MASTER 2 IFI (FILLIERE IAM)

Rapport de stage de fin d'études 2013-2014

Mise en place d'un mécanisme de sélection sémantique de services pour dispositifs

Stagiaire : DAIKHI Rahma

Maitres de stage:

TIGLI Jean-Yves LAVIROTTE Stéphane

Tuteur enseignant : HERMENIER Fabien

Laboratoire: I3S, UNS/CNRS - Equipe RAINBOW

Remerciements

Je tiens à exprimer mes profonds remerciements à Monsieur Jean-Yves TIGLI et Monsieur Stéphane LAVIROTTE pour leurs disponibilités, leurs encadrements de haut niveau et surtout pour m'avoir fait confiance. Qu'ils trouvent ici l'expression de mes sincères reconnaissances.

J'adresse de même mes remerciements à Monsieur Fabien HERMENIER pour sa compréhensivité et ses conseils.

Un grand merci à ma chère mère qui a été toujours présente pour les bons conseils, à mes amis Ines,Lamia et Marwen qui m'ont toujours soutenu et m'ont apporté un grand support moral.

Je tiens également à remercier tous ceux qui ont, de près ou de loin, m'aidé à rendre ce travail possible, que ce soit par des idées ou par des encouragements.

A vous tous, je dis merci.

Table des matières

Table des figures **4**

1 Introduction **6**

1.1 Informatique ambiante 6

1.2 Présentation du stage 7

1.3 Objectif du stage 7

1.4 Scénario d'illustration de l'objectif 8

1.5 Organisation du rapport 8

2 Approche proposée **9**

Introduction 9

2.1 Problématiques sous-jacentes au stage 9

2.1.1 Problématique d'adaptation dynamique 9

2.1.2 Problématique de sélection de services 9

2.1.3 Problématique de services annotés 10

2.2 Technique d'adaptation dynamique 10

2.3 Modèle général de raisonnement sémantique 10

2.3.1 Les services pour dispositifs annotés 11

2.3.2 Gestion de la base de connaissance 13

2.3.3 Interrogation de la base de connaissance et sélection de service . . 14

2.3.4 Composition de service 15

Conclusion 15

3 La mise en œuvre **16**

Introduction 16

3.1 Outils utilisés 16

3.1.1 Conquer 16

3.1.2 WCOMP 17

3.2 Planification du stage 18

3.2.1 Diagramme de Gantt 18

3.2.2 Découpage en lots 19

3.2.3 Synthèse des résultats obtenus 19

3.3 Implémentations 20

3.3.1 Ontologie pour représenter les informations contextuelles 20

3.3.2 Architecture du modèle global 22

3.3.3 Création des composants annotés 23

3.3.4 Enrichissement et gestion de la base de connaissance 24

3.3.5 Création des services pour dispositifs annotés 26

3.3.6 Interrogation de la base de connaissance et sélection de service . . 27

Conclusion 28

4 Conclusion **29**

4.1 Conclusion 29

4.2 Perspectives 30

Annexe 1 **31**

Annexe 2 **32**

Annexe 3 **34**

Annexe 4 **35**

Bibliographie et Références **36**

Table des figures

2.1 Modèle général 11

3.1 Diagramme de Gantt 18
3.2 Ontologie représentant les informations contextuelles 22
3.3 Architecture du modèle global 23
3.4 Requêtes SPARQL 25
3.5 Réponse de Bdc 25
3.6 Requêtes SPARQL/Visualisation de l'ontologie 26
3.7 Device Spy 27
3.8 Schéma de comparaison 28

Chapitre 1

Introduction

1.1 Informatique ambiante

Depuis deux décennies, Mark Weiser a introduit sa vision d'une informatique de futur où elle devient accessible et présente dans notre vie quotidienne, en toute chose et en tout lieu. Il s'agit de l'intelligence ambiante qui vise essentiellement à fournir des objets physiques, intelligents et capables de communiquer et d'interagir les uns avec les autres, avec l'environnement et avec l'individu. Cette interaction impose de prendre en considération certains concepts qui sont obligatoires dans le challenge de la conception logicielle du système [Tigli, 2007].

Le premier concept est l'hétérogénéité qui décrit la diversité des capacités et des fonctionnalités fournies par des objets intelligents et qui consiste dans L'aptitude de faire face à des logiciels, des matériels et même des protocoles. Le deuxième concept est la dynamicité de l'environnement qui illustre la mobilité de dispositif ce qui implique son apparition et sa disparition. Le troisième concept est l'adaptation face à la diversité des situations (multi-dispositifs, multi-utilisateurs, dans un environnement physique variable). En fait, les applications ambiantes doivent être capables de percevoir et de s'adapter dynamiquement aux besoins de l'utilisateur et au contexte de l'exécution. « LE CONTEXTE EST L'ENSEMBLE DES INFORMATIONS QUI SERVENT À CARACTÉRISER UNE ENTITÉ (OBJET, PERSONNE, TEMPS, ETC.) » [Dey, 2001].

La clé du succès serait donc d'offrir à l'utilisateur le meilleur service qui satisfait ses besoins en utilisant un mécanisme de sélection de services, tout en gardant la transparence de l'utilisation de ces services.

1.2 Présentation du stage

J'effectue mon stage au sein de l'équipe RAINBOW du laboratoire I3S (Informatique, Signal et Système de Sophia Antipolis). L'équipe travaille dans le domaine de l'ingénierie des applications logicielles appliquées à l'informatique ambiante.

Une partie des travaux de recherche de l'équipe RAINBOW a été consacrée à la problématique de l'adaptation dynamique des applications logicielles à leur contexte. Cette adaptation permet de rendre les systèmes logiciels plus autonomes et mieux adaptés à leur contexte d'exécution.

Pour répondre à cette problématique, l'équipe Rainbow a introduit une technique d'adaptation dynamique qui nécessite l'intégration d'un mécanisme de sélection de services. Ce mécanisme consiste à sélectionner les services adéquats pour produire des fragments d'application.

Le mécanisme de sélection de service a été au cœur de plusieurs projets de recherche. Les résultats des recherches ont été accomplis pour rendre les systèmes ambiants capables de sélectionner les services selon le contexte d'exécution et selon les besoins de l'utilisateur.

Dans ce cadre, les travaux menés dans l'équipe Rainbow ont pour objectif d'évoluer le mécanisme de sélection de services disponibles. C'est dans cet objectif que s'inscrit mon stage.

1.3 Objectif du stage

L'objectif de ce stage est d'étudier et de mettre en place un mécanisme de sélection sémantique de services qui s'appuiera sur un ensemble de services annotés. Ce mécanisme devra être intégré dans la première partie de schéma d'adaptation. Il devra interroger la base de connaissance qui contient les métadonnées nécessaires des services disponibles afin de sélectionner les services pertinents. Les services sélectionnés vont être envoyés ensuite à la deuxième partie de schéma d'adaptation pour la composition de services. Ce schéma d'adaptation sert à produire un ou plusieurs fragments d'application.

Notre projet s'intègre dans les travaux menés par l'équipe RAINBOW et l'équipe HADAS. Durant le stage, nous avons utilisé différents outils et modèles. Ces derniers vont être détaillés dans les chapitres suivants, parmi lesquels :

– La base de connaissance Conquer [Benyalloul, 2010]
– L'intergiciel WComp [Tigli, 2009]

Le travail à faire durant le stage :

1. Développement d'une ontologie qui modélise les services pour dispositifs avec leurs annotations (localisation, utilisateur, etc).
2. La conception d'un modèle global permettant la résolution de l'intégrité de notre travail.
3. Implémentation de parties logicielles qui permettent d'intégrer notre modèle dans des schémas de sélection et de composition de services.

1.4 Scénario d'illustration de l'objectif

Notre idée pourra être trouvée dans plusieurs domaines tels que le domaine de santé, le domaine des habitats de futur, etc.

Afin d'illustrer notre solution, nous allons prendre comme exemple un scénario dans le domaine des habitats de futur. Nous allons mettre en scène la personne « Alice ».

Dans ce scénario, nous allons décrire comment notre application va répondre au besoin d'Alice. Cette application offre plusieurs services : activer le système du réveil, allumer le téléviseur, ouvrir les volets de la chambre, activer le chauffage ou le climatiseur, déclencher la cafetière, etc.
Alice vit en colocation avec une autre personne. Elle veut contrôler le système d'éclairage dans sa suite. Elle utilise un interrupteur mobile qui peut éteindre ou allumer une lampe spécifique. Par exemple, Alice veut allumer la lampe de sa chambre à coucher. Dans ce cas, l'application active le système d'éclairage qui se trouve dans la chambre d'Alice en choisissant l'interrupteur d'Alice et la lampe qui se trouve dans la chambre d'Alice.

1.5 Organisation du rapport

Ce rapport s'organise comme suit :
– Chapitre 2 : Approche proposée Dans ce chapitre, nous allons présenter notre travail proposé. Nous allons décrire notre modèle global qui permet la résolution de l'intégrité de notre travail.
– Chapitre 3 : Mise en œuvre Dans ce chapitre, nous allons décrire le travail réalisé qui est basé sur l'architecture décrite dans le chapitre 2.
– Chapitre 4 : Conclusion et perspectives

Chapitre 2

Approche proposée

Introduction

Dans ce chapitre, nous commençons tout d'abord par spécifier les problématiques sous-jacentes au stage. Après nous allons présenter notre approche proposée. Par la suite, nous détaillons notre approche en expliquant ces différentes parties.

2.1 Problématiques sous-jacentes au stage

2.1.1 Problématique d'adaptation dynamique

L'environnement de systèmes ambiants se caractérise par la multiplicité des services pour dispositifs. Cette dernière impose de prendre en considération le problème de dynamicité qui challenge la conception logicielle des systèmes ambiants. En effet, ces services peuvent être mobiles ce qui implique leur apparition et leur disparition dynamique.

Le défi est donc que les systèmes ambiants soient capables de s'adapter dynamiquement à l'évolution de l'ensemble des services disponibles à chaque instant.

2.1.2 Problématique de sélection de services

Aujourd'hui, le mécanisme de sélection de services utilisé pour la production des fragments d'application est relativement simple et repose seulement sur une correspondance syntaxique des noms de services par rapport à une expression régulière.

L'idée est donc d'introduire un mécanisme plus évolué et qui se repose non plus sur une comparaison syntaxique mais sur une comparaison sémantique de services.

2.1.3 Problématique de services annotés

Aujourd'hui, les services utilisés dans les schémas d'adaptation sont identifiés seulement par leurs noms et leurs types. De ce fait, le mécanisme de sélection repose seulement sur le nom de service.

Néanmoins, pour introduire un mécanisme de sélection sémantique de services, il est nécessaire d'améliorer la description des services par des annotations plus spécifiques telles que l'utilisateur, le lieu, le temps, etc.

Le défi est donc de trouver une description sémantique afin de décrire les services pour dispositifs.

2.2 Technique d'adaptation dynamique

Face à tous les changements brusques dans les exigences imposées par les utilisateurs et la disponibilité des services pour dispositifs, les systèmes ambiants doivent être capables de se régler eux-mêmes pour s'adapter aux changements de l'environnement contextuel. L'adaptation, dans ce cas, signifie l'ajout et la suppression des composants ainsi que l'exécution des changements au niveau des liaisons entre eux. [Theng, 2008]

La technique d'adaptation dynamique se base sur des schémas d'adaptations ou ce qu'on appelle « les aspects d'assemblages » (AAs) qui sont composés de deux parties :
– Les règles de sélections de services.
– Les règles de compositions de ces services.
Les aspects d'assemblages sont un exemple de composition par assemblage de composants. Les AAs s'inspirent des approches orientées aspects dans lesquels les règles de sélections de services requis sont assimilées à la notion de « Point de coupe » ou « PointCut » [Annexe1] et les règles de compositions sont assimilées à la notion de « Grefon » ou « Advice » [Annexe1].

2.3 Modèle général de raisonnement sémantique

Notre modèle est composé de plusieurs rubriques. Ces derniers vont participer dans la réalisation d'un système de raisonnement sémantique. La figure suivante présente notre modèle.

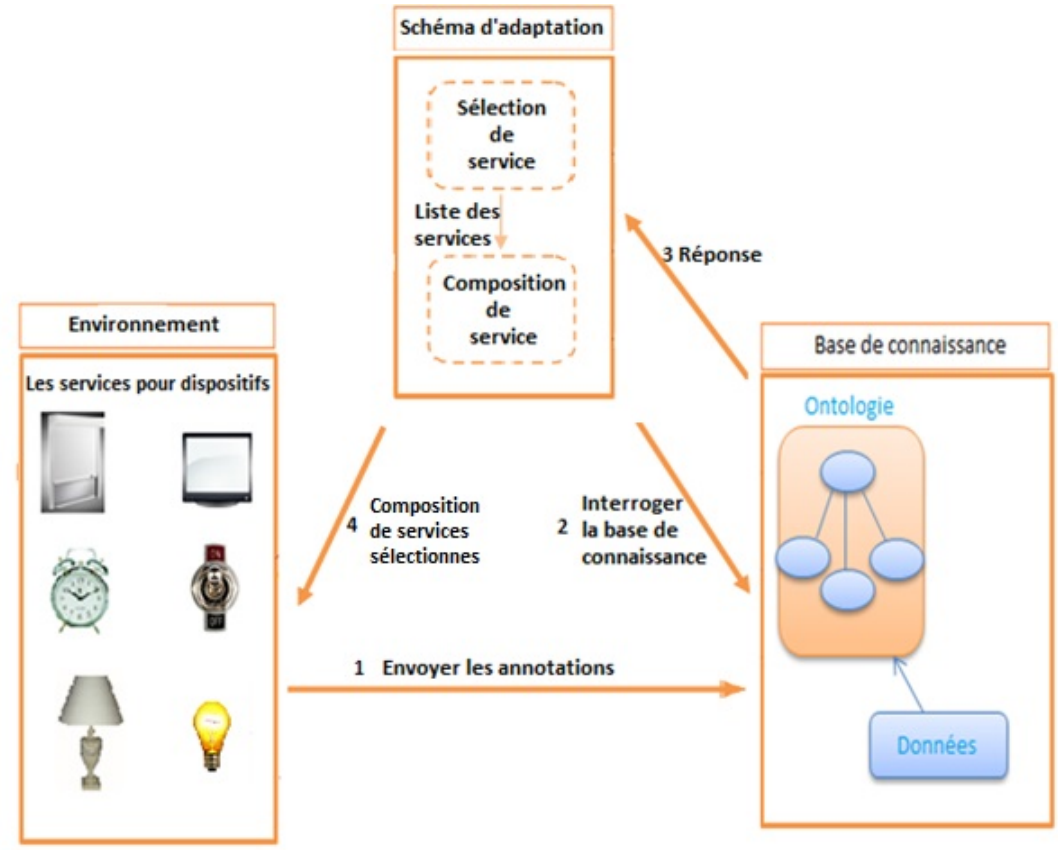


FIGURE 2.1 – Modèle général

2.3.1 Les services pour dispositifs annotés

Afin d'ajouter les annotations au service pour dispositif. Nous avons étudié différents formalismes de descriptions sémantiques de services.

Les formalismes de description sémantique de service

Dans la littérature, il existe plusieurs formalismes, on cite les plus importants. Ces formalismes sont recommandés par W3C.

OWL-S [8] : L'approche OWL-S propose une ontologie supérieure pour les services web motivée. Elle est basée sur OWL, elle décrit le service en termes de ses capacités (service capabilities).

– Service profile : (Qu'est-ce que fournit le service aux clients potentiels ?)Le service profile donne des informations concernant le service et le fournisseur. Il décrit les capacités

de service en termes d'entrées, sorties ou précondition, post-condition, et un ensemble d'informations complémentaires, comme le type de service, catégorie, propriétaire, etc.
– Modèle de Processus : (Comment est-il utilisé ?) Le modèle de processus décrit la conversation de service.
– Le service grounding : (Comment peut-on interagir avec le service) Le service grounding spécifie les informations nécessaires pour l'invocation de service.

WSMO [9] : Cette approche est parmi les formes de base pour la description d'un service web. Web Service Modeling Ontology est une ontologie de services basés sur WSMF (Web Service Modelling Framework).

WSMO définit quatre éléments de haut niveau pour décrire les services web sémantique :
– L'ontologie : définit en utilisant WSML
– But (objectifs) : décrit en termes de précondition et post-condition
– Service web : entité qui permet de répondre aux objectifs de l'utilisateur.
– Médiateur : permet de résoudre l'inadéquation (conflits) entre les trois éléments précédents.

SAWSDL [10] : C'est une approche qui suggère comment ajouter des annotations sémantiques pour les diverses parties d'un document WSDL. SAWSDL définit ainsi un nouvel espace de nommage appelé "sawsdl" et ajoute un attribut d'extension appelé "modelReference" afin que les relations entre les composants WSDL et les concepts dans un autre modèle sémantique d'affaires (ontologie, par exemple) soient traités. [Aljoumaa, 2011]

SAWSDL Permet de définir des annotations sémantique, aux différentes partie d'un document WSDL, ou son schéma XML associé. Les annotations permettent de lier les éléments de WSDL (nom des opérations, entrées, sorties) et les éléments de schéma XML a des concepts existant dans l'ontologie.

La différence entre ces trois approches

OWL-S vise à fournir une spécification haut-niveau de service web sémantique (SWS), alors que WSMO fournit des environnements d'exécution pour les différentes phases de déploiements. SAWSDL ajoute des annotations sémantiques directement à des descriptions de service WSDL sans spécifier le formalisme sémantique.

Langage d'ontologie :

Pour transformer l'information classique en données lisibles par machine, OWL-S, WSMO et SAWSDL annotent des connaissances avec les métadonnées sémantiques. Le langage

d'ontologie que les métadonnées sémantiques sont définies se diffère pour chacun des trois approches. SAWSDL est théoriquement compatible avec tous les langages d'ontologie, tandis qu'OWL-S est étroitement couplé avec OWL. D'autre part, WSMO fournit un appui conceptuel pour une famille de langages d'ontologie, qui comprend la logique de description (DL), la logique du premier ordre et la programmation logique. [Wang, 2013]

Dans notre travail, nous nous sommes inspirés de l'approche OWL-S. Les annotations sont décrites sous forme des triplés qui portent le nom du service pour dispositif, le prédicat (propriété) et l'objet. Les annotations sont les informations qui peuvent décrire un service par exemple la localisation, le type, l'utilisateur, date de fabrication, etc.

Type des annotations

- Les annotations statiques : sont les annotations inchangeables telles que le nom, le type de service, la date de fabrication.
- Les annotations dynamiques : sont les annotations qui changent selon le contexte d'exécution telles que la localisation, l'utilisateur.

Les annotations peuvent être associées au dispositif, aux services pour dispositif ou au domaine d'application. En effet, chaque service pour dispositif annoté contient des informations sur lui-même. En plus, il peut contenir des informations qui décrivent des autres services. Ces informations peuvent anticiper dans l'incrémentation et l'enrichissement de la connaissance.

2.3.2 Gestion de la base de connaissance

La base de connaissance

La base de connaissance est utilisée pour regrouper des connaissances spécifiques à un domaine spécialisé [1]. Elle possède des représentations existantes dans l'environnement. Elle peut contenir des règles et des faits sous forme d'ontologie.

La base de connaissance (BdC) s'appuie sur les standards du web service pour modéliser et interroger les représentations décrites dans l'environnement.

Elle est composée de deux parties l'ontologie qui forme l'ensemble des données et le moteur d'inférence qui présente l'ensemble de règles permettant la déduction d'information.

Ontologie

Une ontologie est un formalisme de description de service web sémantique. Elle est au cœur du web sémantique. Elle s'appuie sur des modélisations de ressources du web à partir de représentations conceptuelles des domaines concernés.

L'ontologie est un modèle de données représentatif d'un ensemble des concepts liés sémantiquement à un domaine pour modéliser un contexte donné. Chaque concept caractérise une classe et possède des liens avec les autres classes. Chaque lien est une propriété d'objet. Les liens entre les concepts sont des relations entre ces classes. Les relations peuvent être héritages, compositions ou affinements. Des instances de ces classes peuvent être déterminées pour faire du raisonnement sur les propriétés du contexte. Par exemple nous pouvons déterminer la localisation de service à l'aide de la relation (propriété) « hasLocation ». Les instances des classes dans l'ontologie sont la partie de règles des faits. Elles représentent l'ensemble des données qui vont être ajoutées à l'ontologie à chaque apparition du service dans l'environnement d'exécution.

Moteur d'inférence

Le moteur d'inférence est l'ensemble des règles d'inférence qui sont intégrées dans l'ontologie. Ces règles sont utilisées pour appliquer des raisonnements sur les données en se basant sur les classes et les relations.

Enrichissement et gestion de la base de connaissance

Parmi les fonctionnalités de notre approche est l'enrichissement de la base de connaissance qui est à l'origine de l'ajout de l'ensemble des faits dans l'ontologie. Ces faits peuvent anticiper dans la sélection de services pertinents. Par conséquent, nous observons une ontologie qui accroit avec chaque apparition de service. Cet enrichissement est accompagné d'une gestion qui sert à mettre à jour la base de connaissance après chaque apparition ou disparition d'un service. En effet, on ajoute les métadonnées de chaque service dès qu'il apparaisse et on les retire de la base de connaissance dès qu'il disparaisse.

2.3.3 Interrogation de la base de connaissance et sélection de service

La sélection de service représente la première étape dans le processus de production des schémas d'adaptation. Elle définit l'ensemble des règles qui vont permettre de choisir les services pertinents. Ces règles sont exprimées sous forme des requêtes qui vont interroger la base de connaissance sur le ou les services pertinents.

Ces services vont être sélectionnés en appliquant un raisonnement sémantique qui s'appuie sur leurs annotations. Par exemple, on sélectionne une lampe qui est la lampe de Alice et qui se trouve dans le salon. Dans ce cas, la sélection va être effectuée selon l'utilisateur et la localisation.

2.3.4 Composition de service

La composition de service est la deuxième étape dans le processus de schéma d'adaptation. Elle représente l'ensemble des règles de composition qui permettent d'effectuer les liaisons entre les services sélectionnés. Elle prend le résultat de la première partie (sélection de service) et produit la composition des services sélectionnés.

Conclusion

Dans ce chapitre, nous avons bien précisé les besoins nécessaires à la mise en œuvre de notre approche, ce qui nous a permet d'entamer la phase de la réalisation qui fera l'objet du prochain chapitre.

Chapitre 3

La mise en œuvre

Introduction

Après avoir défini notre approche, nous allons passer à la phase de mise en œuvre de cette approche. Ce chapitre contient, dans un premier lieu, les outils utilisés durant le stage. Nous présentons dans un deuxième lieu le planning que nous avons suivi pour la mise en œuvre du projet. Par la suite, nous expliquons les travaux réalisés par nos soins.

3.1 Outils utilisés

Durant notre stage, nous avons utilisé :

3.1.1 Conquer

Conquer est une base de connaissance capable de supporter des modèles du contexte décrites en langage RDF, RDFS et OWL à l'aide de l'ontologie. Il s'appuie sur les standards du web sémantique. [Benyalloul, 2010]

Il permet l'utilisation du langage d'interrogation déclaratif SPARQL pour interroger les données du contexte.

Conquer est un web service qui implémente une base de faits et donne les actions suivantes pour manipuler ces faits :

Add : Ajout d'un triplé dans la base.

Delete : Suppression d'un triplé.

Query : Exécution d'une requête SPARQL sur la base.

3.1.2 WCOMP

WCOMP est un intergiciel développé en .Net par l'équipe Rainbow du laboratoire I3S. C'est un environnement d'exécution dynamique dédié pour les applications ambiantes [7]. Il répond aux problématiques posées par l'informatique ambiante telle que la mobilité, la dynamicité de l'environnement d'exécution et l'adaptation au changement du contexte.

En informatique ambiante, l'intergiciel ou « Middleware » représente les couches logicielles du milieu. En effet, il s'insère entre les couches matérielles et les couches logicielles afin d'adapter l'application à un environnement matériel ou logiciel variable.

WCOMP est conçu pour créer des schémas d'adaptation et pour supporter l'assemblage dynamique des services pour dispositifs.
WCOMP se base sur trois paradigmes importants : [Ferry et al, 2011]

* L'architecture orientée service web pour dispositifs (Web Service Oriented Architecture for Device WSOAD), ce sont les services qui fournissent toutes les propriétés nécessaires aux interactions logicielles avec les dispositifs : l'interopérabilité, dynamicité et la réactivité. Les dispositifs sont mobiles, hétérogènes et non permanents. On ignore l'hypothèse que les entités sont connu et toujours disponibles. En effet, les dispositifs peuvent être découverts sans être connus au moment de la conception des applications. DPWS (Device Profile of Web Services) et UPnP (Universal Plug and Play) sont deux implémentations de WSOAD.
* Modèle de composition de services pour dispositifs par composants légers (Service Lightweight Component Architecture ou SLCA). Il prend en compte les variations de l'infrastructure. Il utilise des composants légers pour manipuler l'orchestration des services et pour ajouter de nouvelles fonctionnalités au système. SLCA se base sur LCA (Lightweight Component Architecture) qui permet de créer de nouvelles applications opportunistes et dynamiques à partir des services présents sur l'infrastructure. La prise en compte d'un nouveau service pour dispositif dans la plateforme WComp, qu'il soit réel ou virtuel, est réalisée par la génération automatique d'un composant spécifique appelé composant proxy suite à la recherche et la découverte dynamique d'UPnP.

Lorsqu'un service pour dispositif est découvert, le composant proxy peut être généré à partir de la description du service, chargé dans un container et instancié pour faire partie de l'assemblage de composants. Ces composants sont appelés « Bean ». Dans le WCOMP, il est possible de créer des nouveaux composants. Chacun de ces composants émet des évènements pour informer les autres d'un changement ou pour invoquer une méthode d'un autre composant.

Le modèle SLCA est basé sur une infrastructure de services utilisant des évènements, et découvrable dynamiquement de façon distribuée. Ces services représentent soit des dispositifs utilisés dans les applications ambiantes, soit des autres services composites créés par SLCA. L'infrastructure de services d'une architecture SLCA est donc utilisée pour la découverte et la communication avec les dispositifs et les services composites distribués dans l'environnement.

* Le mécanisme d'adaptation dynamique basé sur l'aspect d'assemblage. Ce mécanisme a été ajouté au modèle SLCA pour exprimer les règles de prise en compte du contexte dans les interactions entre les entités participant à une application. Les règles d'adaptations sont en effet tissées en réaction à l'observabilité des informations contextuelles. Ces règles sont indépendantes et peuvent faire face aux problèmes de la séparation des intérêts.

L'adaptation basée sur l'aspect d'assemblage est conçue pour modifier les services web basés sur les évènements lors de communication en prenant en considération l'apparition et la disparition des dispositifs dans leur environnement. Ce mécanisme est bien adapté pour régler un ensemble de services composites en réaction à une variante particulière de l'infrastructure ou l'évolution des préférences des utilisateurs.

3.2 Planification du stage

3.2.1 Diagramme de Gantt

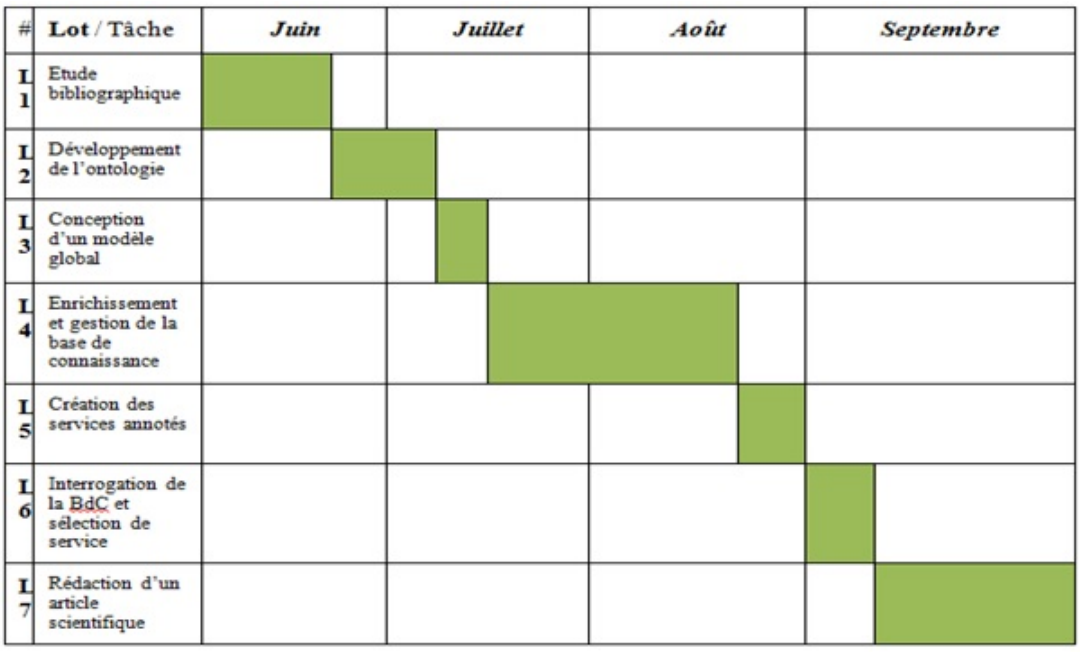


FIGURE 3.1 – Diagramme de Gantt

3.2.2 Découpage en lots

Titre du lot	Type	Début	Fin
L1/T1.1 Etude bibliographique	RECH	02/06/14	20/06/14
L2/T2.1 Développement de l'ontologie	RECH/IMPL	20/06/14	01/07/14
L2/T2.2 Test de l'existant	IMPL	01/07/14	05/07/14
L3/T3.1 Conception du modèle global	IMPL /DEMO	06/07/14	15/07/14
L4/T4.1 Création des composants annotés	RECH/IMPL	15/07/14	25/07/14
L4/T4.2 Peuplement de la Bdc	IMPL /DEMO	28/07/14	18/08/14
Mise à jour et visualisation de la Bdc	IMPL /DEMO	01/08/14	18/08/14
L5/T5.1 Création des services annotés	IMPL	19/08/14	02/09/14
L6/T6.1 Interrogation de la Bdc et sélection de service	IMPL/DEMO	28/09/14	10/09/14
L7/T7.1 Rédaction d'un article scientifique	RECH	11/09/14	30/09/14

Tableau 3.1 – Liste des Lots/Taches

3.2.3 Synthèse des résultats obtenus

#	Objectif	Statut
1	Etude bibliographique	Atteint
2	Développement de l'ontologie	Atteint
3	Conception du modèle global	Atteint
4	Mise en œuvre du modèle global	Atteint
5	Prise en compte des annotations dynamiques	Echec
6	Rédaction de l'article scientifique	

Tableau 3.2 – Synthèse des objectifs

Changement apportés aux objectifs initiaux

Les objectifs initiaux sont subis à un changement. Vu que le temps restant est trop court, nous n'aurons plus l'objectif « Prise en compte des annotations dynamiques dans le modèle ».

L'objectif « mise en œuvre du modèle global » qui comporte plusieurs parties, a pris beaucoup plus du temps que prévu. Dans ce lot, nous avons passé par plusieurs étapes afin d'améliorer notre modèle. Pour chaque étape, nous avons intégré des phases de tests et des démonstrations pour vérifier le bon fonctionnement du système.

Nous avons remplacé l'objectif « Prise en compte des annotations dynamiques dans le modèle » par l'objectif « Rédaction de l'article scientifique » .Le but de cet objectif est de décrire notre modèle et de valider l'ensemble de résultats obtenus.

Difficultés rencontrées La difficulté a été rencontrée au début du stage, dans la phase « test de l'existant ». En effet, nous avons eu quelques difficultés dans le fonctionnement de l'outil « Conquer » vu qu'il n'y avait pas assez de documentation et d'explication sur l'usage de cet outil.

3.3 Implémentations

3.3.1 Ontologie pour représenter les informations contextuelles

Nous avons développé notre ontologie à l'aide de l'outil protégé [6]. C'est un éditeur qui permet de construire des modèles de domaine et d'application. Dans le contexte du web sémantique, des plugins pour les langages RDF, DAML+OIL, OWL ont été développé pour Protégé. Ces plugins permettent de l'utiliser comme un éditeur d'ontologie.

L'ontologie sous Conquer, est l'ensemble d'expression RDF/RDFS. Cette ontologie comprendra les informations nécessaires sur notre contexte. Pour la construction de cette ontologie, nous avons défini les concepts généraux qui sont la base des règles : Services, Person, Location et les relations qui peuvent relier les différentes instances de ces classes « isIn », « hasLocation », « usedBy », etc. Nous avons défini dans cette ontologie le personnage « Alice », c'est une instance de la classe « Person ». Nous avons défini aussi la localisation des services « Bathroom », « BedRoom », « Kitchen », « LivingRoom ». Les instances et les sous-classes de la classe Services vont être ajoutées au moment de l'apparition de services.

Langage RDF RDF (Resource Description Framework) est le langage de base du web sémantique. Il s'agit d'un modèle développé par le W3C, destiné à décrire de façon formelle les ressources web identifiées par des URIs (Uniforme Ressource Identifier) et leurs métadonnées. [2]

L'une des syntaxes de ce langage est RDF/XML.

En RDF, la description d'une métadonnée est exprimée sous forme d'un triplé composé d'un sujet, d'un prédicat et d'un objet.

– Le sujet (ressource) définit l'entité d'information. C'est un URI pointant sur une ressource web.

– Le prédicat (propriété) définit la relation pour décrire le sujet. C'est un URI pointant sur une ressource web.
– L'objet définit la valeur du prédicat attachée à un sujet. Il peut être soit un URI pointant sur une ressource web, soit un simple littéral représentant une valeur.

Langage RDFS RDF Schema ou RDFS est un langage extensible de représentation des connaissances. Il appartient à la famille des langages du Web sémantique publiés par le W3C [3]. C'est une extension du RDF. Il permet d'exprimer les propriétés des ressources ainsi que leurs types.

Parmi les relations sémantiques (les propriétés) que nous avons définies :

« hasLocation » : pour identifier la localisation.

« usedBy » : pour identifier l'utilisateur.

« isIn » : pour identifier la localisation de personne.

Pour les relations sémantiques qui sont définies par les langages d'ontologies, nous avons

« rdf :type» : pour identifier le type de l'instance.

« rdfs :subClassOf » : pour identifier l'héritage.

Par exemple, si nous définissons les triplés suivants :

< A hasLocation B> : signifie que A se trouve dans B.

< A isA B> : signifie que A est un B.

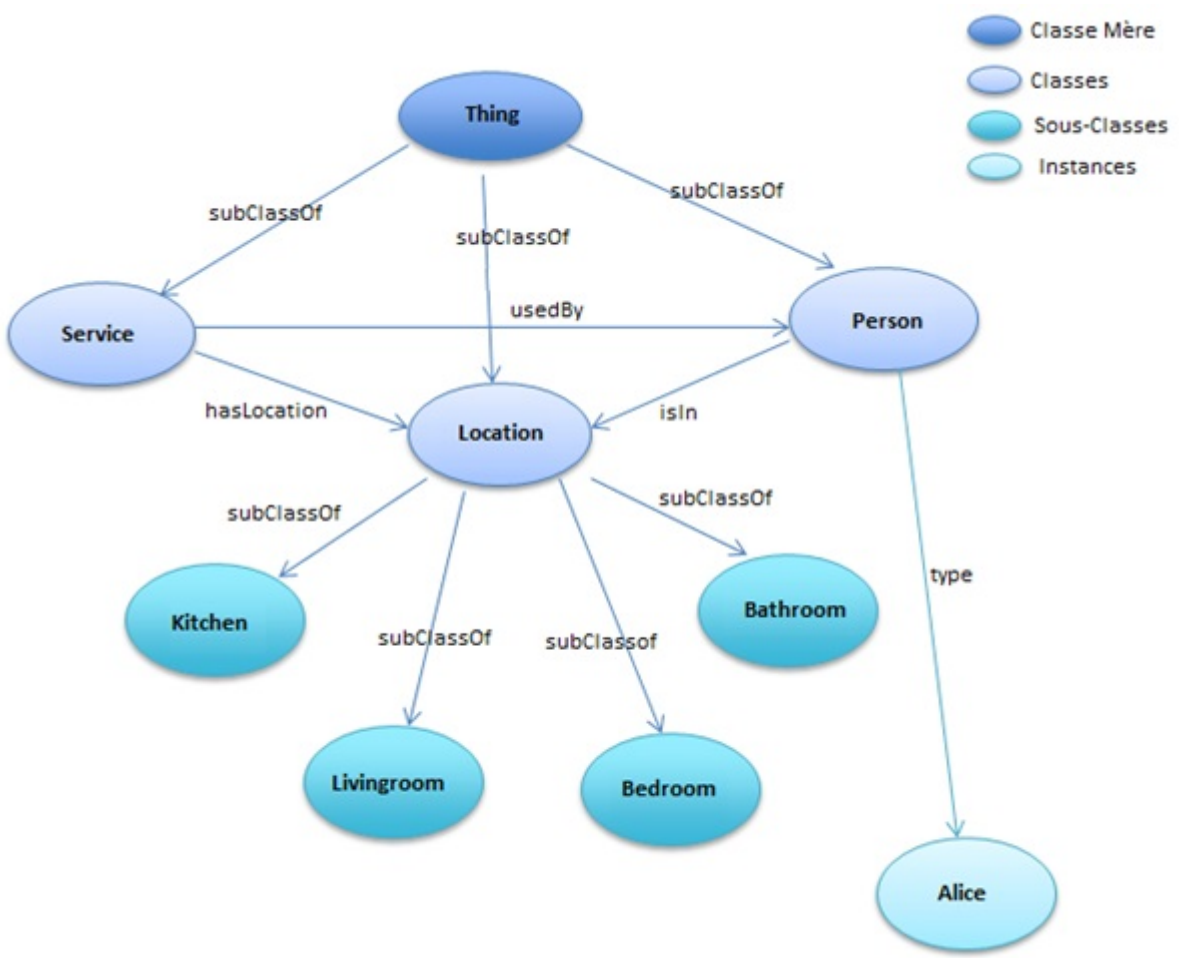


FIGURE 3.2 – Ontologie représentant les informations contextuelles

3.3.2 Architecture du modèle global

Notre modèle est composé de plusieurs rubriques. Ces derniers vont participer dans la réalisation de notre mécanisme. Pour avoir plus de détails voir annexe 2.

* Après l'apparition de chaque service dans l'environnement d'exécution, on récupère la liste des métadonnées qui correspondent à ce service. Dans l'environnement d'exécution, chaque service pour dispositif a son propre composant proxy. Ces composants sont instanciés pour participer dans les schémas d'adaptations.

* Ensuite, les métadonnées seront stockées dans la base de connaissance sous forme de triplé, dans une ontologie qui représente l'ensemble des données à l'aide du composant « PopulateBean ».

* Nous pouvons remarquer l'accroissement de l'ontologie qui est à l'origine de l'ajout des annotations de services.

* En cas de disparition de service, les données de la base de connaissance vont être mises à jour. En effet, les métadonnées associées au service disparu vont être supprimées de la

base de connaissance.
* A partir de ces informations, Nous avons un schéma d'adaptation qui contient les règles de sélection de service et les règles de composition. Les règles de sélection de service interrogent la base de connaissance sur les services disponibles et qui sont pertinents pour établir des assemblages de composants.

Dans notre cas, les règles de sélection de services interrogent la base de connaissance sur la disponibilité de la lampe de Alice qui se trouve dans sa chambre à coucher et sur la disponibilité de son interrupteur afin d'établir un assemblage entre ces deux composants.
* Cette liste sera envoyée à la deuxième partie de schéma d'adaptation pour établir la composition de services sélectionnés.

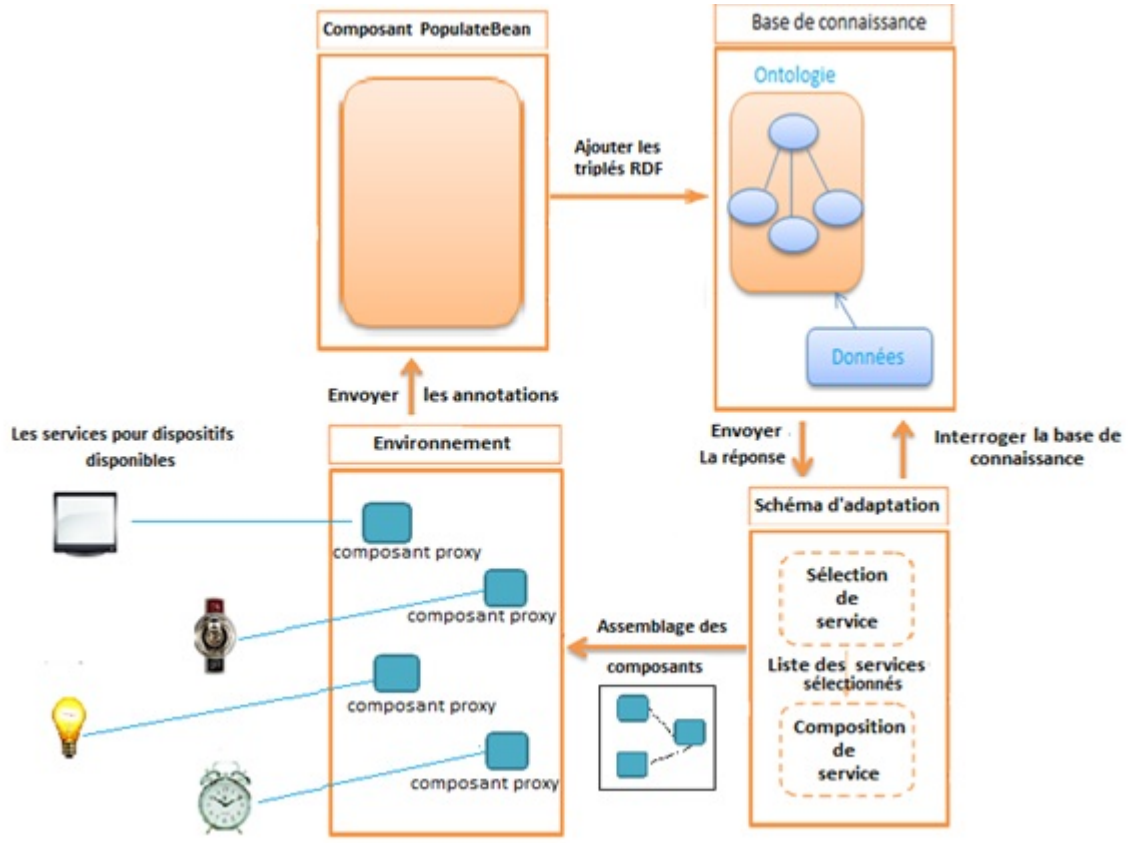


FIGURE 3.3 – Architecture du modèle global

3.3.3 Création des composants annotés

Dans le WCOMP, il est possible de créer des nouveaux composants. Chacun de ces composants contient des propriétés et des méthodes. Il émet des évènements pour informer

les autres d'un changement ou pour invoquer une méthode d'un autre composant.

Dans notre cas, nous avons créé deux composants qui contiennent chacun une propriété « metadata ». Cette dernière est une classe qui sert à gérer la liste des annotations de services (ajouter, modifier, supprimer et lire). Elle émit les annotations dans une DataSet (tableau de données). Cette Dataset sera stockée dans un fichier de format « cnv». La lecture de ces annotations va être à partir de ce fichier.

3.3.4 Enrichissement et gestion de la base de connaissance

Peuplement de la base de connaissance

Le composant « PopultateBean » est le composant responsable du peuplement de la BdC. Il reçoit comme entrée la liste des métadonnées associées à chaque service découvert. Ces métadonnées vont être envoyées à la base de connaissance sous forme des triplés RDF.

Ce composant participe dans l'enrichissement de la base de connaissance.

Mise à jour, interrogation et visualisation de la base de connaissance

La mise à jour

La mise à jour consiste à synchroniser la base de connaissance avec l'environnement d'exécution. En cas de disparition de service, la base de connaissance doit mettre à jour les données stockées. En effet, le système envoie une requête « Remove » au Conquer pour supprimer les données associées au service disparu. Seulement les triplés associés à ce service vont être supprimés.

Interrogation et visualisation de la base de connaissance

La base de connaissance (Bdc) Conquer permet l'utilisation du langage d'interrogations déclaratives SPARQL (SPARQL Protocol And RDF Query Language).

Langage SPARQL

SPARQL est un langage de requêtes et un protocole pour l'accès RDF, il est publié par le W3C. Il est comme tous les langages de requêtes, orienté données : il intègre uniquement les informations contenues dans les modèles. Il est destiné à interroger, extraire et manipuler les données RDF.
SPARQL permet aux utilisateurs d'écrire des requêtes ambiguës sans inconvénient. Il prend simplement la description que l'application nécessite sous forme de requête et retourne cette information sous forme d'un ensemble de liaisons ou d'un graphe RDF. [5]

Nous avons utilisé ce langage pour interroger les données stockées dans l'ontologie. Le composant « Query » est responsable d'interroger la BdC « Conquer » en envoyant une requête SPARQL.
Par exemple pour interroger la base de connaissance Conquer sur la liste des équipements, on exécute la requête suivante :

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ontology: <http://www.owl-
ontologies.com/MyOntology1.xml#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
select ?x where
{?x rdfs:subClassOf ontology:Equipement }
```

FIGURE 3.4 – Requêtes SPARQL

La réponse de la base de connaissance est la suivante :

```
http://www.owl-ontologies.com/MyOntology1.xml#Radio
http://www.owl-ontologies.com/MyOntology1.xml#Alarme
http://www.owl-ontologies.com/MyOntology1.xml#Television
http://www.owl-ontologies.com/MyOntology1.xml#Lampe
http://www.owl-ontologies.com/MyOntology1.xml#Reveil
http://www.owl-ontologies.com/MyOntology1.xml#Equipement
```

FIGURE 3.5 – Réponse de Bdc

La base de connaissance Conquer nous donne la possibilité de visualiser l'ontologie à l'aide d'une interface graphique en utilisant le langage sparql. EN effet, nous pouvons visualiser notre ontologie ou une partie.

Par exemple pour visualiser tous les équipements existant, on exécute la requête suivante sur l'interface graphique du conquer :

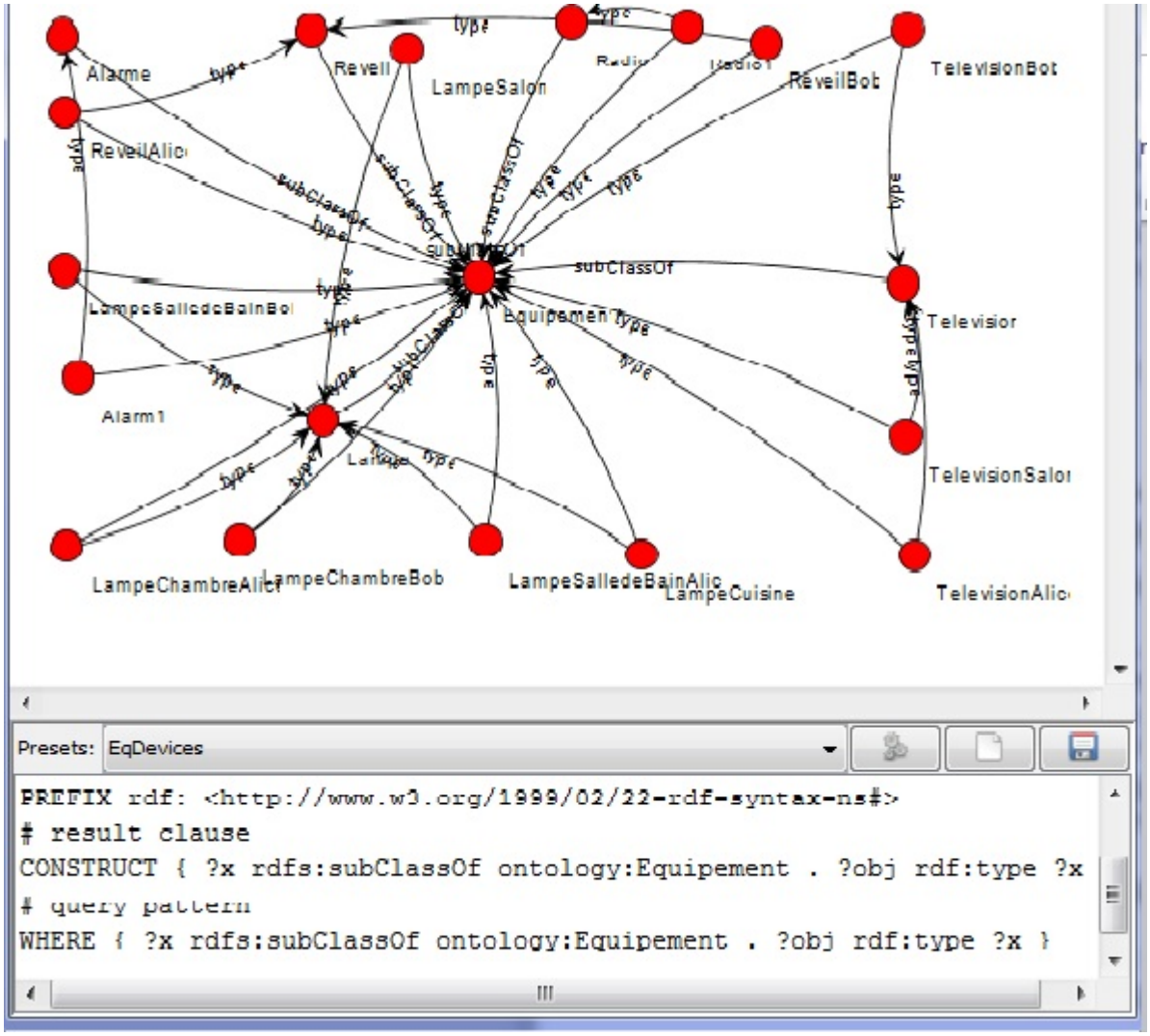


FIGURE 3.6 – Requêtes SPARQL/Visualisation de l'ontologie

3.3.5 Création des services pour dispositifs annotés

Pour faire la mise en œuvre de notre scénario d'illustration. Nous avons créé deux services pour dispositifs annotés « Light » et « switch ». Ces services contiennent une classe « Metadata » comme celle qui a été créée dans les composants annotés.

La différence entre les services pour dispositifs et les composants. C'est que les services pour dispositifs sont créés en dehors du WCOMP alors que les composants beans sont créés dans le Wcomp. Encore, les services pour dispositifs utilisent le standard UPnP pour la communication avec les autres services et avec l'environnement d'exécution. Ces services peuvent être détectés par l'explorateur Réseau Intel pour technologie UPnP le « UPnP Device Spy ». Cet outil peut invoquer les méthodes des services découverts. La figure suivante montre l'interface Device Spy.

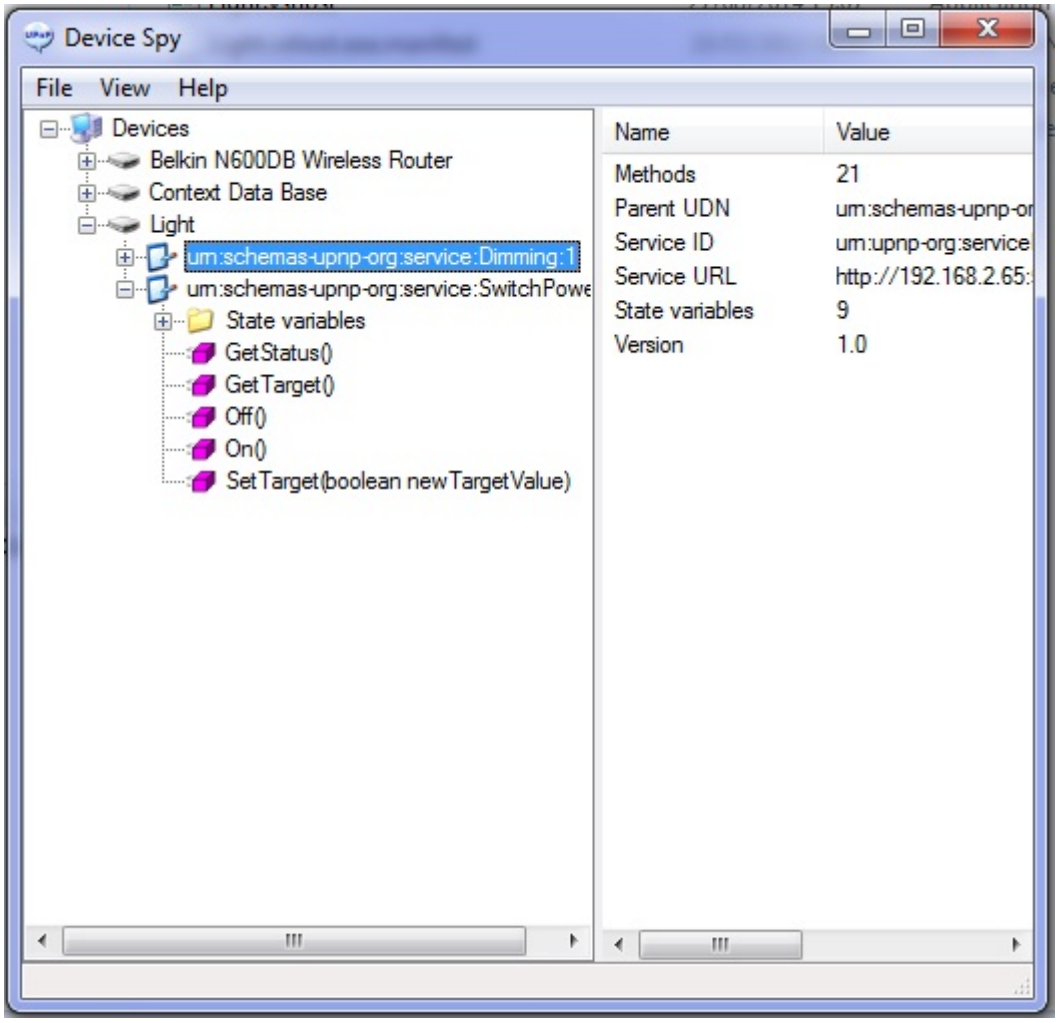


FIGURE 3.7 – Device Spy

Lors de l'apparition de service pour dispositif, nous pouvons générer son composant proxy qui va participer dans la production des schémas d'adaptation. De ce fait, nous avons utilisé l'outil UPnP Wizard Designer qui gère l'apparition et la disparition des services pour dispositifs et qui connecte ces services à l'interface de WComp en créant un composant proxy pour chaque service découvert.

3.3.6 Interrogation de la base de connaissance et sélection de service

Après la découverte de deux services « Light » et « switch », le schéma d'adaptation va vérifier la disponibilité de ses deux services à l'aide de règles de sélection de service. En effet, les règles de sélection de service vont interroger la base de connaissance « Conquer » sur les services demandés et retourne la liste à la deuxième partie du schéma d'adaptation pour établir la composition des services sélectionnés. Le schéma d'adaptation dans notre

projet est le tisseur WComp (Weaver) qui est le responsable de l'adaptation dynamique de l'application logicielle. Voir Annexe 3.

La figure suivante montre la différence entre le mécanisme qui a été implémenté dans les schémas d'adaptation et qui repose seulement sur une comparaison des noms de services par rapport à des expressions régulières « light = light* », et le mécanisme qui va être implémenté et qui repose dans la sélection de service sur le sens et les annotations des services.

Avant Après

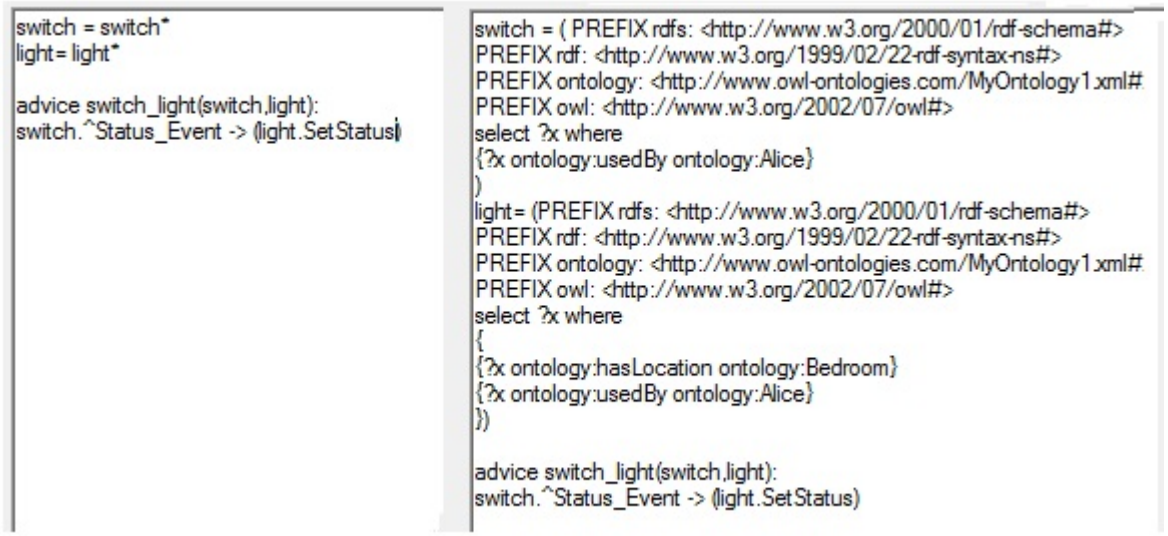


FIGURE 3.8 – Schéma de comparaison

Conclusion

Dans ce chapitre, nous avons présenté l'environnement logiciel pour l'implémentation. Ensuite, nous avons décrit les différentes phases réalisées de notre solution.

Chapitre 4

Conclusion

4.1 Conclusion

Les systèmes ambiants présentent une nouvelle ère de l'informatique qui consiste à utiliser des applications logicielles ambiantes en tout lieu et d'une manière transparente. Les systèmes ambiants sont caractérisés par la diversité et la dynamicité des situations (multi-services, multi-utilisateurs, environnement variable). Toutes ces variations ne peuvent pas être anticipées au moment de la conception. D'où la nécessité de l'adaptation dynamique des applications logicielles. En particulier, l'adaptation dynamique des applications logicielles en fonction de l'évolution de l'ensemble de services disponibles à chaque instant. L'objectif de notre stage est la mise en place d'un mécanisme de sélection sémantique de service basé sur des services annotés.

Dans ce cadre, nous avons présenté un modèle global de raisonnement sémantique qui sert, d'une part, à enrichir et gérer la base de connaissance, et d'autre part, à intégrer un processus de sélection sémantique de service dans le schéma d'adaptation. Ce processus interroge la base de connaissance sur la liste des services pertinents et disponibles. Cette liste sera envoyée à la deuxième partie de schéma d'adaptation pour établir la composition de services sélectionnés et pour fabriquer des fragments d'application.

Ce mécanisme a été mis en place dans le schéma d'adaptation de l'environnement logicielle WCOMP. Enfin nous nous sommes basée sur un scénario que l'on a décrit afin d'illustrer notre objectif et valider notre approche.

4.2 Perspectives

Notre mécanisme peut subir à des améliorations :

- Pour les services annotés : on ajoute les métadonnées dynamiques à la liste des annotations.
- Pour l'ontologie : On pourra considérer que chaque service a sa propre ontologie. Dans ce cas, les métadonnées associées aux services pour dispositifs pourront être différentes à celles qui sont décrites dans l'ontologie générale. Dans ce cas, il faudra aligner les annotations de services pour dispositifs et les annotations décrites dans l'ontologie. On pourra par exemple insérer un dictionnaire « WordNet ».
- Pour les services pour dispositifs : on modifie dans l'implémentation de WComp de telle sorte on aura une unique instance pour chaque dispositif.

Annexe 1 : Définitions

[1] Pointcut ou le point de coupe est un concept utilisé dans la programmation orientée aspect. Le pointcut définie l'expression des règles de recherche des points de jonction dans l'aspect assemblage.

[2] Advice ou Greffon correspond à un ensemble des règles dans la deuxième partie de l'aspect d'assemblage. Ces règles décrivent les modifications appliquées sur les points de jonction (joinpoint).

Annexe 2 : Le code RDF/RDFS correspondant à l'ontologie

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
        <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
        <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
        <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
        <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
        <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
        <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
        <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
        <!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#" >
]>
<rdf:RDF xmlns="http://www.owl-ontologies.com/MyOntology1.xml#"
        xml:base="http://www.owl-ontologies.com/MyOntology1.xml"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
        xmlns:swrl="http://www.w3.org/2003/11/swrl#"
        xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
        xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:owl="http://www.w3.org/2002/07/owl#">
<owl:Ontology rdf:about=""/>
```

<Person rdf :ID=”Alice”>

<IsIn rdf :resource=”#BathRoomAlice”/>

<IsIn rdf :resource=”#BedRoomAlice”/>

<IsIn rdf :resource=”#KitchenHouse”/>

<IsIn rdf :resource=”#LivingRoomHouse”/>

</Person> <rdfs :Class rdf :ID=”BathRoom”>

<rdfs :subClassOf rdf :resource=”#Location”/>

</rdfs :Class> <BathRoom rdf :ID=”BathRoomAlice”/>

<rdfs :Class rdf :ID=”BedRoom”>

<rdfs :subClassOf rdf :resource=”#Location”/>

</rdfs :Class> <BedRoom rdf :ID=”BedRoomAlice”/>

<owl :ObjectProperty rdf :ID=”hasLocation”>

<rdfs :domain rdf :resource=”#Service”/>

<rdfs :range rdf :resource=”#Location”/>

</owl :ObjectProperty> <owl :ObjectProperty rdf :ID=”IsIn”>

<rdfs :domain rdf :resource=”#Person”/>

<rdfs :range rdf :resource=”#Location”/>

</owl :ObjectProperty> <rdfs :Class rdf :ID=”Kitchen”>

<rdfs :subClassOf rdf :resource=”#Location”/>

</rdfs :Class> <Kitchen rdf :ID=”KitchenHouse”/>

<rdfs :Class rdf :ID=”LivingRoom”>

<rdfs :subClassOf rdf :resource=”#Location”/>

</rdfs :Class> <LivingRoom rdf :ID=”LivingRoomHouse”/>

<rdfs :Class rdf :ID=”Location”/>

<rdfs :Class rdf :ID=”Person”/>

<rdfs :Class rdf :ID=”Service”/>

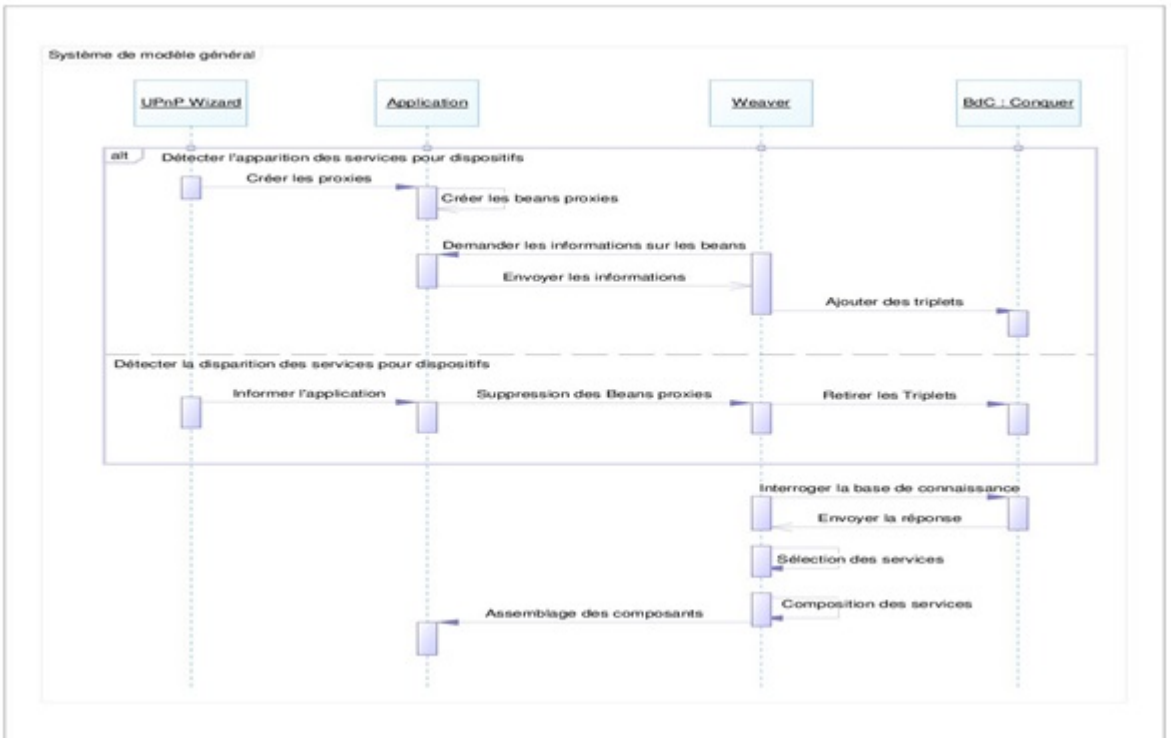
<owl :ObjectProperty rdf :ID=”usedBy”>

<rdfs :domain rdf :resource=”#Service”/>

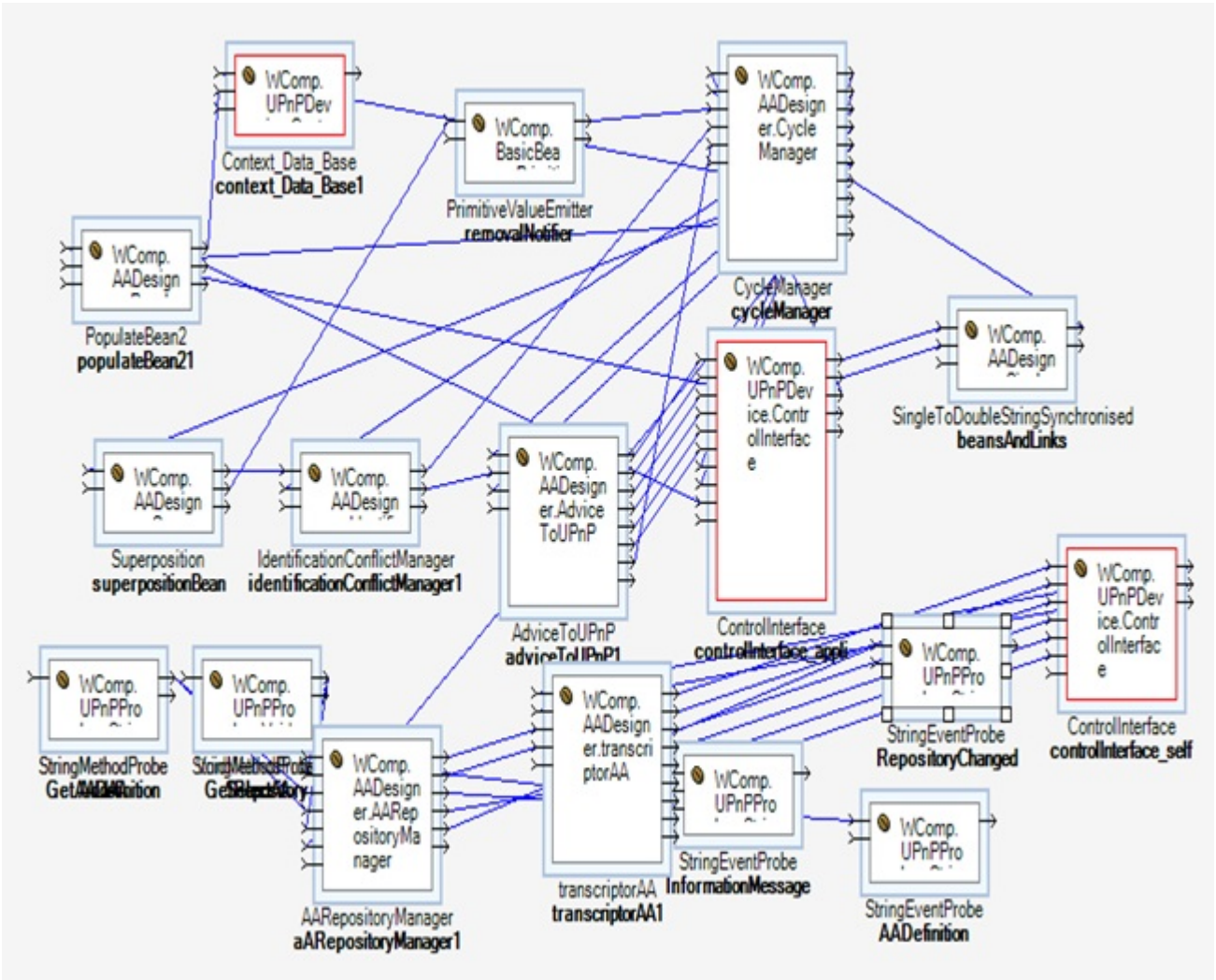
<rdfs :range rdf :resource=”#Person”/>

</owl :ObjectProperty> </rdf :RDF>

Annexe 3 : Diagramme de séquence système



Annexe 4 : Tisseur WComp : Weaver



Bibliographie et Références

[**Aljoumaa, 2011**] Kadan Aljoumaa, Modélisation intentionnelle et annotation sémantique pour la réutilisation de services métiers. Thèse de doctorat, Université PARIS I- PANTHEON-SORBONNE, 2011.

[**Benyelloul, 2010**] Benyelloul, A. (2010). Conquer : an RDFS-based Model for Context Querying RDFS-based Context Model. Ubimob10 (pp. 1–4).

[**Dey, 2001**] Dey, A. Understanding and using context, Personal and Ubiquitous Computing, 5(1), pp. 4-7, 2001.

[**Ferry et al, 2011**] Nicolas Ferry, Vincent Hourdin, Stéphane Lavirotte, Gaëtan Rey, Michel Riveill, Jean-Yves Tigli, feb 2011 « WComp a Middleware for Ubiquitous Computing » in Ubiquitous Computing, chapter 8, pages 151-176. InTech.

[**Theng, 2008**] Yin-Leng Theng et Henry Duh, Ubiquitous computing. Ed. Information Science Reference. 396 pages, 2008

[**Tigli, 2007**] Jean-Yves Tigli , Stéphane Lavirotte , Gaëtan Rey, Vincent Hourdin, Daniel Cheung-Foo-Wo, Eric Callegari, Michel Riveill. “A middleware for ubiquitous computing : WCOMP”.2007

[**Tigli, 2009**] Tigli, J., Lavirotte, S., Rey, G., Hourdin, V., Cheung-Foo-Wo, D., Callegari, E., & Riveill, M. (2009). WComp middleware for ubiquitous computing : Aspects and composite event-based Web services. Annales des Telecommunications

[**Wang et al., 2013**] Wang, Hai H. ; Gibbins, Nick ; Payne, Terry ; Patelli, Alina, “A Survey of semantic web service formalisms”, Semantics, Knowledge and Grids (SKG), 2013 Ninth International Conference on, pages 135-142, 3-4 Oct 2013, Beijing, China.

[1] Bdc, http ://fr.wikipedia.org/wiki/Base_de_connaissance

[2] rdf, http ://www.w3.org/TR/rdf-concepts/

[3] rdfs, http ://www.w3.org/TR/2004/REC-rdf-schema-20040210/

[4] Owl, http ://fr.wikipedia.org/wiki/Web_Ontology_Language

[5] SPARQL, http ://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/

[6] Protege, http ://protegewiki.stanford.edu/

[7] Wcomp, http ://www.wcomp.fr/

[8] OWL-S, http ://www.w3.org/Submission/OWL-S/

[9] WSMO, http ://www.w3.org/Submission/WSMO-primer/

[10] SAWSDL, http ://www.w3.org/2002/ws/sawsdl/

Résumé

Dans l'informatique ambiante, les applications logicielles nécessitent de s'adapter dynamiquement à leur contexte d'exécution face à la diversité des situations (évolution des services, changements de l'environnement physique).

Dans ce projet, nous souhaitons mettre en œuvre un mécanisme de sélection sémantique de services basé sur un ensemble de services annotés. Ce mécanisme se repose sur une comparaison sémantique de services. Il pourra ensuite être intégré au sein des schémas d'adaptation dynamique des applications logicielles.

Abstract

In ubiquitous computing, software applications need to dynamically adapt to their execution context facing the diversity of situations (services evolution, physical environment changes).

In this project, we want to implement a semantic selection mechanism of services based on a set of annotated services. This mechanism is based on a semantic comparison of services. Then, it can be integrated into patterns of dynamic adaptation of software applications.