identifiants (Id) :

Le composant “CityUserAgent” requiert un service du composite “CUCity”. Ce dernier expose un service au composant “CityUserAgent” et requiert trois autres services :

— Le premier service requis par le composite “PublicLighting” pour appliquer la réduction du service de l’éclairage public
— Le deuxième service requis par le composite “AirQuality” pour appliquer la réduction du qualité de l’air.
— Le troisième service requis par le composant “Electric Power Utility” pour prendre la quantité d’énergie consommée.

Le composant “ElectricPowerUtility” expose un service au composant “CUCity” et requiert deux services. Le premier service est exigé du composite “PublicLighting” et le deuxième service est exigé du composite “AirQuality” .

Le modèle SCA ci-dessous (Figure 4) dispose seulement des liaisons REST qui se trouvent entre :

— Le composant “CityUserAgent” et le composant “CUCity” .
— Le composant “CUCity” et le composite “PublicLighting”.

— Le composant "CUCity" et le composite "AirQuality".
— Le composant "CUCity" et le composant "ElectricPowerUtility".
— Le composant "ElectricPowerUtility" et le composite "PublicLighting".
— Le composant "ElectricPowerUtility" et le composite "AirQuality".

5 Mise en oeuvre de la "Smart City"

Dans cette section nous présentons les liaisons entre les opérations de chaque composants dans la ville pour clarifier le fonctionnement de chaque service d'un point de vue implémentation, en se concentrant sur la description d'algorithmes auto-adaptatifs tels que : Le teste d'intensité des lampadaires, la gestion d'énergie, etc. Nous détaillons aussi, le fonctionnement dynamique de la gestion d'énergie, tels que : les conditions du lancement de la réduction de consommation d'énergie, le bon choix du système à réduire, le bon moment de restauration, représenté par les fonctions handle() et reductionConsumptionEnergy().

Le diagramme de la Figure 5 de l'Annexe A illustre les liaisons entre les composants d'éclairage public et les composants de qualité de l'air. Il agrée aussi l'adaptabilité du service d'éclairage public et les algorithmes d'auto-adaptation qui permettent de diminuer la consommation d'énergie tout en gardant le confort de l'utilisateur.

Dans ce qui suit, nous présentons seulement les fonctionnalités qui mettent en oeuvre l'aspect de dynamicité et auto-adaptabilité pour la gestion d'énergie dans la "Smart City".

5.1 Fonctionnement du service de la gestion d'énergie

Le composant "CUCity" contient des opérations de paramétrage, fourni au "CityUserAgent", tel que l'opération (setStabilityThreshold(), setRateLamp(id, rate), setRateRadar(id,rate), etc) pour entrer les valeurs de paramètre nécessaires pour lancer une réduction d'énergie ou une restauration d'état initiale.

Le composant "CityUserAgent" permet de lancer l'opération (Handle()), qui permet de gérer la consommation d'énergie, en invoquant un ensemble d'opérations, à chaque 15 minutes.

L'opération (Handle()) invoque l'opération (getConsumptionEnergy(id)) du composant "ElectricPowerUtility" pour prendre l'énergie consommée et invoque l'opération (getConsumptionThreshold()) pour prendre le seuil à partir de lui et déclencher la réduction.

L'opération (getConsumptionEnergy(id)) du composant "ElectricPowerUtility" invoque l'opération (getConsumptionEnergyPL(id)) du composant "CUPublicLighting" et l'opération (getConsumptionEnergyAQ(id)) du composant "CUAirQuality", pour prendre la consommation d'énergie de ces deux services.

Pour réduire l'énergie dans la "Smart City", nous avons deux services :

— La réduction de la consommation d'énergie du service de la qualité de l'air se faite par l'invocation de l'opération (reduceEnergyConsumption(action)) dont action égale à "reduce", du composant "CUAirQuality", qui permet à son tour d'invoquer l'opération (reduceEnergyConsumption(id,action)),du composant "CUAirQualitySensor", qui permet de fermer un capteur de qualité de l'air en invoquant les opérations (getState(id)) et (setState(id,state)) du composant "AirQualitySensor".
— La réduction de la consommation d'énergie du service de l'éclairage public se faite par l'invocation d'opération (reduceEnergyConsumption(action,level)) dont action égale à

"reduce", du composant "CUPublicLighting", qui permet à son tour d'invoquer l'opération (reduceEnergyConsumption (id,action,level,Rrate), du composant "CULamp-Post", qui permet de diminuer l'intensité de la lampe, si le niveau égale à 1 et de fermer la lampe, si le niveau égale à 2, en invoquant les opérations (getState(id), (On(id,I) et (Off(id)) du composant "Lamp" et les opérations (setState(id, state)) du composant "PresenceSensor" et (setState(id, state)) du composant "Radar".

L'opération (Handle()) invoque l'opération (getStabilityThreshold()) pour prendre le seuil à partir de lui, on restaurer l'état initiale (c-à-d augmenter l'énergie). L'augmentation de l'énergie dans la "Smart City", se faite par l'invocation des mêmes opérations, qu'on a invoqué pour la réduction, mais par un changement de paramètre action égale à "rise".

5.2 Représentation des fonctionnalités d'auto-adaptation de la gestion d'énergie par la modélisation BPMN

Pour atteindre nos objectifs, nous avons choisi de représenter les fonctionnalités d'auto-adaptation de notre "Smart City" en utilisant la représentation graphique permettant de spécifier des processus métier BPMN (Lucidchart).

Dans cette section, nous présentons en BPMN, les opérations importantes qui présentent l'auto-adaptabilité et la dynamicité des services de gestion d'énergie dans la "Smart City".

Le fonctionnement de l'opération "handle" du pool "CUCity" :

Cette opération permet de gérer la consommation d'énergie dans la ville, en utilisant des niveaux de réduction pour garantir le confort utilisateur lorsque une contrainte d'énergie survenu.

Le lancement de cette opération (Figure 6 dans l'Annexe B) est basé sur le temps. Ainsi, toutes les 15 minutes, nous prendrons le seuil de pic consommation (CT) et l'énergie consommée (CE) à partir d'un "Electric Power Utility", puis nous comparons ces deux valeurs.

— Le premier cas est : Si (CE) dépasse (CT), nous commençons une boucle, pour appliquer la réduction de la consommation du service de qualité de l'air pour toutes les zones de la ville, en invoquant l'opération (reduceConsumptionEnergy(id,action)), qui se trouve dans le couloir de CUAirQuality, avec (id=VZone.elementAt(1) et action= "reduce"), mais en vérifiant d'abord la variable booléenne (reductionAirQuality), qui nous a permis de savoir si nous avons appliqué la réduction de ce service ou non. (Remarque : VZone est un vecteur comprendre, à chaque itération, dans la premier case l'identifiant du "CUPublicLighting" et dans la deuxième case l'identifiant du "CUAir-Quality" correspond au i ème zone).
Lorsque la réduction est terminée, c'est à dire que l'opération (reduceConsumptionEnergy(id, action)) renvoie l'action "reduce", nous incrémentons la variable (i), pour passer à réduire les autres zones. Si (i) devient supérieure à (nbZone), autrement dit que la réduction de toutes les zones est terminé, la variable booléenne (reductionAir-Quality) devient vraie.
Ensuite, on passe une autre fois, pour prendre la valeur de l'énergie consommée (CE).

— Si (CE) supérieur à (CT) : Nous passons à la réduction du niveau 1 du service d'éclairage public pour toutes les zones de la ville, en invoquant l'opération (reduceConsumptionEnergy(id, action, level)), du couloir de CUPublicLigthing, avec les paramètres

suivants : (id=VZone.elementAt(0), action="reduce" et level= 1), mais en en vérifiant, d'abord, la variable booléenne (reductionPubliclighting1) qui nous a permis de savoir est ce que nous avons appliqué la réduction du niveau 1 de ce service ou non. Lorsque la réduction du niveau 1 de la première zone est terminé , c'est à dire que l'opération (reduceConsumptionEnergy (id,action, level)) retourne "reduce 1", nous passons à réduire la consommation d'autres zones, en incrémentant la variable (i), pour continuer à réduire les autres zones. Si (i supérieur à nbZone), la variable booléenne (reductionPublicLighting1) prend vraie.

- Si (CE) est encore supérieure à (CT) : Nous passons à suivre le processus de la réduction du deuxième niveau du service d'éclairage public en invoquant l'opération (reduceConsumptionEnergy(id,action, level)) avec les paramètres (id=VZone.elemntAt(0), action="reduce" et level = 2), après la vérification de la variable booléenne (reduction-Publiclighting2). Lorsque la réduction du deuxième niveau du premier zone est terminée, nous passons à appliquer la réduction sur les autres zones. A la fin, nous affectons "vraie" à la variable (reductionPubliclighting2) et le processus de la réduction terminé.
- Le second cas est : Si (CE) inférieur à (CT), nous prenons le seuil de stabilité (ST) et le comparer par (CE).
  - Si (CE) inférieur à (ST) : Nous vérifions que la variable booléenne (reduction-Public2) est vraie, c'est à dire que la réduction de ce niveau est déja faite, nous passons, ensuite à invoquer l'opération (reduceConsumptionEnergy(id, action, level)), avec les paramètres (id= VZone.elemntAt(0), action= "rise" et level= 1). Lorsque l'augmentation est terminée, c'est-à-dire l'opération (reduceConsumptionEnergy(action, level)), retourne "rise 2" , nous passons à changer la variable (reductionPublicLighting2 = false).
    Ensuite, nous reviendrons pour prendre la valeur de l'énergie consommée (CE).
  - Si (CE) inférieur à (ST), nous continuons à augmenter le niveau 1 du service d'éclairage public en suivant les mêmes étapes que celles de l'augmentation précédente, avec des modifications en paramètres.
  - Si (CE) supérieur ou égal à (ST) : le processus d'augmentation est finie.

En conclusion, la réduction et l'augmentation ont le même principe, sauf que la réduction, commence par le service qui a le moins d'importance (qualité de l'air) et l'augmentation commence par le service qui a le plus d'importance (le deuxième niveau de l'éclairage public).

Le fonctionnement de l'opération "reductionConsumptionEnergy" du pool "CULamp-Post" :

Cette opération permet de gérer la consommation d'énergie pour une seul lampadaire. Elle permet de présenter la réduction d'énergie sur deux niveaux pour assurer une bonne qualité d'éclairage.

Le processus (Figure 7 dans l'Annexe C) commence par une invocation d'opération. La première tâche à faire est la vérification de l'action.

- Si action= "reduce" : Nous passons à tester le niveau.
  - Si level= 1 : nous prenons l'état de la lampe et on le mettre dans une variable nommée (stateLamp).
    Si nous trouvons que la lampe allumée (state="On"), nous appliquons une réduction de (Rrate) sur la valeur de l'attribut (ComfortableLuminosity). Ensuite, nous

passons à régler l'intensité de la lampe, en prenant la valeur de luminosité détectée en lemens, du couloir "LuminositySensor" en la mettant dans une variable (SV-LuminositySensor), puis nous invoquons l'opération (regulateIntensity(id, CL, L)) avec le paramètre (id=VLPComponent.elmentAt(1)), le paramètre (CL= ComfortableLuminosity) et le paramètre (L=SVLuminositySensor), pour régler l'intensité de la lampe en Luxe, en suivant l'équation suivante :

$$E(lx) = P(lms)/A(m^2)$$

Avec P(lms) l'intensité de la lampe en lemens et $A(m^2)$ le surface entre les lampadaire (dans notre cas d'étude, nous mettons $A(m^2) = 15^2$).
(Remarque : VLPComponent est un vecteur qui contient, à la fois, les identifiants des composants d'une lampadaire selon l'identifiant du "CULampPost" donné, dont sa première case contient l'identifiant de la lampe, la deuxième case contient l'identifiant du capteur de luminosité, la troisième case contient l'identifiant du capteur de présence et la quatrième case contient l'identifiant du radar).
A la fin, nous retournons le niveau au couloir du "CUPublicLighting" et le processus est terminé.
Si nous trouvons que la lampe est fermée (state="Off"), nous réduisons uniquement la valeur de (ComfortableLuminosity).
Sinon, c'est à dire que la lampe est en panne (state="down'), le processus est terminé.

- Si level= 2 : Nous fermons tous les composants du lampadaire qui sont (PresenceSensor, Radar, LuminositySensor et la lampe) par l'invocation des opération (setState(id,state)) avec le paramètre (id) prend à chaque fois l'identifiant du composant et le paramètre (state = "Off"), puis nous terminons le processus.

- Si l'action égale à "rise" : Nous passons à tester le niveau.
  - Si level= 1 : nous prenons l'état de la lampe.
    Si la lampe est allumée, nous appliquons une augmentation de (Rrate) sur la valeur de l'attribut (ComfortableLuminosity). Ensuite, nous appliquons le sous processus (regulateIntensityLamp) et le processus est terminé.
    Si la lampe est fermée, nous augmentons que la valeur de (ComfortableLuminosity).
    Sinon, le processus est terminé.
  - Si level= 2, nous allumons tous les composants de la lampadaire (PresenceSensor, Radar et LuminositySensor) et on termine le processus.
    En conclusion, le processus de réduction d'énergie, montre l'auto-adaptabilité du service de la consommation d'énergie. La suivi de plusieurs niveaux de réduction, nous permettre à garantir une bonne qualité de service, tout en adaptant aux contraintes énergétique.

5.3 Implémentation des services de la "Smart City"

Dans cette section, nous présentons, les différentes étapes pour implémenter un système de la "Smart City".

5.3.1 Implémentation du système d'éclairage public

Dans cette phase, Nous avons implémenté en premier lieu, les services Web et en deuxième lieu, les orchestrateurs de services Web, à partir des classes Java.

Création des Services Web Dans ce système, nous avons pour chaque zone de la ville douze services Web, décomposés en des groupe de quatre service Web. Chaque groupe de service Web présente une lampadaire, il contient un capteurs de présence, un radar, une lampe et un capteur de luminosité.

Création des orchestrateurs de Services Web Dans ce système, nous avons trois orchestrateurs de services Web nommés comme suit "CUPublicLightingZ1", "CUPublicLightingZ2", "CUPublicLightingZ3" . Chaque orchestrateur d'une zone orchestre un trois autres orchestrateurs de services Web. Chaque orchestrateur de ces derniers orchestre les quatre services Web d'une lampadaire.

5.3.2 Implémentation du système de gestion d'énergie

Dans cette phase, Nous avons implémenté en premier lieu, les orchestrateurs de services Web et en deuxième lieu, le client de service Web, à partir des classes Java.

Création des orchestrateurs de Services Web Dans ce système, nous avons deux orchestrateurs de services Web.

- Un orchestrateur de service Web "CUCity" qui orchestre trois orchestrateurs du service de qualité de l'air responsables à trois zones (CUAirQualityZ1, CUAirQualityZ2, CUAirQualityZ3) et trois orchestrateurs du service d'éclairage public responsables à trois zones (CUPublicLightingZ1, CUPublicLightingZ2, CUPublicLightingZ3).
- Un orchestrateur de Service Web "ElectricPowerUtility" qui orchestre les trois orchestrateurs du service de qualité de l'air et les trois orchestrateurs du service d'éclairage public.

Création du client de Service Web Dans ce système, nous avons implémenté un client java qui présente l'administrateur "CityUserAgent", qui permet d'enregistrer les seuils nécessaire pour la réduction.

6 Conclusion et perspectives

Initialement, la "Smart City" est une ville qui utilise les nouvelles technologies de l'informatique pour son développement.

Dans ce contexte, nous pensons que l'intelligence s'occupe, précisément, par l'optimisation des services et l'adaptation de la ville aux habitants au maximum. Ceci nécessite une bonne gestion d'énergie. Dans une "Smart City", lorsque des contraintes énergétique surviennent, plusieurs événements produisent une perte de qualité de services, à savoir la désactivation des lampadaires, la désactivation des capteurs de qualité de l'air, ou le manque d'énergie qui peut augmenter des risques sur le bien être des citoyens.

Ce papier a définit la “Smart City” comme une ville économique et dynamique, dans le but d'améliorer la qualité de vie de ses habitants, en développant des services auto-adaptables, des algorithmes d'auto-adaptabilité pour la réduction de la consommation d'énergie, à l'aide d'un découpage intelligent en zone, tout en gardant une bonne qualité de service.

Pour atteindre cet objectif, nous avons conçu notre “Smart City” à l'aide des diagrammes de composants qui illustre l'aspect structurel et les modèles SCA qui illustre l'aspect opérationnel. Ensuite, nous avons détaillé les fonctionnalités d'auto-adaptabilité pour la gestion d'énergie en utilisant BPMN. Enfin, nous avons cité les étapes d'implémentation et création des Services Web et orchestrateurs.

En perspective, nous envisageons à moyen terme améliorer la gestion d'énergie en utilisant les zones de la ville, par l'application de la réduction sur les zones les moins peuplées. Pour la conception, nous prévoyons mettre chaque opération dans un service pour faciliter le changement, ou l'ajout des fonctionnalités sans entrer dans le code.

Références

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

Bril, M. (2016). Un jour nous vivrons dans des villes intelligentes. `https://sms.hypotheses.org/8615`.

Chaabane, M., I. Bouassida Rodriguez, et M. Jmaiel (2017). System of systems software architecture description using the iso/iec/ieee 42010 standard. In *Proceedings of the Symposium on Applied Computing*, SAC '17, pp. 1793–1798. ACM.

Chaabane, M., F. Krichen, I. Bouassida Rodriguez, et M. Jmaiel (2015). Monitoring of service-oriented applications for the reconstruction of interactions models. In *Computational Science and Its Applications – ICCSA 2015*, pp. 172–186. Springer International Publishing.

Croës, G. (2011). Qu'est-ce que rest ? `https://www.croes.org/gerald/blog/qu-est-ce-que-rest/447`.

Dutta, J., C. Chowdhury, S. Roy, A. Middya, et F. Gazi (2017). Towards smart city : Sensing air quality in city based on opportunistic crowd-sensing. In *ICDCN*, pp. 1–6.

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

Low, I. (2018). Smart city : les avantages de la ville intelligente. `https://www.globalsign.fr/fr/blog/les-avantages-de-la-ville-intelligente/`.

Lucidchart. Qu'est-ce que la norme de modélisation des processus métier ? `https://www.lucidchart.com/pages/fr/quest-ce-que-le-BPMN`.

Mohandas, P., J. Sheebha Anni, et X.-Z. Gao (2019). Artificial neural network based smart and energy efficient street lighting system : A case study for residential area in hosur. *Sustainable Cities and Society 48*.

Paik, H.-Y., A. L. Lemos, M. C. Barukh, B. Benatallah, et A. Natarajan (2017). *Service Component Architecture (SCA)*, pp. 203–250. Springer International Publishing.

Sanseverino, E. R., G. Scaccianoce, V. Vaccaro, G. Zizzo, et S. Pennisi (2015). Smart city and public lighting. In *2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC)*, pp. 665–670.

Annexe A

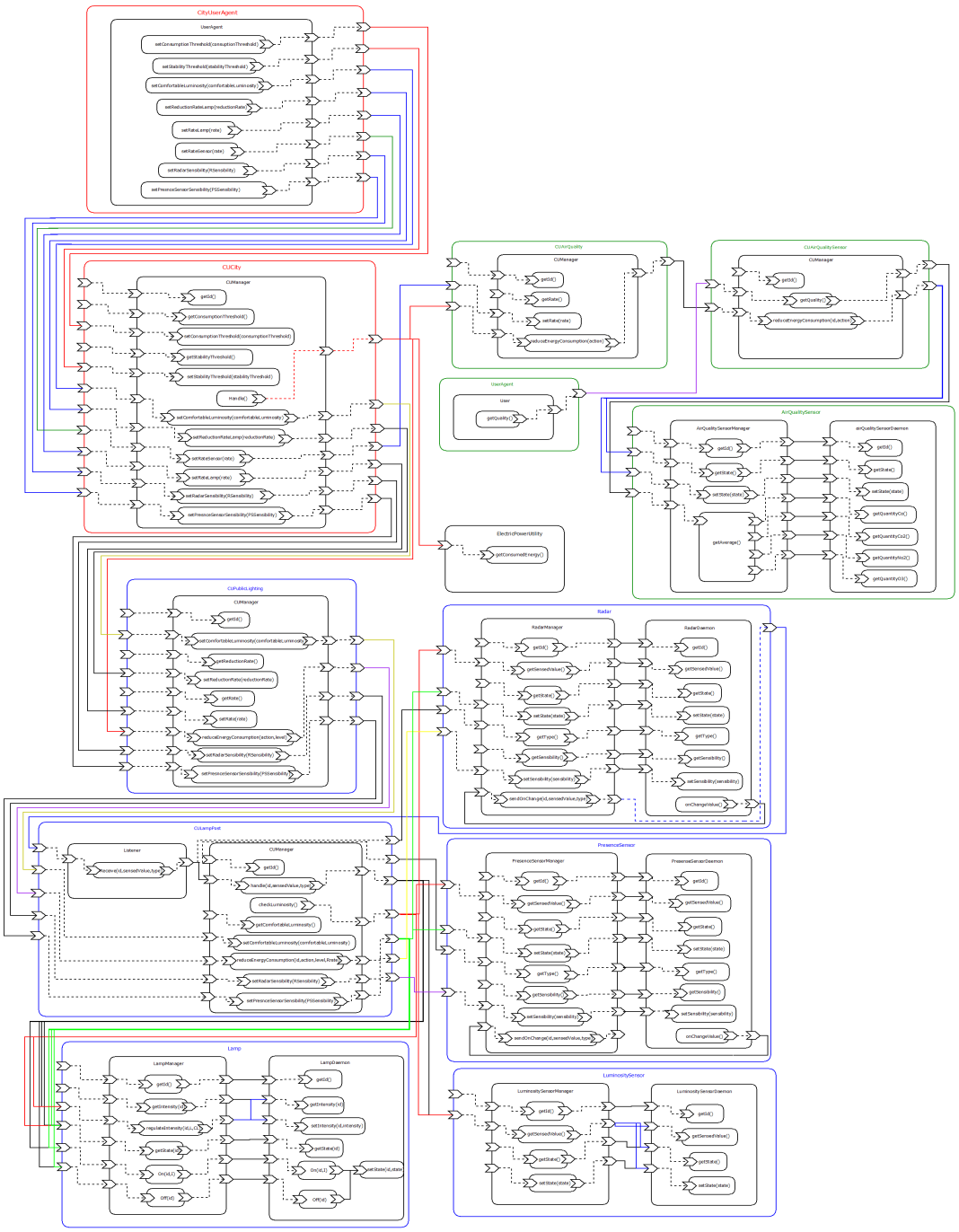

FIG. 5: *Modèle SCA de liaisons entre les composants du système d'éclairage public et les composants du système de qualité de l'air*

Annexe B

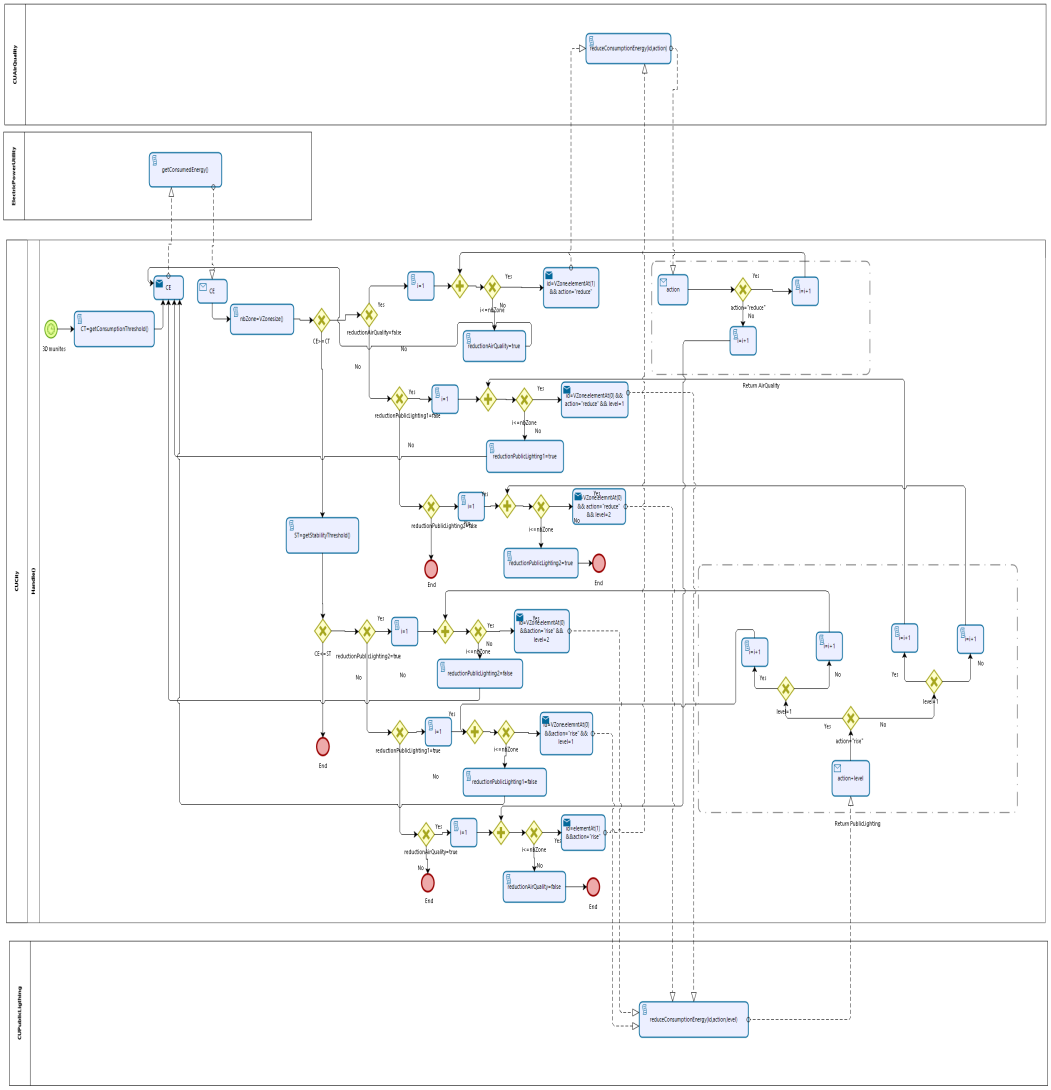

FIG. 6: *Modèle BPMN de l'opération handle() du pool "CUCity"*

Annexe C

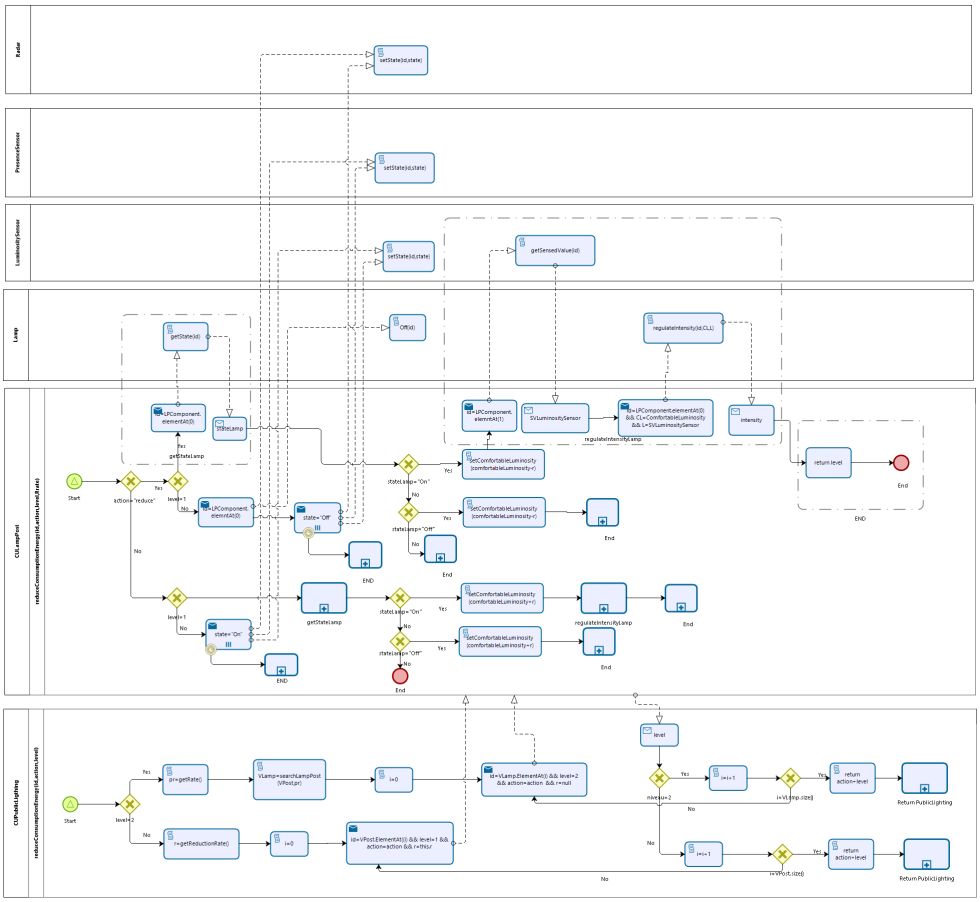

FIG. 7: *Modèle BPMN de l'opération reduceEnergieConsumption(id,action,level) du pool "CUPublicLighting" et l'opération reduceEnergieConsumption(id,action,level,Rrate) du pool "CULampPost"*

Summary

Smart Cities use technology to create sustainable urban comfort when energy resources are depleted. The challenge for urban development is about the sustainability of resources, the adaptability of services and the comfort of users. Therefore, the goal of this paper is to develop a dynamic " Smart City " that uses smart energy-efficient systems that can adapt to the environment changes to ensure users' comfort. The operation of these systems is dependent on energy reduction rules.