Approche de gestion orientée service pour l'Internet des objets (IoT) considérant la Qualité de Service (QoS)

Guillaume Garzone

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *30/11/2018* **par :**

GUILLAUME GARZONE

APPROCHE DE GESTION ORIENTÉE SERVICE POUR L'INTERNET DES OBJETS (IoT) CONSIDÉRANT LA QUALITÉ DE SERVICE (QoS)

JURY

FRANÇOIS CHAROY	Professeur d'Université	Rapporteur
MICHAEL MRISSA	Professeur d'Université	Rapporteur
PASCAL PEREZ	Professeur d'Université	Examinateur
JEAN-MARC PIERSON	Professeur d'Université	Examinateur
THIERRY MONTEIL	Professeur d'Université	Co-directeur de thèse
NAWAL GUERMOUCHE	Maître de Conférences	Co-directrice de thèse

---

École doctorale et spécialité :

ED : MITT – Mathématiques, Informatique et Télécommunications de Toulouse, Spécialité : Informatique et Télécommunications

Unité de Recherche :

LAAS-CNRS – Laboratoire d'Analyse et d'Architecture des Systèmes (UPR 8001)

Directeur(s) de Thèse :

Thierry MONTEIL et *Nawal GUERMOUCHE*

Rapporteurs :

François CHAROY et *Michael MRISSA*

Remerciements

Avant toute chose je tenais à remercier tout particulièrement Nawal et Thierry pour avoir cru en moi et m'avoir sincèrement aidé tout au long de ces années de thèse. Vous m'avez beaucoup appris quant à la Recherche en général mais aussi dans le développement de ces travaux et du point de vue personnel. Malgré le chemin semé d'embûches et les problèmes contraints de planification dont relève la recherche d'intersection des emplois du temps, vous avez su vous impliquer ainsi que m'apporter un support et une collaboration de qualité.

Je remercie également Michael Mrissa et François Charoy d'avoir accepté de rapporter cette thèse, Pascal Perez et Jean-Marc Pierson d'en être examinateurs ainsi que Cyrille Sauvignac et Eric Szymkowiak pour les projets sur lesquels j'ai travaillé avec vous durant ces années.

Pour le soutien et la compagnie quotidienne de qualité, je tiens à remercier mes chers collègues et amis, pour nos échanges riches et constructifs (ou pas). Nicolas, pour ton indéfectible jovialité, et qui plus est toujours prêt à aider : malgré ton stress et tes propres difficultés tu sais être là pour les autres et c'est une grande qualité. Chloé, merci à toi – excepté pour un certain surnom que je citerai pas – qui a contribué à maintenir ma motivation et détermination même après que tu aies pris un autre chemin que celui du labo. Karima, de supporter les trois – que dis-je quatre ! – compères de bureau et pour ton amabilité et ton professionnalisme. Bon courage pour la suite, le relais est entre tes mains désormais. Santi, pour avoir réussi à supporter et à t'intégrer avec les différents compères INSAïens dont tu étais assailli dans ce bureau. Hang, pour ta sympathie et de ton humilité pour avoir eu à cohabiter avec nous tous au commencement de ta thèse alors que tu débutais à peine le français. Bon courage pour ta soutenance à venir !

Je remercie également tous ceux qui ont contribué aux résultats de cette thèse de près ou de loin. Léa, Xavier et Vivien, ce fut un plaisir de travailler avec vous sur vos stages et d'apprendre à vous connaître, je ne vous ai pas oubliés !

À toutes les personnes de l'équipe qui m'ont apporté leur sagesse et leur avis critique sur beaucoup d'éléments, je pense notamment à Khalil que je remercie chaleureusement de m'avoir grandement conseillé avant de prendre la décision de me lancer dans ce marathon mais aussi à des moments critiques de ce dernier. Les anciens de l'équipe et les nouveaux, les collaborateurs, Mahdi, Ghada, Cédric, Yassine, Slim, Christophe, Pascal, Quentin, Rémy, Imane, Clément, Iovi, Tanissia, Nicola, Jean-Fabien, Gabriel, Darius et tant d'autres.

Une pensée amicale également pour toutes ces personnes que l'on oublie trop souvent mais qui au quotidien vous facilitent la tâche avec une amabilité et une humanité sincères dans toutes ces démarches administratives sans fin et récurrentes : Agnès et Martine de l'EDMITT, Daisy et Lynda de l'INSA, et Caroline, Layla, Justine et Marie-Agnès du LAAS, merci à vous. Je pense aussi à Florence et Patricia, qui ont eu à affronter les aléas de l'IoT et qui ont su le faire sans perdre le sourire !

Enfin, un immense et sincère merci à tous ceux qui m'ont supporté (dans tous les sens du terme) à côté du laboratoire et qui ont tous contribué à leur façon à cette

réussite même sans s'en apercevoir. Si la thèse est une épreuve, la colocation en est une autre et, François, un grand merci à toi pour tout ce que l'on a pu partager et d'avoir réussi à ce l'on se supporte pendant d'intenses moments de stress respectif ! Merci à tous mes proches et ma famille sans qui je ne serais pas arrivé où j'en suis aujourd'hui et de m'apporter un support sincère quand il le faut. Une mention spéciale à tous nos moments privilégiés et goûters, un grand merci à Chacha pour tout le soutien et les bulles d'oxygène si vitales que tu as pu m'apporter. Pour tous nos hauts et nos bas, un grand merci Alex pour ton support sans faille malgré tout et tes playlists d'aide à la rédaction. Bon courage à toi pour cette aventure qu'est la thèse !

À ma grande famille, *celle du sang ou de choix* : MERCI.

Aptitude à faire face avec succès à une situation représentant un stress intense en raison de sa nocivité ou du risque qu'elle représente, ainsi qu'à se ressaisir, à s'adapter et à réussir à vivre et à se développer positivement en dépit de ces circonstances défavorables.

– ***Résilience***

Table des matières

Introduction **1**

1 Contexte scientifique et État de l'art **5**
1.1 Les standards, plateformes et protocoles de l'IoT 6
1.1.1 Les protocoles les plus répandus 6
1.1.2 Le consortium et le standard oneM2M 7
1.1.3 D'autres standards et plateformes de l'écosystème IoT 8
1.1.4 *Device Management* : un exemple avec LWM2M 9
1.2 Systèmes orientés services et composition 10
1.2.1 La composition de services : un panel d'approches 11
1.2.2 Conclusions 17
1.3 Les systèmes autonomiques 18
1.3.1 L'informatique autonomique *(autonomic computing)* 18
1.3.2 Quelles implications dans notre contexte IoT ? 20
1.4 Web sémantique, ontologies de l'IoT et des services 22
1.4.1 Web Sémantique, Ontologies, règles SWRL 22
1.4.2 Des ontologies reconnues 23
1.4.3 Conclusions 25
1.5 Grammaires de graphes et réécriture de graphes 26
1.5.1 Définitions : grammaires de graphes et réécriture de graphes . 26
1.5.2 Quelle utilisation ? 27
1.6 Les projets STM & S2C2 28
1.6.1 Présentation générale du projet STM 28
1.6.2 Présentation générale du projet S2C2 29
1.6.3 Des scénarios dans la ville intelligente 29
1.6.4 Le scénario d'utilisation principal 31
1.6.5 Contraintes et problématiques liées à ce contexte 33

2 Vers un système autonome de gestion des services d'une architecture IoT ouverte **37**
2.1 Architecture haut-niveau du système proposé 38
2.1.1 Les composants du système 38
2.1.2 Évènements et comportement associé 40
2.2 Le modèle basé sur une ontologie 40
2.2.1 Instance d'ontologie 41
2.2.2 Règles SWRL et Analyse 45
2.3 Les modèles orientés graphe 47
2.3.1 Requête de changement à effectuer (RFC) 48
2.3.2 Génération de plan d'exécution 49

3 Gestion de services IoT complexes avec paramètres de QoS 55
3.1 Vers une gestion plus haut-niveau des services 56
3.1.1 Principe général 56
3.1.2 Modèle enrichi de RFC 58
3.1.3 Intégration de besoins et de services à la volée 60
3.2 Construction et exécution d'un plan opportuniste 73
3.2.1 Approche générale 73
3.2.2 Algorithme d'exploration et de recherche d'optimum : vers un choix optimisé 79
3.2.3 Autre algorithme témoin pour tests de performance : glouton 82
3.2.4 Évaluation de résultat des algorithmes et validation des contraintes 84
3.2.5 Exemple complet d'application des algorithmes et comparaison de résultats 84

4 Étude d'évaluation du système et prototype 89
4.1 Temps de réponse des composants et performance 90
4.1.1 Configuration du système 90
4.1.2 Analyse sémantique et transformation de graphe 90
4.1.3 Analyse du temps de réponse dans une gestion complexe de services 94
4.1.4 Conclusions des évaluations théoriques 99
4.2 Preuve de concept : prototype et déploiement 100
4.2.1 Eclipse OM2M : un intergiciel standard pour l'IoT 101
4.2.2 Le système IoT de prototype et démonstration 101
4.2.3 Le prototype déployé du système autonomique 104
4.2.4 Analyses et conclusions 107

Conclusions et Perspectives 111

Acronyms 117

A Les standards, consortiums et plateformes de l'IoT 119
A.1 Les standards et consortiums 119
A.1.1 ETSI Smart M2M 119
A.1.2 Open Connectivity Foundation (OCF) 119
A.1.3 ICC *(Industrial internet Consortium* 119
A.1.4 Thread, Weave *(Google)* 120
A.1.5 Homekit *(Apple)* 120
A.1.6 FiWARE 121
A.1.7 OSGi 121

B Compléments système autonomique **123**
B.1 Compléments informations boucle autonomique 123
B.2 Compléments grammaires de graphes 123
B.3 Compléments grammaire de graphes complexe 123

C Compléments algorithmes d'exploration **127**
C.1 Algorithme de recherche d'optimum 127
C.1.1 Structure de données *Result* 127
C.1.2 Pseudo-code recherche d'optimum 128
C.1.3 Autres méthodes 128

D Compléments étude d'évaluation **131**
D.1 Résultats détaillés 131
D.1.1 Analyse de performance de la grammaire avancée et des algorithmes d'exploration 131
D.2 Compléments d'information Plug-in LWM2M 131

Bibliographie **133**

Introduction

L'internet des objets

Ces dernières années les *objets connectés* ont connu une véritable démocratisation avec toujours plus d'accessoires connectés qui débarquent dans notre quotidien, notre domicile, notre environnement de travail mais aussi les villes. Sans le savoir, nous croisons tous les jours des objets connectés intégrés dans notre environnement.

L'**internet des objets** (*en anglais : **Internet of Things (IoT)***), s'est incroyablement développé incluant toujours plus de technologies, d'objets innovants allant des capteurs sans fils à longue autonomie que l'on déploie chez soi à de véritables infrastructures complexes de plus grande ampleur. Au niveau recherche, l'IoT suscite un intérêt fort avec plus de 6000 publications (articles de journal) déjà recensées sur IEEEXplore [1] dont presque 90% ont été publiées ces huit dernières années (2010 – 2018).

L'IoT se définirait d'après [Gubbi 2013] par "***[une] interconnexion de capteurs et d'actionneurs permettant de partager des informations à travers des plateformes via un cadre unifié, en développant une image opérationnelle commune pour permettre le développement d'applications innovantes. [. . .]***"

D'autres terminologies sont parfois utilisées pour parler de ces différents concepts de systèmes *cyberphysiques*, ou même *Internet of Everything*, intégrant à la fois des aspects virtuels et logiciels et des objets physiques en contact avec le monde réel et qui peuvent agir sur ce dernier.

L'idée demeure dans l'expression d'une diversité certaine et d'une dimension toujours plus affolante : d'ici 2020 des dizaines de milliards d'objets connectés déployés sont attendus à l'échelle mondiale [Nordrum 2016]. Tout devient *smart*, intelligent : les objets, les maisons *(smart home)*, les bâtiments (*smart building)*, les usines *(smart factory)*, les villes *(smart city)*, etc. L'intégration de systèmes de capteurs et actionneurs divers, contrôlables à distance rend possible des applications qui étaient encore difficilement envisageables il y a quelques années et débloque la mise en place de nouveaux services complexes. En effet, tous ces objets physiques connectés vont permettre de fournir de nouveaux services par le biais de capteurs (température, présence, pollution, etc.), d'actionneurs (dispositifs contrôlables à distance agissant sur l'environnement) et d'autres entités, et peuvent être considérés en eux-mêmes comme de nouveaux services [Cherrier 2014] dans une vision d'informatique orientée service *(Service Oriented Computing)*. Ces nouveaux services pourraient être intégrés avec de nombreux services déjà présents aujourd'hui comme par exemple des services de logistique ou de gestion de ville, ou encore de prévisions météo ou encore de banques de données accessibles via des interfaces web. De plus, la combinaison de ces technologies avec des approches de Cloud Computing et d'Intelligence

1. `https://ieeexplore.ieee.org`

Artificielle ouvrent de nouvelles perspectives et possibilités de conception et déploiements de nouveaux services applicatifs complexes et composites à haute valeur ajoutée.

Dans le contexte de la ville intelligente, le déploiement à grande ampleur d'objets connectés et d'applications transverses intégrant les besoins et les éléments de différents domaines apporteraient de nombreuses améliorations pour les villes et les citoyens. Certaines de ces pistes ont été explorées dans [Zanella 2014]. Le déploiement de systèmes IoT dans les villes en combinaisons avec des solutions d'analyse de données et l'élaboration de services complexes permettrait d'engendrer de nombreuses améliorations de la qualité de vie et économies pour les agglomérations.

En Europe, des initiatives ont débuté en ce sens afin de travailler à ces problématiques de ville intelligente, par exemple en Espagne la ville de Santander [2] [Cheng 2015] se considère pionnière dans ses initiatives de déploiement à grande échelle d'objets connectés pour l'optimisation du stationnement, de la gestion des déchets, de la qualité de l'air et de l'eau, etc. Bien sûr d'autres initiatives se développent de plus en plus en Europe et ailleurs, et en France où les apports nombreux qu'amènent ces solutions motivent les villes à prendre des initiatives. Quelques exemples de solutions et améliorations rendues possibles par l'installation d'infrastructures IoT résident dans la gestion intelligente des déchets, de la qualité de l'air dans les grandes agglomérations, mais aussi de manière plus générale de la consommation énergétique ou encore des transports et du trafic, en passant par la surveillance de l'état de bâtiments ou des améliorations pour les cas d'urgence. Le nombre d'applications est proche de l'illimité, les limites résidant aujourd'hui dans la complexité de mise en œuvre et les besoins et motivations des entités impliquées.

Cependant, bien qu'ayant réalisé de grandes avancées les années passées, de nombreux défis et problématiques persistent encore aujourd'hui dans le développement de l'IoT.

Problématiques & Défis

Un premier enjeu majeur encore d'actualité réside dans l'***interopérabilité***. En effet, cet écosystème vaste et complexe de technologies, de protocoles et de standards rend complexe le déploiement d'applications intégrant différentes technologies ou différentes sources de données.

Une première étape nécessaire réside dans l'**interopérabilité syntaxique** : être capable d'utiliser différentes technologies de manière transparente et homogène en intégrant des technologies hétérogènes et incompatibles. Un exemple simple est de considérer un bâtiment avec une technologie de gestion de l'éclairage connecté d'une technologie A possédant également une gestion de détection de mouvement assurée par une technologie B non compatible avec A. Le coût de développement d'un service intégrant ces deux technologies peut s'avérer coûteux et ce coût augmente encore dès l'arrivée d'une troisième technologie C incompatible avec les précédentes.

2. `http://www.smartsantander.eu`

Ces problématiques font l'objet de travaux importants qui seront abordés par la suite.

Une étape supplémentaire réside dans l'**interopérabilité sémantique** : être capable d'utiliser différentes technologies de manière harmonisée est une chose, mais permettre aux systèmes de comprendre de manière automatique les différentes données produites par ces capteurs hétérogènes aux formalismes de données disparates demeure un défi. Un exemple de ce type de problématique et de solution a été présenté dans [Aïssaoui 2017].

Outre ces aspects, d'autres enjeux d'importance demeurent ne serait-ce que dans le déploiement physique de toutes ces entités connectées sur des réseaux et de tous les problèmes que cela engendre. La **scalabilité** de solutions IoT reste aujourd'hui encore une problématique de recherche active de par la variété de capacités et de types de communications impliquée dans de tels déploiements.

Au delà de ces enjeux, être capable d'intégrer différents éléments que ce soit objets ou services déjà existants déployés sur internet reste complexe. **La gestion** de ce type de systèmes peut demander beaucoup de ressources et de configurations qu'il serait avantageux de pouvoir automatiser et configurer à l'aide de politiques de gestion haut-niveau. En effet, outre les difficultés de scalabilité et d'interopérabilité, l'IoT présente une certaine **versatilité** avec des objets aux connexions changeantes, aux états variables et parfois mobiles. La définition de systèmes capables de gérer ce type d'architecture demeure complexe et les modèles souvent difficilement adaptables.

Enfin, des difficultés résident dans la capacité d'**intégrer automatiquement les services** fournis par ces objets connectés dynamiques avec des services existants dans le but de répondre à des besoins d'utilisateurs. De plus, être capable de mutualiser le recours à certains objets ou services de manière plus générale comporte un intérêt dans ce contexte afin d'optimiser les appels de services quand cela est possible par exemple.

Cependant il serait intéressant d'être capable de réaliser cette intégration en privilégiant la généricité et la modularité.

Organisation du manuscrit et des contributions

Le reste de cette thèse est organisé comme suit :

Chapitre 1 : Contexte scientifique & État de l'art
Dans ce chapitre, des éléments de contexte dans le domaine de l'IoT sont abordés dans un premier temps, notamment par rapport aux standards et protocoles qui seront récurrents. Dans un deuxième temps, le projet S2C2 dans lequel s'inscrivent ces travaux est présenté pour poser d'autres éléments de contexte et introduire le scénario central dans cette approche.

Ensuite, différents aspects du contexte scientifique et de l'état de l'art sont détaillés en rapport avec les systèmes orientés services et la composition de services, les

systèmes autonomiques, le web sémantique et les ontologies de l'IoT et de services, et enfin les grammaires de graphes et la réécriture de graphes.

Chapitre 2 : Vers un système autonome de gestion des services d'une architecture IoT ouverte
Dans ce deuxième chapitre, la première contribution est détaillée : elle consiste en une architecture de système autonomique de gestion d'environnement IoT avec une vision orientée services. L'idée de cette proposition est de faciliter la gestion de ces systèmes IoT et de permettre la combinaison des services fournis pas les objets avec des services existants. La particularité de cette approche repose dans son aspect multi-modèles, combinant un modèle à base d'ontologie et un modèle à base de grammaire de graphes. Cette démarche permet de manière autonome de réagir à des évènements qui occurrent sur le système supervisé, d'effectuer une analyse avec une inférence de connaissance. De plus, le modèle de transformation de graphe permet de générer un plan d'exécution à base de services et de mutualiser ces derniers lorsque l'opportunité se présente. Des instances de modèles dans notre scénario sont présentées accompagnées d'exemples.

Chapitre 3 : Gestion de services IoT complexes avec paramètres de QoS
Dans ce troisième chapitre, l'approche présentée précédemment est enrichie afin d'inclure plus d'éléments d'autres contributions. Des besoins temporaires sont intégrés au modèle de graphe afin de pouvoir inclure à la volée des instances de services au besoin pour plus de souplesse. Des opérateurs plus complexes d'organisation du plan d'exécution sont également introduits afin de structurer le plan d'exécution en introduisant plus de possibilités d'exécution. De plus, des paramètres non fonctionnels sont intégrés au modèle afin de permettre de réaliser des choix parmi les services disponibles et d'optimiser l'exécution finale. D'un autre côté, des algorithmes d'exploration sont introduits afin d'être appliqués sur les plans d'exécution générés et de réaliser une optimisation du plan final si possible.

Chapitre 4 : Étude d'évaluation du système et prototype
Dans ce chapitre une étude d'évaluation de l'approche proposée est réalisée. Tout d'abord une caractérisation du temps de réponse des différents modèles proposés est détaillée. Par la suite, une preuve de concept est réalisée via l'instance des modèles proposées et leur intégration dans un prototype de système IoT sous forme de maquette incluant un bus connecté.

Enfin pour conclure, un bilan des différentes contributions et problématiques traitées dans cette thèse est abordé accompagné de perspectives de travail en continuité.

Chapitre 1

Contexte scientifique et État de l'art

Sommaire

1.1 Les standards, plateformes et protocoles de l'IoT 6
1.1.1 Les protocoles les plus répandus 6
1.1.2 Le consortium et le standard oneM2M 7
1.1.3 D'autres standards et plateformes de l'écosystème IoT 8
1.1.4 *Device Management* : un exemple avec LWM2M 9
1.2 Systèmes orientés services et composition 10
1.2.1 La composition de services : un panel d'approches 11
1.2.2 Conclusions 17
1.3 Les systèmes autonomiques 18
1.3.1 L'informatique autonomique *(autonomic computing)* 18
1.3.2 Quelles implications dans notre contexte IoT ? 20
1.4 Web sémantique, ontologies de l'IoT et des services 22
1.4.1 Web Sémantique, Ontologies, règles SWRL 22
1.4.2 Des ontologies reconnues 23
1.4.3 Conclusions 25
1.5 Grammaires de graphes et réécriture de graphes 26
1.5.1 Définitions : grammaires de graphes et réécriture de graphes . 26
1.5.2 Quelle utilisation ? 27
1.6 Les projets STM & S2C2 28
1.6.1 Présentation générale du projet STM 28
1.6.2 Présentation générale du projet S2C2 29
1.6.3 Des scénarios dans la ville intelligente 29
1.6.4 Le scénario d'utilisation principal 31
1.6.5 Contraintes et problématiques liées à ce contexte 33

Les domaines d'applications dans l'IoT sont vastes et les défis encore aujourd'hui nombreux : que ce soit tant sur les problèmes d'interopérabilité syntaxique ou sémantique, que sur la complexité de gérer ces systèmes complexes et de les intégrer avec des systèmes et services déjà existants.

Dans un contexte de ville intelligente, de nombreuses problématiques se posent, tant sur le niveau physique (pour le déploiement et la gestion des objets), qu'au

niveau transport de données (*e.g.*, pour le trafic réseau généré par tous ces objets). Des problématiques se posent au niveau applicatif dans le cas d'application complexes incluant différentes technologies et domaines.

En effet, un défi important réside dans la complexité de combiner les différents objets connectés entre eux mais aussi toutes les données générées.

Dans un tel écosystème, différentes entités (protocoles, standards et outils) existent et présentent des intérêts comme l'optimisation de la consommation de données réseau ou l'optimisation d'autonomie énergétique des objets. L'accent est mis dans un premier temps sur ces éléments. Ensuite, le contexte du projet S2C2 et du projet STM dans lesquels s'inscrit cette thèse est abordé afin de poser le cadre d'application utilisé dans ces travaux et le cas d'utilisation considéré afin de poser différents éléments de problématique.

En réponse à ces éléments, différents axes sont étudiés. Tout d'abord la composition de services est explorée, dont les contributions sont nombreuses dans la littérature. L'étude de ces approches permettra de comprendre l'intégration de techniques existantes et éprouvées dans ce domaine. Les systèmes autonomiques représentent des avantages intéressants dans le contexte de l'IoT et du projet S2C2. Ces derniers sont étudiés dans un deuxième temps. Puis, des technologies utiles du Web sémantique sont présentées pour la modélisation de systèmes impliquant des services et des objets connectés avec les avantages qu'elles représentent. Enfin, les technologies de grammaires de graphes et de réécriture de graphes sont abordées pour leurs avantages, notamment en rapport avec la gestion de plans d'exécution incluant des services.

1.1 Les standards, plateformes et protocoles de l'IoT

Afin de contextualiser le travail de cette thèse, différents standards, plateformes et protocoles aujourd'hui considérés dans l'IoT sont abordés.

1.1.1 Les protocoles les plus répandus

Différents protocoles sont utilisés de par l'IoT. Les principaux qui seront cités régulièrement dans la suite de cette thèse sont :

- HyperText Transfer Protocol (HTTP) [1] : ce protocole est au cœur du fonctionnement du Web tel qu'on le connaît. HTTP est basé sur Transmission Control Protocol (TCP) qui permet d'établir une connexion entre les entités concernées. L'établissement de cette connexion peut être coûteuse et HTTP est un protocole assez verbeux ce qui ne le rend pas forcément adapté à toutes les architectures IoT. HTTP est également à la base des architectures REpresentationnal State Transfer (REST) qui se basent sur les opérations du protocole HTTP pour définir les opérations sur les ressources

1. `https://tools.ietf.org/html/rfc2616`

- Constrained Application Protocol (CoAP) [2] : est un protocole inspiré d'HTTP (il se base sur les mêmes types d'opération) mais utilise le protocole User Datagram Protocol (UDP). De plus la taille de paquet applicatif est très limitée afin de minimiser les données échangées sur le réseau.
- Message Queuing Telemetry Transport (MQTT) [3] : est un protocole de type *publish / subscribe*. De fait, des clients MQTT vont pouvoir publier des messages sur un serveur intermédiaire *(broker)* via des sujets *(topics)* auxquels il est également possible de s'abonner. Il est ainsi facile de s'abonner à un sujet et d'être notifié quand un nouveau message est publié. De multiples clients peuvent s'abonner au même sujet et publier dans un même sujet.

1.1.2 Le consortium et le standard oneM2M

oneM2M [4] est un organisme de normalisation fédérant de nombreux organismes régionaux pour les communications de machine à machine (Machine-to-Machine (M2M)), sur le développement d'un système M2M interopérable et a été formé en 2013. Son objectif principal est de développer des spécifications techniques rassemblant celles des organismes de normalisation régionaux : ATIS et TIA en Amérique du Nord, ETSI en Europe, CCSA, ARIB, TTA et TTC en Asie. Ce regroupement a été effectué dans le but d'élargir le marché et de permettre une véritable concurrence sur les services. Le consortium regroupe également plus de 200 industriels. La norme oneM2M développée par le consortium peut s'appliquer à de nombreux domaines d'activité liés à l'IoT comme les villes intelligentes, les transports connectés, la e-santé, la logistique, l'industrie connectée, la maison intelligente, etc. oneM2M répond au besoin d'une couche de services commune pouvant être facilement intégrée à divers matériels et logiciels, et permettant de connecter des périphériques sur le terrain avec des serveurs d'applications.

Le standard définit un ensemble de services qui seront fournis par les plateformes et exposés sous la forme d'une Application Programming Interface (API) REST accessible via divers protocoles (*e.g.*, HTTP, MQTT) et avec une sérialisation eXtensible Markup Language (XML) ou Javascript Object Notation (JSON) des données échangées.

Afin de supporter des services bout à bout, la plateforme oneM2M est structurée en trois couches : la **couche applicative**, la **couche de services**, et la **couche dédiée transport** de données (réseau). Dans la couche applicative, des AE (*Application Entity*) vont représenter différentes applications enregistrées sur le système. Elles peuvent symboliser des objets, des applications mobiles, un gestionnaire greffé sur la plateforme, etc. Dans la couche de services, différents CSE (*Common Services Entity*) permettront aux AE de s'enregistrer sur ces derniers et d'interagir avec les ressources qui y sont stockées. Ces CSE pourront être hébergés sur des serveurs principaux, des serveurs intermédiaires, des passerelles, etc., et représentent

2. `http://coap.technology/`
3. `http://mqtt.org`
4. `http://www.onem2m.org`

un ensemble de services fournis par le nœud (*e.g.*, la possibilité de s'enregistrer, de découvrir des ressources, de s'abonner à certaines ressources et changements, etc.). La couche de transport permet de modéliser les services de transport de données (protocoles) mais aussi d'établir une connexion avec des protocoles comme de la gestion de flotte (*Device Management*).

Le modèle de données de oneM2M se base sur un modèle orienté ressources (REST). Des informations vont ainsi être stockées sous forme de ressources sur les nœuds de la topologie et être accessibles via les interfaces. L'architecture de communication définit différentes interfaces pour communiquer avec la couche applicative, pour la communication entre couches de services et pour la communication avec la couche de transport.

Différents types de nœuds sont ainsi définis en fonction de leur capacité à fournir tout ou partie des services oneM2M ainsi que leur capacité d'héberger l'enregistrement d'AE.

Le protocole de communication oneM2M est défini de manière générique à l'aide de primitives indépendamment du protocole de transport qui sera utilisé. (À ce jour les protocoles HTTP, MQTT, CoAP et Web Socket sont supportés par le standard).

L'**interopérabilité syntaxique** de oneM2M est donc forte d'un côté pour l'interopérabilité entre différentes plateformes implémentant le standard, mais aussi avec des connexions (*binding*) définies et possibles avec des protocoles autres comme LightWeight M2M (LWM2M) par exemple. De plus, le standard définit des *Interworking Proxy Entitiy* (IPE) qui permettent de formaliser une interface avec un autre standard / protocole ou technologie.

Également, dans ses dernières versions (depuis la deuxième parution de 2017), le standard travaille à apporter une **interopérabilité sémantique** en intégrant une gestion d'ontologie et de méta-données sémantiques dans les ressources du standard.

Le standard intègre aussi des mécanismes de sécurité et de gestion de droits d'accès aux différentes ressources.

De nombreuses implémentations du standard existent industrielles et open-source [5].

1.1.3 D'autres standards et plateformes de l'écosystème IoT

L'écosystème IoT est vaste et peuplé de nombreux standards et plateformes. On peut citer (pour plus de détails *cf.* Annexe A) :

- *ETSI Smart M2M* défini par l'ETSI
- le consortium Weave/Thread (supporté par Google)
- OCF
- ICC
- Homekit (supporté par Apple)
- FiWare
- OSGi Alliance

5. Une présentation des implémentations open source est disponible ici :
`http://www.onem2m.org/developers-corner/tools/open-source-projects`

Bien d'autres standards et protocoles sont en jeu dans l'IoT.

1.1.4 *Device Management* : un exemple avec LWM2M

La gestion de flotte d'objets *(device management)* s'avère également une problématique forte dans l'écosystème IoT. Afin de répondre à ce problème différents protocoles de gestion ont été développés.

Un des protocoles principaux de *device management* est OMA LightWeight M2M (LWM2M) qui est déjà vastement utilisé au niveau industriel bien que sa première version standardisée n'ait été publiée qu'en Février 2017.

LWM2M se base sur un modèle orienté ressources (architecture REST) et sur le protocole CoAP afin d'alléger les communications au maximum. Les communications peuvent également être sécurisées par l'échange de clefs et l'utilisation de DTLS. Le protocole met en place une architecture client / serveur avec un serveur permettant de centraliser la gestion de la flotte d'objets, l'état des objets et permettre des opérations comme la mise à jour à distance de micrologiciel, etc. Les différents services sont accessibles via une API REST côté serveur. Les différentes ressources vont permettre d'effectuer des opérations comme la création, la récupération, la mise à jour, la suppression, la configuration ou l'exécution. De fait, via des ressources, les objets (clients) vont pouvoir transmettre des informations (*e.g.*, sur les objets physiques : mémoire, niveau de batterie, etc., et les éléments logiciels exécutés) à un serveur qui va centraliser l'information. Il est également possible d'utiliser un mécanisme de souscription / notification entre le serveur et les clients pour obtenir des mise à jour lors de la modification de certains paramètres.

Différentes implémentations de ce standard existent, notamment en open-source comme Eclipse Leshan [6] (client / serveur, *Java*, licence EPL), Eclipse Wakaama [7] (client, *C*, licence EPL), ARM Mbed [8] (*C*, licence Apache).

Le protocole définit également des opérations de *bootstrap* afin de connecter dynamique des objets et assurer une meilleure sécurité par un système de clefs / token partagés.

En résumé

De nombreux protocoles existent dans l'IoT et des standards se développent pour fournir des solutions aux problèmes d'interopérabilité, et surtout syntaxique. Certains standards s'adressent à des domaines en particulier et d'autres cherchent une vision plus transverse et multi-domaines afin de fournir une solution face aux problèmes d'interopérabilité syntaxique. L'intérêt de ces standards transverses est de pouvoir donner une vision services haut niveau des objets connectés sous-jacents hétérogènes via des API standardisées.

6. `https://www.eclipse.org/leshan/`
7. `http://projects.eclipse.org/projects/technology.wakaama`
8. `https://www.mbed.com/en/`

De plus, des standards comme oneM2M s'intéressent également à des technologies comme le web sémantique afin de répondre aux problématiques d'interopérabilité sémantique dans l'IoT. En effet, ces problématiques s'avèrent primordiales surtout dans le cas d'entités multi-domaines qui partagent des informations via des objets connectés hétérogènes.

Dans notre contexte, il est donc particulièrement intéressant de se baser sur ce genre d'architecture IoT et de standard, d'autant plus qu'il est également possible d'intégrer d'autres types de services comme des services de *Device Management*. De fait, l'API REST fournie par les plateformes oneM2M permettrait d'avoir accès aux objets via l'utilisation de services REST et le protocole oneM2M.

1.2 Systèmes orientés services et composition

Lorsque l'on considère des systèmes orientés services ou que l'on travaille avec des Architectures Orientées Services (SOA), il est souvent question de composition de services.

La composition de services est une thématique de recherche active, avec presque 3600 publications (publications dans journaux) recensées sur IEEEXplore [9] dont plus de 70% de ces publications publiées au cours des 10 dernières années (2008 – 2018). Historiquement, cette thématique provient de problématiques se posant dans l'utilisation des services web et leur composition [Srivastava 2003]. En effet, lors de la définition de processus métiers, l'instanciation de ces processus peut vite devenir très laborieuse en présence de nombreux services hétérogènes à composer à la main. Ces problématiques se sont ensuite enrichies et étendues avec l'arrivée de plus en plus de nouveaux formalismes et de services rendant plus complexe cette composition, ainsi que des problématiques non fonctionnelles liées à des paramètres de Quality of Service (QoS) ainsi que des contraintes sur ces derniers. Par exemple dans certains cas le temps de réponse du système est fortement contraint ou la fiabilité des services utilisés doit être maximale.

De manière générale dans la littérature, la composition de services regroupe différentes approches :

- La **composition de services** [Srivastava 2003] : regroupe à la fois différentes approches mais signifie également l'action de composer différents services pour générer un nouveau service à valeur ajoutée, *i.e.*, utiliser différents services pour répondre à un besoin ou générer un nouveau service à valeur ajoutée par exemple
- L'**orchestration de services** [Gortmaker 2004] : le fait de composer des services dans un processus défini (par exemple dans un plan BPEL instancié), *i.e.*, utiliser différents services dans un ordre séquentiel afin d'effectuer un traitement successif sur un jeu de données. L'exécution est souvent conduite par un orchestrateur qui exécute le plan d'orchestration. L'orchestration est combinable avec d'autres mécanismes.

9. `https://ieeexplore.ieee.org`

- La **chorégraphie de services** [Barros 2005] définit le fait de faire partie d'un plan de composition et de faire des efforts pour faciliter cette composition, et donc que les services ont connaissance de cette composition
- La **coordination de services** [Dunst 2002] : dans ce cas là, il n'y a pas de communication directe entre les services mais elles passent par un coordinateur
- Le ***raccord* de services** (*wiring*) [Campos 2014] est le fait de fournir une interface propre et de décrire les éléments requis afin de pouvoir connecter directement des services entre eux
- La **sélection de services** [Yu 2005] revient à *choisir* les "meilleurs" services pour effectuer la tâche requise parmi un ensemble de services candidats, généralement cette sélection se base sur un plan d'exécution bien défini avec un ensemble de services candidats pour chaque tâche. Le but de la sélection est de nettoyer les services candidats éventuellement et de choisir la meilleure combinaison de services (parfois pour satisfaire des contraintes et/ou pour optimiser l'exécution).

La littérature concernant la composition de services et particulièrement la composition de services dans l'IoT est assez riche d'approches différentes. Un panel des contributions étudiées est présenté dans la section suivante.

1.2.1 La composition de services : un panel d'approches

Une grande partie de la littérature dédiée à la composition de services se concentre sur le développement de méthodes formelles de composition et algorithmes. Le but des approches est de pouvoir instancier automatiquement des plans de composition basés sur un plan *abstrait* de services et un ensemble de services connus, candidats pour les tâches à effectuer voire de composer certains services pour répondre à un besoin. Le plan *abstrait* se définit en un ensemble de tâches ou de services *abstraits* à instancier avec des services réels. Ce type de plans peuvent être par exemple définis dans un formalisme comme Business Process Model and Notation (BPMN) [10] [White 2008], (proposé initialement en 2004 et aujourd'hui maintenu par l'Object Management Group (OMG) [11]), qui permet de définir des processus métiers avec un certain niveau de détails dans les activités possibles, les flux de données / messages, des opérateurs de composition, etc. L'idée est d'aboutir par la suite à des processus implémentés sous forme de services interconnectés, par exemple dans un formalisme Business Process Execution Language (BPEL) [12] [Küster 2009]. BPEL est un langage qui permet de modéliser ces processus de services et les connexions entre ces derniers par la définition d'une orchestration de services (classiquement des services web).

Ce plan va pouvoir être lu par une entité (orchestrateur de services) qui exécutera lesdits services dans l'ordre désiré en réalisant les différentes opérations spé-

10. http://www.bpmn.org
11. https://www.omg.org
12. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

cifiées et en connectant correctement les entrées / sorties des différents services concernés. La difficulté dans ce contexte réside donc à "choisir" les services en fonction de l'activité qui doit être remplie (sélection de services). Effectuer ce choix manuellement peut s'avérer coûteux suivant les différents cas d'utilisation et pouvoir accomplir cette tâche de manière automatique représente un avantage par exemple lors de la mise en place de nouveaux processus ou bien lorsqu'il faut remplacer certains services dans des processus existants, etc.

Différents types d'approches en ce sens ont été réalisées afin de répondre à ces problématiques et sont regroupées dans la Table 1.1. Ces différentes approches vont être détaillées par la suite. Un premier point important réside dans des approches formelles qui fournissent des méthodes et des algorithmes pour générer cette composition de manière automatique ou pseudo-automatique en se basant sur un processus de services abstrait et un ensemble de services connus ou services candidats.

Références	Approche	Techniques / méthodes	QoS	Sémantique	Services	Processus	Mutualisation
[Jin Xiao 2005]	provisionning, réseaux	k-MCOP, graphes de domaines	assurance de QoS bout à bout	–	–	–	–
[Chafle 2006]	adaptation, selec.	separation fonctionnel / non fonctionnel	oui	–	WS	predef.	–
[Ardagna 2007]	selec.	opti pb.	contraintes	–	WS	bpel (flex)	–
[Zhovtobryukh 2007]	composition	réseaux pétri	–	–	WS	goal driven	–
[Wu 2007]	interface matching	ontologie, matching in/out	–	DAML-S	WS	–	–
[Chan 2007]	survey, opt. comp.	IA, HTN	oui	–	WS	–	–
[Liu 2009]	composition	eCPN model + GA	Contraintes globales	–	WS	predef.	–
[Guinard 2010]	prov., selec., découv.	DPWS, integration serv. Obj.	–	–	REST	–	–
[Raj 2010]	selec.	qos manager, ranking serv.	rank, constriants	–	WS	–	–
[Gronvall 2011]	service comp.	end user driven, templates	–	–	REST	templates	–
[Mosincat 2011]	maintenance, selec.	monitoring, performance detection	monitoring	–	WS	BPEL	–
[Shi 2011]	survey, comp. Qos-aware	math. program., algo. Heuristiques	oui	–	–	–	–
[Cai 2014]	Lifecycle mgmt.	modèle abstrait, semantique, ROA	–	ont. Spec.	REST	–	–
[Yu 2014]	selec., serv. Matching	max. weighted bipartite, int. Linear. Prog.	multi param., contraintes	–	–	–	–
[Kim 2014]	orchestration	wsdl, ontologie	–	ont. Modèle	WS	predef.	–
[Yachir 2015]	découv., invocation	découverte, invocation et sélection, adhoc	considérée	ont. Pour classification serv. Et tâche utilisateur.	–	?	–
[Aljawarneh 2016]	survey, comp. Context aware	Context awareness, declarative policies, user centric, heterogeneity of services	oui	–	–	–	–
[Rupasingha 2016]	serv. Clustering	text-mining, ont. Relations	–	ont. Dans modèle	–	–	–
[Fähndrich 2016]	comp., selec.	semantic matching	–	OWL-S	WS	predef., bpmn	–
[Ramírez 2017]	optimiser comp.	evolutionary algo.	opti. Multi paramètres	–	WS	–	–
[Guo 2017]	selec.	spark, covering, particle swarm	multi. Param.	–	WS	predef.	–
[Guidara 2017]	selec.	heuristiques (prunning, contraintes)	contraintes QoS, temporelles	–	–	predef.	–
[Chattopadhyay 2018]	selec.	opti. Pb., multi objectif	multi param., contraintes	–	WS	–	–
[Asghari 2018]	survey, opt. Comp.	ant, bee, bat, GA, greedy, hybrid	oui	–	WS	predef.	–
[Alsaryrah 2018]	selec., optimisation	k-shortest path	énergie, autres param.	–	–	predef.	–

TABLE 1.1 – Récapitulatif des approches étudiées

1.2.1.1 Optimisation combinatoire & intelligence artificielle

Différentes contributions dans ce domaine s'intéressent à des algorithmes basés sur des techniques d'Intelligence Artificielle (IA, ou *Artificial Intelligence, AI* en anglais) et de problèmes d'optimisation.

Un panel des approches basées sur des algorithmes d'optimisation est présentée dans [Asghari 2018]. Dans cette étude comparative, différents algorithmes sont comparés comme les algorithmes de colonie de fourmis, de colonie d'abeilles, génétique, chauve-souris, glouton et hybride. L'étude souligne l'importance et les avantages de ce type d'algorithmes dans un contexte de composition de services à un niveau *Cloud*.

A la différence dans [Aljawarneh 2016] les auteurs se concentrent sur un état de l'art de techniques plus conscientes du contexte de la composition de service.

Dans [Chan 2007], les auteurs combinent une étude de différentes méthodes de planification de la littérature basées sur des techniques d'IA en soulignant la prévalence de la méthode HTN (Hierarchical Task Network) [Erol 1996] par rapport aux autres qui permet d'effectuer de la planification automatique en structurant les dépendances entre les tâches dans un réseau hiérarchique. Différentes contributions dans ce domaines se basent sur des méthodes d'IA également ou des méthode de calculs particulières afin de trouver des solutions à des problèmes d'optimisation que pose la génération de plan d'orchestration.

Par exemple, dans [Liu 2016], un algorithme basé sur une Culture Génétique (CGA) est proposé pour résoudre les problèmes d'optimisation combinatoire et trouver une solution optimale localement au problème de composition donné. Cette approche prend aussi en compte des paramètre QoS et leur prédiction dans le processus de sélection de service.

Dans [Wang 2009], une méthode de composition automatique basée sur la méthode *pi-calculus* [Boudol 1992] appliquée à la composition de services Web sémantiques afin de générer des services web à la demande. La correspondance entre les services web existants et les sous-services sont vérifiés par la formalisation et la vérification avec pi-calculus.

Dans [Ardagna 2007], une approche de résolution de problème d'optimisation dans la sélection de services est proposée ainsi qu'une optimisation des plans d'orchestration se basant sur de la transformation de graphes. Cette approche se place dans un contexte de plans d'exécutions complexes et avec de nombreux éléments et en présence de contraintes de QoS fortes.

Dans [Liu 2009], les auteurs apportent une approche permettant à la fois de modéliser les dépendances et les contraintes entre différents services en se basant sur des réseaux de pétri colorés étendus (*extended Colored Petri Net, eCPN*) [Jensen 1981] [Shu 2010] dans le but de faire de la composition de services semi automatique. Ensuite, un algorithme génétique (*Genetic Algorithm (GA)*) est utilisé afin d'optimiser la solution notamment en vue de l'exécution.

Dans [Jin Xiao 2005], l'approche proposée permet une modélisation par graphes de domaines des problèmes abordés dans une thématique de communication auto-

nomique (réseaux) ainsi qu'une optimisation des paramètres QoS bout à bout via l'utilisation d'algorithmes k-MCOP (*k-multicontrained optimal path*).

Dans [Zhovtobryukh 2007], l'approche proposée permet de réaliser une composition de services en se basant sur des services existants et une modélisation basée sur des réseaux de pétri. Un ensemble d'opérateurs de composition sont définis afin de composer les services web considérés. Cette composition est guidée par un besoin initial qui est raffiné en différentes étapes à remplir comme par exemple dans un processus de réservation complète de déplacement comprenant le transport mais aussi les logements, etc.

1.2.1.2 Sélection de services plus générale

Dans [Chafle 2006] l'approche proposée vise à adapter des processus définis de services pour la composition et durant l'exécution en sélectionnant des services web ainsi qu'en adaptant le processus abstrait sur un processus concret de services. L'approche dissocie l'aspect fonctionnel et non-fonctionnel dans différents pré-requis utilisés. Ainsi l'approche permet de réagir aux changements à l'exécution en se basant sur des gabarits de processus abstraits et un répertoire de services.

Dans [Mosincat 2011] l'approche proposée travaille sur de la sélection de services pour la maintenance d'exécution de plans d'exécution formalisés en BPEL. Pour ce faire, l'approche utilise des systèmes légers de surveillance à l'exécution afin de détecter les baisses de performances et permettre de chercher d'autres services à exécuter en cas de faute afin de réparer l'exécution.

Dans [Yu 2014], l'approche considérée travaille sur de la correspondance de services en terme de besoins QoS afin de trouver des services correspondant au besoin formulé notamment en terme de paramètres QoS avec plusieurs paramètres considérés.

Dans [Guo 2017] une approche est proposée afin de réaliser de sélection de service à plus grande échelle sur de grands espaces de services candidats. Le processus de sélection de services est ainsi réparti en différentes étapes impliquant un algorithme couvrant ainsi qu'un algorithme d'essaim de particules (*swarm particles*) à l'aide de Spark. L'algorithme couvrant est utilisé pour réduire l'espace de recherche initial et l'algorithme à particules est utilisé pour l'optimisation de la sélection de service par rapport aux paramètres QoS considérés.

Dans [Guidara 2016] et [Guidara 2017], l'approche proposée permet une sélection de services dynamiques pour de la composition. Le but est de réaliser des actions de sélection dynamiquement pour éviter des changements majeurs et des fautes dans une exécution de plan pré-établi. De plus, des heuristiques sont proposées afin d'améliorer le processus de sélection de services dans le cas de contraintes QoS et également temporelles globales et locales. Ce processus permet un nettoyage de l'espace des services candidats en fonction des contraintes globales et une meilleure sélection de services pour obtenir une solution quasi-optimale.

Dans [Chattopadhyay 2018], l'approche proposée travaille à réaliser une sélection automatique de services web. Tout d'abord un algorithme de recherche de so-

lution optimale est proposé pour construire une solution satisfaisant les contraintes QoS du problème d'optimisation multi-objectifs posé. Deux heuristiques sont également proposées en se basant sur une stratégie de *beam search* [Tillmann 2003] et une stratégie de tri génétique non déterministe.

Dans [Alsaryrah 2018], l'approche proposée permet de réaliser une recherche de chemin optimisé parmi un ensemble de services candidats pour chaque tâche à effectuer. L'intérêt de cette approche est qu'elle permet de prendre en compte l'énergie consommée par les services qui peuvent être liés à des objets connectés. L'approche permet également de chercher à optimiser plusieurs paramètres en normalisant ces derniers et par l'utilisation d'un algorithme de recherche de k-plus court chemin.

1.2.1.3 Approches appliquées à des services REST

Dans [Gronvall 2011], l'approche proposée se base sur un framework léger (SECREST) de composition de services REST via l'utilisation de guidage utilisateur dans un scénario de réseau de soin. La composition se base sur des gabarits (templates) de services qui sont instanciés en services en fonction du besoin.

Dans [Guinard 2010], l'approche proposée vise à intégrer des services REST fournis par des objets connectés dans des architectures de services existantes en intégrant ces services dans les répertoires de services existants et en permettant de faire de la découverte de services.

1.2.1.4 Approches basées sur le Web sémantique

Dans [Wu 2007], l'approche proposée vise à réaliser être capable de déterminer si des services correspondent, notamment en terme d'entrées / sorties en se basant sur une modélisation sémantique des web services (ontologie DAML-S, qui sera par la suite renommée en OWL-S [13]).

Dans [Kim 2014], la plateforme proposée permet de réaliser de l'orchestration de services en se basant sur une approche centrée utilisateur ainsi qu'un modèle d'ontologie afin de pouvoir orchestrer des web services et des objets connectés. Cependant, la gestion de paramètres QoS n'est pas intégrée.

Dans [Fähndrich 2016], l'approche proposée permet de trouver une correspondance dans un processus spécifié (BPMN) en trouvant les services adaptés à l'aide d'un moteur sémantique (basé sur OWL-S). Un système a été conçu pour réaliser différentes étapes de la spécification de processus jusqu'à exécution des services. Les paramètres QoS ne sont cependant pas considérés.

1.2.1.5 Autres approches connexes

Dans [Rupasingha 2016], l'approche proposée permet de réaliser du *service clustering* afin de classifier de manière automatique des services. Cette méthode se base sur de l'analyse de texte descriptif des services en utilisant une ontologie. Cette

13. `https://www.w3.org/Submission/OWL-S/`

méthode est également comparée à d'autres méthodes afin de mettre en avant les performances de l'approche proposée. Cette méthode permet de classifier de manière efficace les services traités afin de faciliter la composition de services en étant conscient du domaine des services et de pouvoir effectuer de la sélection de services par la suite.

Dans [Yachir 2015], une approche est proposée dans un environnement incluant des services ambiants (objets connectés notamment). Cette approche vise à sélectionner des services pour répondre à un évènement donné en fonction de tâches définies par l'utilisateur et la classification des services disponibles en classes de services qui peuvent répondre à un besoin. Ainsi, la sélection de services peut être automatisée et le temps de réponse optimisé en ayant connaissance du temps de réponse moyen des services considérés.

1.2.2 Conclusions

Après l'étude des différentes contributions présentées précédemment, différents éléments sont à souligner.

De nombreuses méthodologies et approches existantes ont été proposées et améliorées au niveau de la composition de services.

Cependant, un élément généralement récurrent dans la littérature au niveau de la composition de services est la pré-définition du processus d'exécution. Une approche intéressante serait de pouvoir prendre en compte des plans qui seraient générés à la volée en fonction du besoin et du contexte actuel dans lequel se trouve le système. L'idée serait de pouvoir intégrer ensemble différentes approches afin d'être capable à la fois de gérer des processus générés à la volée ainsi que de réaliser des étapes de sélection de services et d'optimisation de paramètres QoS comme le réalisent certaines approches étudiées. Peu d'approches de composition de services se concentrent sur cet aspect, car beaucoup d'approches utilisent un processus prédéfini ou ne spécifient pas le processus utilisé dans la théorie d'optimisation de leur approche. L'utilisation de systèmes autonomes capables de gérer un système orienté services est une direction qui répondrait à certaines problématiques de l'IoT afin d'intégrer cette gestion de processus dynamiquement générés.

De plus, un élément particulièrement intéressant dans notre contexte est également de pouvoir mutualiser des services notamment pour optimiser l'exécution et dans certains scénarios comme la diffusion de contenu, il est intéressant de pouvoir mutualiser l'appel à des services (producteurs ou consommateurs de contenus par exemple). En effet, cela permet de récupérer un contenu en un appel de service par exemple et de le diffuser via différents services consommateurs en évitant plusieurs appels du premier service. Cet aspect mutualisation de service est très peu pris en compte à notre connaissance dans la littérature et s'intègre difficilement dans les modèles proposés. Une approche s'y apparentant est le *service pooling* ou le partage de services (*service sharing*) qui peut se retrouver dans des approches plus orientées gestion logistique par exemple. Ce genre d'approche se retrouve dans des contextes comme de la e-santé ou la logistique (usines, production, etc.). Par exemple, dans

[Helal 2011] une approche est proposée afin de pouvoir partager différents services dans un contexte de télé-santé, et dans [Qiu 2015] une approche est proposée pour partager différents services logistiques dans un contexte IoT et de parc industriel. Dans notre contexte, il serait intéressant de pouvoir intégrer ce type de démarche afin de mutualiser des services, par exemple des services de diffusion de contenu.

1.3 Les systèmes autonomiques

Afin de répondre aux problèmes liés à la dynamicité et à la volonté de pouvoir gérer des processus générés dynamiquement, l'étude se porte sur des système autonomiques capable d'auto-gestion et de gérer un système en autonomie.

1.3.1 L'informatique autonomique *(autonomic computing)*

L'informatique autonomique (*autonomic computing* dans la littérature) est un paradigme introduit en 2001 par IBM [Kephart 2003] et mis à jour dans [IBM 2006]. L'informatique autonomique est un concept inspiré du fonctionnement d'organismes biologiques qui visent à développer un système capable de gérer un système supervisé mais aussi de s'auto-gérer. Les systèmes autonomiques vont être composés d'un ou plusieurs gestionnaires qui seront en charge de gérer le système et ses évolutions et de s'auto-suffire. On retrouve différentes propriétés importantes d'auto-gestion comme définies dans [Kephart 2003] :

[label=•]**auto-configuration** *(self-configuration)* : représente la capacité du système à configurer différents éléments en se basant sur des politiques haut-niveau et s'adapter aux changements causés par ces configurations **auto-optimisation** *(self-optimization)* : les composants et le système cherche continuellement à améliorer sa propre efficacité et performance **auto-soin** *(self-healing)* : en cas de problèmes (matériel ou logiciel) le système détecte ces derniers, les diagnostique et les répare en autonomie **auto-protection** *(self-protection)* : le système se protège contre des attaques malicieuses ou des erreurs cumulatives par exemple, le but étant d'éviter un arrêt total du système

La structuration d'un système autonomique comme proposé dans [Kephart 2003] se définit sur différents composants qui vont faciliter son fonctionnement. Ces composants constituent une boucle MAPE-K comme détaillée en Figure 1.1. La fonction des différents composants sont détaillées rapidement ci-dessous :

Senseur *(sensor)* : ces composants permettent de récupérer des informations sur le système supervisé

Effecteur *(effector)* : ces composants permettent d'agir sur le système supervisé

Moniteur *(monitor)* : le moniteur est en charge de savoir ce qui se passe sur le système supervisé. Il va pouvoir collecter des évènements qui surviennent sur

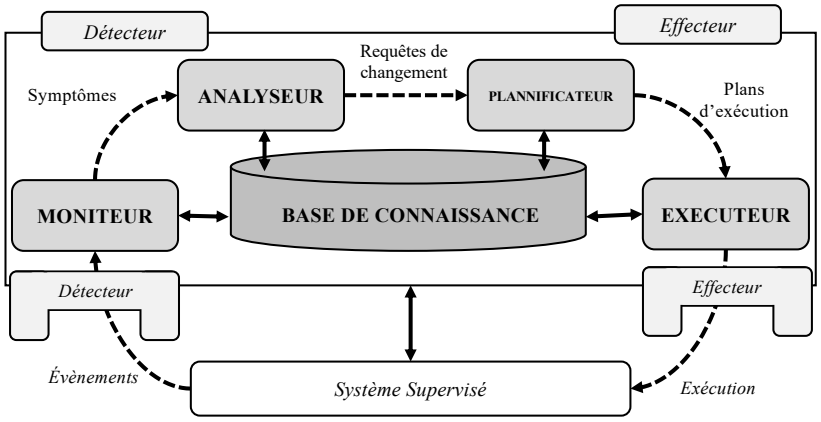


FIGURE 1.1 – Structure d'un système autonomique comme défini par IBM

le système supervisé et générer des *symptômes* en fonction de ces derniers à l'aide de senseurs

Analyseur *(analyseur)* : l'analyseur est en charge de la prise de décision à savoir quelles sont les actions à prendre. L'analyseur va ainsi avoir des capacités d'analyses des symptômes en vue de générer des *requêtes de changement (Request For Change, RFC)* si nécessaire

Planificateur *(planner)* : est en charge de comment effectuer les changements demandés par l'analyseur. Ainsi, le planificateur va générer à partir des RFC des plans d'exécutions à exécuter sur le système supervisé

Exécuteur *(executor)* : est en charge de comment l'exécution a été réalisée. Ainsi ce composant va exécuter les plans d'exécutions fournis par le planificateur sur le système supervisé.

Base de connaissance *(Knowledge Base, KB)* : le composant central du système autonomique. Elle regroupe les informations et modèles nécessaires au traitement de l'information et à l'analyse de symptômes, etc.

Différents niveaux sont définis pour classifier les systèmes de gestion allant de systèmes basiques jusqu'à des systèmes autonomiques [Nami 2007] :

Niveau Basique : À ce niveau les différents éléments du systèmes sont gérés à la main par un professionnel dédié qui se charge de la configuration, l'optimisation, les réparations et la protection

Niveau géré : À ce niveau le système est capable de récupérer de l'information sur le système supervisé et permet à l'administrateur de réaliser de meilleures analyses et d'intégrer une certaine automatisation d'analyse des données

Niveau prédictif : À ce niveau, les différents composants du système de gestion se surveillent eux-mêmes, analysent les changements et offrent des conseils à l'administrateur pour la prise de décision

Niveau adaptatif : À ce niveau, les composants peuvent de manière individuelle et groupée se surveiller, analyser les opérations et offrir des conseils avec une intervention humaine minimale

Niveau autonomique : À ce niveau, les différentes opérations du système sont gérées par des politiques haut-niveau (business) établies par administrateur. En fait, ces politiques vont guider le système de gestion de manière générale là où dans les niveaux précédents il y avait une certaine interaction entre le système de gestion et l'administrateur

1.3.2 Quelles implications dans notre contexte IoT ?

Dans le contexte de systèmes IoT, ce type de gestionnaire autonomique présente un intérêt tout particulier notamment pour la gestion de systèmes distribués ou de systèmes d'objets connectés hétérogènes. L'utilisation de systèmes autonomiques permet de répondre à de nombreuses problématiques liées à l'IoT que ce soit à la fois pour gérer les systèmes eux mêmes avec des composants autonomiques ou des gestionnaires distribués en charge de différents composants IoT.

Ces systèmes peuvent être utilisés pour gérer de manière autonome des systèmes (IoT) via des politiques haut niveau mais il est aussi possible d'utiliser des systèmes autonomiques pour gérer d'autres systèmes autonomiques.

Dans notre contexte, l'utilisation d'une architecture de système autonomique est un axe de solution car cela apporte des éléments de réponse aux problématiques notamment de dynamicité et permet de réagir aux différents évènements qui peuvent survenir sur le système supervisé. De plus, l'architecture autonomique telle que définie dans [Kephart 2003] et [IBM 2006] apporte une modularité du système en découpant les différents traitements et facilitant l'intégration de politiques haut-niveau de gestion.

Dans ce contexte différentes contributions ont été réalisées dont voici quelques exemples.

1.3.2.1 FRAMESELF, un framework autonomique pour le M2M

Dans [Ben Alaya 2014], un framework de gestionnaire autonomique est proposé : FRAMESELF. Ce framework vise à permettre de déployer des systèmes M2M autonomiques. Sa structuration se base sur la boucle MAPE-K avec différents sous-composants ainsi qu'un formalisme de définition des évènement, symptômes, RFC et actions. FRAMESELF intègre une gestion logique de politiques en se basant sur des règles DRL et un moteur d'inférence DROOLS [Bali 2009]. Ce fonctionnement va permettre d'exprimer des politiques haut-niveau de manière logique de telle sorte que s'il y a une certaine combinaison d'évènements qui surviennent dans une fenêtre de temps donnée, un symptôme particulier sera généré, etc. Un ensemble d'exemples de définitions d'évènements, de requêtes de changement et d'action a été instancié dans un contexte d'appartement connecté avec des objets connectés (capteurs et actionneurs) déployés.

Typiquement, ce framework peut être utilisé comme support de déploiement pour une solution à notre problème.

1.3.2.2 AODA : une architecture pour des systèmes orientés services basée sur une ontologie

Dans [Gharbi 2012], une architecture pour des systèmes orientés services et guidés par des évènements est proposée. L'approche se base sur une architecture autonomique et propose un modèle basé ontologie afin de gérer d'un point de vue sémantique les différents évènements qui peuvent survenir sur le système supervisé. L'architecture proposée comporte un gestionnaire autonomique sémantique et un bus sémantique qui assure les interactions entre le gestionnaire et les ressources supervisées. Le modèle sémantique incorpore également un ensemble de règles Semantic Web Rule Language (SWRL) [Horrocks 2004] qui permettent de déterminer automatiquement un ensemble d'éléments comme par exemple les actions à réaliser en fonction des symptômes présents. Les éléments de décision et d'analyse sont ainsi décorrélés des objets physiques auxquels ils vont s'appliquer. Un exemple d'instance d'ontologie incluant des objets gérés par le gestionnaire autonome est également proposé dans un contexte domotique et de compteur connecté. Les plans d'exécution se basent sur des actions à effectuer qui vont être exécutées sur un objet en particulier comme par exemple allumer un ventilateur.

Cette approche apporte des éléments de réponse dans notre problème mais demeure limitée au niveau de la génération de plans d'exécutions. L'idée serait d'aller plus loin pour générer des plans composites dynamiquement et qui ne possèdent pas nécessairement un ou des services à exécuter pour effectuer une action. L'idée est de pouvoir utiliser différents services en combinant à la fois les objets mais pas que.

1.3.2.3 FASEM, un framework de gestion de services dans l'IoT

Dans [Yachir 2015], une approche de système autonome est proposée afin de permettre de gérer des services ambiants fournis par des objets connectés. Cette approche se focalise sur découvrir des services ambiants fournis par des objets connectés, de les classifier, puis de pouvoir les utiliser en fonction de tâches spécifiées par l'utilisateur pour aller jusqu'à une sélection de services pour répondre au besoin. Le système vise à surveiller les évènements qui occurrent sur le système et agir en conséquence. Un processus de sélection locale de services ambiants est également abordé afin de sélectionner les meilleurs services disponibles pour répondre à la tâche. Les processus d'exécution sont cependant basés sur des tâches demandées par l'utilisateur pour lesquelles un service (ou un service composite prédéfini) est sélectionné en fonction de paramètres QoS estimés ou connus.

Par rapport à notre problème, les considérations pour la découverte comporte un intérêt certain ainsi que le processus de sélection de services en se basant sur des paramètres QoS. A terme ce type d'approche pourrait compléter notre solution pour élargir ses capacités.

1.3.2.4 Une architecture de système de gestion de service autonomique

Dans [Dai 2017], l'approche proposée définit une architecture de système de gestion de service autonome dans un contexte de systèmes cyber physiques d'industrie connectée. Le cœur du système proposée se base sur des principes du SOA ainsi qu'une ontologie permettant de constituer la KB du système MAPE-K. Cette base de connaissance possède également un ensemble de règles qui permettent via des mécanismes de raisonnement de déduire différents éléments comme les actions à réaliser sur le système supervisé. Des politiques d'auto-optimisation sont également intégrés dans le système afin d'améliorer l'efficacité du système et la gestion des ressources.

Bien que le cas d'utilisation soit différent, il est intéressant de s'inspirer de cette approche pour la génération de règles sémantiques afin de traiter les symptômes et de prendre des décisions.

1.4 Web sémantique, ontologies de l'IoT et des services

Différentes contributions qui ont été étudiées font appel à des technologies du web sémantique et à des ontologies. Dans cette section, une brève définition de ce type de modèles est détaillée vu que ces modèles seront abordés dans cette thèse. De plus, un panel de différentes ontologies reconnues dans notre contexte IoT et intégrant des services est détaillé pour contextualiser les travaux.

1.4.1 Web Sémantique, Ontologies, règles SWRL

Le **Web Sémantique** a été initialement proposé dans [Berners-Lee 2001] et vise à permettre de développer un Web qui ne soit pas compréhensible que pour des utilisateurs humains, mais aussi pour les machines. De fait, l'idée est d'introduire des mécanismes permettant de rendre les contenus disponibles compréhensibles pour les machines (*machine understandable*) en conservant un aspect compréhensible pour les humains. Dans cette optique, il est nécessaire de permettre aux systèmes de comprendre les données échangées notamment via l'utilisation de méta-données compréhensibles par les machines. Ces méta-données vont se baser sur des vocabulaires partagés qui seront définis par des ontologies.

Les **ontologies** permettent de formaliser des vocabulaires afin de représenter la connaissance. Ce formalisme a été introduit initialement dans [Gruber 1991]. Les ontologies permettent de définir des concepts (classes) ainsi que des attributs *(data property)* et des relations *(object property)* entre ces concepts. La conjonction de ces vocabulaires avec un ensemble de données annotées va représenter de l'information qui sera stockée sous forme de base de connaissance (sémantique) *(semantic knowledge base, SKB)*. Il est également possible d'aligner des ontologies, *i.e.* définir des concepts comme étant similaires et ainsi permette une plus grande expressivité de modèles et/ou aligner des définitions spécifiques sur des vocabulaires standards ou communs. On peut aussi importer des concepts définis dans d'autres vocabulaires

et les étendre. En se basant sur de tels systèmes de la connaissance peut ainsi être inférée à l'aide de **raisonneurs** (sémantiques) et de moteurs d'inférence qui vont se baser sur la connaissance disponible dans la SKB et des règles qui peuvent être incorporées dans le vocabulaire.

SWRL [Hutchison 2005] est un formalisme permettant de définir des règles (logiques) pour compléter la représentation de connaissance dans les ontologies.

Les différentes technologies du Web sémantique citées précédemment (hormis SWRL qui n'est pas un standard mais qui est utilisé pour les avantages et la simplicité de définition apportée) sont standardisées par le W3C (World Wide Web Consortium) [14]. Le consortium publie des recommandations définissant les différents formalismes. Le W3C a défini le formalisme Ressource Description Framework (RDF) qui permet de représenter des graphes sous forme de triplets avec un *sujet*, une *propriété* et un *objet*. Ce formalisme est cependant limité en terme de sémantique et ne suffit pas à représenter des ontologies complexes. Pour ce faire, le W3C a défini RDFS et Web Ontology Language (OWL). RDFS permet de décrire également les relations entre les classes et de représenter des ontologies dites "légères" *(lightweight)*. D'un autre côté, OWL permet d'utiliser des ensembles d'axiomes logiques dans la description des ontologies et d'enrichir l'expressivité des modèles.

1.4.2 Des ontologies reconnues

De nombreuses ontologies ont été développées dans la littérature dont certaines dans le contexte de système IoT et d'autres en rapport avec l'annotation de services Web notamment. Dans les ontologies existantes différentes catégories sont distinguées comme des ontologies de fondation, qui définissent des concept de très haut-niveau qui seront instanciés dans d'autres ontologies, les ontologies de domaine qui définissent des concepts primordiaux dans un domaine particulier et des ontologies spécifiques qui définissent des concepts propres à un contexte particulier ou un scénario spécifique.

Dans notre contexte, différentes ontologies existent et en voici quelques unes des plus utilisées.

1.4.2.1 SOSA / SSN

Semantic Sensor Network (SSN) [15] est une ontologie dédiée à la modélisation de capteurs (plutôt que des objets en général) et d'observations à l'origine. Depuis, SSN a été retravaillée en 2017 et séparée dans un module SOSA (Sensor, Observation, Sample and Actuator) [16] et une extension de cette nouvelle ontologie a été réalisée pour faire la connexion avec l'ancienne terminologie de SSN et les concepts qui n'ont pas été inclus dans SOSA [17].

14. https://www.w3.org
15. http://purl.oclc.org/NET/ssnx/ssn
16. http://www.w3.org/ns/sosa/
17. http://www.w3.org/ns/ssn/

1.4.2.2 oneM2M Base Ontology

Dans les dernières parutions du standard, oneM2M a défini une ontologie afin de pouvoir intégrer des méta-données sémantiques pour décrire les différentes ressources enregistrées dans la plateforme. Cette ontologie, nommée oneM2M Base Ontology [18] est documentée dans une spécification technique dédiée [19]. Cette ontologie est une ontologie de cœur de domaine *(core domain)* : un ensemble minimal de concepts est défini pour permettre un alignement par la suite avec d'autres concepts standards ou spécifiques aux différentes implémentations et technologies. Les concepts définis sont des concepts assez haut-niveau comme Objet ou Service et sont liés aux ressources de l'architecture oneM2M.

1.4.2.3 SAREF

Smart Appliance REFerence, SAREF [20] est à la base une ontologie dédiée à la gestion d'énergie et des services dans les maisons intelligentes. Depuis, l'ontologie a été étendue [Daniele 2016] pour s'appliquer à d'autres domaines de l'IoT comme les bâtiments connectés, etc.

1.4.2.4 IoT-Lite

IoT-Lite [21] est une ontologie de cœur de domaine légère *(lightweight)* définie dans [Bermudez-Edo 2017]. Des concepts haut-niveau sont définis comme objet actionnable ou des services. Elle a été adoptée dans différents projets comme cœur de modèle comme FiWARE par exemple grâce à sa simplicité. [22]

1.4.2.5 MSM

Minimal Service Model, MSM [23] est une ontologie basique qui définit de manière haut-niveau une notion de service couplé à une notion d'opération et de messages entrants ou sortants de ces opérations. Elle est utile pour réaliser notamment de l'alignement lors de la définition de services spécifiques.

1.4.2.6 WSMO & WSMO-Lite

Web Service Modeling Ontology, WSMO, est une ontologie proposée dans [Roman 2011] qui possède une approche *top-down*. Des concepts haut-niveau sont définis afin de décrire des services web en intégrant la possibilité de définir des paramètres non fonctionnels mais aussi des capacités des services. WSMO-Lite est une approche beaucoup plus simpliste qui se base sur les concepts de MSM pour définir un modèle haut-niveau de service. WSMO-Lite [Vitvar 2008] [Roman 2015] définit

18. `https://git.onem2m.org/MAS/BaseOntology/raw/master/base_ontology.owl`
19. `http://www.onem2m.org/technical/published-drafts`
20. `http://ontology.tno.nl/saref/`
21. `http://iot.ee.surrey.ac.uk/fiware/ontologies/iot-lite`
22. `https://www.fiware.org/`
23. `http://iserve.kmi.open.ac.uk/ns/msm`

également des concepts d'effet pour les services et de propriétés fonctionnelles et non-fonctionnelles.

1.4.2.7 hRESTs

hRESTs (HTML for RESTful Services)[Kopecký 2008] [Roman 2015] est un modèle de micro-annotations pour des services REST qui peut être intégré directement dans du HTML descriptif des API. hRESTs permet de définir la notion de service et d'opération en s'appliquant au cas de services REST et en définissant les concepts d'opérations HTML correspondante au service.

Des extensions ont été proposées comme SA-REST et Micro-WSMO [Kopecký 2008] pour fournir plus de support sémantique.

1.4.2.8 IoT-O

IoT-Ontology, IoT-O [24] est une ontologie cœur de domaine modulaire publiée dans [Seydoux 2016]. En effet, IoT-O est une ontologie fédératrice et regroupe de nombreux concepts définis dans des ontologies reconnues. La conception d'IoT-O respecte la méthodologie NeON [del Carmen Suarez de Figueroa Baonza 2010].

Les concepts de *capteurs* se basent sur les définitions de SSN. Les concepts d'*actionneurs* se basent sur des concepts de SAN et SSN. La gestion des services se base notamment sur MSM et WSMO et intègre également des éléments de hRESTs. L'intégration de l'énergie est réalisée grâce aux concepts définis dans PowerOnt, une ontologie référencée par SAREF.

La ré-utilisabilité de tous les concepts forts de différentes ontologies standards ou déjà reconnues dans le domaine fait une force de l'ontologie et il est donc intéressant de pouvoir se baser sur une telle ontologie de domaine fédératrice. De plus, de nombreux alignements sont déjà existants avec les ontologies de référence comme MSM ou SSN et permettent une meilleure intégration d'autres modèles spécifiques suivant le besoin ou le cas d'utilisation.

1.4.3 Conclusions

Comme soulevé précédemment, l'intérêt d'intégrer des modèles à bases de technologies du web sémantique dans un système permet de débloquer des compétences d'analyses de ce système en se basant sur des vocabulaires existants et en utilisant la puissance des raisonneurs sémantiques. De fait, ce genre de modèles permet de stocker des informations sous forme de base de connaissance sémantique ainsi que de réaliser de l'inférence de connaissance si besoin en fonction des différentes informations connues du système.

Dans le contexte de ces travaux, un modèle particulièrement intéressant se dégage : IoT-O. Cette ontologie regroupe des concepts à la fois d'objets connectés (Actionneur, Capteur) et différentes méta-données sur les objets physiques, mais

24. `https://www.irit.fr/recherches/MELODI/ontologies/IoT-O`

fait également le lien avec des services liés à ces objets physiques. De plus, il est possible décrire directement des services REST comme ceux fournis par des plateformes horizontales par exemple. Dans notre cas d'utilisation du projet S2C2, l'intégration de plateformes oneM2M est possible via l'utilisation d'IoT-O (qui apporte des améliorations supplémentaires par rapport à la base ontology de oneM2M).

1.5 Grammaires de graphes et réécriture de graphes

Dans une approche orientée services, il est intéressant de pouvoir travailler sur les plans d'exécution de services. Ces derniers peuvent être représentés aisément sous la forme de graphes de propriétés afin de contenir les informations nécessaires à l'exécution.

1.5.1 Définitions : grammaires de graphes et réécriture de graphes

1.5.1.1 Graphes attribués ou partiellement attribués

Par la suite, les graphes considérés sont des graphes partiellement attribués ou attribués [Eichler 2015]. Ces graphes sont définis de la manière suivante :

Définition 1 *Le graphe dirigé et (partiellement) attribué G est défini comme un système (N, A, Att) où :*

- *N correspond à l'ensemble des nœuds de G*
- *$A \subseteq V * V$ correspond à l'ensemble des arcs de G, de sorte que (v_1, v_2) définit un arc de v_1 vers v_2*

Soit EL^{att} l'ensemble des éléments (arcs ou nœuds, EL^{att}) attribués de G. Pour tout $e \in EL^{att}$, avec $M \in \mathbb{N}$ le nombre d'attributs liés :

- *$Att_i^e = (A_i^e, D_i^e)$ avec $i \in [1, M]$ le i-ème attribut de l'élément e. A_i^e est la valeur de cet attribut et D_i^e le domaine de définition tel que $A_i^e \in D_i^e$.*
- *Att^e est l'ensemble des attributs de l'élément attribué e. La cardinalité de cet ensemble est notée $|Att^e|$ et représente le nombre d'attributs de e, soit M.*

1.5.1.2 Grammaires de graphes et réécriture de graphes

La réécriture de graphes permet de définir des règles de réécriture de graphes qui définissent à la fois une transformation de graphe et les circonstances dans lesquelles elle doit s'appliquer.

Il s'agit de manière générale de remplacer un certain motif m dans un graphe cible par un autre motif d. Pour chaque occurrence de m (image isomorphe) trouvée dans un graphe hôte, ce motif va être supprimé du graphe et une image isomorphe de d est ajoutée dans le graphe hôte.

Différentes façons de spécifier des règles de réécriture et de réaliser de la transformation de graphe ont été proposées dans la littérature [Rozenberg 1997], notamment les approches **Double PushOut (DPO)** et **Single PushOut (SPO)** [Ehrig 1997]. Les motifs considérés m et d sont généralement notés L et R pour

partie gauche *(left)* et partie droite *(right)* de la règle. Dans l'approche SPO, seuls les motifs L et R sont définis avec un morphisme partiel entre ces derniers. La règle est alors notée $L \to R$. Dans le cadre de l'approche DPO, un troisième graphe interface K est explicitement spécifié. La règle s'écrit alors $L \leftarrow K \to R$. Les parties L et R vont être identiques entre les deux formalismes. À l'application de règles de réécriture, parfois la transformation peut aboutir à l'apparition d'arcs dits *suspendus* car une ou deux extrémités a/ont été supprimée·s. Dans le cadre de l'approche SPO ces arcs sont simplement supprimés. Dans le cadre de DPO, la règle ne peut pas être appliquée si de tels arcs doivent apparaître. Cela représente une condition de suspension.

Dans le cadre de l'approche SPO, une grammaire de graphe est définie de la manière qui suit [Eichler 2015] :

Définition 2 ***(Grammaire de graphes et règle de réécriture)***
Une règle de réécriture de graphes $r = L_r \xrightarrow{m_r} R_r$ est caractérisée par un couple de graphes ($L = (N, E, \tau)$, $R = (N', E', \tau')$) et un morphisme $m_r = (f_r, \emptyset)$ d'un sous-graphe $L^I = (N_L^I = Dom(f_r)E_L^I)$ de L dans R.

La partie gauche (L_r) de la règle représente le schéma de graphe qui est cherché dans le graphe à transformer et la partie droite (R_r) représente le schéma transformé. Quand la règle r est appliquée à un graphe G, si une image de L_r existe dans G alors la règle est appliquée et remplace le schéma L_r qui correspond par le schéma R_r.

Parfois une condition de non applicabilité (Not Applicable Condition, NAC) est utilisée pour prévenir l'application de la règle concernée dans le schéma de graphe indiqué dans la NAC est trouvé dans le graphe cible.

1.5.2 Quelle utilisation ?

Cette technologie a été utilisée de manière variée dans la littérature et dans différents cas d'application. Par exemple, l'application de grammaires de graphes peut permettre de réaliser de la validation formelle de structure de systèmes que ce soit à la conception ou pour la gestion de systèmes existants. Dans [Drira 2017], des grammaires de graphe sont utilisées principalement à cette fin pour aider lors de la conception et de la gestion de systèmes logiciels et vérifier et valider leur structure. Dans [Eichler 2014], Eichler *et al.* étudient comment des méthodologies basées sur des graphes typés et des règles de réécriture permettent la reconfiguration de systèmes corrects par construction. Cette approche est détaillée plus en détails dans un contexte de modélisation formelle de systèmes dynamiques autonomes en utilisant des graphes partiellement typés et des ensembles de règles de réécriture [Eichler 2015].

Les différentes approches de la littératures s'appliquent sur des problématiques différentes ou connexes. Cependant, il est intéressant de considérer ces approches pour appliquer des comportements proches dans notre contexte.

Dans [Higashino 2016], les grammaires de graphes sont utilisées pour la gestion d'évènements complexes et de flux de données. Les règles de réécriture vont

permettre par exemple d'optimiser les processus de traitement en mutualisant des opérations (agrégation) ou en optimisant les traitement via la transformation des processus de traitements sous forme de graphe. Dans un contexte plus proche du notre, [Zhao 2012] définit une structure basée sur une grammaire de graphe et des règles de réécriture pour réaliser de la validation de contraintes structurelles sur des systèmes de composition de services. Dans [Xue 2018], les grammaires de graphes sont utilisées pour découper des processus de services dans un contexte de composition de services et optimiser lesdits processus en détectant des motifs dans les processus exprimés sous forme de graphes.

Concrètement, la gestion de systèmes dynamiques autonomes mais aussi de plans de composition sont des applications qui font sens dans notre contexte. De plus, la modélisation sous forme de graphe s'applique particulièrement bien à des plans d'exécution, notamment de services. Des règles de réécriture faciliteront la génération de ces plans d'exécution de manière dynamique en raffinant des plans plus haut-niveau ou en permettant de créer des liens entre services à l'aide d'autres informations connues sur le système à l'aide de la base de connaissance.

1.6 Les projets STM & S2C2

Cette thèse s'inscrit dans le cadre d'un projet de recherche régional **STM** : **S**mart **T**hings **M**anagement, et d'un projet de recherche national **S2C2** : **S**mart **S**ervices for **C**onnected vehi**C**les. Dans cette section, certains éléments de contextualisation et en rapport avec les travaux de cette thèse sont abordés ainsi que certains scénarios qui ont été instanciés par la suite.

1.6.1 Présentation générale du projet STM

Le projet Smart Things Management (STM) est une collaboration entre le LAAS-CNRS et SRC Solution [25], une société française. Ce projet est financé par l'Etat français (région Midi-Pyrénées) et vise à créer de nouvelles solutions pour gérer les objets de l'IoT. Le produit commercial phare de la solution SRC : Pilot Telecom [26] est un outil de gestion des lignes téléphoniques et de la consommation en tant qu'entreprise. Le but du projet est de travailler sur le point de vue de la gestion des périphériques pour créer une solution permettant de surveiller les périphériques déployés et d'éviter des coûts supplémentaires par exemple lors de l'utilisation de forfaits cellulaires.

À cette fin, des normes et des solutions existantes ont été étudiées pour contribuer à leur amélioration. Par exemple, des normes telles que oneM2M (standard de haut niveau) ou LWM2M (protocole de gestion de périphérique) se sont révélées particulièrement appropriées. De plus, l'utilisation d'une plateforme oneM2M permet d'intégrer également une connexion vers LWM2M à travers un IPE.

25. `http://www.src-solution.com`
26. `http://www.src-solution.com/pilot-telecom/`

1.6.2 Présentation générale du projet S2C2

1.6.2.1 Vision haut niveau

Le projet S2C2 est formé par un consortium composé de : AToS [27] (Bordeaux), e-Device [28], M3Systems [29] et le LAAS-CNRS. Le projet se positionne dans un contexte de ville intelligente et de déploiement d'objets connectés au sein de métropoles. Dans ce contexte, de nombreuses entités sont en jeu, autant au niveau des institutions que des véhicules mais aussi des utilisateurs dans la ville. En effet, de nombreuses opportunités et nouvelles possibilités se dessinent autour des véhicules à la fois personnels, des transports en communs, des transports de marchandises, des véhicules d'intervention d'urgence, etc.

L'objectif du projet est de proposer une architecture capable de connecter des objets et des services existants avec de nouveaux systèmes mais aussi de nouvelles technologies hétérogènes. Un aspect intéressant est de pouvoir enrichir et proposer de nouveaux services notamment pour tout ce qui touche aux véhicules connectés. De fait, le projet vise également à travailler sur des scénarios innovants dans ce contexte et proposer des solutions permettant de répondre à ces scénarios tant au niveau objets connectés et capteurs qu'au niveau logiciels et architectures.

1.6.2.2 Architecture générale

L'architecture générale adoptée dans le projet est présentée en Figure 1.2. Cette dernière comporte de nombreux composants liés à différents aspects de la ville intelligente. De plus, l'architecture s'articule autour d'une couche d'interopérabilité intermédiaire qui va faciliter le fonctionnement global du système. En effet, cette couche d'interopérabilité syntaxique et potentiellement sémantique, vient enrichir le système IoT existant et permettre le déploiement de nombreux nouveaux services autour de cette couche intermédiaire.

De nombreux composants peuvent ainsi être interconnectés via la couche d'intergiciels, que ce soit les transports en commun connectés, les utilisateurs de la ville intelligente, les véhicules d'urgence, les véhicules particuliers et de transport de marchandises, les commerces, mais aussi les systèmes d'urgence et de gestion de flotte ainsi que les autorités de la ville intelligente.

L'idée est de pouvoir réaliser des connecteurs vers des systèmes déjà existants afin de permettre une utilisation transverse des différentes richesses et possibilités apportées par chaque domaine.

1.6.3 Des scénarios dans la ville intelligente

Dans le projet, différents scénarios ont été définis :

- **Globalement** : le but global est de définir un contexte général d'interopérabilité autour duquel vont pouvoir s'articuler d'autres scénarios

27. `https://atos.net/fr/`
28. `https://www.edevice.com`
29. `http://m3systems.net`

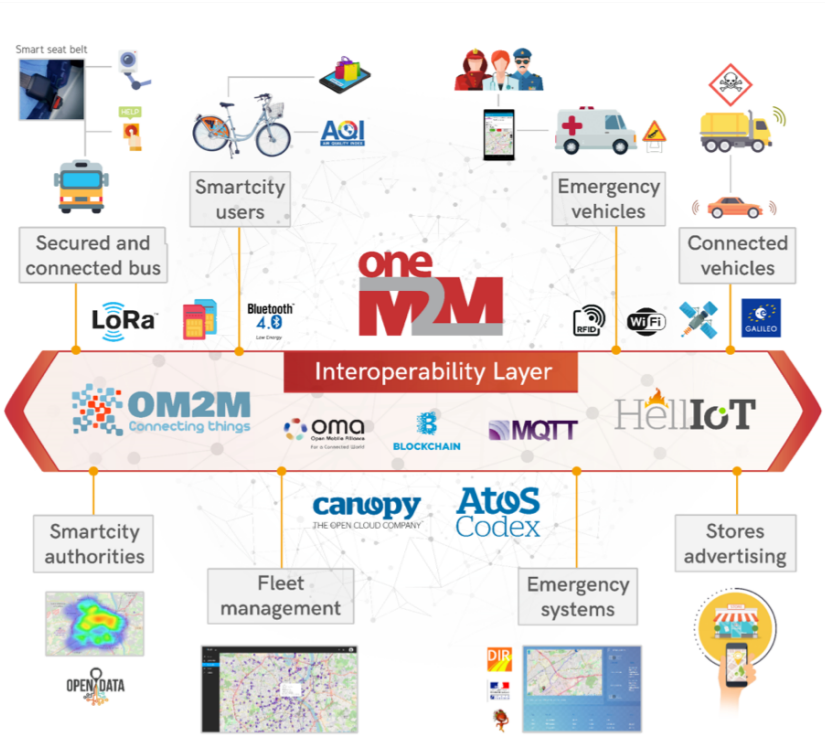


FIGURE 1.2 – Architecture générale du projet S2C2

- Scénario ***Sécurisation de l'intervention des véhicules de secours*** : l'idée du scénario est de déployer des solutions connectées afin de faciliter et sécuriser l'intervention de véhicules de secours en signalant dynamiquement leur position par exemple
- Scénario ***Alerte de la proximité d'un 2 roue*** : l'idée de ce scénario est de réduire les accidents impliquant les deux roues en permettant aux parties prenantes d'être informées de la proximité de véhicules à 2 roues
- Scénario ***Diffusion dynamique d'informations relatives à la localisation dans les transports en commun*** : ce scénario vise à permettre la diffusion de contenus dynamiques en fonction de la position des transports en commun dans la ville intelligente. Ce scénario sera plus détaillé par la suite.
- Scénario ***Plateforme de géolocalisation à la demande*** : l'idée est de permettre la mise en place d'un système de géolocalisation à la demande notamment pour la gestion de flotte (bus).
- Scénario ***Aide à la gestion de flotte via modèles prédictifs et Big Data*** : dans ce scénario, le principe est de mette en place des systèmes de modèles prédictifs et big data afin d'assister la gestion de flotte. En effet, à l'aide de capteurs et de géolocalisation des véhicules, il est possible d'analyser de nombreuses données en les combinant avec des données de trafics, d'évènements, de météo, etc., fournies par les objets connectés déployés dans

la ville.

- Scénario ***Système d'aide au stationnement en agglomération*** : dans ce scénario l'idée est de combiner des sources de données hétérogènes afin d'améliorer et réduire le temps de recherche de stationnement pour atteindre une destination. L'idée est également d'améliorer de manière générale le déplacement péri-urbain et urbain en combinant des modes de déplacements multi-modaux
- Scénario ***Gestion de crise*** : ce scénario vise à assister les différentes entités en cas d'accident et de gestion de crise. Dans ce scénario une plateforme est mise en place pour recevoir différentes informations afin de permettre d'avoir accès aux positions des différents agents d'intervention ainsi que les missions lorsqu'elles apparaissent.

1.6.4 Le scénario d'utilisation principal

Dans cette section, le scénario principalement étudié dans cette thèse est décrit.

1.6.4.1 Contexte

La démocratisation des déplacements en transports en commun permet d'avoir une grande diversité de profils de voyageurs. Ceux-ci passent, de manière quotidienne ou ponctuelle, un certain temps dans les transports ou dans les zones d'embarquements (quais, arrêts de bus...). De nos jours, des informations relatives à la localisation en temps-réel des transports sont diffusées : le temps d'attente avant le prochain bus, le temps d'attente avant le prochain arrêt, parfois messages en cas de perturbation, etc. On appelle cela le Système d'Information Voyageurs Embarqué (SIVE). En général, ces informations sont diffusées sur des écrans et des bornes via une liaison de données de type filaire (Ethernet) ou radio (TETRA, GPRS). Dans le contexte des villes connectées, on peut enrichir ce scénario en amenant l'information aux voyageurs à bord des transports de manière dynamique et interactive : proposer des informations ciblées en fonction du profil du voyageur. Par exemple, si le voyageur est en visite touristique, des informations propres à la ville, fournies par la mairie ou l'office de tourisme pourraient lui être diffusées. En plus d'être un avantage pour le voyageur et enrichir son déplacement, le fait de pouvoir évaluer le profil des voyageurs pourra permettre une amélioration des transports grâce à un retour auprès des compagnies de transport, ainsi que promouvoir les acteurs de la ville.

1.6.4.2 Description fonctionnelle du scénario de diffusion de contenu

Ce scénario propose deux éléments : pouvoir utiliser et rentabiliser un affichage intra-bus afin de fournir plus d'informations aux passagers, et un système plus souple et dynamique accessible permettant (par exemple via une application pour smartphone) de pouvoir obtenir un contenu plus personnalisé pour les utilisateurs

des transports. L'idée est de permettre aux utilisateurs d'exprimer des préférences et des intérêts dynamiquement (via une application mobile par exemple).

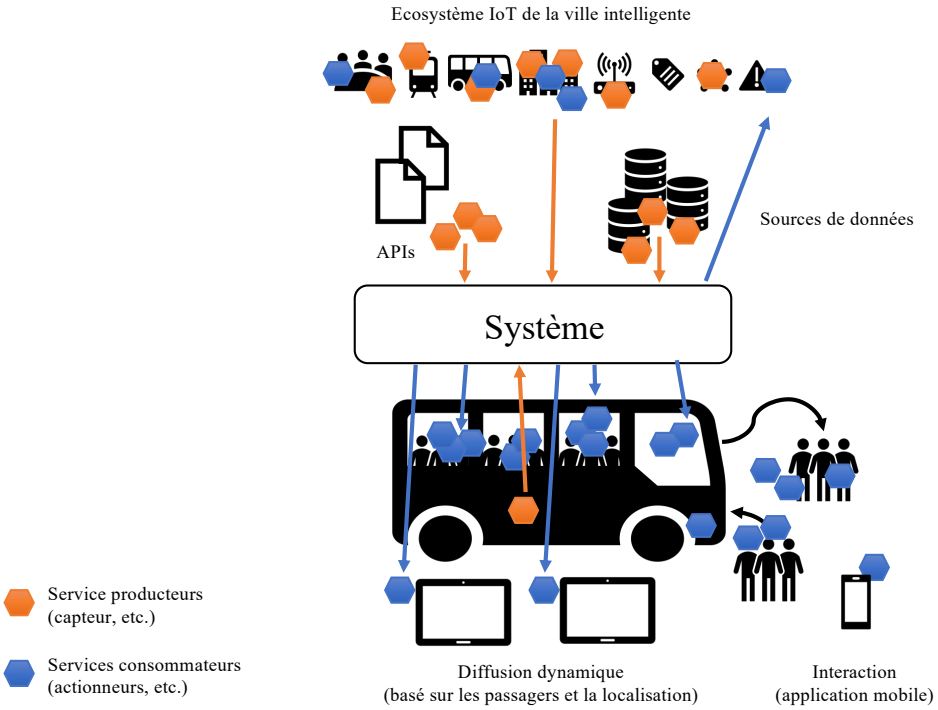


FIGURE 1.3 – Idée du scénario de diffusion de contenu dynamique dans un bus

Le système centralisera des informations relatives à la ville, aux horaires de différentes lignes par exemple, points d'intérêts touristiques, horaires des magasins / grandes surfaces, animations particulières, informations générales, etc. Toutes ces données seront stockées dans le système accessible de manière distante et leur récupération s'effectuera de deux façons différentes. Tout d'abord pour l'affichage interne au bus, des informations pourront être chargées au démarrage du bus (ou au dépôt la veille) dans l'espace de stockage embarqué à bord du bus et être ensuite diffusées sur les écrans (par exemple sur le réseau Tisséo [30] cette opération est réalisable avant la mise en service du bus mais difficilement pendant sa circulation pour ne pas occuper la bande passante). D'un autre côté, une interface serait offerte côté utilisateur pour être accessible depuis une application pour smartphone. Sur cette interface les informations pourront être plus actualisées (temps réel) et personnalisées en fonction du profil utilisateur (couplage avec les réseaux sociaux). Différentes informations comme la localisation du bus pourront être utilisées pour indiquer à l'utilisateur les éléments à proximité qui l'intéresseraient potentiellement, comme le nombre de vélos disponibles à la station la plus proche. Une possibilité intéressante serait également que l'utilisateur puisse se connecter à un réseau dans le bus (WiFi) et avoir accès à ces informations sur son terminal mobile.

30. `www.tisseo.fr`

Le principe du système est présent sur la Figure 1.3. L'idée est que les différents éléments accessibles du système, que ce soient des capteurs connectés ou les terminaux d'affichage dans le bus ou des utilisateurs sont accessibles via des services. Par exemple dans le cas où une plateforme oneM2M est utilisée, les différents éléments pourront être accessibles à travers des services REST. Sur la Figure, on peut distinguer des services qui vont produire du contenu typiquement des capteurs dans la ville (particules, température, etc.) (en orange sur le schéma) et des services qui vont plutôt être consommateurs de contenu (en bleu sur le schéma) comme les terminaux d'affichage du bus ou des utilisateurs. Il également possible qu'il y ait des entités qui soient à la fois consommatrices et productrices de contenu comme le bus qui peut générer des informations sur sa position par exemple mais aussi consommer des informations à afficher.

1.6.5 Contraintes et problématiques liées à ce contexte

Dans ce contexte différents éléments de problématique sont levés.

Tout d'abord il semble difficile d'avoir une approche centralisée dans ce scénario. En effet, il est très difficile de pouvoir centraliser le traitement de ce genre de systèmes vastes (ville intelligente) : dans le cas d'un bâtiment connecté, parfois des milliers d'objets sont en jeu. De fait, sur une ville entière il sera difficile de centraliser la gestion et surtout le traitement.

La gestion de ce type de systèmes IoT en intégrant l'aspect services s'avère complexe ne serait-ce que de par le côté dynamique lié au type de système. La mobilité est un élément important, que ce soit au niveau des utilisateurs ou des objets connectés (comme le bus). De plus il serait intéressant d'être capable de travailler à une gestion de ce type de systèmes en intégrant des paramètres non fonctionnels (QoS) afin de permettre de réaliser des optimisations suivant les scénarios. Dans le cas du scénario de diffusion d'information dynamique une contrainte peut être le temps de diffusion d'une information, notamment dans le cas d'une information urgente par exemple, ou bien le nombre de services utilisés. Des problématiques d'énergie peuvent également se poser par le coût d'exécution de certains ordres suivant les objets utilisés dans ce contexte car certains objets peuvent fonctionner sur batterie et posséder une autonomie limitée en cas de trop forte sollicitation. Enfin, la généricité et l'adaptabilité du ou des modèle·s en jeu doit être forte afin de limiter le coût d'adaptation du système à l'arrivée de nouveaux services, de nouveaux objets, à la déconnexion de certains objets, etc.

La vision orientée service amène également à la possibilité de combiner de manière automatique différents services entre eux (objets connectés avec services logiciels ou accessibles sur internet comme un service de prévision de trafic par exemple ou météo).

Un moyen intéressant est de pouvoir *composer* différents services disponibles pour réaliser certaines actions, *i.e.* être capable de combiner différents services entre eux qui sont disponibles séparément. La combinaison avec des services de gestion d'objets apporte également une richesse supplémentaire et ouvre à de nouvelles

applications. Par exemple il est possible d'utiliser des services afin de récupérer l'état de batterie connu de différents objets afin de prendre une décision en fonction.

Enfin, pouvoir *mutualiser* certains services apporte des améliorations. En effet, dans le cas de diffusion de contenu dynamique la **mutualisation de services** de diffusion de contenu représente une optimisation d'intérêt notamment dans le cas d'informations globales à diffuser à un lot d'utilisateurs. De manière plus générale, la capacité d'avoir recours à des services communs en mutualisant les appels à ces services pour minimiser le nombre d'appels et utiliser les ressources de manière plus optimisée, représente également un avantage.

Conclusion

En considérant différents éléments de contexte, différents axes ont été étudiés pour y répondre : les **standards et protocoles de l'IoT**, les problématiques de **composition de services**, les **systèmes autonomiques** ainsi que des **modèles** capables d'aider à cette gestion de systèmes IoT orientés services. Tout d'abord l'utilisation de plateformes standard (oneM2M notamment) et de protocoles comme LWM2M est particulièrement intéressante. En effet, l'utilisation de ce type de plateformes horizontales est un tremplin pour une plus grande interopérabilité et l'intégration d'applications IoT plus ambitieuses et complexes et engendre une vision plus haut-niveau, orientée services. L'interopérabilité permet également de rassembler des technologies hétérogènes au niveau des objets et de connecter différentes plateformes entre elles (ce qui apporte une valeur ajoutée dans le contexte des projets) mais aussi de donner des éléments pour faciliter l'interopérabilité sémantique.

Afin de répondre aux différentes problématiques qui se posent dans le contexte du scénario de ville intelligente et de diffusion de contenus dynamiques notamment mais pas que, différentes pistes ont été explorées. La richesse de la littérature sur la **composition de services** amène de nombreux moyens d'aider à cela, dans un contexte de vision orientée services. Les **architectures autonomiques** peuvent répondre aux problématiques de dynamicité et d'autonomie demandée en permettant de configurer des systèmes avec des politiques comportementales haut-niveau et que le système fasse preuve de capacités d'auto-gestion.

Enfin, les technologies du **Web sémantique** et notamment les **ontologies** sont plus démocratisées de nos jours et rendent possible la définition de vocabulaires partageables et compréhensibles pour les machines afin d'amener des informations de contextualisation et d'analyse de la connaissance sur un système donné via des mécanismes de raisonnement et d'inférence de connaissance automatique Différentes ontologies reconnues ont été présentées et **IoT-O** se dégage comme un modèle de référence particulièrement intéressant dans notre contexte de travail. Les **grammaires de graphes** sont des solutions permettant à la fois de modéliser des éléments sous formes de graphes typés mais aussi de transformer automatiquement des graphes en plans d'exécution à la volée et ne pas comporter des exécutions de plans figés préétablis en fonction de certains symptômes particuliers.

Dans le chapitre suivant, l'architecture centrale des contributions de cette thèse est abordée ainsi que l'approche multi-modèles mise en œuvre de manière théorique pour répondre aux problématiques d'analyse et de génération de plans d'exécution dynamique. Enfin, des instances des modèles dans le contexte de notre cas d'utilisation sont proposées et détaillées.

Chapitre 2

Vers un système autonome de gestion des services d'une architecture IoT ouverte

Sommaire

2.1 Architecture haut-niveau du système proposé 38
2.1.1 Les composants du système 38
2.1.2 Évènements et comportement associé 40
2.2 Le modèle basé sur une ontologie 40
2.2.1 Instance d'ontologie 41
2.2.2 Règles SWRL et Analyse 45
2.3 Les modèles orientés graphe 47
2.3.1 Requête de changement à effectuer (RFC) 48
2.3.2 Génération de plan d'exécution 49

Dans le contexte de systèmes IoT et plus particulièrement des villes intelligentes, différentes problématiques et enjeux sont à considérer. Tout d'abord, une approche orientée services permet d'avoir une vision haut-niveau et homogène des différents éléments mis en jeu qu'il s'agisse d'objets connectés (accessibles directement ou via des plateformes exposant ces objets via des services (service Web, services REST, etc.), mais aussi d'autres services déployés sur internet (services de géolocalisation, météo, banques de données ouvertes, etc.).

Le système doit être capable de gérer cette diversité avec cette vision orientée services mais aussi de répondre aux problèmes posés par la dynamicité de l'environnement (objets qui changent d'état et / ou de localisation, etc.) ainsi que la dynamicité des besoins utilisateurs et fonctionnels (*e.g.*, gestion dynamique de contenus et d'entités mobiles).

Afin de répondre à ces différents éléments de problématique liés aux systèmes IoT, une **architecture haut-niveau d'un système de gestion autonome** est présentée dans ce chapitre. Ce système se base sur une gestion modulaire et autonomique de systèmes IoT dans une vision orientée services afin d'être capable de surveiller les évènements qui occurrent sur le système supervisé, de prendre des décisions en autonomie ainsi que de générer des actions exécutables sur ledit système.

Tout d'abord l'architecture générale du système proposé est présentée avec ses différents composants ainsi que son mode de fonctionnement général. Ensuite, les

différents modèles en jeu sont abordés. Ils consistent en : un **modèle sémantique à base d'ontologies** permettant de stocker de la connaissance sur le système supervisé ainsi que de réaliser une analyse de cette connaissance ; ainsi qu'**un modèle à base de graphes et de grammaire de graphes** permettant de générer des plans d'exécution exécutables sur le système supervisé. Pour chaque modèle, le principe de ce dernier est mis en avant puis leur mise en œuvre est détaillée. Enfin, des exemples sont présentés afin de comprendre leur instanciation dans le contexte de notre cas d'utilisation.

2.1 Architecture haut-niveau du système proposé

Dans cette section est détaillée l'architecture mise en place pour concevoir un système capable de gérer de manière autonome la dynamicité liée au système IoT avec une vision orientée services. Différents modèles sont concernés dans l'élaboration et l'instanciation de ce système et ils seront détaillés dans les sections suivantes.

2.1.1 Les composants du système

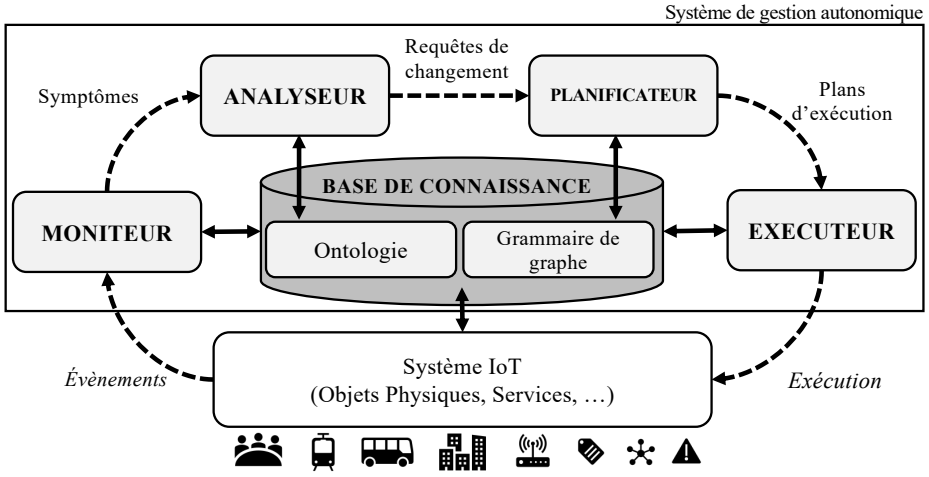


FIGURE 2.1 – Structure globale du système

Afin de traiter le problème présenté précédemment, le système de gestion conçu est présenté en Figure 2.1. Il se compose de trois éléments principaux : le **gestionnaire autonome** (*Autonomic Manager, AM*), la **Base de Connaissance** (*Knowledge base (KB)*), et le **système IoT** (*IoT System, IoTS*) sous-jacent.

2.1.1.1 Le système IoT géré

Le système IoT regroupe les objets connectés considérés qui vont être déployés dans la ville par exemple, les véhicules connectés, et les autres entités connectées, mais aussi les services associés aux objets ou les services logiciels (*e.g.*, déployés

sur le Cloud). Étant donné que l'approche est orientée service il est important de pouvoir faire un lien entre une vision services et les objets physiques accessibles et déployés, et de stocker ce lien dans la base de connaissance.

2.1.1.2 Le gestionnaire autonome (AM)

Le gestionnaire autonome représente l'entité haut-niveau qui est en charge de surveiller et gérer la partie sous-jacente. Ce gestionnaire se base sur un modèle de type MAPE-K introduit dans [Kephart 2003] et présenté plus en détails dans le Chapitre 1, Section 1.3.1. Ce paradigme présente une architecture haut-niveau pour un système autonome spécifiée en quatre composants principaux :

- le **Moniteur** (*monitor*)
- l'**Analyseur** (*analyser*)
- le **Planificateur** (*planner*)
- l'**Exécuteur** (*executor*)

Cette architecture permet au système de répondre de manière autonome aux différents évènements pouvant survenir sur le système supervisé par l'intégration de comportements à travers notamment la base de connaissance. L'idée est de pouvoir intégrer une gestion de comportements haut-niveau définis en fonction de symptômes et pouvant émettre des actions à effectuer.

Dans notre cas, le système se base sur une vision orientée services afin de faciliter l'interaction avec le système supervisé, que ce soit par l'utilisation de plateformes (intergiciels, plateformes Cloud, etc.) ou via l'interaction directe avec les objets concernés.

Dans le système présenté, le gestionnaire interagit avec à la fois la base de connaissance et le système IoT.

2.1.1.3 La base de connaissance

La base de connaissance du système MAPE-K est un composant clef du système. Elle permet de stocker et de partager de la connaissance sur le système contrôlé, mais aussi les politiques haut-niveau de configuration ce qui permet au système d'agir en fonction des événements qui surviennent.

Dans notre approche la base de connaissance est principalement composée de deux modèles différents : une **ontologie** et une **grammaire de graphes** associée à un moteur de transformation de graphes.

Une **ontologie** est une "*spécification explicite d'une conceptualisation*" (*cf.* Section 1.4.1 et [Gruber 1991] pour définition). Elle constitue un modèle sémantique de données d'un domaine et permet de définir des relations entre ces différents concepts. De plus, l'ontologie permet de réaliser du raisonnement sur la connaissance à propos des concepts concernés et donc d'enrichir la connaissance détenue.

Une **grammaire de graphes** définit un ensemble d'éléments comme un graphe type, et un ensemble de règles de réécriture qui vont associer des motifs de graphes à remplacer par de nouveau motifs dans les graphes cibles (*cf.* Section 1.5.1.2 et [Rozenberg 1997] pour définition complète).

Il est important de noter ici que ces modèles peuvent évoluer ou être modifiés en fonction des besoins et du contexte ou même du comportement attendu du système de gestion.

L'ontologie est utilisée pour stocker des méta-données sémantiques en rapport avec le système IoT. Elle permet notamment de réaliser du raisonnement qui va aider l'analyseur à prendre des décisions (*cf.* Section 2.2) via l'inférence de connaissance sur le système supervisé.

Le planificateur instancié pour la boucle autonomique repose quant à lui sur le modèle basé sur une grammaire de graphes pour générer les plans d'exécution (*cf.* Section 2.3.2). En fonction de requêtes générées par l'analyseur, une transformation de graphe va être effectuée sur les requêtes générées par l'analyseur afin de produire un plan d'exécution.

Dans cette thèse, nous nous intéressons particulièrement à la phase d'***analyse*** et de ***planification*** où les décisions sont prises et les actions à effectuer sont générées pour être transformées en plan d'exécution. Les composants *moniteur* et *exécuteur* ne seront pas étudiés en détail. Cependant, des techniques avancée de monitoring peuvent être utilisées en complément telles celles présentées dans [Delgado 2004] [Palacios 2011] [Mosincat 2011].

2.1.2 Évènements et comportement associé

Dans le contexte de notre cas d'utilisation de diffusion de contenu, un ensemble d'évènements ainsi que les symptômes associés (non exhaustif) est présenté en Annexe B, dans la Table B.1. Par exemple, un évènement survenant sur le système peut être : *"Un nouveau contenu c est disponible"*. De fait, le *Moniteur* va associer un symptôme à cet évènement : *"nouveau contenu disponible"*. Lorsque l'analyseur détecte ce symptôme il va ainsi déclencher l'analyse afin de déterminer s'il doit ou non diffuser ce contenu et de quelle manière.

Dans les sections qui suivent, les différents modèles utilisés par le système sont présentés ainsi qu'une instance de ces derniers dans un cas d'utilisation.

2.2 Le modèle basé sur une ontologie

Dans cette section la partie sémantique des modèles en jeu dans la base de connaissance est abordée avec tout d'abord le principe du modèle sémantique, puis une instance de ce modèle.

Différents éléments sont nécessaires à l'élaboration du modèle sémantique ainsi qu'à l'intégration de comportements haut-niveau nécessaires pour la phase d'analyse. Ces derniers vont se composer de concepts, de relations entre eux et de règles d'inférence.

Concepts et relations

Afin de mettre en place un modèle à base d'ontologie pour l'*Analyseur* de notre AM, il est important de définir les différents concepts mis en jeux dans le cas applicatif. Ces concepts vont représenter des entités physiques ou virtuelles ainsi que différents éléments de contexte pour avoir suffisamment de connaissances sur le système surveillé. Par exemple on peut définir un concept représentant une personne ou un bus.

De plus, des relations peuvent être définies entre ces concepts comme par exemple une relation permettant d'indiquer qu'une personne donnée se trouve dans un bus donné.

Règles et inférence

Une base de connaissance sémantique permet au gestionnaire d'être capable d'inférer automatiquement de la connaissance à partir des données qu'il possède déjà sur le système supervisé. L'idée est donc de travailler avec une ontologie de domaine propre au domaine d'application et aux scénarios étudiés. De plus un comportement haut-niveau est implémenté directement dans l'ontologie à l'aide de règles d'inférence. Par exemple, il est possible d'implémenter dans l'ontologie une règle qui indique que si une personne se trouve dans un bus alors elle se trouve à la position géographique du bus.

L'avantage est que le modèle peut aisément être incrémenté par l'ajout de nouveaux concepts ou l'utilisation d'autres ontologies si besoin. De plus, de nouvelles règles peuvent être facilement ajoutées à l'ontologie existante pour inférer de nouvelles connaissances et enrichir le comportement du système ou introduire de nouvelles prises de décision.

2.2.1 Instance d'ontologie

Le modèle basé sur une ontologie utilisé par le système de gestion est présenté dans la Figure 2.2. Le cœur de cette ontologie a été développé pour caractériser le système IoT de notre cas d'utilisation. L'ontologie étend les concepts définis dans l'ontologie IOT-O [1] [Seydoux 2016] (*cf.* Section 1.4.2.8) afin d'avoir des méta-données sur des objets physiques (*devices*) et des services (notamment services REST) (pour plus de détails *cf.* Section 1.4.2.8).

2.2.1.1 Les entités du modèle

Les extensions proposées des concepts d'IoT-O sont présentées ci-dessous :

Content (contenu) : Ce concept peut représenter n'importe quelle donnée brute ou information dans différent formats (document HTML, extrait vidéo, fichier audio, donnée de température, etc.) qui sont produites par des entités productrices de contenus (content producer).

1. `https://www.irit.fr/recherches/MELODI/ontologies/IoT-O.html`

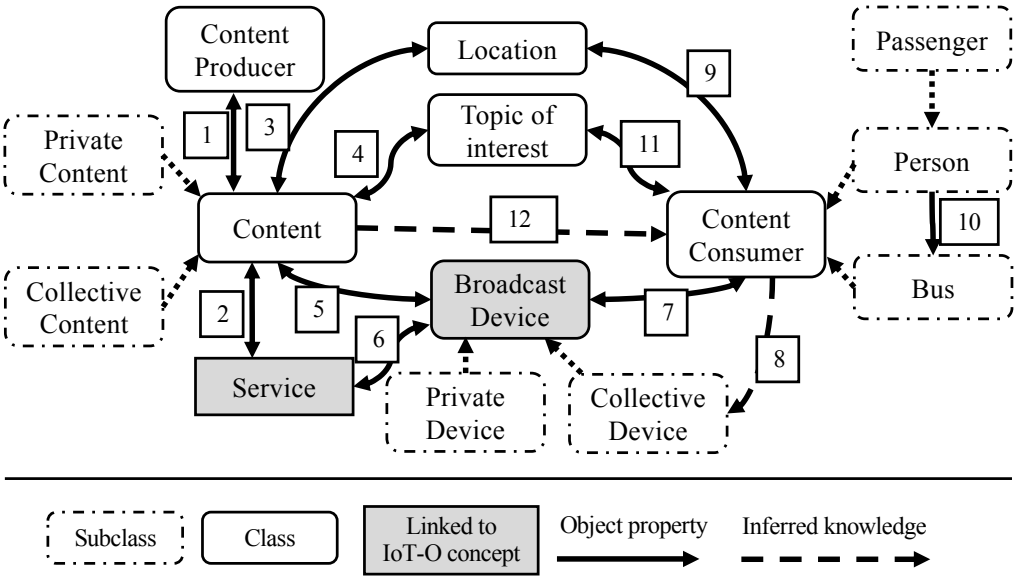


FIGURE 2.2 – Le modèle basé sur une ontologie pour la diffusion de contenu dans la ville intelligente

Topics of interest (sujet d'intérêt) : Ce concept représente tout type de sujet qui peut catégoriser un contenu et qui peut potentiellement intéresser des entités connectées au système. Par exemple, un sujet d'intérêt peut être "*qualité de l'air*", donc les parties tierces peuvent ainsi exprimer leur intérêt pour des informations relatives à la qualité de l'air que ce soient de nouvelles informations, des mises à jour de données, etc.

Content consumer (consommateur de contenu) : Ce concept représente les entités dans le système qui sont intéressées dans le fait d'avoir accès à du contenu. Dans notre cas d'utilisation, les consommateurs de contenu peuvent représenter des personnes ou des véhicules connectés (comme les bus par exemple).

Content producer (producteur de contenu) : Ce concept représente les entités qui produisent des contenus. Dans notre cas d'utilisation, cela peut représenter des capteurs connectés ou même le pôle de communication de la ville intelligente par exemple.

Location (localisation) : Le concept de localisation est une représentation de lieu qui permet de distinguer différents endroits géographiques dans la ville ou même de représenter la ville elle-même (d'un point de vue géographique).

Broadcast devices (terminaux de diffusion, appelés aussi terminaux d'affichage) : Ce concept représente des objets physiques (réels) qui diffusent du contenu à certaines entités. Par exemple, si une personne possède un smartphone ce dernier est considéré comme un terminal d'affichage et est accessible afin d'envoyer du contenu à afficher (à travers une application mobile ou des

notifications par exemple).

2.2.1.2 Relations

Les relations entre les concepts (*object properties*) définies dans l'ontologie sont représentées par des flèches sur la Figure 2.2. Certaines flèches sont bidirectionnelles : cela traduit le fait qu'il existe deux relations qui sont définies comme des relations inverses.

Voici une description de ce qu'elles représentent suivant la numérotation de la Figure 2.2 :

1. Ces relations décrivent le lien entre un *contenu* et l'entité qui le produit, le *producteur de contenu*. Cela permet de mettre en évidence différents contenus produits par le même producteur de contenu si besoin.
2. Ces relations permettent de modéliser le lien entre un contenu et le ou les service·s à utiliser pour récupérer le contenu en soi.
3. Ces relations permettent de modéliser l'attachement symbolique ou physique d'un contenu à une zone géographique : par exemple un capteur de qualité de l'air sera rattaché à une certaine zone géographique.
4. Ces relations permettent de modéliser les liens entre des sujets d'intérêt et des contenus. Un contenu peut être associé à plusieurs sujets d'intérêt et de même un sujet d'intérêt peut être lié à divers contenus différents.
5. Ces relations traduisent le lien entre un terminal d'affichage et un contenu qui a été affiché sur ce dernier.
6. Ces relations modélisent le lien entre un terminal d'affichage et le·s service·s associé·s qui permet·tent de diffuser du contenu sur ledit terminal
7. Ces relations modélisent le lien entre un consommateur de contenu et un terminal d'affichage. Le consommateur de contenu peut en effet avoir un ou plusieurs terminaux associés (comme un smartphone par exemple).
8. Cette propriété représente le fait qu'une entité donnée ait accès à un terminal d'affichage. Cette relation est complémentaire à la propriété 7 : elle permet d'indiquer qu'une personne a accès à un terminal qui n'est pas sien, par exemple quand une personne est passager d'un bus elle a accès au terminal du bus cependant ce terminal est le terminal du bus et non pas de la personne.
9. Ces relations traduisent l'association entre des consommateurs de contenus et de lieux. Cela permet de positionner géographiquement grossièrement ou finement des entités.
10. Cette relation permet de traduire le fait qu'une entité consommatrice de contenu de type *Personne* est passagère d'un *Bus*. Cette relation est notamment utilisée dans certains mécanismes d'inférence par la suite.
11. Ces relations traduisent les associations entre consommateurs de contenu et les sujets d'intérêt. Cela permet de modéliser les intérêts d'une entité

pour des sujets mais aussi le fait qu'un sujet particulier intéresse un certain ensemble d'entités.

12. Cette propriété est primordiale pour caractériser le fait qu'un contenu devrait être envoyé à une certaine entité. La détermination de cet élément se fait grâce à la position connue et aux intérêts exprimés de l'entité ciblée. Cette connaissance est un élément clef pour la prise de décision dans la phase d'analyse et cet aspect est présenté plus en détail dans la Section 2.2.2.

Exemple 1 ***Exemple d'instance simple***
Un exemple d'instance simple de l'ontologie présentée précédemment est schématisé dans la Figure 2.3 ci-dessous.

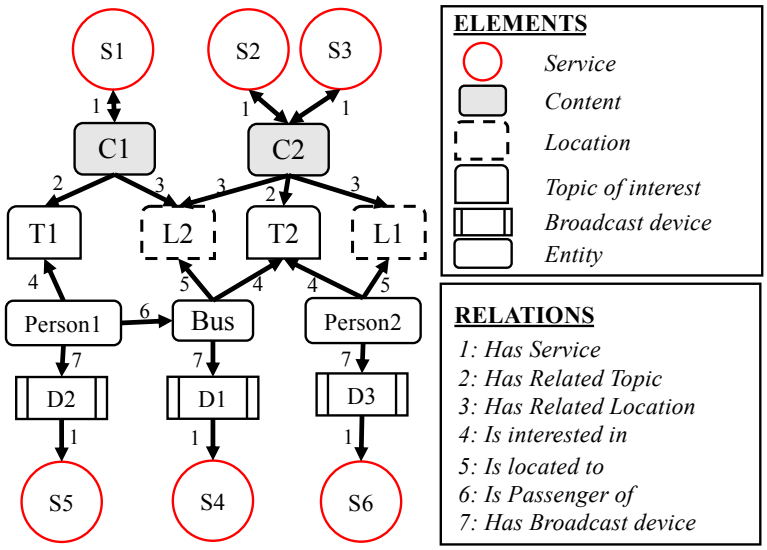


FIGURE 2.3 – Une instance basique de l'ontologie

Dans cet exemple, deux contenus différents sont identifiés C_1 et C_2 (afin de simplifier l'exemple et la clarté du schéma, les producteurs des contenus en question n'ont pas été représentés). Il y a également un bus Bus ainsi que deux personnes différentes $Person_1$ et $Person_2$ qui sont considérés. La personne 1 est un passager du bus tandis que la personne 2 n'est pas encore un passager du bus à cet instant. Deux sujets d'intérêts différents T_1 et T_2 sont également instanciés ainsi que deux lieux différents L_1 et L_2. C_1 est un contenu lié au sujet T_1 et est lié à la position géographique L_2, et C_2 est lié au sujet T_2 et aux deux positions géographiques. (Par exemple C_2 peut représenter une donnée météorologique qui va affecter plusieurs lieux différents.) Les services associés sont également représentés sur le schéma ($S_1..S_6$). On remarquera que C_2 est lié à deux services S_2 et S_3 : cela peut représenter par exemple deux possibilités de récupération du contenu différentes via deux protocoles différents. Les relations instanciées sont décrites dans la légende à droite de la Figure.

2.2.2 Règles SWRL et Analyse

L'analyseur détermine quelles sont les entités intéressées dans les contenus cibles grâce à la base de connaissances sémantiques. Cette fonctionnalité est caractérisée par des règles SWRL [Horrocks 2004] (*cf.* Section 1.4.1) embarquées dans la base de connaissances sémantique. L'ensemble de règles permet d'inférer de la connaissance à travers l'utilisation d'un raisonneur. Par exemple, cela va permettre d'inférer qu'un contenu devrait être envoyé à un ensemble d'entités de manière automatique. Cependant, cela n'inclut pas la manière dont le contenu doit être envoyé (quel service utiliser, etc.). Le rôle de l'analyseur est d'extraire cette connaissance de la base de connaissance et de générer des requêtes de changement (RFC) à transmettre au planificateur qui sera en charge de déterminer comment effectuer les changements ou actions voulus. Par exemple, l'analyseur va déterminer qu'un certain contenu doit être envoyé à un certain nombre d'entités et le planificateur va déterminer comment l'envoyer à toutes ces entités avec le meilleur moyen en se basant sur les terminaux auxquels les entités ont accès.

Différentes règles sont exploitées dans l'inférence de connaissance. Les règles implémentées dans l'ontologie utilisée pour notre cas d'utilisation sont présentées ci-dessous :

La règle présentée dans le Listing 2.1 permet de déterminer si un contenu devrait ou non être envoyé à une personne que le contenu soit d'ordre collectif ou privatisé. En effet, un contenu à envoyer à une personne en particulier peut être d'ordre privatisé ou collectif.

```
Person(?p) ^ isInterestedIn(?p, ?topic) ^
Content(?info) ^
hasRelatedTopic(?info, ?topic) ^
hasRelatedLocation(?info, ?loc) ^
isLocatedTo(?p, ?loc) ^
hasDisplayDevice(?p, ?d)
-> shouldBeSentTo(?info, ?p)
```

Listing 2.1 – Règle de détermination de l'envoi d'un contenu à une personne

La règle présentée dans le Listing 2.2 permet d'ajouter un type à un individu représentant une personne dans le cas où cette personne est dans un bus l'individu est également de type *Passager*.

```
Person(?p) ^ isPassengerOf(?p, ?b)
-> Passenger(?p)
```

Listing 2.2 – Règle inférant un type supplémentaire pour une personne passager d'un bus

La règle présentée dans le Listing 2.3 permet de déterminer si une personne a accès à un terminal qui n'est pas le sien. De fait, cette inférence se base sur le fait

que la personne est passager d'un bus qui a un terminal d'affichage. Si tel est le cas, la personne a alors automatiquement accès au terminal du bus.

```
Person(?p) ^ isPassengerOf(?p, ?b) ^
hasDisplayDevice(?b, ?d) -> hasAccessTo(?p, ?d)
```

Listing 2.3 – Règle de détermination de l'accès à un terminal d'affichage

De manière similaire à la règle précédente, la règle présentée dans le Listing 2.4 permet à un passager d'un bus donné d'hériter de la position du bus. Cela peut simplifier l'inférence de connaissance notamment dans le cas où la position du passager est moins précise ou envoyée moins souvent que celle du bus.

```
Passenger(?p) ^ isPassengerOf(?p, ?b) ^
isLocatedTo(?b, ?l) -> isLocatedTo(?p, ?l)
```

Listing 2.4 – Règle impliquant l'héritage de la position d'un bus pour ses passagers

La règle présentée dans le Listing 2.5 permet de déterminer à quelles entités un contenu collectif doit être envoyé. Elle se base sur le(s) sujet(s) lié(s) à ce contenu ainsi que les localisations pour déterminer à quelles entités (consommateurs de contenu) il devrait être envoyé.

```
ContentConsumer(?e) ^
isInterestedIn(?e, ?topic) ^
CollectiveContent(?c) ^
hasRelatedTopic(?c, ?topic) ^
hasRelatedLocation(?c, ?loc) ^
isLocatedTo(?e, ?loc) -> shouldBeSentTo(?c, ?e)
```

Listing 2.5 – Règle de détermination d'envoi de contenu collectif

Exemple

En se basant sur les règles SWRL présentées ci-dessus et le reste de l'ontologie, l'analyseur va décider de déclencher le raisonneur en fonction des évènements qui surviennent dans le système (apparition de nouveaux contenus par exemple). Le déclenchement de ce raisonneur va permettre l'inférence de nouvelles connaissances en fonction des éléments présents dans la base de connaissance sémantique. Ensuite, l'analyseur va générer une RFC basée sur la connaissance incluant les éléments inférés si besoin. Par exemple, dans notre cas d'utilisation de gestion de contenu, il va considérer les relations qui indiquent à quelles entités les nouveaux contenus devraient être envoyés. Ces relations vont permettre de faire le lien ensuite entre les données à récupérer produites par les services producteurs et les services liés aux terminaux des entités à qui le contenu devrait être envoyé.

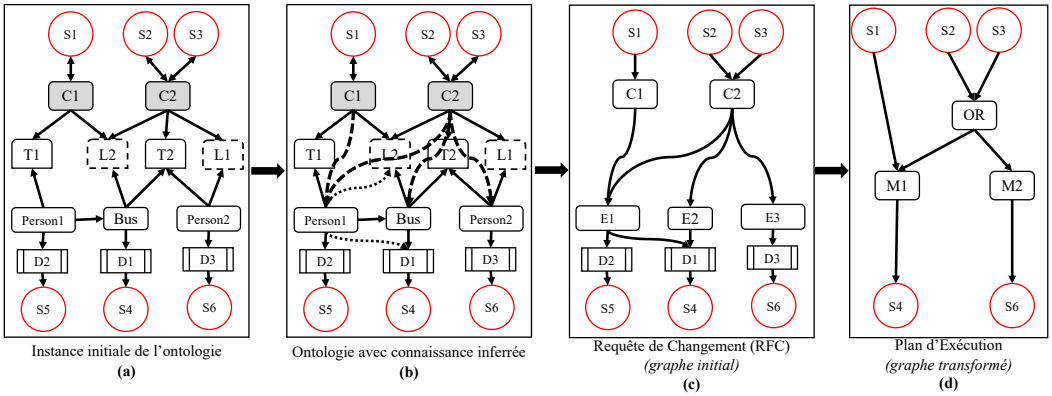


FIGURE 2.4 – Une instance complète du système et des différentes étapes suivies

Exemple 2 ***Inférence de connaissance à partir d'une instance simple***
Un exemple d'inférence de connaissance basé sur l'instance précédente de l'ontologie (Exemple 1) est présentée dans la Figure 2.4.(b).

Ici, l'analyseur déclenche le raisonneur de la KB sémantique (telle qu'instanciée dans la Figure 2.4.(a)) et va observer les changements qui vont s'opérer dans la base. Ce comportement est basé sur les symptômes reçus en conséquence de la mise à disposition de nouveaux contenus (ici C_1 et C_2). De la connaissance est automatiquement inférée telle que montrée dans la Figure 2.4.(b) : les nouveaux contenus sont intéressants pour certaines entités et devraient être envoyés à ces dernières. La connaissance inférée est visible sur la Figure 2.4.(b) en lignes épaisses discontinues pour le contenu qui devrait être envoyé à des entités, et en pointillés pour des propriétés inférées en rapport avec les passagers.

Dans la section suivante, les modèles pour la RFC, le plan d'exécution ainsi que celui pour la transformation de graphe sont détaillés.

2.3 Les modèles orientés graphe

Dans cette section les modèles basés sur des graphes utilisés par le système sont détaillés : les modèles de la RFC, de la transformation de graphes et du plan d'exécution.

Le principe de ce modèle est d'apporter un complément par rapport au modèle sémantique. En effet, l'ontologie permet de réaliser une phase d'analyse ainsi que de regrouper un ensemble de connaissances sur le système supervisé. Cependant, ce modèle a des limites notamment lorsqu'il s'agit de générer des plans d'exécution à base de services à exécuter sur le système réel. De fait, pour compléter ce comportement, un modèle à base de grammaire de graphes a été développé. Il permet à partir de requêtes de changement haut niveau (RFC) générées par le composant d'analyse (via l'analyse sémantique) de réaliser des opérations de transformation de

graphes afin de transformer un ensemble d'informations en un plan exécutable sous forme de services.

2.3.1 Requête de changement à effectuer (RFC)

Quand une action doit être effectuée sur le système supervisé, qu'elle relève de la diffusion de contenu, de la récupération et traitement d'information ou autre, l'analyseur génère une spécification pour l'action à effectuer. Cette dernière est représentée sous la forme d'un graphe de RFC accompagné d'une grammaire de graphes ([Rozenberg 1997]) afin de générer le plan d'exécution à partir du graphe de la RFC qui est transformé.

La RFC est un graphe qui contient les informations relatives aux différents services à utiliser et d'autres éléments liés aux services. Dans notre cas d'utilisation, la RFC contient par exemple les contenus à envoyer et les informations intéressantes extraites de la KB qui vont permettre de trouver un moyen de les transmettre. Ce graphe RFC est transmis ensuite au planificateur qui va le transformer afin de générer le plan d'exécution qui lie les différents services (producteurs et consommateurs de contenus). Un point important à souligner ici est l'aspect générique et modulaire de l'analyseur : de nouvelles fonctionnalités et de nouveaux comportements peuvent être facilement intégrés dans le système à travers de nouveaux ensembles de symptômes associés à un modèle de RFC et une grammaire de graphe pour générer les plans d'exécutions.

La RFC est le graphe initial nécessaire pour permettre au planificateur de transformer ce graphe et fournir un plan à exécuter.

Définition 3 *(**Request for Change : RFC**) Une spécification haut-niveau est composée de deux éléments : une RFC sous forme de graphe typé (ou semi-typé) et une grammaire de graphe associée.*

Soit Γ *l'ensemble des types de nœuds. Un graphe RFC est un tuple* (N, E, τ) *dans lequel :*

- N *est l'ensemble de nœuds qui caractérisent les services et les entités en jeu dans l'élaboration du plan d'exécution*
- $E \subseteq N * N$ *est l'ensemble d'arcs qui caractérisent les flux de données (contenus) qui vont être récupérées via des services, éventuellement transformées, puis envoyées à des services (consommateurs)*
- $\tau : N \rightarrow \Gamma$ *est une fonction qui associe les nœuds à leur type*

Ce modèle de RFC peut être instancié de la façon qui suit avec différents types de nœuds, par exemple pour la gestion de contenu :

Services notés S : ces nœuds représentent les différents services en jeu dans l'action à effectuer (service de production de contenu ou de consommation de contenu)

Contenus notés C : ces nœuds représentent des données intéressantes (fournies par des services producteurs de contenu) qui peuvent être consommées par un ensemble d'entités

Entités notées E : ces nœuds représentent les personnes, les bus, ou toute autre entité du système intéressée dans le fait d'avoir accès à du contenu fourni par des services

Terminaux d'affichage notés D (device) : ces nœuds représentent tout type d'objet connecté qui permet au système de diffuser des informations aux entités concernées. Cela peut consister en un smartphone à travers une application mobile par exemple, ou des écrans connectés dans les bus de la ville. Ces objets possèdent un ensemble associé de services qui vont *consommer* les données fournies par d'autres services pour les afficher (*i.e.* qui vont recevoir le contenu en entrée (input) et l'afficher sur le terminal). De plus, les entités peuvent avoir accès à plus d'un terminal d'affichage. Cette connaissance est représentée par un arc entre le nœud représentant l'entité et le nœuds représentant le terminal.

En résumé, l'analyseur identifie quels sont les éléments à inclure dans le graphe de la RFC en se basant sur les symptômes en provenance du moniteur et la connaissance dans la KB sémantique. Dans notre cas d'utilisation, il considère quels contenus devraient être envoyés à quelles entités, ainsi que les éléments nécessaires pour permettre au planificateur de générer les plans d'exécution correspondants. Pour ce faire, l'analyseur rassemble des informations en rapport avec les entités : quels sont les terminaux auxquels elles ont accès, et quels sont les services producteurs et consommateurs de contenu associés respectivement aux-dits contenus et aux-dits terminaux d'affichage.

Exemple 3 ***Génération de RFC***
Un exemple de transformation de graphe à partir de connaissance extraite d'une instance d'ontologie est présentée en Figure 2.4.(c).

Dans la continuité de l'Exemple 2, dans l'Exemple 3 l'analyseur génère la RFC correspondante telle que montrée dans la Figure 2.4.(c). L'idée est d'extraire les informations nécessaires pour le planificateur et de générer un graphe en fonction. Comme on peut le constater, seules les informations nécessaires pour que le planificateur puisse déterminer quels terminaux utiliser sont incluses dans la RFC. D'autres informations comme les lieux et les sujets d'intérêt ne sont pas retenus dans la RFC car inutiles pour la planification.

Dans la prochaine section, le modèle du *plan d'exécution* basé sur les *grammaires de graphes* est défini. Un exemple d'instance est également présenté.

2.3.2 Génération de plan d'exécution

Dans la continuité de ce qui a été mentionné précédemment, le planificateur reçoit une RFC en provenance de l'analyseur sous forme de graphe. L'objectif est de générer un plan d'exécution à partir des informations disponibles et du graphe RFC

fourni par l'analyseur. Cette génération s'effectue à travers une série de transformations qui vont être effectuées sur la RFC afin de générer un graphe qui représente le plan d'exécution. En effet, le plan d'exécution comporte un ensemble de services qui doivent être exécutés suivant un ordre d'appel ainsi que les connections entre les données produites par certains services et les données consommées par d'autres.

De telles transformations sont formalisées à l'aide du formalisme des grammaires de graphes. Un ensemble de règles de réécriture de graphes (règles de transformation) est défini. Le résultat de la transformation est un graphe modélisant un plan d'exécution qui caractérise l'ensemble de liens concrets entre les différents services en fonction des informations données par l'analyseur. Ainsi les contenus récupérés à travers les services producteurs peuvent être soit assemblés ou sélectionnés entre deux services producteurs qui fournissent le même contenu sous des formats différents.

2.3.2.1 Graphe de plan d'exécution

Définition 4 ***Modèle de graphe de plan d'exécution*** *Le modèle du plan d'exécution est en partie similaire à celui de la RFC étant donné qu'il va se composer de certains nœuds provenant de la RFC et en introduire de nouveaux. Une instance du modèle de plan d'exécution est la suivante :*

Services *S : ces nœuds représentent les différents services en jeu dans l'action à effectuer (service de production de contenu ou de consommation de contenu)*

Nœuds d'agrégation (merge) *notés M : indiquent que des entrées ou sorties de services doivent être agrégées avant d'être envoyées à un autre service, ainsi les données des arcs entrants sur le nœud sont agrégées avant d'être envoyées sur les arcs sortants*

Nœuds de choix (OR) *notés OR : indiquent que plusieurs services sont disponibles pour effectuer l'action désirée et l'un d'entre eux peut être choisi à l'exécution pour un résultat équivalent*

Donc, un graphe RFC est transformé en un graphe caractérisant un plan d'exécution en se basant sur le formalisme de grammaire de graphe présenté Section 1.5.1.2.

2.3.2.2 Grammaire de graphe et règles de réécritures

Différentes façons de spécifier des règles de transformations de graphes ont été proposées (*cf.* [Rozenberg 1997] et Section 1.5.1.2). Dans ces travaux, l'approche *Single Push-Out* (SPO) est utilisée.

L'instance de grammaire de graphe est composée de différents éléments : un graphe type permet de définir tous les nœuds possibles des différents graphes ainsi que les liens possibles (arcs) entre les nœuds considérés et leur cardinalité.

Graphe Type Le graphe type de l'instance de grammaire de graphe développée en rapport avec notre cas d'utilisation est présenté en Figure 2.5. Ce graphe définit

l'ensemble des arcs possibles entre les différents nœuds pouvant composer un graphe de RFC et sa version transformée. Par exemple, le lien est clairement directionnel entre les objets (*devices, d*) et les services, là où dans le cas d'utilisation d'opérateurs (*Merge, OR*) les liens peuvent être multiples (entrants ou sortants) car étant en amont ou en aval des services dans le flux d'exécution.

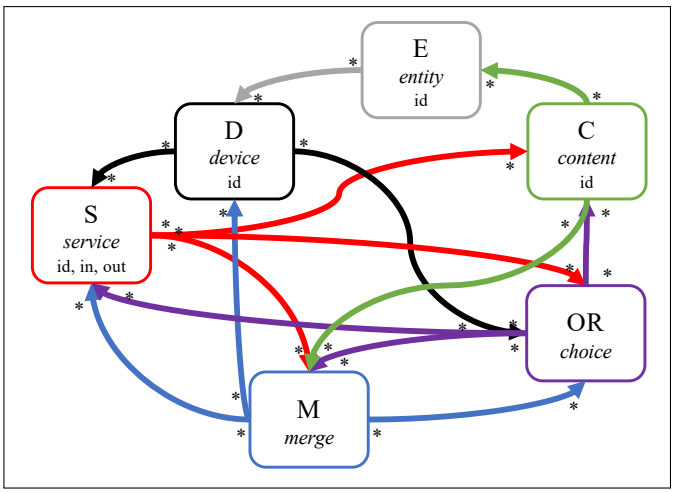


FIGURE 2.5 – Graphe type de l'instance de grammaire de graphe

Règles de réécriture Dans la Figure 2.6, les graphes L et R de chaque règle sont schématisés, et la fonction f_r associée au morphisme m_r de la règle concernée est matérialisée par une flèche de gauche à droite. L'ensemble de règles présenté permet au planificateur de décider quels services producteurs agréger avant d'envoyer les données récupérées aux bons services consommateurs. Les règles sont séparées en plusieurs couches.

On distingue trois couches de règles :

Couche 1 (règles 1 à 6) : Cette couche est dédiée à l'introduction de nœuds M dans le graphe afin d'agréger les contenus en se basant sur le fait que certaines entités devraient recevoir le même contenu. Cette couche permet la création de nœuds intermédiaires dans le graphe (agrégation M) dans le but d'agréger les sorties de plusieurs services avant de l'envoyer aux services consommateurs. L'idée est d'être capable de regrouper différents contenus intéressants pour plusieurs entités qui ont accès à un terminal d'affichage commun. Ces contenus, récupérés à partir de services producteurs, seront assemblés durant la phase d'exécution et utilisés en entrée des services associés au terminal d'affichage.

Les règles 1 et 2 créent des nœuds M dans le graphe. Les entités demeurent liées aux terminaux afin d'éviter la destruction de la connexion avec un autre contenu et ce terminal à travers les entités. Cependant, les entités disparaissent totalement du graphe à terme. La première règle crée des nœuds

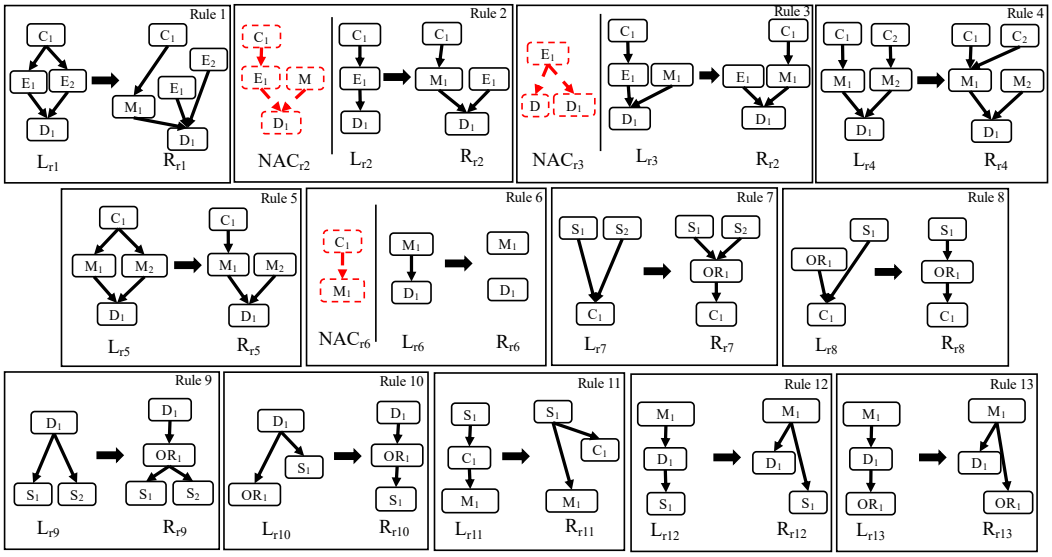


FIGURE 2.6 – Un ensemble de règles de transformation de la grammaire de graphe instanciée

M en se basant sur le fait que deux entités différentes devraient recevoir le même contenu et ont accès à un terminal d'affichage en commun (par exemple, si les entités représentent deux passagers d'un même bus). La stratégie de la deuxième règle est similaire mais elle crée des nœuds M pour chaque lien restant entre les contenus et les entités après application de la première règle.

Les règles 3, 4 et 5, se basent sur les nœuds M existants pour connecter les contenus aux terminaux ou bien pour fusionner différents nœuds M pour éviter une duplication de nœuds M liés au même terminal d'affichage. En effet, les règles 4 et 5 vont permettre de regrouper les liens entre les contenus et certains nœuds d'agrégation pour pouvoir regrouper l'envoi sur un terminal.

La règle 6 va détacher les nœuds M qui ne sont plus connectés à aucun contenu ce qui permettra par la suite des les supprimer du graphe.

Couche 2 (règles 7 à 13) : Cette couche travaille sur ce qui résulte de la couche précédente et va faire la connexion entre les nœuds M créés et les services. Les règles 7 et 8 permettent d'introduire des nœuds OR pour indiquer qu'il y a plusieurs choix possibles parmi les services associés à un contenu. De manière similaire, les règles 9 et 10 permettent de faire la même action avec les services liés à un terminal d'affichage.

Les règles 11 et 12 permettent de créer le lien entre les services et les nœuds M. Des règles similaires comme la règle 13 (pas toutes schématisées ici) permettent d'obtenir le même comportement avec des nœuds OR s'ils sont présents.

Couche 3 : La troisième couche est dédiée à nettoyer le graphe des nœuds isolés restants : les entités E, les terminaux D, et les nœuds représentant les contenus C. En effet, dans le plan d'exécution final, seuls resteront les services et les nœuds intermédiaires (M et OR). Ces règles n'ont pas été schématisées ici par souci de clarté : elles sont simples et consistent uniquement à isoler les nœuds inutiles (comme les nœuds M détachés par la règle 6) et à les supprimer ensuite.

Exemple 4 ***Transformation de graphe RFC***
Dans la continuité de l'Exemple 3, dans la Figure 2.4.(c) et 2.4.(d) sont représentées une instance d'un graphe RFC simple qui a été généré par l'analyseur et sa transformation.

La Figure 2.4.(d) représente le graphe de plan d'exécution qui résulte de la transformation du graphe RFC d'après les règles de la grammaire de graphes présentée précédemment. Le plan obtenu établit que le contenu produit par le service S_1 agrégé avec, soit le contenu de S_2 ou soit de S_3 sont à envoyer au service S_4. De plus, le contenu produit par les services S_2 ou S_3 doit être envoyé au service S_6.

De ce fait, cela permet de faire parvenir le contenu C_1 (produit par S_1) et C_2 (produit par S_2ou S_3) à toutes les entités concernées (le bus et le passager) en une seule opération (appel de S_4). E_3 reçoit tout de même le contenu qui l'intéresse cependant cela passe par le seul terminal accessible pour cette entité. Dans le cas où E_3 est une personne qui deviendrait passager du bus (E_2) alors le contenu pourrait ne plus être envoyé via S_6 mais via S_4 à E_3 vu que l'entité aurait alors accès au terminal du bus (D_1) comme E_1.

(Un exemple de transformation plus complexe est présenté en Annexe B.2.)

Donc, de par l'application de règles de transformation spécifiées dans la Figure 2.6 le graphe de RFC généré par l'analyseur est transformé par le planificateur en un plan d'exécution qui est transmis à l'exécuteur pour une exécution sur le système réel.

Conclusion

Différentes problématiques liées à l'environnement IoT se posent dans notre contexte, à savoir : une dynamicité et une mobilité de certains éléments (objets connecté, mobilité dans un environnement de ville intelligente, etc.), mais aussi la possibilité d'intégrer de nouveaux comportements à la volée ainsi qu'être capable d'utiliser les différents éléments (services) disponibles de manière dynamique.

Pour répondre à ces éléments de problématique, une **architecture haut-niveau générique basée sur la boucle autonomique MAPE-K** a été proposée et détaillée. Cette architecture permet de pallier les problèmes liés à la dynamicité ainsi qu'à la variabilité des usages et des fonctionnalités disponibles de par sa structure ainsi que les différents modèles mis en œuvre.

En effet, un premier **modèle couplant sémantique et ontologies** a été défini puis instancié dans un cas d'utilisation (bus connecté dans une ville intelligente avec diffusion adaptative et dynamique de contenu). Ce modèle permet de réaliser une analyse des différents évènements et changements qui occurrent dans le système IoT supervisé ainsi que d'inférer de nouvelles connaissances en se basant sur la connaissance actuelle. Du plus un modèle complémentaire à base de **grammaires de graphes** et de transformation de graphes a été défini puis instancié dans le cas d'utilisation de cette thèse. Ce modèle permet de compléter le comportement du modèle sémantique afin de générer des plans d'exécution à base de services exécutables sur le système supervisé.

Des exemples de mise en œuvre ont également été présentés pour illustrer le fonctionnement du système et des différents modèles.

Ces travaux ont donné lieu à une publication scientifique [Garzone 2018] ainsi qu'à l'élaboration d'un prototype dans le contexte du projet S2C2. Ces différents éléments seront intégrés par la suite au prototype du système réalisé afin d'évaluer son comportement et ses performances qui sera détaillé dans le Chapitre 4.

Dans le chapitre suivant, afin de permettre l'utilisation de ce système dans des cas plus complexes, les modèles proposés vont être enrichis afin de permettre de prendre en considération des paramètres non fonctionnels jusqu'à présent non considérés dans les modèles. Un algorithme d'exploration de graphe est également proposé afin de réaliser des optimisation sur ces paramètres non fonctionnels dans le cas où il y a plusieurs choix possibles de plans d'exécution.

Chapitre 3

Gestion de services IoT complexes avec paramètres de QoS

Sommaire

3.1 Vers une gestion plus haut-niveau des services 56
3.1.1 Principe général 56
3.1.2 Modèle enrichi de RFC 58
3.1.3 Intégration de besoins et de services à la volée 60
3.2 Construction et exécution d'un plan opportuniste 73
3.2.1 Approche générale 73
3.2.2 Algorithme d'exploration et de recherche d'optimum : vers un choix optimisé 79
3.2.3 Autre algorithme témoin pour tests de performance : glouton 82
3.2.4 Évaluation de résultat des algorithmes et validation des contraintes 84
3.2.5 Exemple complet d'application des algorithmes et comparaison de résultats 84

Une première étape nécessaire pour le système était qu'il soit capable de répondre à différentes problématiques, notamment la gestion de la dynamicité et la possibilité de définir des comportements haut-niveau. De plus, un point primordial est d'analyser les évènements qui occurrent sur le système supervisé et de déduire de manière autonome quels sont les éléments à récupérer, envoyer, contrôler, etc. Cet aspect fonctionnel a été mis en place à travers une architecture adaptée et l'instanciation de modèles sémantiques et de grammaires de graphes pour que le système puisse automatiquement analyser et déduire les informations nécessaires jusqu'à l'élaboration de plans d'exécution à base de services. Le système proposé est également capable de gérer la mutualisation de services dans les plans d'exécution générés à l'aide d'un processus de transformation de graphes.

Cependant, les services étaient considérés jusqu'à présent comme interopérables et étant directement utilisables. Par exemple, dans le cas où une personne est intéressée par une donnée de qualité de l'air à une adresse spécifique, cette donnée est accessible directement par l'appel du service associé. Dans certains cas il se peut qu'un service (capteur) soit directement accessible à la position géographique

concernée mais dans d'autres situations cela peut être différent. Dans le cas où il n'y a pas de service directement accessible, il est cependant envisageable d'utiliser un service *complexe* qui devra être déterminé automatiquement par le système en se basant sur des services disponibles et des informations comme la position de l'utilisateur que ce dernier aura communiquée.

Ce comportement va se traduire par une intégration de *besoins* dans la RFC afin de remplir l'objectif initial dans le cas où il n'y a pas de services et de solution exécutable en l'état. Par exemple, si une personne est intéressée par une donnée de qualité de l'air à sa position, le système pourra générer un besoin possédant en entrée cette position et qui attend une donnée de qualité de l'air.

De plus, un aspect important qui n'était pas considéré jusqu'à présent sont les propriétés non fonctionnelles : comment intégrer dans cette architecture et ces modèles une considération de paramètres QoS comme le temps d'exécution, le coût énergétique ou encore le coût monétaire de l'exécution des plans produits.

Pour ce faire des enrichissement des modèles proposés dans le chapitre précédent sont nécessaires pour compléter le comportement existant et définir une grammaire de plus haut-niveau pour la gestion des services. Cette grammaire de graphes avancée est présentée afin de gérer la résolution de besoins générés à la volée par le système et d'intégrer des services complexes composites dans le plan d'exécution.

De plus, suivant les scénarios d'utilisation, il peut être important de réduire (optimiser) certains paramètres QoS mais aussi de contraindre le système. En considérant différents paramètres non fonctionnels, il est alors possible de réaliser des choix optimisés lors de l'exécution des plans générés automatiquement pas le système. Pour ce faire des algorithmes d'exploration des graphes transformés sont proposés afin de déterminer quel plan exécuter dans le cas où il y aurait plusieurs possibilités. Un premier algorithme dit *glouton* permet de trouver de manière opportune une solution dans le cas où il y a un plan exécutable et permet de servir de référence en terme de performance de résultat, et un algorithme plus complexe *optimisé* permet de trouver un *optimum* (global, et local dans un cas particulier) dans les plans d'exécutions. Des contraintes sur ces paramètres sont également intégrées dans le modèle.

3.1 Vers une gestion plus haut-niveau des services

Ici, on se focalisera sur l'étape de génération de plans d'exécution qui implique le modèle à base de grammaires de graphes. Le principe étant à partir du graphe généré par l'analyseur, d'être capable de construire un pseudo-plan de composition et ensuite d'exécuter un plan optimisé si possible.

3.1.1 Principe général

Jusqu'à présent les scénarios traités se concentraient sur une gestion de contenu que ce soit récupération de contenu ou diffusion aux cibles concernées en utilisant

les moyens à disposition et en considérant que les contenus étaient récupérables (resp. affichables) via l'utilisation directe d'un seul service.

Afin d'enrichir ce scénario il est possible de généraliser cette approche en incluant d'autres comportements liés à des services : services de déplacement, réservation, etc. De plus, il est également possible que les services de production de contenu nécessitent le recours à des services de traitement intermédiaire avant de transmettre le contenu aux services de consommation de contenu. Enfin, dans le cas où il n'y a pas de service qui réponde directement au besoin, le système peut chercher à construire un service composite dynamiquement en se basant sur les informations connues et le besoin à cet instant particulier. Un exemple simple réside dans le cas où un utilisateur souhaite accéder à une donnée de qualité de l'air à son adresse mais qu'aucun capteur ne couvre cette zone. Il peut alors partager sa position et le système va générer un besoin à l'aide de cette position qui attend une donnée de qualité de l'air. Ce besoin sera instancié via l'utilisation de plusieurs services qui vont permettre de récupérer la zone géographique correspondant à l'adresse donnée, puis récupérer la valeur de qualité de l'air connue ou estimée sur cette zone. (Principe présenté en Figure 3.1.)

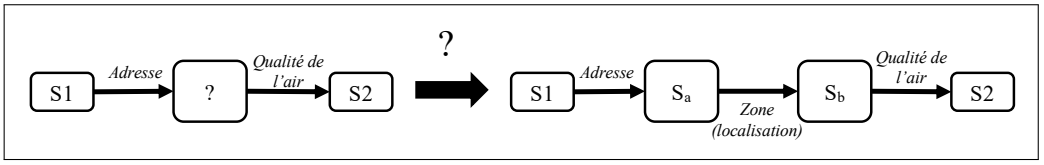


FIGURE 3.1 – Principe de besoin et de résolution dynamique

Dans ce contexte plus complexe, non seulement des contenus peuvent être utilisés et diffusés mais aussi des besoins vont être récupérés et devront être traités. Par exemple un passager d'un bus intéressé dans certaines informations va pouvoir recevoir le contenu suivant différents moyens mais aussi découvrir un nouvel évènement qui l'intéresse particulièrement et décider de s'y rendre. Pour ce faire il va générer dynamiquement un nouveau besoin de déplacement depuis sa position actuelle jusqu'au lieu de l'évènement via les services de déplacement accessibles. Cependant, il est nécessaire qu'il soit sur place avant une certaine heure pour pouvoir accéder à l'évènement tant que celui-ci est accessible. De ce fait cela va pouvoir inclure des contraintes temporelles dans les différents services qui seraient utilisés.

Afin d'être capable de traiter ces différents éléments, le système a besoin d'une gestion plus complexe des services considérés ainsi que de prendre en compte des paramètres non-fonctionnels ou de QoS. De plus, les graphes vont être complexifiés en intégrant des possibilités plus riches entre les différents services via l'utilisation de nœuds supplémentaires dans les graphes à transformer et les plans d'exécution générés.

De nouveaux éléments vont être inclus dans le modèle : prise en compte de besoins à instancier, d'objectifs et de contraintes.

3.1.2 Modèle enrichi de RFC

La RFC est le graphe initial nécessaire pour permettre au planificateur de transformer ce graphe et fournir un plan à exécuter. Ce graphe va ainsi comporter un ensemble d'informations permettant au planificateur de générer des plans d'exécution. Ces informations seront fournies à la fois par l'*analyseur* et la *base de connaissance* partagée.

Définition 5 ***Spécification haut-niveau : RFC enrichie***

Une spécification haut-niveau est composée de deux éléments : une RFC sous forme de graphe et une grammaire de graphe associée.

Soit Γ *l'ensemble des types de nœuds. Un graphe RFC est un tuple* (N, E, τ) *dans lequel :*

- N *est l'ensemble de nœuds qui caractérisent les services, les nœuds temporaires et les entités en jeu dans l'élaboration du plan d'exécution*

- $E \subseteq N * N$ *est l'ensemble d'arcs qui caractérisent les flux de données (contenus) qui vont être récupérées via des services, éventuellement transformées, puis envoyées à des services (consommateurs). Ces arcs possèdent un attribut spécifiant quel contenu est représenté dans ce flux.*

- $\tau : N \rightarrow \Gamma$ *est une fonction qui associe les nœuds à leur type*

Ce modèle de RFC est instancié de la façon qui suit avec différents types de nœuds :

Nœuds terminaux (début et fin), notés *begin* et *end* : ces nœuds permettent respectivement de représenter le début d'un flux de services (actions) à exécuter et d'en représenter la fin. Cela permet au composant *Exécuteur* de faire une lecture du plan d'exécution dirigé dont le nœud source est le nœud *begin*.

Services, notés s : ces nœuds représentent les différents services en jeu dans l'action à effectuer (service de production de contenu ou de consommation de contenu par exemple). Ces nœuds ont un ensemble d'attributs associé noté $A_s = \{in_0..in_n, out_0..out_m, qos_0..qos_p\}$ où *in* représente un type d'entrée (*input*) et de même *out* représente un type de sortie (*output*) du service. De plus, les attributs *qos* vont représenter des paramètres non-fonctionnels afin de pouvoir comparer ces paramètres entre les différents services ainsi que de calculer les coûts totaux de l'exécution d'un ensemble de services. Par exemple un service s_1 ayant deux entrées et une sortie avec un paramètre *qos* (temps d'exécution par exemple) aura comme attributs : $A_{s_1} = \{in_0, in_1, out_0, qos_0\}$. De plus, les arcs entrants et sortant du nœud s_1 sont annotés avec le type caractérisant l'input et l'output du service. En l'occurrence, un nœud service s_1 ayant comme attributs A_{s_1} aura deux arcs entrants avec in_0 comme attribut du premier arc, et in_1 comme attribut du deuxième arc. L'arc sortant aura comme attribut la valeur out_0.

Temporaires (Temp.), notés t : ces nœuds sont des nœuds temporaires dans le flux de services à exécuter. Ces nœuds vont permettre d'indiquer au système ou à ce dernier de déterminer qu'il est nécessaire de remplacer le nœud t par un service ou un ensemble de services qui vont remplir la fonction voulue. Le nœud t possède un ensemble d'attributs $A_t = \{in_0..in_n, out_0..out_m, done\}, done \in \{true, false\}$. L'attribut *done* permet à la grammaire de graphe de stocker l'information lorsque le nœud temporaire a été traité au moins une fois dans un cycle de transformation.

Impasse (Deadend), notés *deadend* : ces nœuds représentent une impossibilité de résoudre une requête de nœud temp. et indique à l'exécuteur que cette branche dans le plan d'exécution est une impasse à cet instant donné.

Opérateur, noté o : ces nœuds servent d'opérateurs entre les différents éléments du graphe afin de structurer le plan d'exécution. Ces opérateurs s'inspirent notamment des opérateurs définis et utilisés dans les langages BPMN et BPEL (*cf.* Section 1.2.1 pour plus de détails). Les nœuds O ont un attribut noté $A_o = type$, $type \in \{seq, or, par, join, undef\}$. Cet attribut permet d'indiquer le type d'opérateur (par défaut ce type est instancié sur indéfini : $undef$). En effet, les services peuvent être exécutés de différentes manières :

- *Exécution séquentielle :* cette exécution entre deux services est représentée par le type *seq*. Ce type d'opérateur représente l'action d'exécuteur deux services de manière séquentielle, l'un après l'autre. Cet opérateur est utilisé quand il n'y a pas d'autres options possibles que d'exécuter les services spécifiés dans un ordre déterminé.
- *Exécution en parallèle :* cette exécution particulière est représentée par le type *par*. Ce type d'opérateur représente l'action d'exécuter en parallèle deux branches ou plus, c'est-à-dire simultanément de manière indépendante. Généralement, cet opérateur particulier va de pair avec l'opérateur de type *join* qui permet de représenter la jointure des branches de l'opérateur *par*, c'est-à-dire qu'il va représenter le fait que l'exécution doit attendre à ce point la finalisation de l'exécution de toutes les branches entrantes sur l'opérateur *join*.
- *Choix (OR) :* cet opérateur permet de modéliser la possibilité de choix lors de l'exécution lorsqu'il y a plusieurs possibilités qui permettent de répondre au besoin via différentes exécutions possibles. Cet opérateur permet également d'indiquer la possibilité d'avoir recours à différents services pour répondre à un besoin identique dans le cas où il y a plusieurs actions identiques à traiter. Par exemple dans le cas où il y a n données à traiter de manière identique, l'exécution peut être parallélisée en se fiant aux branches *or*, chaque branche étant indépendante.

Soient les ensembles suivants :

- E_S l'ensemble des services dans le graphe considéré
- E_T l'ensemble des nœuds temporaires dans le graphe considéré
- E_O l'ensemble des nœuds opérateurs dans le graphe considéré

- A_s l'ensemble des attributs du service $s \in E_S$, avec I_s l'ensemble des entrées (input) de s tel que $I_s \subset A_s$ et O_s l'ensemble des sorties (output) de s tel que $O_s \subset A_s$
- A_t, l'ensemble des attributs du nœud temporaire $t \in E_T$, avec I_t l'ensemble des entrées (input) attendues de t tel que $I_t \subset A_t$ et O_t l'ensemble des sorties (output) attendues de t tel que $O_t \subset A_t$

3.1.3 Intégration de besoins et de services à la volée

Suite aux différentes décisions prises par l'*analyseur*, il est possible que d'autres actions soient nécessaires : il est possible que des besoins dynamiques soient insérés dans la graphe.

Le système va ensuite traiter le·s nœud·s temporaire·s généré·s, *i.e.* utiliser les services à disposition pour le·s remplacer par des instances concrètes de services en les articulant correctement à l'aide d'opérateurs (*seq*, *or*, *par*, *etc.*).

Afin d'être capable de réaliser cette opération, un ensemble de règles de transformation a été établi. L'ensemble de services à disposition est déterminé grâce à la *base de connaissance* du système autonomique.

3.1.3.1 Graphe type général

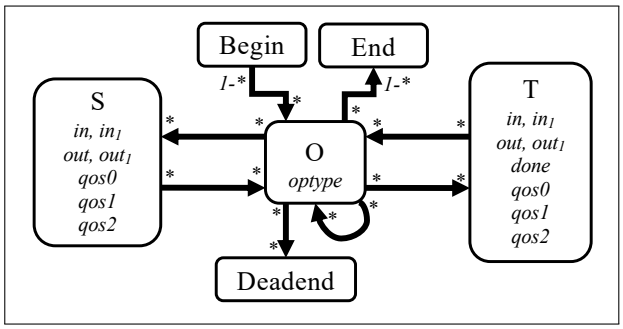


FIGURE 3.2 – Graphe type de la grammaire de graphe

Le graphe type RFC décrit précédemment est présenté en Figure 3.2. Ce graphe définit les différents types de nœuds en jeu dans la grammaire de graphe, leurs attributs, ainsi que la cardinalité des arcs possibles dans le graphe. Ce dernier intègre la possibilité d'avoir plus d'un flux de données entrant / sortant des services et des nœuds temporaires. Les opérateurs vont structurer le graphe en n'ayant des liens que vers des opérateurs pour faciliter la transformation notamment.

3.1.3.2 Règles de transformation générales

Les règles de transformation du graphe RFC vont suivre différents principes généraux en fonction de différents critères :

Différents cas à considérer

- Un service correspond parfaitement au besoin :

$$\exists t \in E_T, \exists s \in E_S \ / \ (I_s = I_t) \wedge (O_s = O_t) \tag{3.1}$$

- Un service correspond unilatéralement en terme d'entrées (entrées correspondantes mais aucune sortie) :

$$\exists t \in E_T, \exists s \in E_S \ / \ (I_s = I_t) \wedge (O_s \cap O_t = \emptyset) \tag{3.2}$$

- Un service correspondant unilatéralement en terme de sorties (sorties correspondantes mais aucune entrée) :

$$\exists t \in E_T, \exists s \in E_S \ / \ (I_s \cap I_t = \emptyset) \wedge (O_s = O_t) \tag{3.3}$$

- Un service correspondant partiellement unilatéralement en terme d'entrées (une ou plusieurs entrées parmi les requises correspondent, mais aucune sortie) :

$$\exists t \in E_T, \exists s \in E_S \ / \ (I_s \neq I_t) \wedge (I_s \cap I_t = I' \neq \emptyset) \wedge (O_s \cap O_t = \emptyset) \tag{3.4}$$

- Un service correspondant partiellement unilatéralement en terme de sorties (une ou plusieurs sorties parmi les requises correspondent, mais aucune entrée) :

$$\exists t \in E_T, \exists s \in E_S \ / \ (I_s \cap I_t = \emptyset) \wedge (O_s \neq O_t) \wedge (O_s \cap O_t \neq \emptyset) \tag{3.5}$$

- Un service correspondant partiellement (une ou plusieurs entrées et sorties parmi les requises correspondent) :

$$\exists t \in E_T, \exists s \in E_S \ / \ (I_s \neq I_t) \wedge (I_s \cap I_t = I' \neq \emptyset) \wedge (O_s \neq O_t) \wedge (O_s \cap O_t = O' \neq \emptyset) \tag{3.6}$$

- Aucun service disponible ne correspond même partiellement :

$$\exists t \in E_T, \forall s \in E_S \ / \ (I_s \cap I_t = \emptyset) \wedge (O_s \cap O_t = \emptyset) \tag{3.7}$$

Ces différents cas vont permettre de discriminer différents comportement que la grammaire de graphe devra avoir. La complexité des règles correspondantes dépendra notamment de la taille des ensembles I_t, O_t, I_s et O_s. En effet, plus il y aura d'entrée/sortie pour un nœud temporaire ou pour les services, plus les règles seront complexes à mettre en œuvre afin de gérer tous les cas possibles.

Comportement haut-niveau de la grammaire de graphes Le comportement haut-niveau de la grammaire de graphe est spécifié ci-dessous pour chaque cas :

- Cas 1 :

$$3.1 \ \Rightarrow \ s \equiv t$$

Pour un nœud temporaire, s'il existe un service disponible tel que le service possède les entrées et sorties attendues (input / output), alors ce service est inséré dans le plan entre les opérateurs encadrant le nœud temporaire et le nœud temporaire est marqué comme traité au moins une fois

- Cas 2 :

$$3.2 \Rightarrow s + t' \equiv t \ / \ (I_{t'} = O_s) \wedge (O_{t'} = O_t)$$

Pour un nœud temporaire, s'il existe un service disponible tel que le service possède les entrées attendues mais que les sorties du service ne correspondent à aucune du nœud temporaire, alors le service est équivalent au nœud temporaire si l'on ajoute un nouveau nœud temporaire possédant un ensemble d'entrées égal à l'ensemble de sorties du service, et l'ensemble de sorties égal à l'ensemble de sorties attendues du nœud temporaire traité. Cela va se traduire par l'insertion du service suivi d'un opérateur séquentiel puis du nouveau nœud temporaire entre les opérateurs encadrant le nœud temporaire traité.

- Cas 3 :

$$3.3 \Rightarrow t' + s \equiv t \ / \ (I_{t'} = I_t) \wedge (O_{t'} = I_s)$$

De manière équivalente, pour un nœud temporaire, s'il existe un service disponible tel que le service possède les sorties attendues mais que les entrées du service ne correspondent à aucune du nœud temporaire, alors le service est équivalent au nœud temporaire si l'on ajoute un nouveau nœud temporaire possédant un ensemble d'entrées égal à l'ensemble d'entrées du nœud temporaire traité, et l'ensemble de sorties égal à l'ensemble d'entrées attendues du service considéré. Cela va se traduire par l'insertion du nouveau nœud temporaire suivi d'un opérateur séquentiel puis du service, entre les opérateurs encadrant le nœud temporaire traité.

- Cas 4 :

$$3.4 \Rightarrow t \equiv s + t' \ / \ (I_{t'} = I_t - I' + O_s) \wedge (O_{t'} = O_t)$$

Pour un nœud temp. t s'il existe un service s possédant une ou plusieurs des entrées attendues du nœud temp., alors il est possible d'insérer ce service dans le graphe en ajoutant un deuxième nœud temp. t' à la suite du service et qui aura comme caractéristiques les mêmes entrées que le nœud temp. t en remplaçant les entrées du service par ses sorties. Les sorties attendues de t' demeurent identiques à t.

- Cas 5 :

$$3.5 \Rightarrow t \equiv t' + s \ / \ (I_{t'} = I_t) \wedge (O_{t'} = O_t - O_s + I_s)$$

Pour un nœud temp. t s'il existe un service s possédant une ou plusieurs des sorties attendues du nœud temp., alors il est possible d'insérer ce service dans le graphe en ajoutant un deuxième nœud temp. t' qui aura comme caractéristiques les mêmes sorties que le nœud temp. t en remplaçant les sorties

du service présentes par ses sorties. Les sorties attendues de t' demeurent identiques à t.

- Cas 6 :

$$3.6 \Rightarrow (s) \times (t') \equiv t \;/\; (I_{t'} = I_t - I_s) \wedge (O_{t'} = O_t - O_s)$$

Pour un nœud temp. t, s'il existe un service s qui possède une ou plusieurs des entrées et des sorties requises, alors il est possible de répondre au besoin par l'exécution en parallèle (opérateur de type *par* représenté par $\times$) dudit service s et d'un nouveau nœud temporaire t' ayant en entrée et sortie les attributs qui ne sont pas en commun.

- Cas 7 :

$$3.7 \Rightarrow t \equiv deadend$$

Si pour un nœud temp. t il n'existe aucun service ayant au moins une entrée ou une sortie en commun avec t, alors le nœud temporaire est considéré comme un nœud mort qu'il n'est pas possible de traiter actuellement. Le nœud t est alors remplacé dans le graphe par un nœud *deadend* pour signifier qu'il n'est pas possible de répondre à ce besoin précis à cet instant.

Le comportement haut-niveau attendu est donc spécifié. Maintenant, ce comportement haut-niveau va pouvoir être traduit par un ensemble de règles de transformation de graphe. Les règles instanciées dans notre cas d'utilisation sont spécifiées ci-dessous.

Grammaire de graphe et règles de réécriture Par la suite, les règles de transformation présentées sont représentées de la manière suivante comme défini dans la Section 1.5.1.2.

Le mode d'applicabilité des règles est défini par priorité d'application de règles. Cela permet de différencier les règles par leur priorité et ainsi d'appliquer en priorité certaines règles avant d'autres quand cela est possible. Ainsi, les règles les plus prioritaires seront appliquées dès que possible puis les autres jusqu'à ce que plus aucune règle ne soit applicable alors la transformation se termine.

3.1.3.3 Règles de transformation simples

Dans un premier temps afin de considérer un problème simplifié, l'hypothèse suivante est posée : *les règles élaborées ne vont considérer que les services ne possédant qu'une seule entrée et qu'une seule sortie. Cela implique :* $\forall s \in E_s, |I_s| = 1, |O_s| = 1$ et $\forall t \in E_t, |I_t| = 1, |O_t| = 1$.

Dans cette hypothèse particulière, les équations 3.2 et 3.4 sont identiques ainsi que 3.3 et 3.5 car si $|I_s| = |I_t| = 1$ alors $I_s \subset I_t \Rightarrow I_s = I_t$. Il en va de même pour O_s et O_t. Dans ce contexte, le cas spécifié en 3.6 est également similaire aux cas 3.3 ou 3.5 suivant la correspondance rencontrée.

De plus, on considère que tous les services ont une entrée et une sortie. Pour la sortie il peut s'agir d'un acquittement dans le cas d'une action à effectuer par

exemple. Pour l'entrée il peut s'agir de l'état à atteindre ou de paramètres d'une action à effectuer, d'un contenu à diffuser, etc. Dans le cas d'une récupération de donnée, usuellement il y aura une requête particulière à effectuer qui permettra de récupérer la donnée souhaitée dans un formalisme particulier.

Conditions de non applicabilité Afin de clarifier les schémas, les conditions de non applicabilité des règles 1 à 3 sont regroupées en Figure 3.3. Le principe des conditions de non applicabilité des règles 1 à 3 a pour objectif d'éviter d'insérer de multiples fois le même service (au même endroit notamment). Un critère permettant de discriminer le cas où le service a déjà été inséré est l'existence de lien·s avec des opérateurs. En effet, un service ou un nœud temp. va toujours être relié à un opérateur dès lors qu'il est inséré dans le plan (même si l'opérateur est juste de type séquentiel afin de structurer le plan d'exécution).

Les *NAC* sont présentées en Figure 3.3 et sont appliquées aux règles 1 à 3.

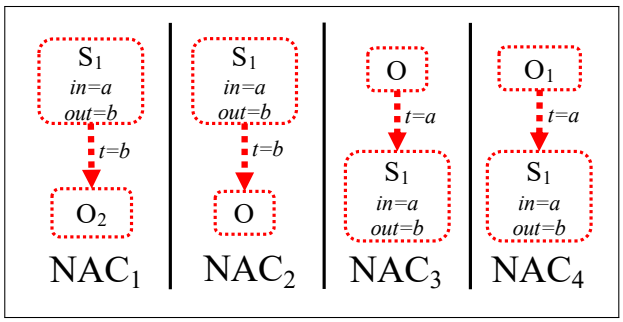


FIGURE 3.3 – *NAC* : Conditions ne non applicabilité des règles 1 à 3

Cas (1) **: règle** 1_a Dans ce cas, étant donné que le système va considérer le service comme étant adéquat et ne nécessitant aucun autre traitement, il va être inséré directement dans le graphe. La règle correspondante est représentée en Figure 3.4

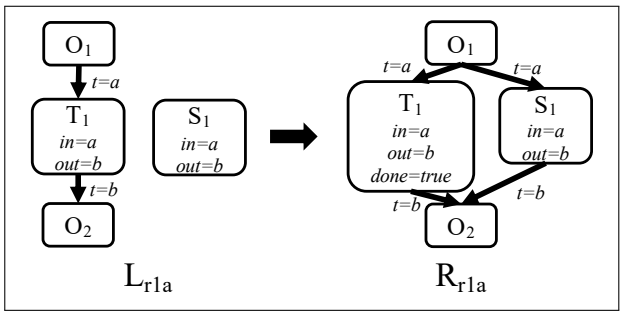


FIGURE 3.4 – Règle 1_a : insertion d'un service dont les entrées et les sorties correspondent

Le principe est d'insérer le service comme un choix équivalent au nœud temp. présent dans le graphe. Le nœud temp. est marqué comme traité au moins une fois

afin de le retirer du graphe une fois les règles appliquées. Cependant, le nœud temp. n'est pas immédiatement retiré du graphe : cela va permettre d'insérer d'autres possibilités s'il y en a et d'effectuer certaines optimisations si possible.

Cette règle fait partie des règles à plus haute priorité d'application, *i.e.* qu'elle sera appliquée d'abord si cela est possible avant d'appliquer d'autres règles. Cette propriété peut être reconfigurée en fonction du comportement souhaité de la grammaire de graphes, cependant une priorité élevée sur cette règle assure l'insertion de services qui correspondent à un nœud temp. dès que cela est possible.

Cas (2)**,** (4) **et** (6) **: règle** 2_a Dans ce cas, s'il y a des services qui répondent partiellement au besoin en terme d'entrées, le système va considérer que ce dernier est équivalent à un nœud temp. t par l'insertion d'un nouveau nœud temp. t' qui devra être traité par la grammaire. L'enchaînement entre le service et le nouveau nœud temporaire est considéré comme séquentiel par défaut mais pourra évoluer en fonction du traitement du nœud temp. t'. Cette règle est présentée dans la Figure 3.12.

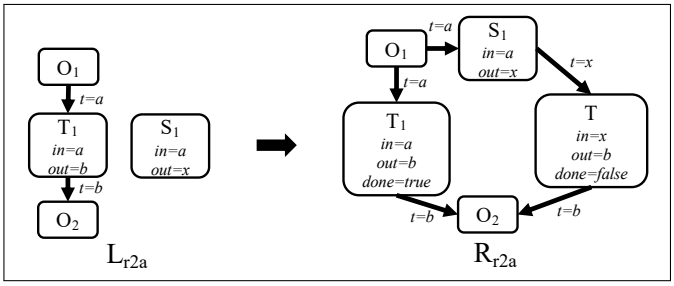


FIGURE 3.5 – Règle 2_a : insertion d'un service dont l'entrée correspond mais pas la sortie

Cette règle va être configurée avec une priorité intermédiaire, inférieure à la règle 1_a de sorte que le système va privilégier les services qui correspondent parfaitement au besoin. Cependant, s'il y a des services correspondant partiellement ils seront insérés également.

Cas (3)**,** (5) **et** (6) **: règle** 3_a De même que pour le cas précédant, cette règle va permettre d'insérer dans le graphe un service en cas de correspondance partielle en terme de sortie. La règle est présentée en Figure 3.6.

Cette règle va avoir une priorité inférieure à celle de la règle 1_a. En revanche elle peut avoir la même priorité que la règle 2 ou une priorité légèrement inférieure afin de donner la priorité à la résolution des entrées d'abord. Une priorité égale à celle de la règle 2_a entraînera un choix arbitraire de services quand il y a des services qui correspondent au nœud temp. en terme d'entrées ou de sorties uniquement.

Cas (7) **: règle** 4_a Dans le cas où un nœud temp. présent initialement ou généré par la grammaire de graphe au cours de la transformation n'a pas pu être traité

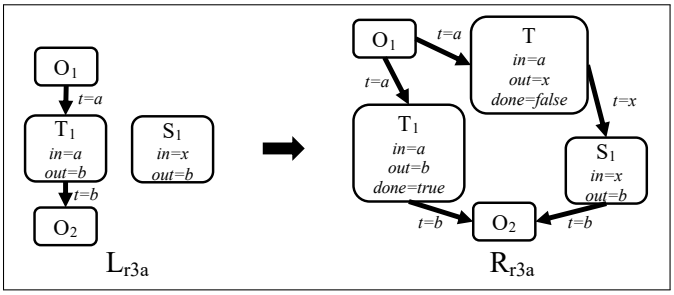


FIGURE 3.6 – Règle 3_a : insertion d'un service dont la sortie correspond mais pas l'entrée

au moins une fois, ce dernier est remplacé par un nœud représentant une impasse (*deadend*) dans le plan d'exécution. Cela signifie qu'il n'est pas possible d'exécuter cette branche à l'instant donné avec les services disponibles car il n'y aucun service permettant de traiter ce nœud temp..

Cette règle est présentée en Figure 3.7 et possède également une priorité la plus basse.

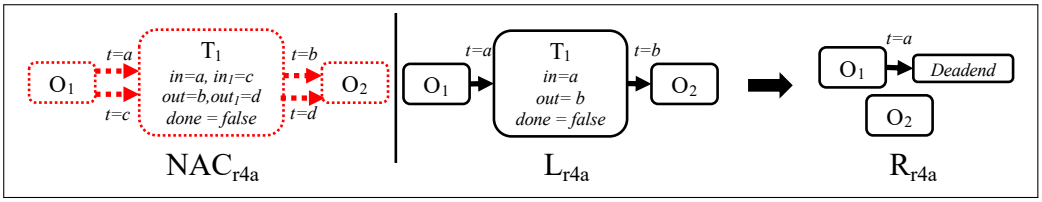


FIGURE 3.7 – Règle 4_a : Remplacement d'un nœud temporaire non traité par un nœud *deadend*

Règles supplémentaires Enfin, afin d'assurer le bon fonctionnement de la grammaire de graphes, d'autres règles sont nécessaires.

Tout d'abord, suite à une insertion de service dans le plan d'exécution, il est possible que le nouveau nœud temp. t' inséré possède les mêmes entrées que ses sorties : $I_{t'} = O_{t'}$. Cela signifie que la grammaire a réussi à connecter deux parties de plan d'exécution à l'aide de services. Dans ce cas, la grammaire va supprimer ce nœud temp. afin d'éviter d'avoir un nœud temporaire à traiter inutile. Cette action est réalisée par la règle 5_a, présentée en Figure 3.8. Cette règle va posséder une priorité d'application la plus élevée afin d'éviter le traitement de ce nœud temp. particulier lorsqu'il apparaît dans le graphe. En effet, le cas échéant, il pourrait être traité comme une correspondance unilatérale (entrée ou sortie) avec un service existant disponible et cela introduirait des étapes inutiles dans le plan d'exécution, voire pourrait conduire à un plan d'exécution non exécutable là où une solution potentielle a déjà été trouvée.

La règle 5_a va se compléter par une autre qui va permettre de fusionner des opérateurs séquentiels. En effet, la suppression d'un nœud temp. peut potentielle-

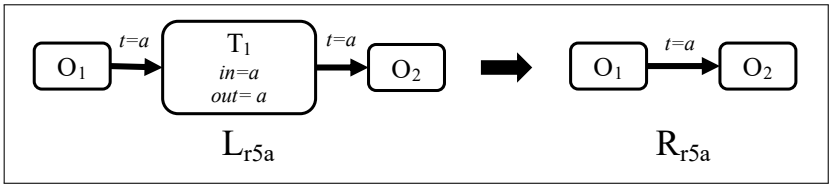


FIGURE 3.8 – Règle 5_a : Reconnexion de pans de plan d'exécution en supprimant un nœud temporaire avec entrée et sortie identiques

ment entraîner une succession de deux opérateurs sans aucun autre branchement. Dans certains cas particuliers, il est cependant possible que les opérateurs concernés soient impliqués dans d'autres branches du plan d'exécution et il ne faut donc pas les supprimer. Cela se traduit par des conditions de non applicabilité de la règle dans la grammaire de graphes. La règle et ses conditions de non applicabilité sont présentées en Figure 3.9.

En effet, il est possible qu'il y ait plusieurs opérateurs successifs dans un plan d'exécution notamment par exemple des opérateurs encadrant un choix, suivis d'un opérateur de jointure pour clore un ensemble de branches à exécuter en parallèle, ou bien un opérateur de choix permettant de séparer les différents "chemins" d'exécution possibles.

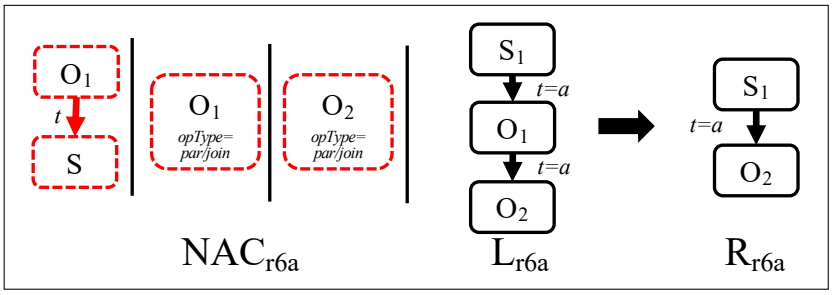


FIGURE 3.9 – Règle 6_a : Suppression d'un opérateur inutile (suite à une suppression de nœud temporaire)

De plus, les nœuds temp. vont être supprimés lorsqu'ils ont pu être traités et cela se traduit par la règle présentée en Figure 3.10. On notera que les opérateurs encadrant un nœud temp. traité ne sont pas altérés : en effet, il peut s'agir de nœud départ de différentes intersections de choix possibles d'exécution ou encore le regroupement de différentes options équivalentes.

Cette règle va avoir une priorité la plus basse afin de s'appliquer uniquement lorsque toutes les autres règles ne peuvent plus être appliquées.

3.1.3.4 Règles de transformation plus complexes

Afin de complexifier le problème étudié précédemment, une nouvelle hypothèse de travail est posée : *les règles élaborées vont considérer les services ne possédant qu'une seule entrée et qu'une seule sortie ou deux entrées et deux sorties. Cela*

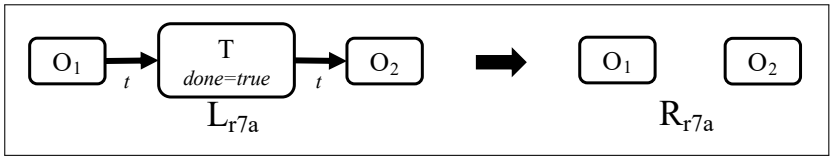


FIGURE 3.10 – Règle 7_a : Suppression d'un nœud temporaire qui a été traité au moins une fois

implique : $\forall s \in E_s, |I_s| \leq 2, |O_s| = |I_s|$ et $\forall t \in E_t, |I_t| \leq 2, |O_t| = |I_s|$. Le cas où le nombre d'entrée(s) est différent du nombre de sortie(s) n'est pas traité dans ces règles. Cependant, par extension des règles présentées dans la section précédente et celle-ci, ce cas peut être traité.

Nouvelles conditions de non applicabilité (*NAC*) Afin de traiter les cas avec deux entrées et deux sorties, les conditions de non applicabilité présentées en Figure 3.3 sont appliquées également mais dans le cas de services avec deux entrées et deux sorties.

Cas (1) **: Règle** 1_b Dans le cas où un service correspond parfaitement au besoin, le comportement va être similaire à la règle 1 vue précédemment. Le service va être inséré dans le plan et le nœud temp. considéré va être marqué comme traité. La règle 1.b est présentée en Figure 3.11.

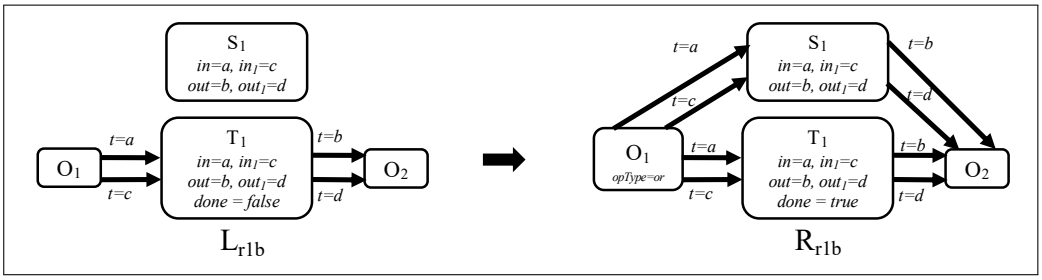


FIGURE 3.11 – Règle 1_b : insertion d'un service dont les entrées et les sorties correspondent au besoin

Cette règle va être de même priorité que la règle 1_a afin qu'elle soit appliquée dès que possible. À la différence de précédemment, on distingue ici les deux arcs entrants et sortants du nœud temp. et du nœud service après son insertion étant donné qu'il y a deux contenus différents en entrée et en sortie.

En effet, ces contenus pourraient être séparés suivant la construction du plan et les services disponibles.

Cas (2) **: règle** 2_b Dans ce cas, en plus de la règle 2_b, une nouvelle règle est nécessaire. Cette règle permet d'insérer dans le plan un service correspondant en terme d'entrées attendues. Cela va se traduire par l'insertion également d'un nouveau

nœud temp. qui va traduire un nouveau besoin afin d'obtenir les sorties attendues initialement. La règle 2_b est présentée en Figure 3.12.

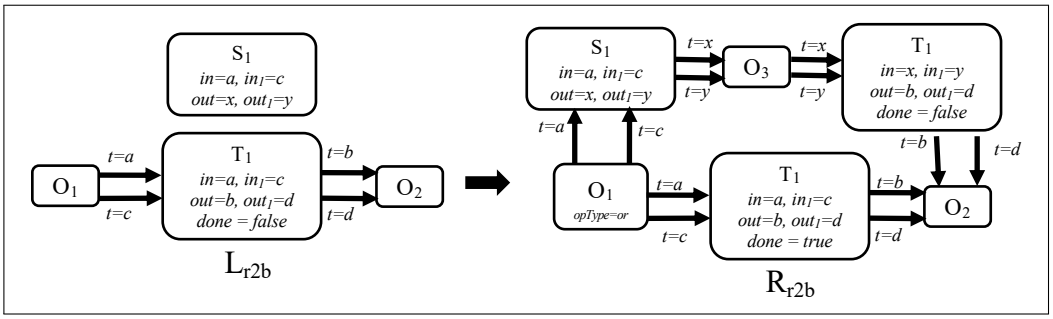


FIGURE 3.12 – Règle 2_b : insertion d'un service dont les entrées correspondent au besoin

Cette règle va avoir une priorité équivalente à la règle 2_a.

Cas (3) **: règle** 3_b**,** Dans ce cas, en plus de la règle 4_a, une nouvelle règle est nécessaire. Elle permet d'insérer dans le plan un service correspondant en terme de sorties attendues de manière similaire à la règle 2_b. Cette règle est similaire à la règle 2_b et a également une priorité équivalente à la règle 3_a et 2_b.

Cas (4) **: règle** 4_b Dans ce cas, s'il existe un service qui peut remplir partiellement le besoin en terme d'entrées, il peut être inséré dans le graphe. L'insertion va être plus complexe que pour les règles précédentes. La règle est présentée en Figure 3.13.

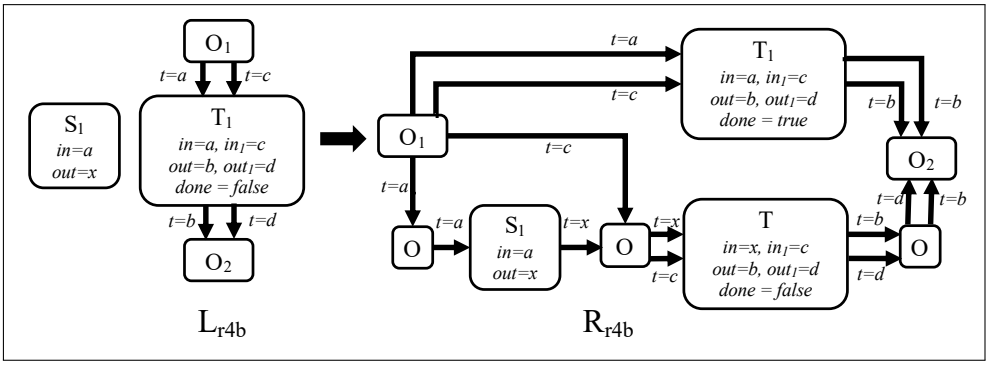


FIGURE 3.13 – Règle 4_b : insertion d'un service dont les entrées correspondent partiellement au besoin

Une autre version de cette règle est nécessaire pour traiter le cas où l'autre entrée du nœud temp. correspond à l'entrée du service. (Cette variante n'est pas schématisée car similaire).

Ces deux règles vont être de priorité intermédiaire afin de privilégier le traitement de nœuds où les services correspondent mieux au besoin.

Cas (5) **: Règle** 5_b De manière équivalente aux règles 4_b, des règles (5_b et $5'_b$) permettent de traiter le cas où les sorties correspondent partiellement au besoin mais pas les entrées. Ces règles sont similaires à la règle 4_b.

Ces règles vont également être de priorité intermédiaire afin de privilégier l'application de règles avec une meilleure correspondance.

Cas (6) **: Règles** $6_b, 7_b, 8_b, 9_b$ Dans ce cas là différentes possibilités doivent être considérés de par les combinaisons possible d'entrées / sorties. La règle 6_b et ses variations traitent ces différents cas. L'insertion dans le plan est plus complexe que pour les règles précédentes car en présence de services à une entrée et une sortie, il va être possible d'obtenir un comportement équivalent au besoin de par l'exécution en parallèle de services qui vont chacun remplir une partie du besoin et être complétés par l'insertion d'un nouveau nœud temp.

La règle 6_b est présentée en Figure 3.14. (Les variantes ne sont pas représentées car de simple variations en terme d'entrée / sortie des services de par la combinatoire possible.)

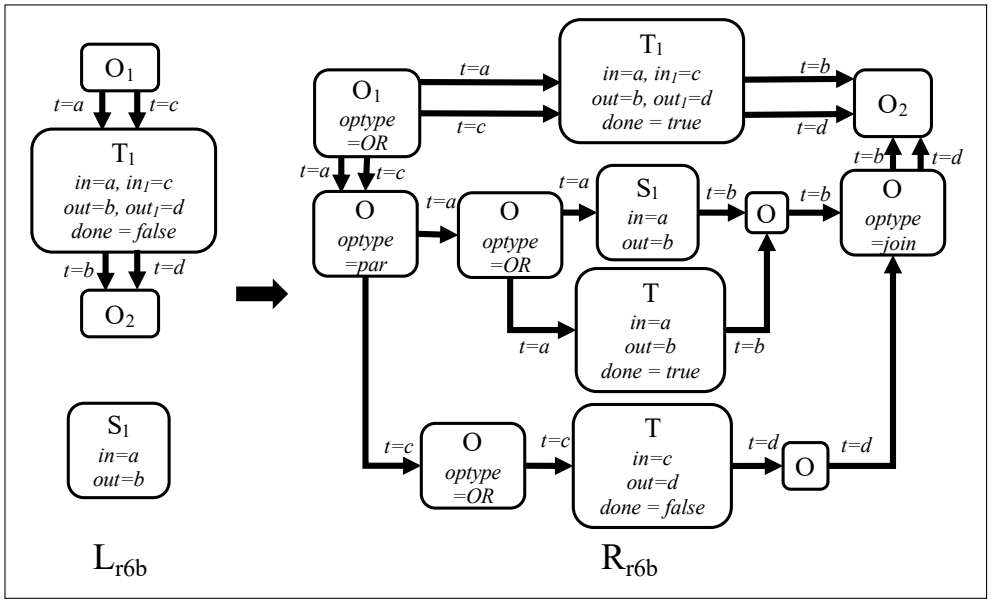


FIGURE 3.14 – Règle 6_b : insertion d'un service dont les entrées et les sorties correspondent partiellement

Le principe des règles consiste à introduire un choix alternatif au nœud temp. t considéré par l'insertion d'un opérateur parallèle (*par*) qui sépare les entrées et sorties attendues sur deux branches. Dans une des deux branches générées le service considéré va être inséré, et dans l'autre branche un nœud temp. t' complémentaire sera introduit et à traiter par la grammaire. Un nœud temp. t'' équivalent au service introduit est également inséré dans le graphe afin de permettre de trouver des alternatives à ce services si possible. Cependant, ce nœud temp. t'' est déjà marqué comme traité au moins une fois car il y a déjà un service correspondant dans le

plan.

Ces règles vont posséder une priorité similaire aux règles traitant un cas identique dans la section précédente : les règles 2_a et 3_a.

Cas (7) **: règle** 9_b Dans ce cas-là, le comportement de la grammaire de graphe va être similaire au comportement précédemment présenté avec la règle 4_a en s'adaptant aux nouveaux nœuds temp. pouvant posséder deux entrées et deux sorties.

Cette règle possède la priorité la plus basse afin de ne l'appliquer qu'une fois que toutes les autres règles l'ont été.

Cette règle peut être complétée par la règle $9'_b$ qui va permettre de "remonter" le nœud *deadend* le plus haut possible dans la branche concernée. Cette règle est présentée en Annexe B.3, Figure B.2. (Une variante est nécessaire pour traiter les cas avec multiples entrées et sorties. Étant identique elle n'est pas schématisée.)

Les règles 4_a et $4'_a$ vont également recevoir une nouvelle (NAC) afin d'éviter son application en présence d'un nœud avec deux entrées et deux sorties. En effet, cela entraînerait une mauvaise connexion des arcs en cas d'application de la règle sans considérer les arcs multiples. Une alternative est de paramétrer la priorité des règles concernées pour qu'elles s'appliquent systématiquement après la règle 9_b.

Règles supplémentaires pour le bon fonctionnement de la grammaire :

Suppression de nœud temporaire inutile : Ainsi, de manière similaire à la règle 5_a, la règle 10_b permet de retirer du graphe des nœuds temp. dont les entrées sont identiques aux sorties. Ces nœuds peuvent être insérés lors de l'insertion de services dans le graphe et ne doivent pas être transformés. La règle est présentée en Annexe B.3, Figures B.3. (Une variante de la règle est nécessaire pour traiter tous les cas combinatoires d'entrée et sortie identique. Cette règle étant similaire, elle n'est pas schématisée ici.)

Suppression de chaîne d'opérateurs séquentiels : En effet, suite à la suppression de nœuds temp. (traités ou inutiles), il est possible d'avoir plusieurs opérateurs (séquentiels) qui sont à la suite dans la même branche du plan d'exécution. De fait, il est possible de supprimer l'opérateur inutile. Cette règle est présentée en Annexe B.3, Figure B.4. Cependant, il est faut éviter la suppression de l'opérateur si d'autres branches en sortent ou arrivent sur ce dernier.

Exemple 5 ***Exemple de transformation de graphe***
En reprenant le principe présenté en Figure 3.1, un exemple de transformation de graphe est présenté dans la Figure 3.15.

L'exemple 5 reprend le principe qui a été présenté en Figure 3.1 mais en utilisant le modèle de grammaire de graphes incluant les besoin temp. et les opérateurs. Le graphe serait alors instancié avec un service S_1 permettant de récupérer la position de l'utilisateur (ce service peut consister à récupérer la position partagée par l'utilisateur) et un service S_2 permettant de transmettre l'information à l'utilisateur.

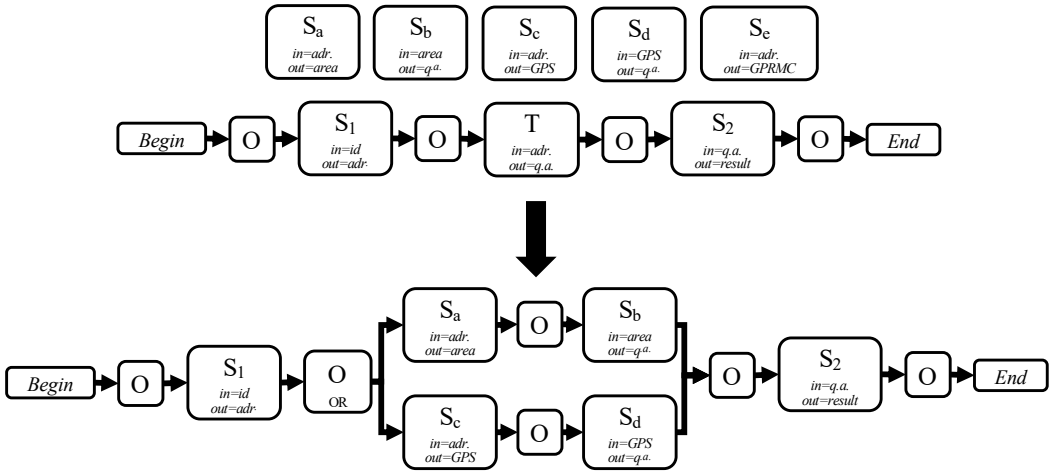


FIGURE 3.15 – Exemple de graphe avec besoin et de transformation

Un nœud temporaire est généré et inséré dans le graphe. Ce nœud spécifie que l'information disponible sera l'adresse donnée par l'utilisateur et qu'une donnée de qualité de l'air est attendue. Différents services sont déterminés comme disponibles par la base de connaissances et une représentation est insérée dans le graphe. S_a permet de récupérer une zone géographique dans laquelle se trouve l'adresse fournie. S_b permet de récupérer la qualité de l'air (q.a.) connue dans cette zone. S_c permet de récupérer des coordonnées GPS à partir d'une adresse et S_d permet de récupérer la valeur de qualité de l'air la plus proche. S_e retourne une position au format GPRMC à partir de l'adresse fournie.

La transformation a lieu et va déterminer que le service S_a sera utilisé pour récupérer une zone géographique dans la ville à partir de l'adresse et que deux services sont disponibles pour obtenir une donnée de qualité de l'air dans cette zone : S_b et S_d. Le plan final comporte donc les services initiaux ainsi qu'un choix possible lors de l'exécution sur le service de récupération de qualité de l'air dans la zone géographique.

Dans ce cas-là, différentes options sont envisageables dans les choix dans le graphe transformé. Ce choix peut être opportuniste ou guidé éventuellement.

En résumé

La grammaire de graphe va être capable de transformer un graphe initial composé de un ou plusieurs nœuds temporaires à transformer et instancier par des vrais services ainsi que les services à disposition dont la sélection a été effectuée par l'étape d'analyse. La grammaire instanciée dans notre cas d'utilisation permet de gérer des besoins et des services possédant une entrée et une sortie ou deux entrées et deux sorties. De manière similaire il est possible d'intégrer plus d'entrées / sorties dans le modèle par extension de la grammaire de graphes.

Une fois le graphe transformé, le système dispose d'un graphe comprenant un

ou des plans d'exécution potentiels avec éventuellement plusieurs possibilités ou contraintes d'exécution (choix / parallèle, etc.). Afin de pouvoir effectuer une exécution intelligente du plan lorsque cela est possible, un algorithme va être utilisé afin de choisir les solutions les plus avantageuses dans les solutions proposées par la grammaire de graphe.

3.2 Construction et exécution d'un plan opportuniste

De fait, la transformation de graphe va générer automatiquement des graphes qui représentent les possibilités d'exécution pour répondre au besoin demandé. Dès lors il est possible d'essayer d'exécuter le plan en l'état c'est-à-dire sans aucun autre traitement, cependant il peut être intéressant dans certains cas de travailler à optimiser les choix d'exécution.

En effet, le graphe généré regroupe différentes options d'exécution en terme de services et de choix et / ou contraintes sur ces derniers. Ce plan potentiel est composé de différents nœuds en plus des services concernés afin de déterminer les étapes d'exécution via des opérateurs que ce soit des choix possibles ou des exécutions à mener en parallèle par exemple.

L'idée est donc désormais de guider l'*exécuteur* sur les choix à effectuer notamment au niveau des branches à exécuter, des choix de branches quand il y a différentes options (opérateur *or* par exemple). De plus, il est intéressant d'être capable de minimiser ou maximiser certains paramètres suivant les objectifs et/ou contraintes qui peuvent être établis en fonction du cas d'utilisation.

3.2.1 Approche générale

L'approche générale est de guider les choix dans le plan lorsque cela est possible via l'utilisation d'un algorithme adéquat. Le graphe transformé par la grammaire de graphes va être composé de possibilités d'exécution (s'il y en a plusieurs) et il est donc possible d'optimiser l'exécution des services avant qu'elle ait lieu en explorant ce graphe. Cette optimisation se base sur les objectifs définis comme par exemple minimiser l'impact énergétique, ou des contraintes comme par exemple le plan doit être exécuté en moins de x secondes. Il est également possible d'avoir une exécution totalement opportuniste et se basant sur des choix arbitraires lors du parcours du plan d'exécution dans le cas où il n'y a pas de contraintes ni d'objectifs particuliers.

3.2.1.1 Optimisation du plan d'exécution

La structure du graphe généré par la transformation de la *RFC* possède un nœud de départ indiquant le début de l'exécution et un nœud de fin, qui indique que le plan a été exécuté jusqu'à sa terminaison de manière similaire au *BPMN*. Les opérateurs structurent le graphe en indiquant quelles seront les modalités d'exécution, s'il s'agit d'un choix de service à effectuer ou s'il faut exécuter plusieurs branches à la fois (opérateur *par*) pour parvenir à l'objectif.

De fait, la structure ici s'apparente à celle d'un graphe dans lequel on recherche le *plus court chemin* : un chemin d'exécution va être composé de la suite de services à exécuter, dans quel ordre, et s'il y en a, quelles suites en parallèle. Cependant, afin de déterminer quel est le plus court chemin dans cet ensemble de branches d'opérateurs et de services, il est nécessaire de définir un coût dans ce chemin.

3.2.1.2 Quels paramètres considérer

Afin de procéder à des optimisations dans l'exécution, il faut définir quel paramètre considérer. Les paramètres non fonctionnels (ou paramètres *QoS*) considérables sont assez nombreux. Différents exemples de paramètres considérables sont exposés dans [Guérout 2014]. Par exemple on peut considérer lors de l'exécution d'un service des paramètres liés à la performance, la sécurité (fonctionnement, données, etc.) ou même son coût que ce soit monétaire ou énergétique.

Dans notre cas nous allons nous focaliser sur trois paramètres non fonctionnels : le **temps d'exécution** (performance) ou **coût temporel**, le **coût énergétique** qui représente l'énergie consommée pour exécuter le service considéré sur l'objet ou la plateforme considéré·e et le **coût monétaire**. Le temps d'exécution peut avoir une incidence importante notamment dans des scénarios d'urgence où il est nécessaire d'avoir une réponse rapide du système. Le coût monétaire est intéressant à considérer pour minimiser ou optimiser les dépenses en fonction des contraintes dans le système et en fonction de l'utilisateur. Le coût énergétique peut avoir une incidence non négligeable dans le cas d'utilisation d'objets connectés déployés sur batteries et il est intéressant de pouvoir réduire l'impact sur le réseau d'objets si possible lorsque différents choix sont disponibles.

D'autres paramètres non fonctionnels peuvent également être intégrés dans les modèles de manière purement incrémentale, notamment dans le modèle de grammaires de graphes.

3.2.1.3 Objectifs, contraintes et coût d'exécution

Lors d'une action ou d'un ensemble d'actions à effectuer, il est intéressant de considérer l'établissement d'objectifs et / ou de contraintes. En effet, dans certains scénarios il sera préférable d'avoir comme objectif de minimiser le temps de réponse par exemple, là où dans d'autres cas cet objectif deviendra une contrainte avec un temps de réponse maximal à ne pas dépasser. Pour ce faire, la RFC doit être accompagnée d'un besoin (ou objectif) et d'une contrainte si nécessaire.

Objectif et coût unitaire d'exécution

Définition 6 ***Objectif*** *L'objectif est modélisé sous la forme d'un vecteur de n coefficients entiers permettant de déterminer quels paramètres non-fonctionnels doivent être pris en compte sur les n paramètres pris en compte dans les modèles. Un coefficient nul indique que le paramètre ne sera pas pris en compte lors de l'optimisation.*

L'objectif est défini de la forme suivante :

$$objectif = (a_0, a_1, .., a_n), a_i \in \mathbb{R}$$

Exemple 6 *Dans le cas où l'on ne veut optimiser que le premier paramètre de qos et qu'il y a seulement deux paramètres considérés dans le modèle (coût temporel et coût énergétique par exemple) on a :*

$$objectif = (1, 0)$$

Définition 7 ***Coût normalisé et coût unitaire*** *Soit $\{qos_a\}$ représentant l'ensemble des valeurs possibles du paramètre qos_a considéré. Par exemple $\{qos_t\}$va représenter l'ensemble des valeurs connues du paramètre de temps d'exécution.*

Pour un paramètre qos_{a_i} donné (paramètre a de qos considéré), la valeur normalisée $normqos_{a_i}$de ce paramètre est définie comme suit :

$$normqos_{a_i} = \frac{(qos_{a_i} - \min(\{qos_a\}))}{(\max(\{qos_a\}) - \min(\{qos_a\}))}$$

Cet objectif va permettre de calculer un coût symbolique pour les différents nœuds du graphe résultant de la transformation. Le coût unitaire d'exécution va permettre de discriminer différents nœuds du graphe comme étant des choix plus intéressants / optimaux en fonction de l'objectif d'optimisation.

Le coût unitaire d'un élément va être défini de la manière suivante avec $normqos_{a_i}$ la valeur normalisée du paramètre qos_{a_i} :

$$cout_i = \sum_{i=0}^{n} (a_i \cdot normqos_i)$$

De fait, quand un paramètre n'est pas considéré comme à être optimisé, il sera annulé par le coefficient dans le calcul de coût unitaire d'exécution.

En revanche, il est également possible via cette méthode d'optimiser plusieurs paramètres simultanément. Le prérequis pour ce faire est cependant de normaliser les différents attributs qos considérés dans le modèle puis d'ajuster les coefficients de corrélation afin de mettre une priorité plus importante sur un paramètre en particulier si besoin.

Exemple 7 ***Objectif avec deux paramètres***
Si l'on revient sur le cas où l'on ne considère que deux paramètres qos (temps et énergie) et que l'on veut calculer le coût en tenant compte des deux paramètres mais en considérant que le coût temporel est 10 fois plus coûteux que le deuxième, on a :

$$objectif = (10, 1)$$

De fait, dans le calcul de coût le deuxième paramètre de qos ne sera pas ignoré, cependant le temps d'exécution sera considéré dix fois plus coûteux pour le système que l'énergie.

Cette particularité permet de faire varier les coefficients lors du calcul de coût et ainsi de donner la priorité à certains paramètres sur d'autres tout en les considérant tous si besoin.

Contrainte Dans le cas d'une contrainte, un seuil maximal à ne pas dépasser peut être exprimé pour chaque paramètre de *qos*.

Définition 8 ***Contrainte***
Une contrainte sur un paramètre qos est définie comme un vecteur de $n \in \mathbb{N}$ valeurs seuil à ne pas dépasser au total pour chaque paramètre de qos considéré dans les modèles. On a alors :

$$contrainte_{qos} = (a'_0, a'_1, .., a'_n), a'_i \in \mathbb{R}$$

Il est également possible d'établir d'autres contraintes comme un nombre total de services à exécuter maximal. Ces contraintes spécifiques sont traitées séparément du vecteur qos.

Exemple 8 ***Contrainte à deux paramètres***
Par exemple, dans le cas où l'on considère deux paramètres de qos, (qos_a, qos_b), que qos_a représente le temps d'exécution en secondes et qos_b représente l'énergie consommée en watts et que l'exécution totale du plan ne doit pas excéder 60 secondes on a :

$$contrainte = (60, 0)$$

Ces éléments vont permettre à l'algorithme d'exploration d'optimiser le plan en fonction de l'objectif voulu grâce aux coefficients fournis et va également déterminer si le plan fourni est valide par rapport aux contraintes ou ne pourra pas toutes les respecter.

3.2.1.4 Coût d'exécution total : définition et calcul

Le coût d'exécution total d'un plan va pouvoir être estimé et calculé en fonction du type de paramètre considéré et du coût unitaire des différents services à exécuter potentiellement. Dans le cas où l'on cherche à optimiser le temps que va prendre le plan à s'exécuter, le coût considéré sera le coût cumulatif en temps du nœud de départ jusqu'à atteindre le nœud de fin. Ce coût va pouvoir être estimé lors de l'exploration du graphe en sommant le coût individuel des services d'une branche séquentielle ou en ne retenant que le plus grand coût cumulatif de plusieurs branches exécutées en parallèle. En revanche, dans le cas d'un coût qui soit monétaire ou énergétique, le calcul devra être différent : en effet, lors de l'exécution en parallèle de plusieurs branches de services, le coût monétaire / énergétique va être cumulatif entre les branches à la différence du coût temporel.

De ce fait, il est nécessaire de définir une méthode de calcul de coût en fonction du paramètre non fonctionnel considéré.

Définition 9 ***Calcul du coût d'exécution***

Soit f la fonction qui permet de calculer le coût d'exécution cumulatif / total. f aura comme arguments :
- c, le coût cumulatif déjà calculé pour l'ensemble de services et d'opérateurs sur la branche jusqu'au nœud considéré
- n, le nœud considéré qui peut être : un opérateur o avec o_t le type d'opérateur considéré $(or, seq, par, join, undef)$ ou s, un service
- qos_a, le type de paramètre considéré pour le coût (qos)

N.B. : le cas où $o_t = undef$ est considéré par défaut comme équivalent à $o_t = seq$ pour le calcul de coût.

Cas où qos_a représente un paramètre de coût temporel : $qos_a = qos_t$
f est définie de la manière suivante pour $n \in E_O$ en fonction du type o_t :
$o_t = seq$: $f(c, n, qos_a) = c$
$o_t = or$: $f(c, n, qos_a) = c$
$o_t = par$: dans ce cas particulier, c est stocké et un nouveau calcul de coût sera relancé sur chaque branche sortante du nœud par jusqu'à atteindre le $join$ associé
$o_t = join$: $f(c, n, qos_a) = \max_i^m(c_i)$ pour les m branches arrivant sur le nœud $join$, en effet dans le cas d'une exécution en parallèle le temps total d'exécution de toutes les branches sera égal au temps d'exécution le plus grand parmi toutes les branches

Cas où qos_a représente un paramètre de coût monétaire ou énergétique : $qos_a = qos_m$
f est définie de la manière suivante pour $n \in E_O$ en fonction du type o_t :
$o_t = seq \Rightarrow f(c, n, qos_a) = c$
$o_t = or \Rightarrow f(c, n, qos_a) = c$
$o_t = par \Rightarrow$ cas particulier, c est stocké et un nouveau calcul de coût sera relancé sur chaque branche sortante du nœud par jusqu'à atteindre le $join$ associé
$o_t = join \Rightarrow f(c, n, qos_a) = \sum_{i=0}^m(c_i)$ pour les m branches arrivant sur le nœud $join$, en effet en cas d'exécution en parallèle, le coût monétaire ou énergétique total est égal à la somme globale du coût de chaque branche à la différence du temps d'exécution

f est définie de la manière suivante pour $n \in E_S$ en fonction du type o_t :
$f(c, n, qos_a) = c + cout_s[qos_a]$ où $cout_s[qos_a]$ représente le coût unitaire du service s pour le paramètre qos_a.

f est définie de la manière suivante dans le cas où $n \in E_D$:
$f(c, n, qos_a) = +\infty$
Cela indique qu'il n'est pas possible d'exécuter un plan comportant un nœud d'impasse dans les nœuds à exécuter car son coût total est infini.

Exemple 9 ***Comparaison de coûts et calcul intuitif***

Un exemple de graphe simple est présenté en Figure 3.16 avec le détail de calcul de coût global d'exécution en fonction du coût unitaire de chaque service sur deux paramètres différents : QoS_0 représente le temps d'exécution et QoS_1 le coût monétaire unitaire. (Pour simplifier le schéma le type sur les arcs n'est pas indiqué).

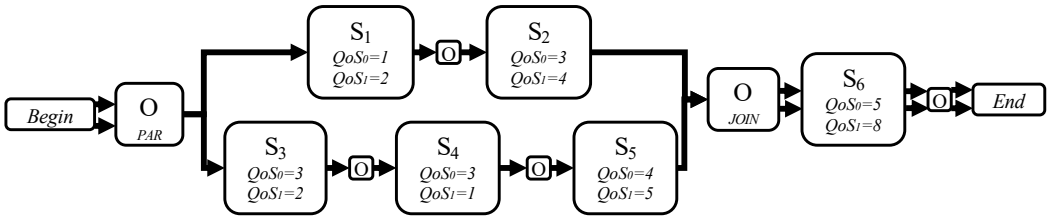


FIGURE 3.16 – Exemple de graphe incluant un nœud parallèle

Comme on peut le remarquer sur la Figure 3.16, dans cet exemple il y a un plan d'exécution composé de deux branches à exécuter en parallèle chacune composée respectivement de deux services et trois services à exécuter séquentiellement. Les paramètres non fonctionnels pour chaque service sont indiqués sur chaque nœud. Dans cet exemple, le coût global d'exécution de ce plan va donc pouvoir être calculé de la manière suivante :

Le coût temporel total est noté $cout_t$ et le coût monétaire total est noté $cout_m$. Le coût unitaire pour un paramètre qos_a d'un service est noté $cout_{s_i}[qos_a]$.

$$\begin{aligned} cout_t &= \max((\sum_{i=1}^{i=2} cout_{s_i}[qos_0]); (\sum_{i=3}^{i=5} cout_{s_i}[qos_0])) + cout_{s_6}[qos_0] \\ &= \max((1+3); (3+3+4)) + 5 \\ &= 10 + 5 \\ &= 15 \end{aligned}$$

En effet, le coût temporel d'exécution de ce plan va dépendre du temps d'exécution le plus long entre les différentes branches exécutées en parallèle.

$$\begin{aligned} cout_m &= \sum_{i=1}^{i=6} cout_{s_i}[qos_1] \\ &= 2 + 4 + 2 + 1 + 5 + 8 \\ &= 22 \end{aligned}$$

Ce coût ne dépend pas du mode d'exécution car il va se sommer de manière globale.

Concrètement cette méthode de calcul de coût va pouvoir être implémentée dans l'Algorithme 1.

3.2.2 Algorithme d'exploration et de recherche d'optimum : vers un choix optimisé

Dans le but d'effectuer un meilleur choix lors de l'exécution du plan généré par la transformation du graphe *RFC* par la grammaire de graphes, un algorithme d'exploration va explorer le graphe généré afin de proposer d'exécuter les branches les plus intéressantes suivant le cas.

3.2.2.1 Fonctionnement haut niveau

En s'inspirant d'un algorithme de recherche de plus court chemin [Dijkstra 1959], il va être possible d'explorer le graphe et de déterminer le meilleur chemin d'exécution à exécuter. Le principe est donc de démarrer d'un nœud de départ connu et d'atteindre le nœud de fin du plan en empruntant le *plus court chemin*. La particularité présente ici est que certaines branches devront être traitées de manière particulière : en effet, dans le cas où il y a un ou des opérateurs *par* dans le graphe, une recherche sera effectuée sur chaque branche sortante du nœud *par*. En effet, l'exécution de chaque branche sortante d'un nœud *par* étant obligatoire pour remplir le besoin initial, il est nécessaire de trouver un chemin optimisé pour chaque branche qui sera exécutée en parallèle des autres. Cela permettra également de calculer le coût total de l'exécution en parallèle des différentes branches. Cependant, le principe de recherche demeure similaire étant donné que le point de départ des branches parallèles est indiqué par un nœud *par* connu et la jointure (nœud de fin de l'algorithme) de ces branches est indiquée par un nœud *join*. Dans le cas de branches parallèles imbriquées dans une branche parallèle le cas sera récursivement similaire.

L'algorithme ne comporte pas d'heuristique spécifique afin de guider les choix d'exploration. En effet, étant donné que les graphes à explorer auront une profondeur et une largeur limitée de par la complexité de chaque graphe géré par le système, une recherche de chemin optimal est intéressante. Cela permet d'avoir la certitude que l'algorithme a trouvé le meilleur plan parmi les différentes possibilités sans avoir un impact fort en terme d'exécution de l'algorithme comme on pourra le constater dans le Chapitre 4.

3.2.2.2 Algorithme haut-niveau

Le graphe généré par la grammaire de graphe est noté $G = (N, A)$ où :

- l'ensemble N est l'ensemble fini des nœuds (sommets) de G
- l'ensemble A est l'ensemble des arcs de G tel que si $(n_1, n_2) \in A$, alors il existe un arc depuis le nœud $n_1 \in N$ vers le nœud $n_2 \in N$ dans G
- la fonction *cout* est définie sur $N \times N$ dans R et qui à un couple (n_1, n_2) associe un coût positif $cout(n_1, n_2)$ de l'arc reliant n_1 à n_2 (et $+\infty$ s'il n'y a pas d'arc reliant n_1 à n_2). Dans notre cas, si n_1 représente un service, le coût de l'arc sera celui d'exécution du service en question.

Le coût d'un chemin entre deux nœuds (sommets) est la somme du coût des arcs qui le composent. Un cas particulier sera à traiter en présence de branches parallèles. Le but de l'algorithme est donc de trouver un chemin d'exécution de moindre coût depuis un sommet de départ (n_{deb}) et un sommet d'arrivée (n_{fin}) appartenant à N et prédéterminés.

Pour ce faire l'algorithme fonctionne en construisant un sous-graphe P de manière à ce que le coût total entre un sommet n de P depuis n_{deb} soit connu et minimal dans G. Le coût total connu, (cumulatif depuis le nœud de départ jusqu'à un nœud n), est noté $c(n)$. Au commencement, P ne contient que le nœud de départ n_{deb} et le coût de ce nœud jusqu'à lui-même (qui est nul). A chaque étape d'exploration des arcs vont être ajoutés à P :

1. en identifiant les arcs $a_i = (n_{i1}, n_{i2})$ dans $P \times G$
2. en choisissant l'arc $a_j = (n_{i1}, n_{i2})$ dans $P \times G$ qui ait le coût minimum depuis n_{deb} à n_{j2} en passant par tous les chemins créés menant à ce nœud, le coût est noté $cout(n_{deb}, n_{fin})$.

L'algorithme prend fin quand P devient un arbre couvrant de G, ou que le plus court chemin jusqu'au nœud d'arrivée est trouvé.

L'algorithme haut-niveau peut donc s'écrire comme présenté ci-dessous dans l'Algorithme 1 :

Entrées : $G = (N, A)$ un graphe avec pondération positive *cout* des arcs / nœuds, n_{deb} et n_{fin} des sommets de N considérés comme nœud de départ et nœud de fin du chemin à trouver, connus à l'avance.

La particularité de l'algorithme par rapport à un algorithme de recherche de chemin "classique" réside dans la présence potentielle de nœuds *par* qui vont impliquer une recherche de chemin supplémentaire à effectuer sur chaque arc sortant du nœud parallèle jusqu'au nœud de jointure correspondant.

3.2.2.3 Implémentation et pseudo-code

L'implémentation et pseudo-code de l'algorithme utilisé dans le cas d'utilisation sont fournis en Annexe C. L'implémentation considère trois paramètres de *qos* différents : le temps d'exécution qos_0, le coût monétaire qos_1 et le coût énergétique qos_2. Une structure particulière est définie pour stocker le coût cumulatif calculé depuis le nœud de départ ainsi qu'un tas trié afin de récupérer à chaque itération le nœud avec le coût cumulatif moindre.

Dans l'implémentation de l'algorithme, ce dernier produit également une liste de nœuds sélectionnés parmi les nœuds du graphe. Ce sont les nœuds qui devront être exécutés. Dans le cas où aucun chemin n'a pu être obtenu, cela indique qu'il n'est pas possible d'exécuter ce plan d'exécution.

3.2.2.4 Particularité avec nœuds parallèles et de calcul temporel

L'algorithme présenté est un algorithme de recherche d'optimum. Cependant, dans le cas particulier où l'on cherche à optimiser le temps d'exécution du plan

Algorithm 1 Recherche d'optimum (meilleur plan d'exécution)

$P := \emptyset$
$\forall a \in N, c(a) := +\infty$
$c(n_{deb}) := 0$
while $\exists a \in N$ / $a \notin P$ avec le coût le plus bas depuis n_{deb} et que $a \neq n_{fin}$ **do**
  Choix de ce sommet a hors de P de plus petit coût cumulatif (distance) $c(a)$
  Mettre a dans P
  if a n'est pas un opérateur *par* **then**
    for Pour chaque sommet b hors de P voisin de a **do**
      $x := \min(c(b), c(a) + cout(a, b))$
      if $x < c(b)$ **then**
        $c(b) := x$
        Prédécesseur de $b := a$ dans un meilleur chemin
      end if
    end for
  else
    Sauvegarde de $c(a)$
    $c(join_a) := \infty$
    for Pour chaque sommet b hors de P voisin de a **do**
      Nouvelle recherche avec : $n_{deb} := b$ et $n_{fin} := join_a$
      Récupération du coût de b jusqu'à $join_a$ pour la branche explorée avec résultat de l'exploration : $c(b, join_a)$
      Mise à jour de $c(join_a)$ en fonction des paramètres considérés avec la valeur déjà connue de $c(join_a)$ (ou 0 si première exploration jusqu'au $join$) et le résultat de l'exploration de la branche : $c(join_a) := c(join_a) + cout(b, join_a)$
    end for
    $c(join_a) := c(join_a) + c(a)$
  end if
end while

généré et que le graphe comporte des branches parallèles, le plan trouvé par l'algorithme sera un optimum local mais n'est pas garanti comme étant un optimum global. En effet, la particularité liée au calcul en coût temporel avec les branches parallèles implique que pour être certain de trouver l'optimum global il faut considérer toutes les possibilités combinatoires, toutes branches confondues car le coût total d'exécution d'une branche parallèle se base sur le maximum des coûts temporels unitaires de chaque branche. Cependant, dès lors que d'autres paramètres sont considérés (coût monétaire et / ou énergétique par exemple), le résultat trouvé sera l'optimum global. En effet, étant donné que le coût sera incrémental et cumulatif entre les branches, les minimums locaux trouvés par branches constitueront un minimum global une fois cumulés.

3.2.3 Autre algorithme témoin pour tests de performance : glouton

Afin d'avoir un résultat témoin pour les tests de performance et évaluation des résultats obtenus, un choix opportuniste de services est utilisé pour des métriques de référence.

Pour ce faire, le système va suivre le plan d'exécution indiqué par le graphe transformé et dans le cas où plusieurs choix sont possibles le choix sera déterminé arbitrairement. En cas d'impasse, l'algorithme va effectuer une opération de retour en arrière (*backtracking*) afin de se repositionner au dernier choix possible (opérateur *or*) rencontré. Ensuite un choix aléatoire peut être effectué sur les branches restantes non explorées. Dans le cas d'un opérateur de type *par* si une des branches ne peut pas être exécutée tout l'ensemble à exécuter en parallèle sera considéré non exécutable, similairement à précédemment. Une fois l'exploration effectuée d'un ensemble de branches *par*, l'exploration se poursuit à partir du nœud *join* associé.

L'algorithme est présenté dans l'Algorithme 2 présenté ci-dessous.

L'algorithme glouton d'exploration en profondeur se base sur le graphe fourni, un nœud de départ d'exploration et un nœud cible à atteindre. Avant première exécution, les variables sont initialisées de la manière suivante :
$s := start$, $e := end$

Le *chemin* trouvé par cet algorithme ne sera pas nécessairement le plus optimisé suivant les paramètres de *qos* considérés, cependant il permet de trouver rapidement un chemin exécutable. Si l'algorithme ne trouve aucun chemin exécutable, alors le plan produit par la grammaire de graphe n'est pas exécutable.

Désormais, deux algorithmes différents peuvent être utilisés pour explorer le graphe produit par la grammaire de graphe et indiquer quels services exécuter parmi les différentes possibilités. Un algorithme glouton peut être utilisé afin d'obtenir un plan exécutable sans aucune contrainte ou objectif particulier, et un algorithme de recherche de plan optimisé peut être utilisé en présence de contraintes et d'objectifs notamment au niveau des différents paramètres de *qos* considérés.

Algorithm 2 Recherche gloutonne de plan exécutable par exploration en profondeur

Bool $pathFound$ **explorer**(Graphe G, nœud s, nœud e)

```
marquer s comme exploré
pathFound := false
if (s = e) then
  un chemin a été trouvé, fin de l'exploration
  pathFound := true
else if (s est un opérateur par) then
  bool temp := true
  for (tout voisin v du par) do
    temp := temp AND explorer(G, v, join)
  end for
  if (chemin possible (temp = true) then
    pathFound := explorer (G, join, e)
  else
    marquer s comme mort, fin de l'exploration
  end if
else if (s est un nœud deadend) then
  marquer s comme mort, fin de l'exploration de cette branche
else
  while pathFound = false AND il y a des voisins de s encore non explorés do
    sélection aléatoire d'un voisin v de s non marqué
    pathFound := explorer(G, v, e)
  end while
end if
renvoi de pathFound
```

3.2.4 Évaluation de résultat des algorithmes et validation des contraintes

Afin d'évaluer les résultats obtenus, une fonction d'évaluation est nécessaire. Cette dernière se base sur la sélection de nœuds a exécuter déterminée par l'algorithme d'exploration utilisé ou *chemin*.

La fonction d'évaluation a pour but de vérifier si les contraintes peuvent être exprimées et de calculer le coût total d'exécution du plan pour chaque paramètre *qos* considéré. Les entrées de la fonction d'évaluation se basent sur le graphe concerné, une liste de nœuds sélectionnés avec le nœud précédent dans l'exécution, et les contraintes à vérifier. La particularité dans notre cas est que certains nœuds (*join*) peuvent avoir plusieurs prédécesseurs dans le plan d'exécution. En effet, en présence d'un nœud *join*, ce dernier aura comme prédécesseur chaque dernier nœud de chaque branche parmi les branches à exécuter en parallèle. Cela sera modélisé par un ensemble de plusieurs listes de chemins en présence de nœuds *par* et *join*. Ainsi, un coût peut être calculé pour chaque morceau de chemin : du nœud terminal *end* jusqu'à un *join*, puis calcul sur chaque branche parallèle, puis calcul du coût total de toute la branche sur le nœud *par* puis calcul du coût entre le nœud *par* et le nœud initial *begin*.

L'algorithme d'évaluation est présenté ci-dessous dans l'Algorithme 3. Le nœud utilisé par l'algorithme d'évaluation pour démarrer l'évaluation sera le nœud *end*.

3.2.5 Exemple complet d'application des algorithmes et comparaison de résultats

Un exemple complet est maintenant présenté avec exécution des deux algorithmes sur un graphe transformé donné.

Exemple 10 ***Exemple de résultats d'exploration***
Un exemple de différence d'exploration et de chemin final choisi est présenté en Figure 3.17. Le plan final sélectionné par l'algorithme est indiqué avec des nœuds en ***bleu*** *(et des arcs épais). Les nœuds explorés sont indiqués en* ***orange*** *et cerclés (dans le cas où ils ne font pas partie du plan final). Les nœuds en* ***rouge*** *sont les autres nœuds du graphe qui n'ont pas été explorés par l'algorithme concerné. Pour des soucis de clarté, les valeurs de paramètres QoS ne sont pas affichées.*

L'exemple 10 se base sur graphe à explorer avec application des deux algorithmes d'exploration présentés précédemment. Comme on peut le constater, le plan final sélectionné par les algorithmes sont différents entre la Figure 3.17a et la Figure 3.17b. L'algorithme glouton s'est contenté de choisir le premier "chemin" valide trouvé : $begin \rightarrow S_{14} \rightarrow S_{18} \rightarrow end$. À la différence, l'algorithme de recherche d'optimum a exploré plus de nœuds du graphe et a choisi un plan différent : $begin \rightarrow S_{22} \rightarrow S_5 \rightarrow end$.

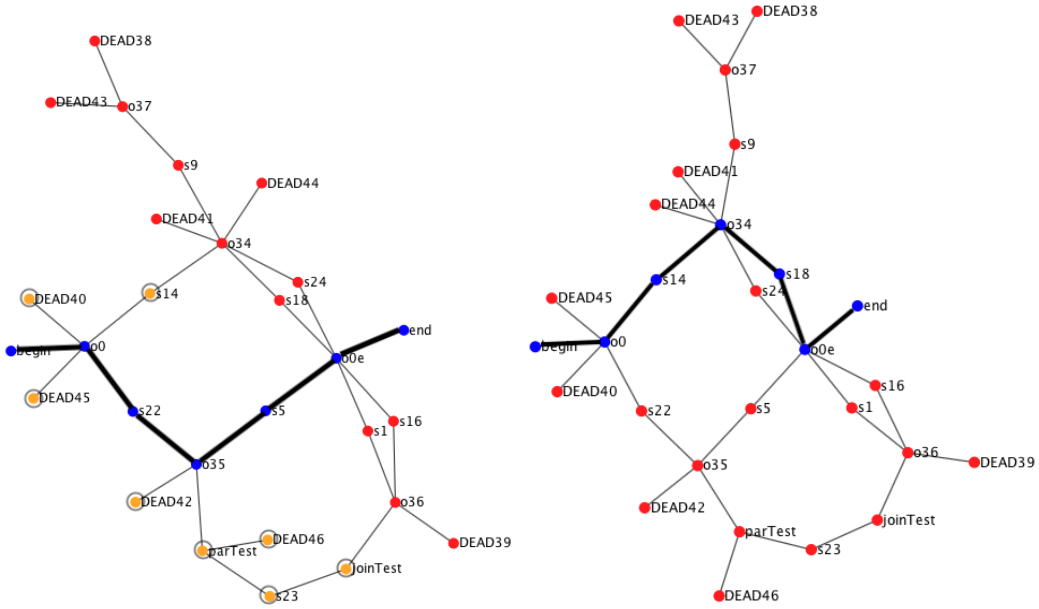
En effet, le coût final (en temps) du plan choisi par la recherche d'optimum est égal à : $cout_{opti} = cout[S_{22}] + cout[S_5] = 5 + 7 = 12$

Algorithm 3 Évaluation de plan d'exécution (*chemin*) déterminé par la transformation de graphe puis exploration

Bool *valide*, Cost coutTotal **evaluation**(Graphe G, List<List<String> > predecesseurs, contrainte (...))

totalQos := $(0, 0, 0)$
for (chaque liste de nœuds, et chaque nœud n dans *predecesseurs* **do**
  if (n est un service) **then**
    incrémentation du cout total pour chaque paramètre
  else if (n est un nœud *join*) **then**
    if (un coût total est connu du nœud *end* à n **then**
      récupération du coût connu du nœud *end* à n
    else
      stockage du coût total du nœud *end* à n
    end if
  else if (n est un nœud *par*) **then**
    if (un cout total est connu pour n) **then**
      if ($coutTotalQos[0](n) < totalQos[0]$ **then**
        $coutTotalQos[0](n) = totalQos[0]$
      end if
      $coutTotalQos[1](n) += totalQos[1]$
      $coutTotalQos[2](n) += totalQos[2]$
    end if
    stockage de $coutTotalQos(n)$
  end if
end for
le résultat est considéré valide si le total sur chaque paramètre *qos* respecte la contrainte
renvoi de $totalQos_{0..2}$

Là où le coût final du plan choisi par la recherche gloutonne est égal à : $cout_{glou} = cout[S_{14}] + cout[S_{18}] = 15 + 6 = 21$



(a) Optimum (b) Glouton

FIGURE 3.17 – Exemple de différents résultats sur un graphe donné en fonction de l'algorithme d'exploration utilisé

Conclusion

Afin d'enrichir le comportement du système pour qu'il réponde à plus de cas différents et soit capable de répondre à des besoins, intègre des paramètres QoS et des contraintes, le modèle de graphe proposé dans le chapitre précédent a été enrichi. En effet, afin de permettre au système de générer des besoins temporaires à la volée en fonction des résultats afin de compléter le comportement présenté dans le Chapitre 2, de nouveaux éléments ont été mis en œuvre pour **résoudre des besoins temporaires** par la génération de services complexes composites. Ce chapitre visait également à prendre en compte des **paramètres non fonctionnels (QoS)** pour intégrer un aspect guidage de choix de services disponibles et intégration de contraintes dans la construction et l'exécution des plans d'exécution.

Pour ce faire, un modèle de grammaire de graphe a été défini afin d'intégrer la gestion des différents types de services. De plus, des **opérateurs** structurant les plans d'exécution ont été définis, permettant la construction de plans d'exécution plus complexes en intégrant une gestion de choix de services mais aussi la possibilité d'exécuter des branches de plan en parallèle ou en séquentiel.

De plus, **deux algorithmes d'exploration** des graphes transformés ont été proposés. Le but des algorithmes est de déterminer rapidement si un plan exécutable a été trouvé et le cas échéant de les fournir au composant d'exécution. Mais aussi, un algorithme permet d'optimiser les choix d'exécution dans le cas où il y aurait plusieurs possibilités.

En effet, le premier algorithme proposé (*recherche d'optimum*) se base sur un algorithme de recherche de chemin adapté afin de pouvoir explorer le graphe transformé et guider l'exécution du plan en optimisant les choix suivant les paramètres non fonctionnels considérés. Le deuxième algorithme (*glouton*) permet de trouver rapidement une solution (s'il en existe) de manière opportuniste sans considérer les paramètres non fonctionnels. Cet algorithme sera plus adapté pour des graphes comportant peu de solutions voire une seule mais servira également de témoin pour comparer les performances entre les deux algorithmes.

Enfin, une **fonction d'évaluation** a été modélisée pour pouvoir valider ou non le plan d'exécution final produit et notamment comparer ses caractéristiques (avec l'utilisation de l'algorithme glouton).

Ces travaux ont donné lieu à une publication scientifique [1].

Dans le chapitre suivant, une présentation du prototype implémenté est réalisée ainsi qu'un ensemble d'évaluations de performance et de comportement du système dans différents cas d'utilisation du théorique à l'appliqué à notre cas d'utilisation. Notamment, une étude d'évaluation des modèles et algorithmes présentés dans ce chapitre y est réalisée.

1. *en cours de soumission*

Chapitre 4

Étude d'évaluation du système et prototype

Sommaire

4.1 Temps de réponse des composants et performance 90
4.1.1 Configuration du système 90
4.1.2 Analyse sémantique et transformation de graphe 90
4.1.3 Analyse du temps de réponse dans une gestion complexe de services 94
4.1.4 Conclusions des évaluations théoriques 99
4.2 Preuve de concept : prototype et déploiement 100
4.2.1 Eclipse OM2M : un intergiciel standard pour l'IoT 101
4.2.2 Le système IoT de prototype et démonstration 101
4.2.3 Le prototype déployé du système autonomique 104
4.2.4 Analyses et conclusions 107

Différents éléments de problématique liés au contexte IoT de ces travaux se posent comme la dynamicité et la versatilité des système IoT considérés mais aussi l'intérêt de pouvoir mutualiser des services, notamment dans un contexte de récupération / diffusion ou traitement de contenu. Pour répondre à ces différents éléments problématiques, une architecture de système de gestion autonomique de systèmes IoT orientés services a été proposée. Cette approche permet de répondre de manière autonome aux changements survenant dans le système IoT supervisé et de prendre des décisions en autonomie à partir de configurations haut-niveau. Cette architecture se base sur différents modèles (ontologie et grammaires de graphes) et sur un modèle de boucle autonomique MAPE-K.

Après avoir effectué des simulations (*cf.* exemples des chapitres précédents) sur le comportement des différents composants et des différents modèles, une étude d'évaluation de l'approche est réalisée dans ce chapitre. Dans un premier temps, une évaluation théorique est détaillée : l'objectif est de caractériser la réponse du système de manière générale avec une variation des paramètres considérés de manière non fonctionnelle. D'un côté, le temps de réponse de la partie analyse et inférence sémantique couplée à la transformation de graphe est étudié. Ensuite, la grammaire de graphe avancée et les algorithmes d'exploration sont traités afin de caractériser la réponse temporelle mais aussi la qualité des résultats produits par les algorithmes.

Dans un deuxième temps, une évaluation du comportement fonctionnel du système est réalisée. En effet, un prototype fonctionnel a été implémenté et déployé sur une architecture IoT réaliste dans un contexte de ville intelligente. Les déploiements et configurations du système sont détaillés ainsi que les scénarios réalisés accompagnés des résultats correspondants afin de valider l'approche proposée. Ce prototype a été déployé et utilisé dans le cadre de démonstrations comme preuve de concept de l'approche.

4.1 Temps de réponse des composants et performance

Des expérimentations ont été conduites afin d'étudier et de caractériser le temps de réponse des différents éléments de l'approche proposée. En effet, dans notre cas d'utilisation le système devrait être capable de répondre dans un temps raisonnable (moins d'une minute notamment) et ce afin d'éviter le cas où les entités concernées ne recevraient pas le contenu à temps. Par exemple, il faut éviter l'affichage de contenus intéressants pour certains passagers sur les terminaux d'un bus alors que les passagers ont déjà quitté le bus.

4.1.1 Configuration du système

Pour la partie sémantique de l'analyseur, (fichiers RDF et OWL, règles SWRL, inférence) différents outils ont été utilisés :

- OWLAPI (v. 4.2.7) [1]
- SWRLAPI (v. 2.0.0) [2]
- Drools engine (v. 6.5.0) [3]
- JFact reasoner (v. 4.0.4) [4]

Pour la partie orientée graphe, le moteur AGG [5] (v. 2.1) a été utilisé pour la transformation des graphes (RFC) générés par l'analyseur (ou manuellement pour certains tests).

Les mesures ont été réalisées sur un serveur Ubuntu (équipé de : Intel Xeon CPU (3 Ghz), 32 GB of RAM, OpenJDK version 1.8).

Le déploiement considéré se concentre sur la gestion d'un bus à la fois et de ses passagers. En effet, ce déploiement permet au système d'être plus rapide en réponse mais aussi de s'approcher plus d'un déploiement réaliste avec des passerelles déployées dans les bus. Les exécutions de tests n'ont pas été parallélisées.

4.1.2 Analyse sémantique et transformation de graphe

Pour réaliser les tests suivants, un ensemble de fichiers RDF a été généré. Dans ces derniers, les sujets d'intérêts de chaque personne sont établis de manière aléa-

1. https://github.com/owlcs/owlapi
2. https://github.com/protegeproject/swrlapi
3. https://github.com/protegeproject/swrlapi-drools-engine
4. http://jfact.sourceforge.net/
5. http://www.user.tu-berlin.de/o.runge/agg/

toire. Les sujets et les lieux en lien avec les contenus sont établis de manière aléatoire aussi afin d'obtenir des ensemble de données variables.

Les expériences ont été conduites avec un nombre de passagers variable (10 à 60, capacité maximale du bus), différents nombre de contenus à diffuser (10 à 100) et différents nombres de sujet d'intérêt possibles (de 10 à 50). Le nombre de lieux possibles a été fixé à 25 pour ces tests.

Les différentes données extraites de ces expériences sont présentées en Figure 4.1 et Figure 4.2. Le temps de réponse moyen y est présenté en fonction du nombre de passagers du bus (avec un nombre de sujets d'intérêt fixé dans le premier cas, et variable dans le second cas).

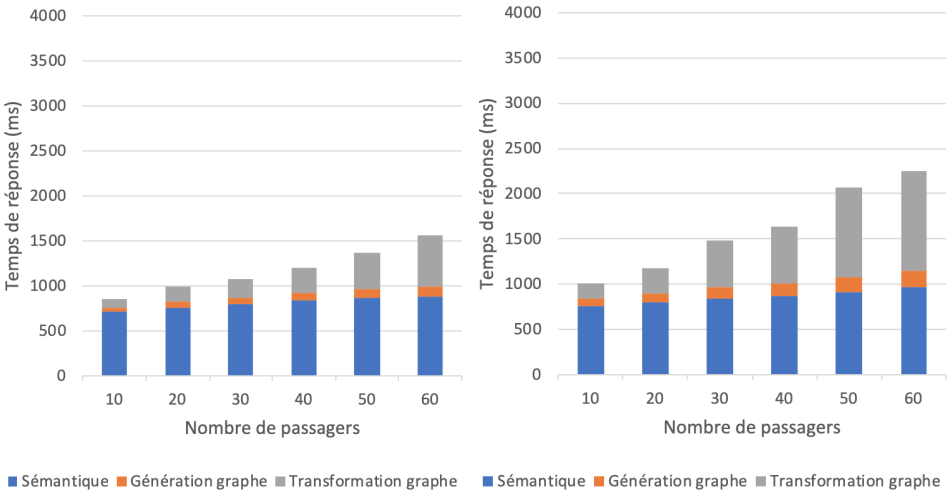
Le temps de réponse pour *l'inférence sémantique* est montrée en *bleu foncé*. Le temps de réponse de la partie *graphe* est montré en *orange* pour le temps de génération du graphe à partir de l'ontologie et en *gris* pour le temps de transformation du graphe.

4.1.2.1 Analyse du temps de réponse pour une variation de nombre de contenus à diffuser

Si l'on s'intéresse au temps de réponse de la partie purement *sémantique*, différents point sont à considérer. Tout d'abord, il faut noter que plus le nombre de contenus à gérer à la fois augmente, plus le temps de réponse augmente proportionnellement. Cela est dû à la complexité à gérer pour déterminer quel contenu devrait être diffusé pour quelles entités. En effet, plus il y a de contenus à considérer à la fois dans le système, plus les éléments liés à considérer (lieux, sujets d'intérêt) sont nombreux et vont ralentir le traitement.

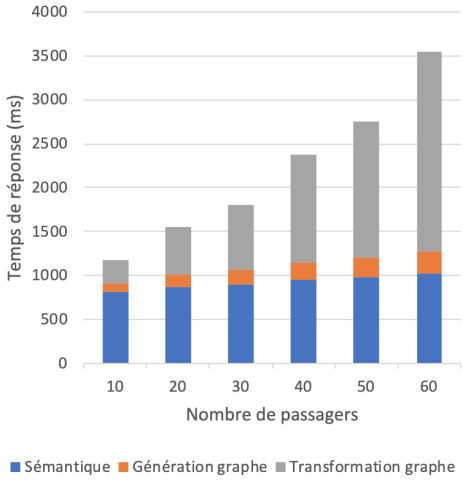
D'un autre côté, le temps de réponse de la partie *graphe* (relativement à la transformation, la génération ayant un temps presque négligeable par rapport au reste) augmente proportionnellement au nombre d'entités à gérer (ici il s'agit des passagers et du bus). En effet, pour un ensemble de contenus donné et de sujets d'intérêt liés, le temps de réponse de la transformation de graphe augmente de moins de 250 ms en moyenne (Figure 4.1a) jusqu'à un peu plus de 2 s quand il y a 100 contenus à diffuser en un seule fois et dans le pire cas (Figure 4.1c). Cela est dû au besoin d'agréger plus de contenus étant donné que potentiellement il y a plus de personnes intéressées par le même type de contenu dans ce cas précis. De plus, cette opération est prise en charge par une couche de règles qui sera répétée jusqu'à non applicabilité. De fait, plus y a de contenu à agréger plus le temps de réponse va augmenter car la règle s'appliquera plus de fois.

Le pire cas dans la Figure 4.1c prend au total jusqu'à presque 3,5 secondes pour s'exécuter avec 60 passagers considérés et 100 contenus à diffuser à la fois. Dans ce cas précis, environ 5340 axiomes sont considérés dans l'ontologie et le graphe RFC à transformer contient plus de 500 éléments. Cependant, il faut noter que dans le cas d'utilisation considéré dans cette thèse, de tels nombre ne devraient pas être atteints : la diffusion de plus de 50 contenus à la fois (en une seule itération) sur terminal d'affichage saturerait complètement les utilisateurs. Une possibilité serait de



(a) 10 contenus

(b) 50 contenus



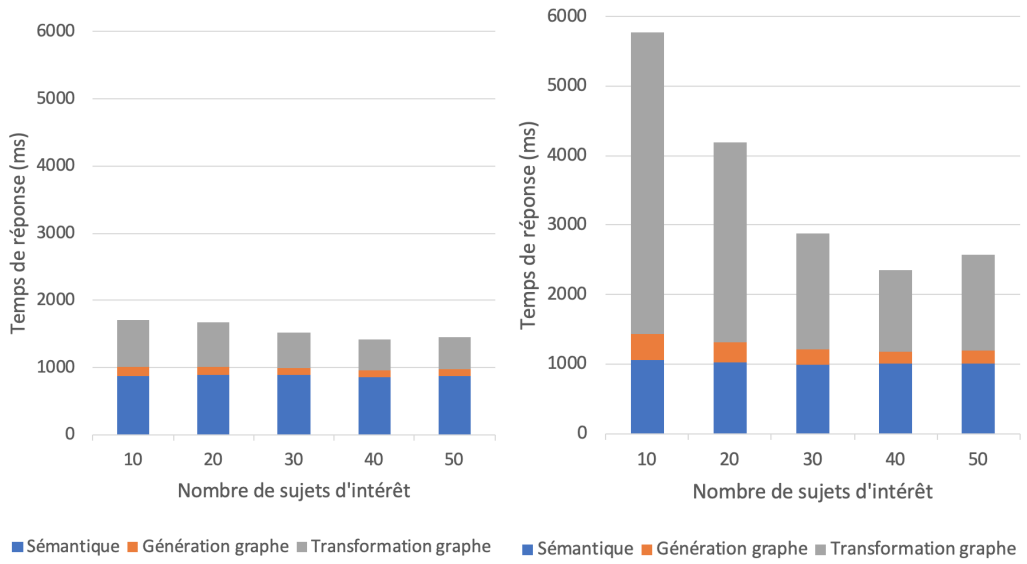
(c) 100 contenus

FIGURE 4.1 – Temps de réponse moyen basé sur le nombre de passagers pour différents nombres de contenus à diffuser en même temps (10 / 50 / 100)

multiplier les terminaux d'affichage qui afficheraient en parallèle plusieurs contenus, mais même dans ce cas-là, il serait difficile d'afficher plus de 50 contenus différents à la fois. De plus, ce cas extrême dans les résultats est atteint en considérant une seule et unique boucle du système autonomique. Cependant, le système est sensé effectuer des boucles en permanence de manière autonome et devrait prendre les décisions avant que de tels nombres soient atteints.

4.1.2.2 Analyse du temps de réponse pour une variation du nombre de sujets d'intérêt

De plus, il est intéressant de remarquer un autre comportement du système dans d'autres cas comme présenté en Figure 4.2.



(a) 10 contenus (b) 100 contenus

FIGURE 4.2 – Temps de réponse moyen basé sur le nombre de passagers pour différents nombres de contenus à diffuser (10 et 100) en fonction du nombre de sujets d'intérêt possibles

En effet, même si en moyenne sur tous les tests le pire cas peut prendre jusqu'à 3,5 secondes (*cf.* Figure 4.1c), si l'on se concentre sur un cas en particulier où il y a 100 contenus à diffuser pour 60 passagers et que l'on considère non pas le nombre de passagers mais le nombre de sujets d'intérêts possibles dans le système ; on observe un comportement différent du système et notamment de la grammaire de graphes. Le temps de réponse dans ce cas – et particulièrement celui de la transformation de graphe – peut prendre jusqu'à 5,8 secondes en moyenne pour 10 sujet d'intérêts possibles là où avec 50 sujets d'intérêts possibles le système ne prendra qu'en moyenne 2,5 secondes à s'exécuter (*cf.* Figure 4.2b).

Ce comportement peut être expliqué par le mode d'application nécessaire des règles de la grammaire de graphes utilisée. En effet, les règles de la grammaire sont appliquées par couches de règles ce qui implique que tant qu'une règle peut s'appliquer, les règles suivantes ne seront pas appliquée et cela peut ralentir le système qui ne peut appliquer qu'une seule couche de règles à la fois. De fait, quand il y a peu de sujets d'intérêts, le système se retrouve à agréger potentiellement beaucoup de contenus : la transformation de graphe détermine quels contenus envoyer au terminal du bus et quels contenus envoyer individuellement. Et donc, dans ce cas le nombre de contenus à diffuser étant élevé avec 10 sujets d'intérêt possibles, il y aura potentiellement une proportion significative de passagers intéressés par le même contenu, ce qui entraîne une transformation longue car il faut faire le lien entre ce contenu et toutes les entités ou presque.

D'un autre côté, dès qu'un nombre plus important de sujets d'intérêts est consi-

déré (50 par exemple), le temps de réponse moyen de la partie graphe diminue grandement (environ 1,5 secondes contre plus de 4,5 secondes auparavant) (Figure 4.2b). Cela est dû au fait qu'il y a une plus grande diversité de sujets d'intérêt et que donc le contenu agrégé sera moins important à cause d'une plus grande diversité d'intérêts chez les passagers.

On peut également observer cette tendance dans le cas où l'on ne diffuse que 10 contenus à la fois (4.2a) mais dans une moindre mesure étant donné que le temps de réponse global demeure autour de 1,5 secondes en moyenne.

4.1.3 Analyse du temps de réponse dans une gestion complexe de services

Afin d'évaluer les performances de l'approche de gestion complexe des services ainsi que les différents algorithmes, un ensemble de fichiers a été généré pour caractériser le comportement du système. Plus de 1000 fichiers comportant un graphe initial à transformer ont été générés avec un besoin aléatoirement défini (entrée·s et sortie·s aléatoirement définies parmi un ensemble de types possibles) ainsi qu'un nombre variable de services disponibles au moment de la génération du graphe (entrée·s et sortie·s aléatoirement définies également). De fait, potentiellement il est possible qu'il n'y ait aucune solution au besoin initial, et ce même une fois le graphe transformé. Les deux algorithmes d'exploration vont déterminer qu'il n'y a aucun plan d'exécution disponible à cet instant là. Dans le cas où un plan est disponible, l'algorithme glouton va retourner le premier plan qu'il trouve et l'algorithme optimisé va retourner le meilleur plan qu'il a trouvé. Dans le cas où un seul plan est possible les deux algorithmes vont trouver la même solution avec des temps de réponse différents : en effet, l'algorithme optimisé va prendre plus de temps à s'exécuter de manière générale car il explore tous les chemins possibles jusqu'à trouver un optimum (local dans certains cas particuliers (considération du temps d'exécution de branches parallèles), et global dans les autres).

4.1.3.1 Analyse du temps de réponse moyen en fonction du nombre de services disponibles dans le graphe initial

Pour produire les résultats présentés par la suite, chaque graphe a été transformé puis exploré au moins 10 fois pour produire une moyenne par fichier ainsi qu'une moyenne globale.

Transformation de graphes : Par l'analyse de ces résultats on remarque que le temps de réponse, notamment de la transformation de graphes va avoir une complexité polynomiale plus le graphe à transformer sera complexe. Cela est dû au moteur de transformation de graphe qui applique les règles de manière non linéaire (fonctionnement par priorité d'application) et qui tente d'appliquer toutes les règles dans un certain ordre jusqu'à non applicabilité de toutes les règles.

Comme on peut le remarquer sur la Figure 4.4, le temps de transformation peut aller de 257 ms en moyenne jusqu'à un peu plus de 10 secondes dans le pire des cas.

Pour plus de précision dans l'analyse du comportement de graphe d'autres résultats ont été mesurés. Dans la Figure 4.3 (détail des valeurs en Annexe D.1, Table D.1), on peut observer le temps moyen de transformation de graphes dans différents cas (global, avec un besoin simple ou un besoin double) en fonction du nombre de services initialement présents dans le graphe. Un besoin *simple* va être représenté par un nœud temp. ne comportant qu'une seule entrée et une seule sortie. *A contrario*, un nœud double comportera lui deux entrées et deux sorties.

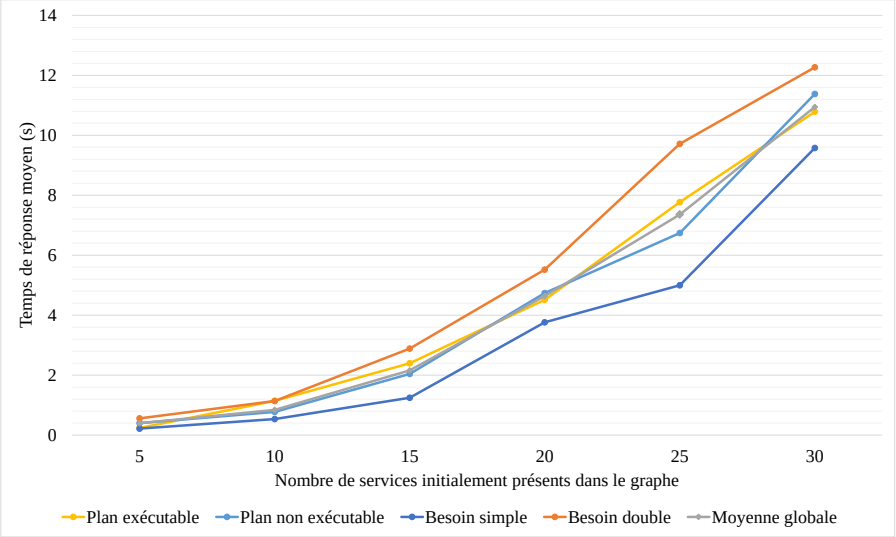


FIGURE 4.3 – Temps moyen de transformation de graphe dans le cas global et dans le cas où le besoin est simple ou double

Le comportement de la transformation s'avère en effet plus laborieux dans le cas où il y a beaucoup de services disponibles pour un besoin donné et également en fonction du nombre de types de contenus considérés (entrée·s / sortie·s). Plus le nombre de services disponibles est élevé et moins ces services répondent de manière proche au besoin, plus le temps de transformation sera long. En effet, dans le cas où de nombreux services sont disponibles et ne correspondent que partiellement au besoin (voire pas directement), un grand nombre de transformations intermédiaires sont nécessaires pour les intégrer au plan de composition. Également, dans le cas où il y a beaucoup de services qui correspondent au besoin, la grammaire va appliquer un plus de fois les règles permettant d'insérer ces nœuds dans le graphe. De plus, il arrive que des branches soient considérées comme mortes à la fin de la transformation car elle ont été raffinées avec de nouveaux besoins auxquels il n'existe finalement pas de réponse possible, même indirectement.

D'un autre côté, la présence d'un nœud temp. double dans le graphe initial va rallonger le temps d'exécution. En effet, dans ce cas, la transformation de ce nœud peut conduire à l'introduction de plusieurs nouveaux nœud temp. dans le graphe en cours de transformation si aucun service ne peut répondre tel quel au besoin, comme on peut le constater sur le temps moyen d'exécution.

Exploration des graphes transformés : Dans la Figure 4.4, le temps de réponse moyen des graphes d'exploration (exploration du graphe avec les deux algorithmes proposés) est présenté en fonction du nombre de services initialement disponibles dans le graphe à transformer.

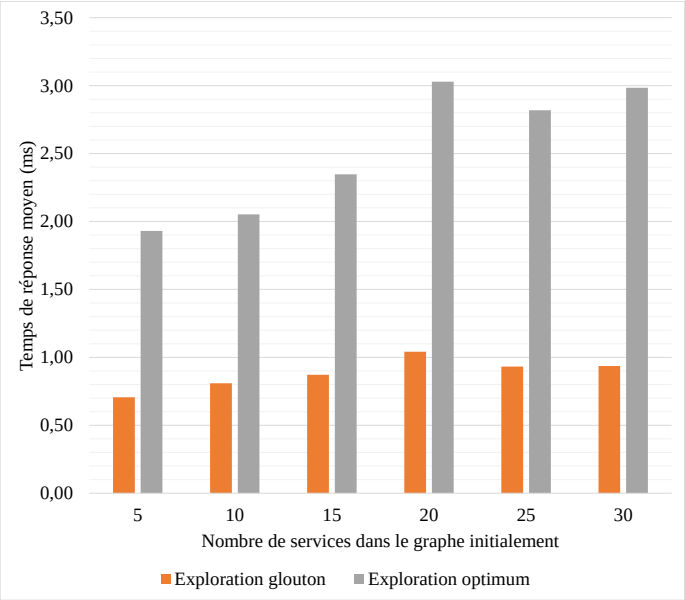


FIGURE 4.4 – Temps de réponse moyen des algorithmes d'exploration en fonction du nombre de services disponibles dans le graphe initial

On remarque que le glouton va avoir un temps d'exploration inférieur à l'algorithme de recherche d'optimum. Cela s'explique par le fait que le glouton va accepter le premier plan trouvé là où l'autre algorithme va explorer plus de possibilités pour rechercher un optimum. En revanche, lorsque l'on regarde ces temps d'exécutions sur une échelle logarithmique on remarque que le temps de réponse moyen des algorithmes va être d'un ordre 100 à 1000 fois plus rapide que la transformation. La complexité de ces algorithmes est en effet limitée par la profondeur et la largeur peu élevées des graphes explorés.

De ce fait, suivant les cas d'utilisation on va pouvoir considérer que ce temps d'exécution sont relativement négligeables par rapport à la transformation de graphe

(notamment dans le cas de graphes complexes à transformer).

Il est important de noter cependant qu'il sera rare d'atteindre de telles proportions en terme de complexité de transformation dans un cas réaliste car la complexité à gérer sera limitée par la phase d'analyse et le nombre de types de contenus sera limité en fonction du scénario géré lors de la prise de décision. Ces tests sont théoriques afin de dimensionner la réponse du système.

4.1.3.2 Comparaison des résultats produits et de performance temporelle des algorithmes d'exploration

Sur la base des graphes générés pour les tests précédents, d'autres résultats ont été exploités à savoir les résultats des explorations, *i.e.* les plans d'exécutions "choisis" par les algorithmes et une comparaison de résultat entre les deux algorithmes. De plus, une comparaison de performances (en terme de temps d'exécution) est réalisée afin de tempérer l'efficacité de l'algorithme de recherche d'optimum par rapport au glouton.

Ces résultats sont présentés dans la Table 4.1. Les comparaisons de performances réalisées sont définies de la manière suivante :

- les *améliorations de résultats* représentent l'amélioration de résultat (optimalité du plan choisi) par l'algorithme de recherche d'optimum par rapport à la solution fournie par le glouton, elles sont notées $perf_r$ avec $cout[glou]$ le coût total du résultat trouvé par l'algorithme glouton, et $cout[opti]$ le coût total du résultat trouvé par l'algorithme de recherche d'optimum :

$$perf_r = \frac{cout[glou] - cout[opti]}{cout[glou]}$$

- les améliorations de performances temporelles représentent le gain en temps d'exécution qu'offre le glouton par rapport à l'exécution de l'algorithme de recherche d'optimum, elles sont notées $perf_t$ avec $temps[glou]$ le temps total d'exécution de l'algorithme glouton, et $temps[opt]$ le temps total d'exécution de la recherche d'optimum :

$$perf_t = \frac{temps[opt] - temps[glou]}{temps[opti]}$$

Certains résultats ne sont calculés que pour les cas où une solution a été trouvée (cas *résolu*).

Comparaison des résultats produits (plan d'exécution final : Pour certains cas, notamment les cas où il n'y a qu'une seule solution possible ou aucune solution, les deux algorithmes vont retourner le même résultat. De ce fait, l'algorithme de recherche d'optimum n'apportera aucune amélioration par rapport à la solution fournie par le glouton (0% d'amélioration de résultat). *N.B. :* Dans les tests théoriques ici réalisés, de nombreux graphes ne comportent pas de solution :

Nb. Services	**5**	**10**	**15**	**20**	**25**	**30**
Moyenne *temps*[*glou*]	0,71	0,81	0,87	1,04	0,93	0,94
Moyenne *temps*[*opt*]	1,93	2,05	2,35	3,03	2,82	2,98
Moyenne $perf_t$	63%	60%	62%	64%	67%	67%
Min. $perf_t$	13%	-111%	-64%	-27%	14%	12%
Max. $perf_t$	82%	81%	77%	93%	82%	87%
Min. $perf_t$ *(résolu)*	68%	29%	18%	29%	37%	12%
Moyenne $perf_r$	0%	1%	3%	3%	5%	6%
Max. $perf_r$	0%	32%	60%	92%	68%	79%
Moyenne $perf_r$ *(résolu)*	0%	3%	8%	7%	8%	8%

TABLE 4.1 – Comparaison de performance temporelle (ms) et de qualité des résultats des algorithmes d'exploration (recherche gloutonne et recherche d'optimum)

en effet, l'ensemble de services générés aléatoirement ne permettent pas d'obtenir un plan d'exécution répondant au besoin aléatoirement généré. Cependant dans un cas réaliste, le graphe généré par l'analyseur comporterait plus de services potentiellement utilisables et disponibles dans l'environnement à l'instant considéré que dans les tests théoriques où les services sont aléatoirement générés de par la présence de la phase d'analyse en amont.

Ainsi, dans le cas de petits graphes (5 à 10 services disponibles) l'algorithme de recherche d'optimum va souvent (voire toujours) trouver la même solution que le glouton. Les améliorations de solution fournie seront donc quasi nulles.

Cependant, dans le cas de graphes plus complexes, l'algorithme de recherche d'optimum va fournir une meilleure solution que l'algorithme glouton au prix d'un temps d'exécution plus long en moyenne. Ainsi, on peut observer dans certains cas extrêmes une amélioration de presque 100% du résultat fourni (ce qui veut dire que le coût total d'exécution du plan sélectionné par la recherche d'optimum est réduit là où le coût d'exécution total du plan fourni par le glouton est presque deux fois supérieur). Dans les cas où une solution existe, (cas *résolu*), la recherche d'optimum va permettre d'améliorer le résultat en moyenne de 3 à 8% quand il y a plusieurs solutions.

Comparaison des performances temporelles : Par rapport au temps d'exécution de ces algorithmes, le glouton va s'exécuter en moyenne en moins de 1 milliseconde et la recherche d'optimum en 2 à 3 ms. En effet, étant donné que le glouton va explorer d'abord en profondeur les graphes là où la recherche d'optimum va explorer en largeur en priorité, le glouton aura un temps d'exécution plus court. Cependant, de manière globale, le temps d'exécution des algorithmes d'exécution va être environ 100 à 1000 fois plus rapide que la transformation de graphe qui sera de l'ordre de la seconde à la dizaine de secondes dans le pire cas.

D'un autre côté, si l'on compare le temps d'exécution de l'algorithme glouton par rapport au temps pris par la recherche d'optimum, le glouton présente un gain de 65% en moyenne en terme de temps d'exécution, ce qui veut dire que le glouton a été

plus d'1.6 fois plus rapide. En effet, le glouton se contentant de prendre la première solution trouvée en explorant en profondeur, ce dernier va en général s'exécuter beaucoup plus rapidement que la recherche d'optimum.

Cependant, dans certains cas particuliers, notamment dans le cas de graphes plus complexes comportant des branches mortes, le glouton peut prendre plus de temps à s'exécuter que la recherche d'optimum de par son mécanisme d'exploration en profondeur et de retour en arrière. Par exemple, sur un graphe particulier, le glouton a pris plus de 2 fois le temps pris par la recherche d'optimum (-111%) pour s'exécuter.

Cependant, les cas où le glouton prendra plus de temps à s'exécuter ne se rencontre que lorsqu'il n'y a pas de solution trouvée. En effet, lorsque l'on compare les résultats sur $\min perf_t$ dans le cas général et le cas où il y a une ou plusieurs solution·s, il n'y a pas de cas avec un ration négatif.

En bref, de manière générale, l'algorithme glouton va représenter un gain en temps d'exécution de plus de 60% par rapport à la recherche d'optimum ce qui représentera quelques *ms* de moins là où dans certains cas l'optimum sera plus rapide mais surtout peut apporter un gain significatif d'optimisation du coût d'exécution des services retenus dans le plan final.

4.1.4 Conclusions des évaluations théoriques

Pour résumer cette analyse, différents points sont à souligner.

Performances de la partie sémantique associé à la transformation de graphes : Comme l'ont démontré les cas théoriques considérés dans les tests présentés précédemment, il est important de noter que le nombre de contenus à diffuser en une seule itération de boucle autonomique sera déterminant pour les performances du système ainsi que le nombre de sujets d'intérêts considérés. Si l'on considère beaucoup de contenus avec également beaucoup de passagers et peu de sujets d'intérêt, le temps de réponse de la transformation de graphe sera grandement augmenté. D'un autre côté, dans un cas plus *normal* d'utilisation, le temps de réponse reste adapté aux conditions du cas d'utilisation avec une réponse du système en quelques secondes (moins de 3 secondes).

Il est également intéressant de noter que le déploiement décentralisé est particulièrement adapté à notre cas d'utilisation. En effet, les contenus considérés dans ces tests ne le sont que pour une seule itération de la boucle autonomique, *i.e.* qu'ils seront traités en une seule fois dans une seule boucle du système autonomique. Dans un déploiement réel, le système devrait boucler bien avant que de tels nombres ne soient atteints.

Enfin dans le cas de besoins intermédiaires, le temps de transformation sera lié à la quantité de services disponibles dans le contexte considéré, et à la présence de besoins plus complexes requérant plusieurs contenus en entrées et plusieurs contenus en sorties. Cependant ces nombres seront relativement limités dans un cas réel inscrit dans la boucle autonomique : en effet les services disponibles seront déter-

minés par l'état actuel du système et l'analyseur permettra d'indiquer des services potentiellement à utiliser. De plus, la présence de nœuds temporaires doubles dans le graphe initial ralentit la transformation de manière théorique mais cet impact devrait être limité dans le cas réel où l'analyseur permettrait d'obtenir des services répondant au besoin ou proches de ce dernier. Les cas où un grand nombre de services seront disponibles et considérés comme intéressants par l'analyseur pourra représenter un temps de génération de plan concret plus long mais il sera possible de réaliser des optimisations intéressantes par la suite.

Performances des algorithmes d'exploration et quel algorithme privilégier : Comme vu dans les résultats présentés, dans des cas simples avec peu de choix ou dans le cas où le temps de prise de décision et d'exécution des services doit être le plus rapide possible, il serait intéressant de privilégier l'algorithme glouton pour trouver un plan d'exécution exécutable le plus rapidement possible quitte à avoir une solution totalement opportuniste. D'un autre côté, quand les cas sont plus complexes avec des choix multiples et/ou que l'on souhaite minimiser un ou plusieurs paramètre·s non-fonctionnel·s comme l'impact énergétique / temporel / monétaire, il est intéressant d'utiliser l'algorithme de recherche d'optimum afin d'obtenir potentiellement un meilleur résultat. De plus, dans un cas plus complexe le temps d'exécution de l'algorithme de recherche d'optimum sera alors presque négligeable par rapport à la transformation de graphes pour le gain potentiel au niveau du résultat produit.

En résumé : Le comportement des différents composants du système est cohérent avec les différentes caractéristiques qui le composent à savoir raisonnement et inférence sémantique couplé à de la transformation de graphe. Bien que ces méthodologies peuvent s'avérer coûteuses suivant les cas considérés lorsque la taille des graphes à transformer augmente, dans notre cas l'architecture même du système et son mode de fonctionnement limiteront l'impact de ces technologies sur la réactivité globale du système en limitant le nombre de cas traités à la fois lors d'une seule boucle du système autonomique. Cependant, si des cas plus larges ou complexes veulent être traités cela sera possible par l'adaptation du système autonomique au contexte et au besoin.

4.2 Preuve de concept : prototype et déploiement

Un prototype complet de la boucle MAPE-K a été implémenté et déployé sur un système IoT pour démontrer son bon fonctionnement et l'inclure dans une maquette. L'idée est de représenter un bus connecté dans une ville intelligente avec différents passagers possibles et plusieurs zones dans la ville avec différents types de contenus disponible suivant l'endroit.

4.2.1 Eclipse OM2M : un intergiciel standard pour l'IoT

Depuis 2013, le LAAS est le principal développeur d'une plateforme horizontale pour l'IoT : OM2M [6]. Le projet a débuté en tant qu'implémentation du standard européen ETSI SmartM2M et implémente depuis 2015 le standard international oneM2M [7]. OM2M est un projet open source hébergé par la fondation Eclipse et fait partie du groupe de travail dédié à l'IoT [8]. OM2M est une plateforme de services horizontale pour l'IoT qui fournit une couche d'interopérabilité (d'abord syntaxique mais aussi potentiellement sémantique) par une interface (API) REST avec un ensemble de services génériques. Son architecture est basée sur le framework OSGi et est extensible via un système de plugins. Le but de ce type de plateformes est de permettre le développement de services et d'applications indépendamment du réseau hétérogène d'objets sous-jacent. De fait, cela facilite le déploiement d'applications IoT en créant une couche d'abstraction standard des objets de telle sorte que les applications peuvent être développées indépendamment des objets ou des instances de plateformes. OM2M peut être utilisée à différents niveaux dans une architecture IoT : au plus haut niveau, *e.g.*, au niveau du serveur principal, ou sur des nœuds intermédiaires de la topologie (serveurs intermédiaires ou passerelles), ou même sur les nœuds les plus bas directement connectés aux objets (passerelles). De plus, de par son implémentation du standard, OM2M est interopérable avec d'autres implémentations du standard oneM2M.

En résumé, OM2M fournit une couche d'interopérabilité (syntaxique et potentiellement sémantique) concernant l'architecture et les protocoles, grâce aux spécifications standard oneM2M. La plateforme peut être exécutée à différents niveaux dans une topologie IoT et est extensible. L'utilisation de cette plateforme permet donc un déploiement facilité et générique de notre système sur un système réaliste tout en étant interopérable avec d'autres implémentations standards.

Dans le cadre du projet S2C2, les instances d'OM2M ont été déployées ainsi que la plateforme HELLIOT implémentée par ATOS avec une interconnexion entre les deux plateformes via l'interface standardisée.

Dans le cadre du projet STM, un plug-in spécifique (IPE) permettant l'interconnexion avec le protocole de *Device Management* LWM2M a été développé. Son fonctionnement se base sur l'utilisation d'un serveur LWM2M (Eclipse Leshan [9]). Les différents composants du plug-in ainsi que son processus de fonctionnement sont détaillés en Annexe D.2

4.2.2 Le système IoT de prototype et démonstration

Les différentes contributions mises en avant précédemment ont été mises en œuvre dans un prototype notamment dans le cadre du projet S2C2 et du scénario de diffusion de contenu dynamique avec des bus connectés dans une ville intelligente.

6. `http://www.eclipse.org/om2m`
7. `http://www.onem2m.org`
8. `https://iot.eclipse.org`
9. `https://www.eclipse.org/leshan/`

4.2.2.1 Déploiement général

Le déploiement général des différents éléments du système IoT déployé sont présentés sur les figures 4.5 et 4.6. Ce déploiement repose sur une architecture standardisée oneM2M afin de définir une couche de services standardisée et un déploiement de différents nœud dans une topologie définie.

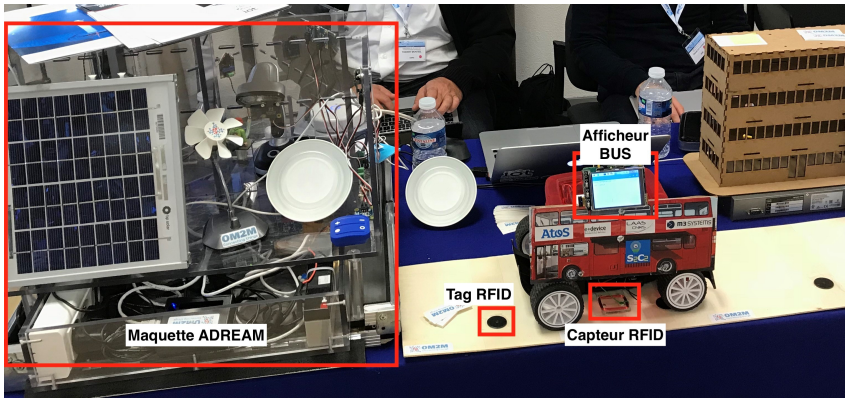


FIGURE 4.5 – Aperçu du prototype de démonstration

Le prototype est composé de plusieurs éléments :

- un PC (serveur) avec l'intergiciel Eclipse OM2M [10] afin de connecter les objets et d'exposer leurs données via une API REST standardisée (standard oneM2M [11])
- Maquette de bâtiment connecté (LAAS-CNRS) avec une passerelle oneM2M dédiée : une Beagle Bone Black (BBB) exécutant OM2M (MN-CSE) ainsi qu'un client LWM2M pour remonter des informations sur la BBB
- Maquette de bus connecté avec une RaspberryPi 2 pour simuler le bus connecté et son affichage connecté (petit écran) incluant un client LWM2M pour remonter des informations sur l'objet physique (mémoire, position, etc.)
- Capteurs divers qui seraient déployés dans la ville via une carte Intel Edison afin de les connecter à l'intergiciel IoT (OM2M) du serveur, accompagné d'un client LWM2M pour remonter des informations sur l'objet physique
- Connexion externe avec autre·s plateforme·s (ici plateforme ATOS HellIoT)

Ces différents éléments et leurs interactions sont schématisés sur la Figure 4.6. L'instance d'entités oneM2M dans les différents nœuds de la topologie IoT sont représentés par les différents systèmes logiciels exécutés par les objets physiques (CSE / AE) ainsi que les connexions (IPE) vers des technologies autres.

Un client LWM2M a également été instancié sur le bus afin de remonter des informations sur la carte et l'écran mais aussi comme la position. Cette connexion est réalisée grâce au plug-in d'interconnexion (IPE) avec LWM2M.

10. http ://www.eclipe.org/om2m
11. `http://www.onem2m.org`

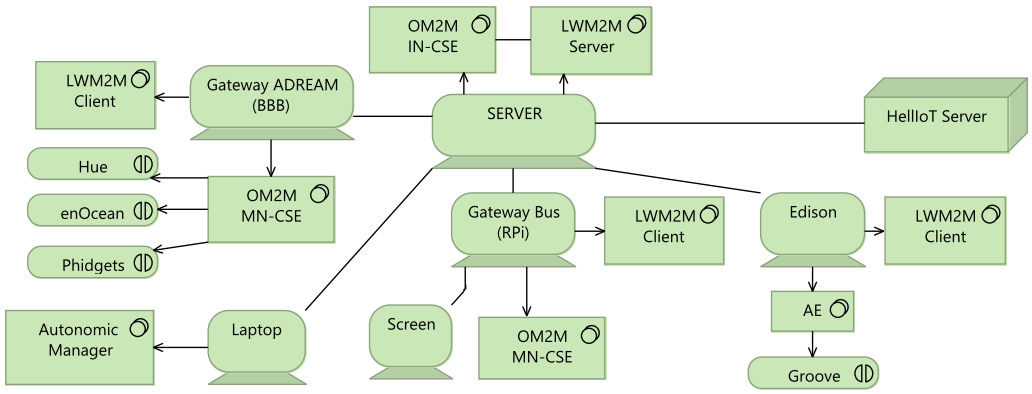


FIGURE 4.6 – Déploiement général du système IoT et du prototype de démonstration

4.2.2.2 Maquette : Smart City & bâtiment connecté

Afin de représenter des éléments qui peuvent être déployés dans une ville intelligente, différents composants ont été utilisés.

Capteurs de la ville : Quelques capteurs sont déployés sur le prototype afin de mesurer notamment des données liées à la qualité de l'air : des capteurs de gaz et de particule. Des capteurs de luminosité et de température sont utilisés également pour apporter d'autres sources de données. Ces capteurs sont déployés sur la carte Intel Edison afin de connecter divers capteurs à l'intergiciel IoT (OM2M).

Bâtiment connecté : La maquette de bâtiment connecté se base sur une représentation du bâtiment connecté (*ADREAM*) du LAAS-CNRS [12] (Figure 4.7). Ce bâtiment est un bâtiment à énergie optimisée qui comporte environ 6500 objets connectés (capteurs et actionneurs) déployés dans tout le bâtiment.

FIGURE 4.7 – Le bâtiment ADREAM, au LAAS-CNRS de Toulouse

La maquette quant à elle reprend l'idée déployée dans le bâtiment réel avec

12. `https://www.laas.fr/public/fr/le-projet-adream`

un ensemble de capteurs et d'actionneurs pour surveiller des données comme la température, l'humidité, la luminosité, la présence, des lampes d'ambiance, un ventilateurs, etc. Ces différents objets sont reliés à une passerelle déployée sur une Beagle Bone Black qui sera en charge de l'interface entre la plateforme standardisée et les objets spécifiques.

4.2.2.3 Maquette de bus connecté

La maquette de bus connecté reprend l'idée d'un bus connecté dans la ville intelligente qui peut changer de position et va remonter cette information via un capteur, et qui possède également des objets connectés (écrans de diffusion de contenu) via des services (ici via la plateforme OM2M). La maquette est composé de différents éléments :

- La Raspberry Pi connectée à la plateforme OM2M en tant qu'AE (Application Entity) afin de pouvoir recevoir de nouveaux contenus à afficher
- L'Intel Edison (utilisée également pour le déploiement d'autres capteurs du scénario) avec un capteur RFID afin de pouvoir lire les cartes des passagers qui valident leur titre de transport en montant dans le bus mais aussi lire les tags RFID utilisés pour simuler la position géographique

Les informations remontées par les capteurs vont être stockées dans la plateforme OM2M avec des conteneurs de données spécifiques.

Certains capteurs sont utilisés essentiellement à des fins de détection d'évènements comme des capteurs RFID qui permettent de détecter quelle personne monte dans le bus ou les changements de positions du bus basés sur la position GPS du bus (simulée par des tags RFID dans la maquette).

4.2.3 Le prototype déployé du système autonomique

Dans le contexte de la maquette de système IoT présenté ci-dessus, une instance du système de gestion autonome a été implémentée et déployée.

4.2.3.1 Diagramme de déploiement global et architecture (intégration avec oneM2M)

Le système de gestion autonomique implémenté en Java est déployé et intégré dans le système comme présenté dans la Figure 4.6.

Le système va être connecté à l'intergiciel OM2M via l'interface applicative (*mca*). L'intérêt du déploiement oneM2M est également de pouvoir réaliser des découvertes d'objets connectés au système mais aussi de s'abonner à ces derniers afin d'observer leurs changements d'état notamment, l'arrivée de nouveaux objets, etc.

Le gestionnaire autonome implémenté va se structure suivant différents composants présentés en Figure 4.8.

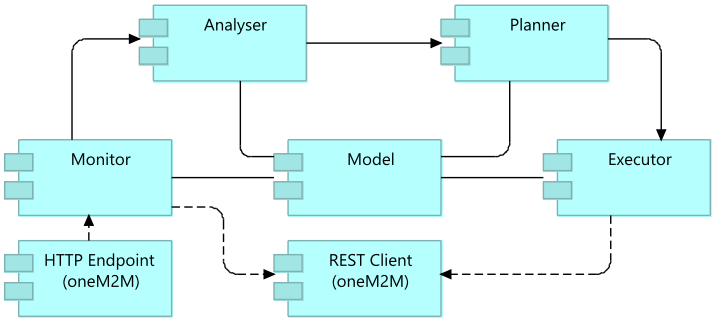


FIGURE 4.8 – Diagramme de composants haut-niveau du prototype de système autonomique implémenté

4.2.3.2 Les différents composants mis en œuvre

On peut ainsi voir comment les différents éléments de la boucle MAPE-K ainsi que les modèles sont instanciés. Le composant *Model* comporte le modèle qui inclue l'ontologie avec les différents individus et les règles SWRL, ainsi que le modèle de grammaire de graphes.

Le *Moniteur* est connecté au système via des souscriptions (*Subscription* avec un client REST oneM2M) afin d'être notifié lors des changements d'états des capteurs considérés (nouvelles valeurs disponibles, nouvel objet connecté, objet déconnecté, etc.). Par exemple, lorsque le capteur RFID va lire une nouvelle valeur, cet évènement sera détecté de même lorsqu'un capteur va produire un nouveau contenu (nouvelle valeur de température par exemple, un changement d'état au niveau du bâtiment connecté, etc.) et une notification sera reçue (HTTP). En fonction des différents évènements qui occurrent sur le système supervisé, le moniteur va mettre à jour la base de connaissance sémantique avec de nouvelles méta-données actualisées ou non via le Contrôleur. Enfin, en fonction de ces évènements il va pouvoir générer des symptômes (un lot d'exemples est fourni en Annexe B dans la Table B.1). Ces symptômes peuvent amener l'*analyseur* à déclencher certaines actions comme l'inférence sémantique. Dans le cas où un symptôme est levé qui nécessite une inférence, l'analyseur va travailler sur un instantané (*snapshot*) de l'ontologie à cet instant à l'aide de la OWL API et réaliser une inférence sémantique à l'aide d'un raisonneur. Une fois cette inférence réalisée, suivant les résultats obtenus un graphe de RFC sera généré à partir des informations connues dans la base de connaissance.

Le *planificateur* va recevoir ces graphes RFC par le biais d'un *buffer* permettant de stocker les graphes à traiter dans le cas où il y aurait plusieurs graphes générés rapidement. Ces graphes vont être transformés et ensuite transmis à l'*exécuteur* (via un buffer également).

L'*exécuteur* va lire les plans produits par le *planificateur* et récupérer les méta-données nécessaire à l'exécution (via la base de connaissance) à l'aide de la OWL API avant de l'exécuter littéralement. Ce dernier sera formé ici de services REST avec le protocole défini par oneM2M (en HTTP), et d'opérateurs permettant d'arti-

culer l'exécution. Les services utilisés ici sont ceux fournis via la plateforme OM2M et permettant à la fois de récupérer les données fournies par les objets (capteurs, etc.) mais aussi d'envoyer des contenus comme par exemple le contenu à afficher sur le terminal du bus. De fait l'interopérabilité syntaxique s'en trouve grandement facilitée. Cependant, il également possible de faire appel à d'autres types de services, il est juste nécessaire de disposer d'un client (REST, SOAP, etc.) implémenté dans l'exécuteur pour l'appel de ces services dont les méta-données sont disponibles dans la base de connaissance.

4.2.3.3 Configuration du système autonomique et du prototype

Zone	**1**	**2**	**3**	**4**
Nom	Init	Industrial	City Centre	Museum
Contenu	Temperature	Temp. Particules Gaz	Temp. Particules	Temp. État du musée

Table 4.2 – Configuration du système de démonstration

Le système simulé est configuré tel que présenté dans la Table 4.2 avec différentes zones géographiques possible dans la ville intelligente et différents services fournissant des contenus (capteurs ou autres) afin de mettre en œuvre le scénario de diffusion autonome de contenu dynamique.

Les différents contenus fournis sont liés à différents sujets :

- *informations générales*, pour des données générales comme la température, des informations globales sur la ville, etc.
- *la qualité de l'air*, pour les données en rapport avec les particules présentes dans l'air, les mesures de gaz polluants, pollens, etc.
- *musées*, afin de regrouper des informations en rapport avec des institutions comme des musées (qui représentera ici le bâtiment connecté disponible)

Dans le prototype, deux personnes sont considérées : P_1 et P_2.

- P_1 est intéressée par les *informations générales* et la *qualité de l'air*
- P_2 est intéressée également par les *informations générales* et les *musées*

De ce fait, lorsque de nouveaux contenus sont disponibles, le système va automatiquement décider comment faire parvenir les contenus aux entités concernées.

4.2.3.4 Résultats en fonctionnement

Des résultats ont été obtenus en jouant le scénario sur le système déployé pour montrer son bon fonctionnement. Ces résultats sont présentés dans la Table 4.3 ci-dessous. Le contenu affiché est obtenu quand le système va exécuter le plan d'exécution de services en récupérant les données à partir des services producteurs de contenu et envoyé les résultats agrégés (ou non suivant les cas) au service d'affichage du bus si nécessaire.

Zone	**1**	**1**	**2**	**3**	**3**	**4**
Passagers	$\emptyset$	P_1	P_1, P_2	P_1, P_2	P_2	P_2
Affichage bus	$\emptyset$	Temp., Lum.	Temp., Lum., Gaz, Particules	Temp., Lum., Particules	Temp., Lum.	Temp., Lum., Musée
Affichage P_1	Temp., Lum.	$\emptyset$	$\emptyset$	$\emptyset$	Temp., Lum.	Temp., Lum.
Affichage P_2	Temp., Lum.	Temp., Lum.	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

TABLE 4.3 – Contenu affiché sur l'écran connecté du bus et sur les terminaux des personnes en fonction des passagers dans le bus et de sa position

Ainsi on peut voir que lorsque le bus se trouve dans la zone 1, aucun contenu ne s'affiche car il n'y a pas de passagers dans le bus. Lorsque le passager P_1 monte dans le bus, le système décide alors d'envoyer les informations générales accessibles (température, luminosité) de la zone concernée à l'écran du bus directement. De fait, les informations générales diffusées auparavant sur le terminal privé de l'utilisateur car il n'avait accès à aucun autre terminal sont désormais diffusées sur le terminal collectif du bus.

Lorsque le passager P_2 monte dans le bus dans la zone 2, le système envoie désormais les nouvelles informations de température, de gaz et de particules sur l'écran car il s'agit d'informations d'ordre collectif et qu'il y a des personnes intéressées dans le bus. Lorsque le bus passe dans la zone 3, il n'y a plus d'informations liées au capteur de gaz d'affichées car il n'y a pas de capteur de gaz déployé dans cette zone. Enfin, quand le passager P_1 descend du bus en zone 3, les données liées à la *qualité de l'air* ne sont plus affichées car cela n'intéresse pas P_2 qui est alors le seul passager restant. De manière similaire, dans la zone 4 les informations concernant le musée à proximité sont affichées sur le terminal du bus car P_2 est toujours dans le bus et est intéressé par ce genre de contenu.

4.2.4 Analyses et conclusions

Grâce à ce déploiement réel dans un prototype de démonstration, l'approche proposée a pu être validée. Les fonctionnalités mises en œuvre permettent de gérer de manière autonome et automatique comment utiliser les différents services disponibles en fonction de l'état du système à un instant donné et de fournir les bons contenus à afficher aux différents services dynamiquement et à la volée sans pré-définition du plan d'exécution. De plus, le système décide en autonomie quels services mutualiser en fonction des types de contenus mis à disposition et des préférences des utilisateurs présents dans le bus connecté ou non. Ce scénario est un des exemples d'applications possibles de l'approche sur lesquels instancier la configuration de la boucle autonomique.

Conclusion

Afin de caractériser le comportement de l'approche proposée dans cette thèse et valider son bon fonctionnement, différentes évaluations ont été conduites.

Tout d'abord une évaluation théorique des différents composants mis en jeux dans l'approche a été réalisée. Les différents tests ont permis de caractériser notamment le temps de réponse des différents éléments du système que ce soit la partie analyse et inférence sémantique basée sur l'ontologie ou la partie transformation de graphes avec les grammaires de graphes. Dans cette étude, les performances du modèle sémantique ont été évaluées notamment pour le temps d'inférence de connaissance à partir de fichiers aléatoirement définis dans le contexte de notre cas d'utilisation. Cette évaluation a permis de caractériser ce temps de réponse en fonction de différents critères comme le nombre de contenus considérés à la fois.

Le temps nécessaire pour l'extraction de connaissance et la génération de graphe à partir de l'ontologie a également été évalué comme étant peu consommateur de ressources.

D'un autre côté les performances du modèle à base de grammaires de graphes a été également évalué afin de dimensionner la réponse temporelle de ce dernier. Également, des tests ont été réalisés avec les algorithmes d'exploration sur des résultats produits par la grammaire de graphes afin de comparer leurs temps d'exécution mais aussi les résultats qu'ils peuvent produire et notamment les améliorations possibles.

Ces différents tests ont permis de tirer différentes conclusions :

- Les performances du modèle sémantique sont acceptables avec un temps de réponse qui va croître en fonction du nombre d'éléments à considérer à la fois dans l'ontologie
- Les performances du modèle à base de graphe sont variables avec un temps de réponse presque négligeable (6 à 7 fois inférieur) par rapport à la sémantique dans le cas de systèmes avec peu d'éléments à considérer à la fois là où les performances du modèle à base de graphes peuvent prendre deux à trois fois plus de temps que la sémantique dans des cas extrêmes
- Les algorithmes d'exploration vont tous deux présenter des avantages et des inconvénients en fonction des cas dans lesquels ils sont utilisés : en présence de graphes simples et sans contraintes particulières, l'algorithme glouton permet d'obtenir un plan d'exécution rapidement là où dans le cas de graphes plus complexes et / ou de contraintes, l'algorithme de recherche d'optimum permet d'optimiser la solution retournée pour un temps d'exécution légèrement plus long
- Dans un cas réel, une distribution décentralisée est préférable afin d'optimiser le temps de réponse de la boucle complète comprenant sémantique et transformation de graphe

Dans un deuxième temps, un prototype de l'approche a été implémenté et déployé sur une architecture réaliste IoT à l'aide de l'intergiciel Eclipse OM2M, plateforme standard (oneM2M) afin de réaliser une preuve de concept.

Les différents tests réalisés sur cette instance du système ont permis de valider

son comportement et de l'articuler autour de scénarios du projet S2C2 dans le cadre de démonstrations.

Ces différents résultats ont fait l'objet de publication avec les contributions précédemment présentées [13] [14].

Dans le chapitre suivant, les conclusions générales de la thèse sont détaillées, regroupant les différentes contributions des chapitres précédents, ainsi que des perspectives de travail liées à ces contributions.

13. [Garzone 2018]
14. *publication(s) en cours de soumission*

Conclusions et Perspectives

Conclusions

L'IoT représente encore aujourd'hui un nombre important de défis et de problématiques complexes. L'écosystème est vaste et varié avec de nombreux standards, protocoles et technologies dont le nombre ne cesse de croître.

Dans ce contexte, le projet S2C2 cherche à amener des éléments de réponses à différentes problématiques et scénarios liés à la ville intelligente et à au déploiement de nouveaux objets connectés. Afin de faciliter le déploiement d'applications haut-niveau composites, une couche d'interopérabilité (syntaxique et à l'avenir sémantique) a été définie à l'aide d'implémentations de plateformes oneM2M. (Eclipse OM2M [15], open source initialement publié par le LAAS-CNRS & HellIoT, implémentation d'AToS). En effet, l'utilisation de ce type de plateformes horizontales est un tremplin pour une plus grande interopérabilité et l'intégration d'applications IoT plus ambitieuses, multi-domaines, transverses et complexes. Cela est rendu possible grâce à une vision orientée services qui permet de prendre en compte les services fournis par ce type de plateformes ainsi que d'autres services (web, logiciels, etc.) déjà existants et à venir.

Un scénario central dans ce contexte est l'implication de nombreux objets et services déjà disponibles dans la ville dans un but de diffusion dynamique de contenus pour les usagers de la ville intelligente. La vision orientée services permet de considérer un grand nombre d'éléments accessibles via différents moyens et de permettre leur utilisation en créant de nouveaux services haut-niveau composites par exemple.

Dans ce contexte, différents éléments de **problématique** se posent :

- comment structurer le système pour qu'il soit capable de réagir en **autonomie** le plus possible face à la versatilité d'environnements IoT et ce même avec des changements en terme de services (nouveaux objets, mobilité d'objets et versatilité avec possibilité d'objets dormants ou désactivés, etc.)
- comment permettre cette **utilisation composite** de manière la plus automatique possible sans définir à l'avance des processus spécifiques à exécuter sur le système
- être capable d'intégrer dans cette approche des **propriétés non fonctionnelles (QoS)** afin de réaliser des optimisations suivant des contraintes ou des objectifs particuliers qui peuvent survenir dans certains cas dans le contexte de la ville intelligente

Afin de répondre aux différentes problématiques qui se posent dans le contexte du scénario de ville intelligente et de diffusion de contenus dynamique (mais pas uniquement), différentes pistes ont été explorées.

15. `http://www.eclipse.org/om2m/`

La richesse de la littérature sur la **composition de services** amène de nombreux moyens d'aider à cela, dans un contexte de vision orientée services. Les **systèmes autonomiques** peuvent répondre aux problématiques de dynamicité et d'autonomie demandée en permettant de configurer des systèmes avec des politiques comportementales haut-niveau et que le système fasse preuve de capacités d'auto-gestion. Les technologies du **web sémantique** (ontologies) permettent de définir des vocabulaires partageables et compréhensibles pour les machines afin d'amener des informations de contextualisation et d'analyse de la connaissance sur un système (raisonnement et d'inférence de connaissance automatique). Différentes ontologies reconnues ont été étudiées et **IoT-O** [16] s'est dégagée comme un modèle de référence particulièrement intéressant dans notre contexte de travail. Enfin, les **grammaires de graphes** sont des solutions permettant à la fois de modéliser des éléments sous formes de graphes typés mais aussi de transformer automatiquement des graphes en plans d'exécution à la volée et ne pas comporter des exécutions de plans figés préétablis en fonction de certains symptômes particuliers.

En considération de ces éléments, différentes contributions ont été réalisées :

- une **architecture de système autonomique multi-modèles** se basant sur le modèle MAPE-K a été proposée et détaillée, particulièrement pour la phase d'analyse et de planification. Les modèles de cette approche ont été formalisés et instanciés dans notre cas d'utilisation
- Une instance de **modèle sémantique d'ontologie** a été proposé : il se base sur une extension de l'ontologie IoT-O ainsi qu'un ensemble de règles SWRL permettant d'ajouter de l'automatisme et des informations supplémentaires qui seront inférées automatiquement par le système.
- Un **modèle de grammaire de graphes** couplé au modèle d'ontologie a été défini puis instancié dans notre cas d'utilisation. L'extraction de connaissance à partir de l'ontologie permet d'établir un graphe primitif qui est ensuite transformé à l'aide d'un lot de règles de réécriture de graphe qui permettent d'automatiquement établir un plan d'exécution de services qui sera exécuté sur le système supervisé.
- Des **exemples** de mise en œuvre ont également été présentés pour illustrer le fonctionnement du système et des différents modèles.

Ces travaux ont donné lieu à une publication scientifique [Garzone 2018].

Dans un deuxième temps, afin d'enrichir cette approche et de permettre d'intégrer plus de diversité dans l'élaboration des plans d'exécutions ainsi que des paramètres non-fonctionnels liés à la QoS pour permettre un choix de services et une certaine optimisation de l'ordre de la sélection de services. Différentes contributions ont donc été réalisées :

- Afin de permettre au système de générer des **besoins temporaires** générés à la volée en fonction des résultats produits par le comportement précédent, de nouveaux éléments on été mis en œuvre comme des nœuds temporaires insérables dans le graphe dans le cas où un besoin apparaît durant l'analyse.

16. `https://www.irit.fr/recherches/MELODI/ontologies/IoT-O.html`

- La possibilité d'intégrer des services possédant plusieurs entrées et plusieurs sorties a été développée
- Des **opérateurs plus complexes** ont été définis afin de permettre une plus grande diversité de génération de plan d'exécution en intégrant plus de choix et la possibilité d'exécuter des processus en parallèle
- Un **modèle de grammaire de graphe** intégrant ces différents éléments a été défini afin d'intégrer la gestion des différents types de services, des besoins temporaires, des opérateurs plus complexes.

Afin de réaliser des choix et des optimisations lorsque plusieurs choix sont disponibles dans le plan d'exécution, des algorithmes d'exploration du graphe transformé ont été proposés.

- un **algorithme glouton opportuniste** performant qui permet d'obtenir un plan final à exécuter directement rapidement dans le cas où il n'y a pas de volonté d'optimiser ou de contraintes
- un **algorithme de recherche d'optimum** qui permet d'effectuer des choix optimisés afin de minimiser le coût d'exécution suivant un ou plusieurs paramètres non fonctionnels.

Ces différentes contributions ont fait l'objet de publication [17].

En résumé, les différentes contributions proposées dans cette thèse vont s'articuler en différentes étapes :

1. Surveillance des évènements sur le système et maintient à jour de la base de connaissance
2. En cas de symptômes particuliers, le système peut utiliser la puissance d'analyse grâce au modèle sémantique et à l'inférence pour générer des requêtes de changement à effectuer sur le système sous forme de graphe.
3. En combinaison avec le modèle sémantique, le modèle de grammaire de graphe va récupérer les informations nécessaires à partir de la base de connaissance pour générer un graphe primitif qui est ensuite transformé par le moteur de réécriture de graphe afin de générer un plan d'exécution à base de services incluant des opérateurs articulant le processus d'exécution.
4. En cas de contraintes et/ou de volonté d'optimiser l'exécution, un algorithme d'exploration du graphe transformé est appliqué pour sélectionner le plan final optimisé qui sera exécuté sur le système réel, sinon un algorithme opportuniste est utilisé pour obtenir un plan exécutable non optimisé.

Enfin, pour valider cette approche et les différentes contributions, une étude d'évaluation a été réalisée.

- Une **évaluation des performances** des modèles proposées a été réalisée (théorie) afin de dimensionner les temps de réponse et capacités de ces modèles comme le temps d'inférence sémantique ou de transformation de graphe

17. *(en cours de soumission)*

- Un **prototype dans le contexte du projet S2C2** et de notre cas d'utilisation a été conçu et implémenté dans une maquette de ville intelligente et de bus connecté. Son bon fonctionnement a validé les fonctionnalités avancées par l'approche.
- Une **étude comparative des deux algorithmes** d'exploration proposés a permis de mettre en avant les différences de résultats produits avec les optimisations possibles en terme de coût d'exécution du plan final de services

Ces différents éléments ont permis de valider l'approche abordée et les différentes contributions dans notre contexte et d'intégrer cela dans le système développé pour le projet S2C2. Les performances des modèles complexes mis en œuvre permettent de répondre au besoin mais demeurent limités en terme de capacités : un déploiement réel d'ampleur devrait prendre en compte des problématiques liées à la scalabilité et au déploiement distribué.

Les résultats de l'étude d'évaluation ont été publiés conjointement aux autres contributions dans [Garzone 2018] et dans d'autres publications [18]. Ces travaux ont également fait l'objet de présentations au cours des **Showcases oneM2M à l'ETSI** (Sophia Antipolis) (2016 – 2018) autour du système autonomique et gestion de services IoT dans les villes intelligentes ainsi que l'intégration de gestion d'objets (*Device Management*) avec LWM2M. Ces présentations étaient en collaboration avec différents partenaires industriels dans le cadre du projet S2C2 et du projet STM (AToS, M3Systems, eDevice, Orange, SRC Solutions).

18. *(en cours de soumission)*

Perspectives

Court-terme : améliorations et développements

Instance d'un modèle plus générique (ontologie)

Afin de pouvoir traiter des scénarios plus divers et travailler à un modèle d'ontologie plus générique, des pistes d'améliorations de l'instance d'ontologie réalisée sont à étudier. Le but serait toujours de traiter des cas dans un contexte de ville intelligente et de systèmes connectés mais de généraliser l'instance de modèle proposé dans le cas d'utilisation du bus connecté pour une approche similaire plus générale. En effet, le comportement de mutualisation de services et de planification dynamique permettent le développement de nouvelles applications composites dynamiques haut-niveau particulièrement intéressantes dans un contexte de ville intelligente. Pour ce faire, une généralisation des concepts instanciés dans l'extension d'IoT-O proposée dans le Chapitre 2, Section 2.2 est en cours de réflexion.

Une ébauche de travail dans cette optique a débuté afin de généraliser les concepts de *producteur de contenu* et de *consommateur de contenu* en considérant des concepts plus généraux d'*activateurs (enabler)* permettant de récupérer des informations, réaliser certaines actions, etc.

L'amélioration de la conscience du contexte *(context awareness)* dans l'environnement est également importante afin d'enrichir les capacités d'analyse du modèle sémantique.

Intégration de paramètres QoS dans le modèle sémantique

À ce jour, la gestion des paramètres a été intégrée dans l'instance de modèle de grammaire de graphe afin de permettre une optimisation des plans d'exécutions. Cependant, cette gestion n'a pas encore été intégrée dans l'instance d'ontologie utilisée pour l'analyse. Une piste intéressante qui se dégage serait de rajouter une extension d'IoT-O et de la notion de *service* afin d'intégrer des paramètres QoS rattachés à un service. Des contributions en ce sens ont été développées dans [Chhun 2016] qui semble un modèle (ontologie) prometteur pour permettre cette intégration. L'étude détaillée de cette contribution ainsi que sa potentielle intégration dans le modèle sémantique est une piste intéressante.

Intégration d'autres opérateurs

Des pistes de travail sur d'autres opérateurs sont à explorer et à intégrer au modèle.

Choix exclusif (OU exclusif) : cet opérateur noté *xor* serait un *or* particulier indiquant que si l'une des branches est sélectionnée il n'est pas possible d'avoir recours aux autres. Par exemple cet opérateur peut être utilisé dans le cas où deux services interdépendants (exécutés en monopolisant le même objet par exemple) sont utilisés dans un enchaînement différent dans deux branches et il ne sera pas possible

d'exécuter une branche si l'autre s'exécute car cela va condamner son exécution, le service en question étant alors indisponible.

Condition (IF) : il serait intéressant d'intégrer un opérateur conditionnel afin de dérouter automatiquement un flux suivant une condition. Cet opérateur serait plus complexe à mettre en œuvre dans la grammaire mais apporterait plus de souplesse pour l'exécution.

Plus long terme

Pour un travail à plus long terme, différentes pistes sont considérées.

Raffinement de besoin utilisateur haut niveau

L'idée est d'enrichir l'approche pour permettre une génération automatisée des politiques haut-niveau à partir d'un besoin exprimé par l'utilisateur. Ce mécanisme se baserait sur une requête utilisateur exprimée sous forme logique ou en langage naturel.

Le principe serait le suivant :

1. Raffinement du besoin utilisateur (des mécanismes similaires ont été utilisés dans [Zhovtobryukh 2007]). Dans le cas d'utilisation d'un besoin utilisateur, des techniques de la littérature comme dans [Pradel 2014] pourraient être utilisées afin de permettre de traduire ce besoin en quelque chose de compréhensible par le système
2. Instanciation de règles sémantiques et de grammaire de graphe de manière automatique à partir de patrons haut-niveau de règles afin de pouvoir traiter un besoin ou cas plus spécifique allant jusqu'à la génération du plan d'exécution final

Une autre approche pourrait être de définir un modèle (ontologie) pour le besoin et réaliser un alignement de concepts suivant les éléments du besoin et la correspondance dans le modèle de système autonomique.

Scalabilité et déploiement

Afin de pouvoir déployer l'approche proposée dans la thèse à une plus grande échelle, il est intéressant de considérer des problématiques de scalabilité et de déploiement distribué. La combinaison de systèmes autonomiques pouvant également se gérer entre-eux (sous la forme d'un contrôleur et de sous systèmes autonomiques en charge de sous parties du système, *e.g.*, embarqué dans les bus) est une possibilité. En parallèle, une intégration de l'approche proposée dans [Seydoux 2018] permettrait de distribuer la partie concernant le raisonnement sémantique et où se situeraient les différents éléments de raisonnement dans la topologie distribuée. L'intégration de telles mécaniques permettrait un déploiement plus vaste en réalisant un compromis de déploiement au niveau des composants de raisonnement sémantique notamment. L'approche pourrait être étendue également à la transformation de graphes.

Acronymes

API Application Programming Interface. 7, 100, 102, 120

BPEL Business Process Execution Language. 11, 15
BPMN Business Process Model and Notation. 11, 16

CoAP Constrained Application Protocol. 6

GA Genetic Algorithm. 14

HTTP HyperText Transfer Protocol. 6, 7

IoT Internet of Things. 1

JSON Javascript Object Notation. 7

KB Knowledge base. 38

LWM2M LightWeight M2M. 8, 101

M2M Machine-to-Machine. 7, 20, 119
MQTT Message Queuing Telemetry Transport. 7

OWL Web Ontology Language. 23, 90

QoS Quality of Service. 10, 13, 14, 15, 16, 17, 33, 56, 57, 84

RDF Ressource Description Framework. 23, 90
REST REpresentationnal State Transfer. 6, 7, 8, 9, 16, 37, 41, 100, 105

SWRL Semantic Web Rule Language. 21, 23, 44, 90

TCP Transmission Control Protocol. 6

UDP User Datagram Protocol. 6

XML eXtensible Markup Language. 7

Annexe A

Les standards, consortiums et plateformes de l'IoT

A.1 Les standards et consortiums

A.1.1 ETSI Smart M2M

SmartM2M est un standard transverse pour la standardisation de communications M2M. Ce standard a été proposé originalement par l'ETSI en 2009 et a abouti à une première spécification en 2011 [1] [2] Le standard est toujours supporté par l'ETSI mais sa progression a été suspendue afin de supporter pleinement l'initiative oneM2M.

A.1.2 Open Connectivity Foundation (OCF)

L'Open Connectivity Foundation (OCF) est l'une des plus grandes organisations de normes de connectivité industrielle pour l'IoT qui vise à développer des normes et des certifications pour les dispositifs impliqués dans l'IoT en se basant sur le protocole CoAP afin d'assurer une certaine interopérabilité. L'OCF unifie l'intégralité de l'ancien Open Interconnect Consortium (OIC) [3]. L'OCF offre un cadre qui permet une découverte facile et une connectivité fiable entre les objets via une spécification, une implémentation de référence et un programme de certification. L'OCF crée une spécification et sponsorise un projet open source pour permettre la communication entre des milliards de périphériques connectés (téléphones, capteurs, ordinateurs,...) quel que soit le fabricant, le système d'exploitation, le chipset ou le transport physique. L'OCF sponsorise le projet open source IoTivity. Récemment, l'OCF a approuvé une fusion avec AllSeen Alliance (qui fournit le framework open source IoTALLJoyn).

A.1.3 ICC *(Industrial internet Consortium*

L'IIC (Industrial Internet Consortium) [4] est un partenariat mondial sans but lucratif de l'industrie, gouvernement et universités qui vise à accélérer le développement et l'adoption de dispositifs et de machines interconnectés. Fondée en Mars

1. http://www.etsi.org/technologies-clusters/technologies/internet-of-things
2. http://www.smartm2msolutions.com/
3. https://openconnectivity.org/
4. http://www.iiconsortium.org/

2014 par AT&T, Cisco, General Electric, IBM et Intel, l'IIC permet de rassembler les organisations et les technologies nécessaires pour accélérer la croissance de l'Internet industriel en identifiant, rassemblant et en promouvant les meilleures pratiques et en coordonnant les priorités et les technologies habilitantes de l'Internet industriel.

A.1.4 Thread, Weave *(Google)*

Thread [5] est un réseau maillé (tous les appareils servent de relais pour les autres) basé sur IPv6 qui permet de connecter et contrôler plusieurs produits dans une maison. Il peut connecter en toute sécurité plus de 250 périphériques dans une maison en un seul réseau maillé. Il assure une meilleure fiabilité puisque le réseau est de type maillé (tous les appareils servent de relai pour les autres) ; une latence intéressante (moins de 100 ms) ; une meilleure portée (puisque chaque appareil relai le signal en Thread) ; une faible consommation énergétique (plusieurs années d'autonomie), mais aussi de mémoire vive (64ko de RAM minimum) ; une meilleure sécurité et une configuration simple. Initialement annoncé en Juillet 2014 et publié en Juillet 2015, Thread a été développé par un groupe de sociétés, dont Nest, qui a vu la nécessité d'un nouveau protocole de réseau IP maillé pour la maison connectée. Le Groupe Thread a maintenant plus de 200 sociétés membres.

NestWeave [6] est un protocole d'application conçu pour fonctionner sur les réseaux IPv6, tels que Thread. Il est utilisé par Google afin de permettre l'interaction et la communication directe entre plusieurs produits Nest. Le protocole se base sur deux normes de communication sans fil existantes : le Wi-Fi 802.11, mais également le standard Thread 802.15.4 aussi utilisé par ZigBee par exemple.

A.1.5 Homekit *(Apple)*

Homekit [7] est un framework qui a pour but permettre la communication et le contrôle d'accessoires connectés notamment dans une maison connectée. Les utilisateurs peuvent ainsi interagir de manière simplifiée et configurer les différents accessoires labellisés Homekit dans leur maison et créer des scénarios intégrant différents objets et différentes technologies.

Le but de ce framework est de permettre un regroupement d'informations via une application iOS et faciliter la communication entre différents objets. Homekit fournit également une API pour la configuration et la communication avec les accessoires pour permettre leur intégration facilitée dans des applications.

5. `https://threadgroup.org`
6. `https://nest.com/weave/`
7. `https://www.apple.com/fr/ios/home/`

A.1.6 FiWARE

FiWARE [8] est à la fois une communauté ouverte mais aussi une plateforme. La communauté FiWARE se veut indépendante et ouverte afin de construire un écosystème ouvert dans l'IoT et de faciliter le développement d'applications innovantes IoT. La plateforme FiWARE regroupe des outils et des API ouvertes afin de faciliter le développement d'applications impliquant différents secteurs verticaux.

A.1.7 OSGi

L'*Open Service Gateway initiative* (OSGi) *Alliance* [9] définit un ensemble de services pour le développement de plateformes en se basant sur le langage Java pour la portabilité. L'Alliance OSGi définit un framework avec différentes couches d'API et de services. L'intérêt de ce type d'architecture est qu'elle permet le développement de *bundles* (composants logiciels déployables à chaud) qui peuvent implémenter des services définis par le standard ou personnalisés. Il est également possible de gérer le cycle de vie des *bundles* (installation, désinstallation, mise à jour, démarrage, arrêt, et ce même à chaud). Un groupe de travail dédié à l'IoT [10] a également été créé dans le but de normaliser l'interconnexion avec des technologies de l'IoT sous forme de plug-ins OSGi.

8. `https://www.fiware.org`
9. `https://www.osgi.org`
10. `https://www.osgi.org/about-us/working-groups/internet-of-things/`

Annexe B

Compléments système autonomique

B.1 Compléments informations boucle autonomique

B.1.0.1 Exemple d'ensemble d'évènements et de symptômes associés

B.2 Compléments grammaires de graphes

Exemple 11 ***Exemple de transformation de graphe plus complexe***
Dans la Figure B.1, un exemple de transformation est détaillé sur un graphe type.

Ici, un exemple de transformation partielle est présenté. Le graphe initial est présenté en haut de la Figure et le graphe présenté en bas est le graphe transformé. Ce graphe transformé est obtenu en appliquant les règle 1 à 6 de manière séquentielle (*i.e.*, chaque règle est appliquée autant que possible et quand elle n'est plus applicable le système passe à la règle suivante). Certains nœuds, (montrés en lignes discontinues sur le schéma) sont voués à être supprimés ensuite du graphe par la troisième couche de règles de la grammaire.

B.3 Compléments grammaire de graphes complexe

ÉVÉNEMENTS	SYMPTÔMES ASSOCIÉS
Nouvelle entité dans le système (Producteur ou Consommateur)	Nouvelle entité considérée
Nouvel intérêt exprimé par une entité	Nouvel intérêt exprimé (potentiellement contenu à envoyer)
Nouvel intérêt associé à un contenu	Nouvel intérêt exprimé (potentiellement contenu à envoyer)
Nouveau contenu mis à disposition / répertorié	Si intérêts exprimés : même symptôme que dessus
Nouveau terminal d'affichage connecté	Nouveau terminal disponible (potentiellement nouveau contenu à envoyer)
Terminal d'affichage déconnecté	Terminal déconnecté
Retrait d'une entité (P / C) du système	Entité retirée (potentiellement besoin de ne plus prendre en compte les contenus produits / les terminaux enregistrés)
Une entité change de position	Nouvelle position associée (si ce changement de position est significatif, sinon non pris en compte)
Une entité exprime un nouvel intérêt	Nouvel intérêt exprimé (potentiellement du contenu à envoyer)
Une personne monte dans un bus / tram	Personne dans bus x
Une personne descend d'un bus / tram	Personne n'est plus dans bus x
Mise à jour des métadonnées sur un contenu (intérêt, position liée, ...)	Dépend du type de méta donnée mise à jour : Si intérêt : nouvel intérêt exprimé, Si position : nouvelle position (contenu, pas entité)
Un contenu a été affiché sur un terminal avec succès	Contenu C affiché sur terminal T
Une erreur est survenue lors de l'affichage sur un terminal	Erreur durant exécution (contenu c)

TABLE B.1 – Exemples d'évènements pouvant survenir sur un système IoT et les symptômes associés dans la boucle MAPE-K

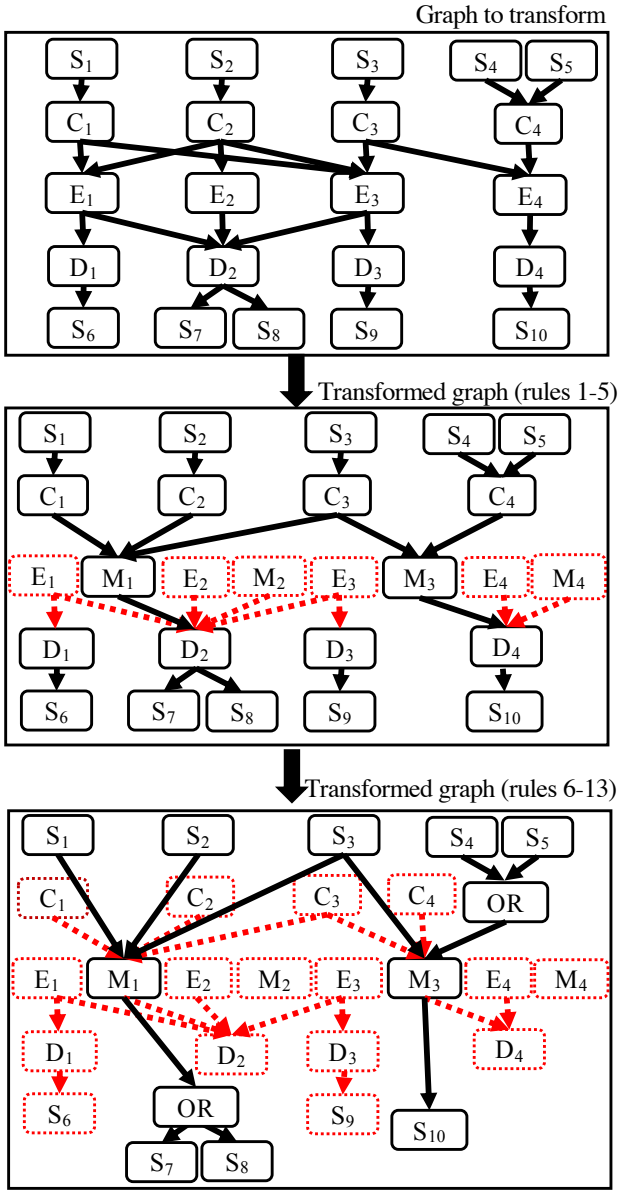


FIGURE B.1 – Exemple de transformation de graphe

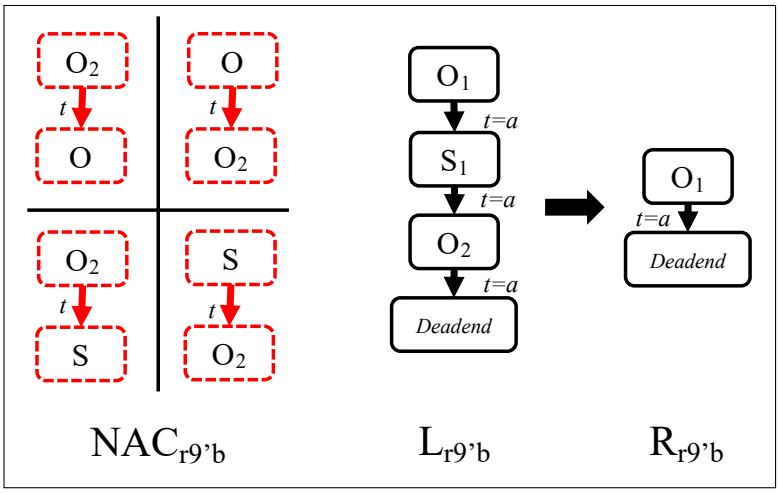


FIGURE B.2 – Règle $9'_b$: Propagation d'un nœud *deadend* dans la branche concernée.

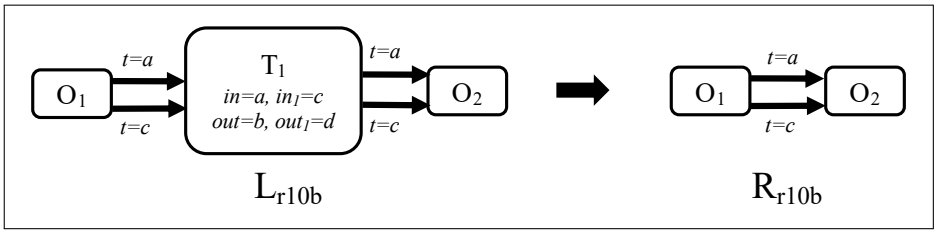


FIGURE B.3 – Règle 10_b : Suppression d'un nœud temporaire avec entrées et sorties identiques.

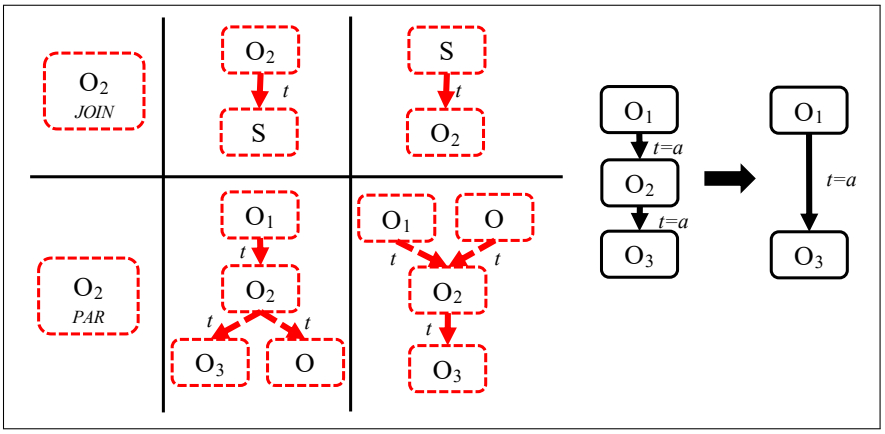


FIGURE B.4 – Règle 11_b : Suppression d'un opérateur inutile

Annexe C

Compléments algorithmes d'exploration

C.1 Algorithme de recherche d'optimum

C.1.1 Structure de données *Result*

La structure de données *Result* permet de stocker les nœuds sous la forme d'une *HashMap* permettant de récupérer le coût connu depuis le nœud *Begin* jusqu'à ce nœud. La structure comporte également un tas trié (*SortedSet*) dans lequel les nœuds à traiter sont insérés en triant le tas par coût du plus bas au plus élevé.

Différentes méthodes sont disponibles :

updateCost(id, cost) : cette méthode permet de mettre à jour le coût stocké connu du nœud concerné. Cela ne concerne pas le tas trié qui ne contient que les nœuds encore à traiter.

getKnownCost(node) : permet de récupérer le coût connu stocké dans la HashMap pour le nœud concerné.

setPrevious[id_1, id_2 :] permet d'enregistrer le meilleur nœud qui doit précéder le nœud cible dans le plan final.

remove(id) : cette méthode permet de retirer un nœud du tas de nœuds à trier.

getLightestNode() : cette méthode permet de récupérer l'identifiant du nœud ayant le coût le plus faible dans le tas de nœuds encore à traiter.

C.1.2 Pseudo-code recherche d'optimum

Signature de l'algorithme :

Algorithm 4 Pseudo code d'implémentation de l'algorithme de meilleur plan

```
Result optimumSearch(Graph graph, Node start, Node end,
Map<String, String> parJoinMap) {
  Result res := new Result() ;
  res.updateCost(start, new Cost(0)) ;
  String toProcessId := res.getLightestNode() ;
  while (toProcessId != null AND !toProcessId.equals(end)) do
    Node source := graphe.getNode(toProcessId) ;
    res.remove(toProcessId) ;
    if (source.isOperator() AND source.getOpType().equals("par")) then
      computeNode(source, res, parJoinMap) ;
    else
      for (Edge e : source.getLeavingEdges() do
        Node target := e.getTargetNode() ;
        Cost knownCost := res.getKnownCost(target) ;
        Cost toalCost := res.getKnownCost(source) ;
        Cost targetCost := target.getCost() ;
        targetCost.setCost(targetCost + totalCost) ;
        if ( !target.isDeadEnd() AND targetCost < knownCost) then
          res.updateCost(target, targetCost) ;
          res.setPrevious(target.getId(), source.getId()) ;
        end if
      end for
    end if
    toProcess := res.getLightestNode() ;
  end while
  return res ;
}
```

C.1.3 Autres méthodes

Algorithm 5 Traitement récursif d'un nœud (cas parallèle)

```
void computeNode(Node par, Result currentRes,
Map<String, String> parJoinMap) {
  if ((start.isOperator()) then
    structure.updateCost(start.getId(), new Cost(0, 0, 0));
  else if (start.isService()) then
    Cost startCost := getCost(start);
    currentRes.updateCost(start.getId(), startCost);
  else
    return false; // deadend case
  end if
  Cost toStore := structure.getKnownCost(source);
  Cost joinCost := null;
  Cost temp := new Cost(0);
  String joinId := parJoinMap.get(source.getId);
  Node join := graph.getNode(joinId);
  String previousId := "";
  Result resultPar := new Result();
  for (Edge e : source.getLeavingEdges() do
    resultPar.remove(joinId);
    resultPar.updateCost(joinId, +∞);
    boolean pathFound;
    pathFound := computeNode(graph, e.getTargetNode(), join, resultPar, par-
    JoinMap); //recursive search
    if (!pathFound) then
      temp := new Cost(+∞);
      break;
    else
      joinCost := resultPar.getKnownCost(join);
      if (joinCost.getQos0() > temp.getQos0()) then
        temp.setQos0(joinCost.getQos0());
      end if
      temp.setQos1(temp.getQos1() + joinCost.getQos1());
      temp.setQos2(temp.getQos2() + joinCost.getQos2());
      previousId += resultPar.getPrevious(joinId) +",";
    end if
  end for
  Cost newJoinCost := new Cost(0);
  newJoinCost := temp + toStore;
  res.updateCost(join, newJoinCost);
  if (newJoinCost < +∞) then
    res.setPrevious(joinId, previousId);
  else
    res.remove(joinId);
  end if
}
```

Annexe D

Compléments étude d'évaluation

D.1 Résultats détaillés

D.1.1 Analyse de performance de la grammaire avancée et des algorithmes d'exploration

Nb services	5	10	15	20	25	30
Besoin simple	217,3	534,3	1245,9	3760,2	4996,2	9575,7
Besoin double	558,1	1136,7	2884,7	5519,1	9716,2	12266,2
Global	396,6	840,6	2156,3	4639,6	7356,2	10943,7
Plan exécutable	237,0	1141,6	2394,0	4513,1	7768,0	10782,8
Pas de plan exécutable	399,5	774,2	2040,8	4734,2	6738,5	11376,3

Table D.1 – Valeurs de temps moyen (ms) de transformation de graphe dans le cas global et dans le cas où le besoin est simple ou double

D.2 Compléments d'information Plug-in LWM2M

Les différents composants de l'IPE LWM2M (Figure D.1) vont permettre de réaliser l'interface entre la plateforme OM2M et un serveur LWM2M, Leshan, embarqué dans l'IPE. À l'activation, l'IPE va lancer le serveur LWM2M embarqué et attendre d'éventuelles connexions de clients LWM2M. Ces clients sont exécutés sur les objets physiques et vont permettre de remonter les informations de *Device Management* au serveur Leshan. Comme indiqué sur la Figure D.2, quand un nouveau client s'enregistre sur le serveur, l'IPE va instancier des ressources oneM2M dans la plateforme OM2M et s'abonner aux futures modifications. Dans le cas où une nouvelle valeur d'observation est envoyée au serveur Leshan par un client, l'IPE va mettre à jour les valeurs côté OM2M.

Ce fonctionnement permet de centraliser des informations de gestion des objets dans la plateforme oneM2M peu importe le protocole (propriétaire, LWM2M, autre, etc.) utilisé.

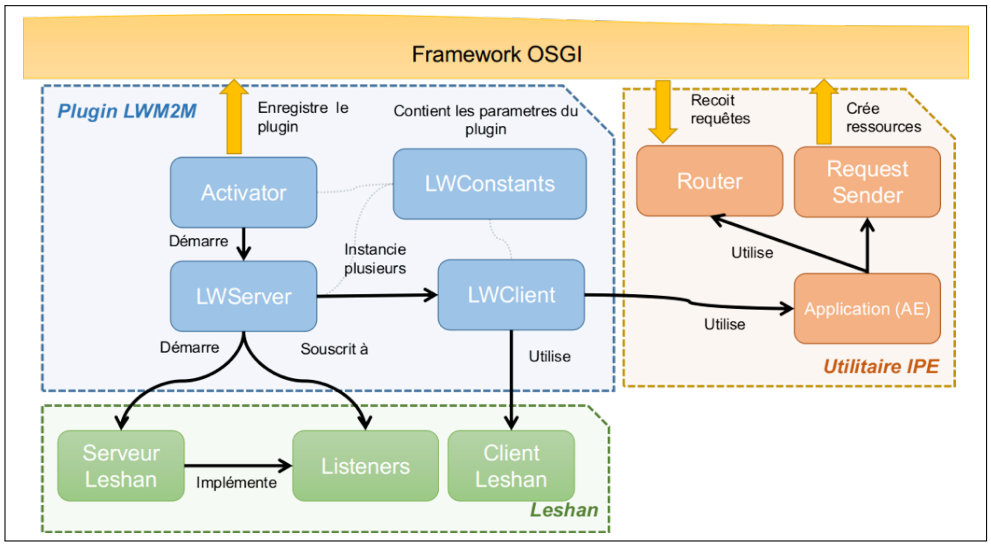


FIGURE D.1 – Composants du plug-in (IPE) LWM2M

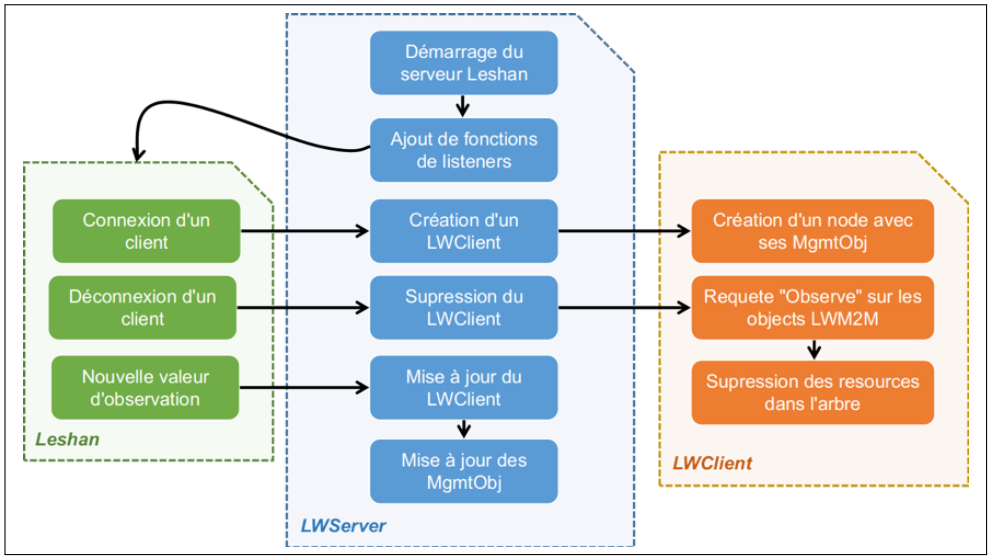


FIGURE D.2 – Processus de fonctionnement de l'IPE LWM2M

Bibliographie

[Aïssaoui 2017] François Aïssaoui, Guillaume Garzone et Nicolas Seydoux. *Providing Interoperability for Autonomic Control of Connected Devices.* Dans Nathalie Mitton, Hakima Chaouchi, Thomas Noel, Thomas Watteyne, Alban Gabillon et Patrick Capolsini, editeurs, Interoperability, Safety and Security in IoT. InterIoT 2016, SaSeIoT 2016. LNICST, volume 190, chapitre InterIoT, pages 33–40. Springer, Cham, Paris, 2017. (Cité en page 3.)

[Aljawarneh 2016] Mahmoud Aljawarneh, Lachhman Das Dhomeja et Yasir Arfat Malkani. *Context-aware Service Composition of Heterogeneous Services in Pervasive Computing Environments : A Review.* no. 1, pages 0–5, 2016. (Cité en pages 13 et 14.)

[Alsaryrah 2018] Osama Alsaryrah, Ibrahim Mashal et Tein Yaw Chung. *Energy-aware services composition for Internet of Things.* IEEE World Forum on Internet of Things, WF-IoT 2018 - Proceedings, vol. 2018-Janua, pages 604–608, 2018. (Cité en pages 13 et 16.)

[Ardagna 2007] Danilo Ardagna et Barbara Pernici. *Adaptive Service Composition in Flexible Processes.* IEEE Transactions on Software Engineering, vol. 33, no. 6, pages 369–384, 6 2007. (Cité en pages 13 et 14.)

[Asghari 2018] Saied Asghari et Nima Jafari Navimipour. *Nature inspired meta-heuristic algorithms for solving the service composition problem in the cloud environments.* International Journal of Communication Systems, no. January, page e3708, 5 2018. (Cité en pages 13 et 14.)

[Bali 2009] Michal Bali. Drools JBoss Rules 5.0 Developer's Guide. Packt Publishing Ltd, 2009. (Cité en page 20.)

[Barros 2005] Alistair Barros, Marlon Dumas et Phillipa Oaks. *Standards for web service choreography and orchestration : Status and perspectives.* Dans International Conference on Business Process Management, pages 61–74. Springer, 2005. (Cité en page 11.)

[Ben Alaya 2014] Mahdi Ben Alaya. *Towards Interoperability, Self-Management, and scalability for Machine-to-Machine Systems.* PhD thesis, INSA de Toulouse, 2014. (Cité en page 20.)

[Bermudez-Edo 2017] Maria Bermudez-Edo, Tarek Elsaleh, Payam Barnaghi et Kerry Taylor. *IoT-Lite : a lightweight semantic model for the internet of things and its use with dynamic semantics.* Personal and Ubiquitous Computing, vol. 21, no. 3, pages 475–487, 2017. (Cité en page 24.)

[Berners-Lee 2001] Tim Berners-Lee, James Hendler et Ora Lassila. *THE SEMANTIC WEB.* Scientific american, vol. 284, no. 5, pages 34–43, 2001. (Cité en page 22.)

[Boudol 1992] Gérard Boudol. *Asynchrony and the pi-calculus.* PhD thesis, INRIA, 1992. (Cité en page 14.)

[Cai 2014] Hongming Cai, Lida Xu, Boyi Xu, Cheng Xie, Shaojun Qin et Lihong Jiang. *IOT-based Configurable Information Service Platform for Product Lifecycle Management.* IEEE Transactions on Industrial Informatics, vol. PP, no. 2, pages 1–1, 2014. (Cité en page 13.)

[Campos 2014] Glaucia Melissa Medeiros Campos, Nelson Souto Rosa et Luis Ferreira Pires. *A survey of formalization approaches to service composition.* Dans 2014 IEEE International Conference on Services Computing (SCC), pages 179–186. IEEE, 2014. (Cité en page 11.)

[Chafle 2006] Girish Chafle, Koustuv Dasgupta, Arun Kumar, Sumit Mittal et Biplav Srivastava. *Adaptation in Web Service composition and execution.* Proceedings - ICWS 2006 : 2006 IEEE International Conference on Web Services, pages 549–557, 2006. (Cité en pages 13 et 15.)

[Chan 2007] K S May Chan, Judith Bishop et Luciano Baresi. *Survey and comparison of planning techniques for web services composition.* Africa, no. October, pages 43–54, 2007. (Cité en pages 13 et 14.)

[Chattopadhyay 2018] Soumi Chattopadhyay et Ansuman Banerjee. *QoS aware Automatic Web Service Composition with Multiple objectives.* pages 1–15, 2018. (Cité en pages 13 et 15.)

[Cheng 2015] Bin Cheng, Salvatore Longo, Flavio Cirillo, Martin Bauer et Ernoe Kovacs. *Building a Big Data Platform for Smart Cities : Experience and Lessons from Santander.* Dans 2015 IEEE International Congress on Big Data, pages 592–599. IEEE, 6 2015. (Cité en page 2.)

[Cherrier 2014] Sylvain Cherrier, Yacine Ghamri-doudane, Sylvain Cherrier, Yacine Ghamri-doudane The et Object-as-a-service Global. *The " Object-as-a-Service " paradigm.* Dans Global Information Infrastructure and Networking Symposium 2014, page 1, Montréal, Canada, 2014. (Cité en page 1.)

[Chhun 2016] Sophea Chhun, Néjib Moalla et Yacine Ouzrout. *QoS ontology for service selection and reuse.* Journal of Intelligent Manufacturing, vol. 27, no. 1, pages 187–199, 2 2016. (Cité en page 115.)

[Dai 2017] Wenbin Dai, Victor N. Dubinin, James H. Christensen, Valeriy Vyatkin et Xinping Guan. *Toward Self-Manageable and Adaptive Industrial Cyber-Physical Systems With Knowledge-Driven Autonomic Service Management.* IEEE Transactions on Industrial Informatics, vol. 13, no. 2, pages 725–736, 4 2017. (Cité en page 22.)

[Daniele 2016] Laura Daniele, Monika Solanki, Frank Den Hartog et Jasper Roes. *Interoperability for smart appliances in the IoT world.* Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9982 LNCS, pages 21–29, 2016. (Cité en page 24.)

[del Carmen Suarez de Figueroa Baonza 2010] Maria del Carmen Suarez de Figueroa Baonza. *Neon methodology for building ontology networks : Specification, Scheduling and Reuse*. PhD thesis, 2010. (Cité en page 25.)

[Delgado 2004] Nelly Delgado, A.Q. Gates et Steve Roach. *A taxonomy and catalog of runtime software-fault monitoring tools*. IEEE Transactions on Software Engineering, vol. 30, no. 12, pages 859–872, 12 2004. (Cité en page 40.)

[Dijkstra 1959] Edsger W Dijkstra. *A note on two problems in connexion with graphs*. Numerische mathematik, vol. 1, no. 1, pages 269–271, 1959. (Cité en page 79.)

[Drira 2017] Khalil Drira. *Multiscale and Multiobjective modelling : a perspective for mastering the design and operation complexity of IoT Systems*. Dans Modeling System Behaviour session Mipro The 40th Jubilee International ICT Convention – MIPRO 2017, editeur, Mipro - The 40th Jubilee International ICT Conven- tion – MIPRO 2017, Modeling System Behaviour session, Opattija, 2017. (Cité en page 27.)

[Dunst 2002] Carl J Dunst et Mary Beth Bruder. *Valued outcomes of service coordination, early intervention, and natural environments*. Exceptional children, vol. 68, no. 3, pages 361–375, 2002. (Cité en page 11.)

[Ehrig 1997] H Ehrig, R Heckel, M Korff, M Löwe, L Ribeiro et A Wagner. *Algebraic Approaches to Graph Transformation : Part II : Single Pushout Approach and Comparison with Double Pushout Approach*. Handbook of graph grammars and computing by graph transformation, pages 247 – 312, 1997. (Cité en page 26.)

[Eichler 2014] Cédric Eichler, Thierry Monteil, Patricia Stolf, Luigi Alfredo Grieco et Khalil Drira. *Enhanced graph rewriting systems for complex software domains*. Software {&} Systems Modeling, pages 1–21, 2014. (Cité en page 27.)

[Eichler 2015] Cedric Eichler. *Modélisation formelle de systèmes dynamiques autonomes : graphe, réécriture et grammaire*. PhD thesis, 2015. (Cité en pages 26 et 27.)

[Erol 1996] Kutluhan Erol. *Hierarchical task network planning : formalization, analysis, and implementation*. PhD thesis, 1996. (Cité en page 14.)

[Fähndrich 2016] Johannes Fähndrich, Tobias Küster, F Johannes, K Tobias et Nils Masuch. *Semantic Service Management and Orchestration for Adaptive and Evolving Processes*. International Journal on Advances in Internet Technology, no. January, 2016. (Cité en pages 13 et 16.)

[Garzone 2018] Guillaume Garzone, Nawal Guermouche et Thierry Monteil. *Autonomic Management Approach for Dynamic Service Based IoT Systems*. Dans 2018 International Symposium on Networks, Computers and Communications (ISNCC) : Internet of Everything, Data Analytics and Smart Cities (ISNCC-2018 IoE-DASC), Rome, Italy, 2018. (Cité en pages 54, 109, 112 et 114.)

[Gharbi 2012] Ghada Gharbi, Mahdi Ben Alaya, Codé Diop et Ernesto Exposito. *AODA : An autonomic and ontology-driven architecture for service-oriented and event-driven systems.* Proceedings of the Workshop on Enabling Technologies : Infrastructure for Collaborative Enterprises, WETICE, pages 72–77, 2012. (Cité en page 21.)

[Gortmaker 2004] Jeffrey Gortmaker, Marijn Janssen et René W. Wagenaar. *The advantages of web service orchestration in perspective.* Proceedings of the 6th international conference on Electronic commerce - ICEC '04, page 506, 2004. (Cité en page 10.)

[Gronvall 2011] E Gronvall, M Ingstrup, M Ploger et M Rasmussen. *REST based service composition : Exemplified in a care network scenario.* Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on, pages 251–252, 2011. (Cité en pages 13 et 16.)

[Gruber 1991] Tr Gruber. *The role of common ontology in achieving sharable, reusable knowledge bases.* Principles of Knowledge Representation and Reasoning : Proceedings of the Second International Conference, pages 601–602, 1991. (Cité en pages 22 et 39.)

[Gubbi 2013] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic et Marimuthu Palaniswami. *Internet of Things (IoT) : A vision, architectural elements, and future directions.* Future generation computer systems, vol. 29, no. 7, pages 1645–1660, 2013. (Cité en page 1.)

[Guérout 2014] Tom Guérout. *Ordonnancement sous contraintes de qualité de service dans les clouds. (Cloud scheduling under quality of service constraints).* PhD thesis, {INSA} Toulouse, France, 2014. (Cité en page 74.)

[Guidara 2016] Ikbel Guidara, Imane Al Jaouhari et Nawal Guermouche. *Dynamic Selection for Service Composition Based on Temporal and QoS Constraints.* Dans 2016 IEEE International Conference on Services Computing (SCC), pages 267–274. IEEE, 6 2016. (Cité en page 15.)

[Guidara 2017] Ikbel Guidara, Nawal Guermouche, Tarak Chaari, Ikbel Guidara et Nawal Guermouche. *Heuristic based Time-aware Service Selection Approach.* 2017. (Cité en pages 13 et 15.)

[Guinard 2010] Dominique Guinard, Vlad Trifa, Stamatis Karnouskos, Patrik Spiess et Domnic Savio. *Interacting with the SOA-Based Internet of Things : Discovery, Query, Selection, and On-Demand Provisioning of Web Services.* IEEE Transactions on Services Computing, vol. 3, no. 3, pages 223–235, 7 2010. (Cité en pages 13 et 16.)

[Guo 2017] Xing Guo, Shanshan Chen, Yiwen Zhang et Wei Li. *Service Composition Optimization Method Based on Parallel Particle Swarm Algorithm on Spark.* Security and Communication Networks, vol. 2017, no. 1, pages 1–9, 2017. (Cité en pages 13 et 15.)

[Helal 2011] Sumi Helal, Raja Bose, Chao Chen, Andy Smith, Scott De Deugd et Diane Cook. *STEPSTONE : An Intelligent Integration Architecture for*

Personal Tele-Health. Journal of Computing Science and Engineering, vol. 5, no. 3, pages 269–281, 2011. (Cité en page 18.)

[Higashino 2016] Wilson A. Higashino, Cedric Eichler, Miriam A. M. Capretz, Luiz F. Bittencourt et Thierry Monteil. *Attributed Graph Rewriting for Complex Event Processing Self-Management.* ACM Transactions on Autonomous and Adaptive Systems, vol. 11, no. 3, pages 1–39, 2016. (Cité en page 27.)

[Horrocks 2004] Ian Horrocks, Peter F Patel-schneider, Harold Boley, Said Tabet, Benjamin Grosof et Mike Dean. *SWRL : A Semantic Web Rule Language Combining OWL and RuleML.* W3C Member submission 21, no. May 2004, pages 1–20, 2004. (Cité en pages 21 et 45.)

[Hutchison 2005] David Hutchison et John C Mitchell. Semantic Web Services and Web Process Composition, volume 3387 of *Lecture Notes in Computer Science.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. (Cité en page 23.)

[IBM 2006] IBM. *An architectural blueprint for autonomic computing.* IBM White Paper, vol. 31, pages 1–6, 2006. (Cité en pages 18 et 20.)

[Jensen 1981] Kurt Jensen. *Coloured Petri nets and the invariant-method.* Theoretical computer science, vol. 14, no. 3, pages 317–336, 1981. (Cité en page 14.)

[Jin Xiao 2005] Jin Xiao et Raouf Boutaba. *QoS-aware service composition and adaptation in autonomic communication.* IEEE Journal on Selected Areas in Communications, vol. 23, no. 12, pages 2344–2360, 12 2005. (Cité en pages 13 et 14.)

[Kephart 2003] Jeffrey O. Kephart et David M. Chess. *The vision of autonomic computing.* Computer, vol. 36, no. 1, pages 41–50, 1 2003. (Cité en pages 18, 20 et 39.)

[Kim 2014] Youngjun Kim, Sanghum Lee et Ilyoung Chong. *Orchestration in Distributed Web-of-Objects for Creation of User-Centered IoT Service Capability.* Wireless Personal Communications, vol. 78, no. 4, pages 1965–1980, 2014. (Cité en pages 13 et 16.)

[Kopecký 2008] Jacek Kopecký, Karthik Gomadam et Tomas Vitvar. *hRESTS : An HTML microformat for describing RESTful Web services.* Proceedings - 2008 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2008, pages 619–625, 2008. (Cité en page 25.)

[Küster 2009] Tobias Küster et Axel Heßler. *Towards transformations from bpmn to heterogeneous systems.* Lecture Notes in Business Information Processing, vol. 17 LNBIP, pages 200–211, 2009. (Cité en page 11.)

[Liu 2009] Xiangwei Liu, Zhicai Xu et Li Yang. *Independent Global Constraints-aware Web Service Composition Optimization Based on Genetic Algorithm.* Dans 2009 International Conference on Industrial and Information Systems, pages 52–55. IEEE, 4 2009. (Cité en pages 13 et 14.)

[Liu 2016] Zhi-zhong Liu, Dian-hui Chu, Zong-pu Jia, Ji-quan Shen et Lei Wang. *Two-stage approach for reliable dynamic Web service composition.* Knowledge-Based Systems, vol. 97, pages 123–143, 4 2016. (Cité en page 14.)

[Mosincat 2011] Adina Mosincat et Walter Binder. *Automated maintenance of service compositions with SLA violation detection and dynamic binding.* International Journal on Software Tools for Technology Transfer, vol. 13, no. 2, pages 167–179, 4 2011. (Cité en pages 13, 15 et 40.)

[Nami 2007] Mohammad Reza Nami et Koen Bertels. *A Survey of Autonomic Computing Systems.* Dans Third International Conference on Autonomic and Autonomous Systems (ICAS'07), volume 228, pages 26–26. IEEE, 6 2007. (Cité en page 19.)

[Nordrum 2016] Amy Nordrum. *Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated.* IEEE Spectrum, 2016. (Cité en page 1.)

[Palacios 2011] Marcos Palacios, José García-Fanjul et Javier Tuya. *Testing in Service Oriented Architectures with dynamic binding : A mapping study.* Information and Software Technology, vol. 53, no. 3, pages 171–189, 3 2011. (Cité en page 40.)

[Pradel 2014] Camille Pradel, Ollivier Haemmerlé et Nathalie Hernandez. *Swip : A Natural Language to SPARQL Interface Implemented with SPARQL.* vol. 8577, pages 260–274, 2014. (Cité en page 116.)

[Qiu 2015] Xuan Qiu, Hao Luo, Gangyan Xu, Runyang Zhong et George Q. Huang. *Physical assets and service sharing for IoT-enabled Supply Hub in Industrial Park (SHIP).* International Journal of Production Economics, vol. 159, pages 4–15, 2015. (Cité en page 18.)

[Raj 2010] R J R Raj et T Sasipraba. *Web service selection based on QoS Constraints.* Trendz in Information Sciences & Computing (TISC), 2010, vol. 6, pages 156–162, 2010. (Cité en page 13.)

[Ramírez 2017] Aurora Ramírez, José Antonio Parejo, José Raúl Romero, Sergio Segura et Antonio Ruiz-Cortés. *Evolutionary composition of QoS-aware web services : A many-objective perspective.* Expert Systems with Applications, vol. 72, pages 357–370, 4 2017. (Cité en page 13.)

[Roman 2011] Dumitru Roman, Uwe Keller, Holger Lausen, Jos De Bruijn, Ruben Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Cristoph Bussler et Dieter Fensel. *Web Service Modeling Ontology.* Applied Ontology, vol. 1, no. 1, pages 77–106, 2011. (Cité en page 24.)

[Roman 2015] Dumitru Roman, Jacek Kopecký, Tomas Vitvar, John Domingue et Dieter Fensel. *WSMO-Lite and hRESTS : Lightweight semantic annotations for Web services and RESTful APIs.* Journal of Web Semantics, vol. 31, pages 39–58, 2015. (Cité en pages 24 et 25.)

[Rozenberg 1997] Grzegorz Rozenberg et H Ehrig. Handbook of graph grammars and computing by graph transformation, volume 1. World Scientific, 1997. (Cité en pages 26, 39, 48 et 50.)

[Rupasingha 2016] Rupasingha A H M Rupasingha, Incheon Paik et Banage T G S Kumara. *Domain-aware Web Service Clustering based on Ontology Generation by Text Mining.* 2016. (Cité en pages 13 et 16.)

[Seydoux 2016] Nicolas Seydoux, Khalil Drira, Nathalie Hernandez et Thierry Monteil. *Iot-O, a core-domain IoT ontology to represent connected devices networks.* Dans E. Blomqvist, P. Ciancarini, F. Poggi et F. Vitali, editeurs, Knowledge Engineering and Knowledge Management. EKAW 2016. LNCS, volume 10024 LNAI, pages 561–576. Springer, Cham, 11 2016. (Cité en pages 25 et 41.)

[Seydoux 2018] Nicolas Seydoux, Khalil Drira, Nathalie Hernandez et Thierry Monteil. *Towards Cooperative Semantic Computing : a Distributed Reasoning approach for Fog-enabled SWoT.* Dans International Conference on Cooperative Information Systems, 2018. (Cité en page 116.)

[Shi 2011] Yulu Shi et Xi Chen. *A survey on QoS-aware web service composition.* Proceedings - 3rd International Conference on Multimedia Information Networking and Security, MINES 2011, pages 283–287, 2011. (Cité en page 13.)

[Shu 2010] Yuan-Zhong Shu, Yan-Pei Liu, Xiao-Hong Peng et Zhi-Yong Chen. *Survey on object-oriented Petri net modeling.* Computer Engineering and Design, vol. 31, no. 15, pages 3432–3435, 2010. (Cité en page 14.)

[Srivastava 2003] Biplav Srivastava et Jana Koehler. *Web Service Composition : Current Solutions and Open Problems.* ICAPS 2003 Workshop on Planning for Web Services, no. August 2003, pages 28–35, 2003. (Cité en page 10.)

[Tillmann 2003] Christoph Tillmann et Hermann Ney. *Word reordering and a dynamic programming beam search algorithm for statistical machine translation.* Computational linguistics, vol. 29, no. 1, pages 97–133, 2003. (Cité en page 16.)

[Vitvar 2008] Tomas Vitvar, Jacek Kopecký, Jana Viskova et Dieter Fensel. *WSMO-lite annotations for web services.* Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 5021 LNCS, pages 674–689, 2008. (Cité en page 24.)

[Wang 2009] Yong Lian Wang et Xue L. Yu. *Formalization and verification of automatic composition based on Pi-Calculus for semantic web service.* 2009 2nd International Symposium on Knowledge Acquisition and Modeling, KAM 2009, vol. 1, pages 103–106, 2009. (Cité en page 14.)

[White 2008] Stephen A White. BPMN modeling and reference guide : understanding and using BPMN. Future Strategies Inc., 2008. (Cité en page 11.)

[Wu 2007] Z Wu, AH Ranabahu et K Gomadam. *Automatic composition of semantic web services using process and data mediation.* 2007. (Cité en pages 13 et 16.)

[Xue 2018] Gang Xue, Jing Liu, Liwen Wu et Shaowen Yao. *A graph based technique of process partitioning.* Journal of Web Engineering, vol. 17, no. 1-2, pages 121–140, 2018. (Cité en page 28.)

[Yachir 2015] A. Yachir, Y. Amirat, A. Chibani et N. Badache. *Event-Aware Framework for Dynamic Services Discovery and Selection in the Context of Ambient Intelligence and Internet of Things.* IEEE Transactions on Automation Science and Engineering, vol. 13, no. 1, pages 85–102, 1 2015. (Cité en pages 13, 17 et 21.)

[Yu 2005] Tao Yu et Kwei-Jay Lin. *Service selection algorithms for Web services with end-to-end QoS constraints.* Information systems and e-business management, vol. 3, no. 2, pages 103–126, 2005. (Cité en page 11.)

[Yu 2014] Shih Yuan Yu, Chi Sheng Shih, Jane Yung Jen Hsu, Zhenqiu Huang et Kwei Jay Lin. *QoS oriented sensor selection in IoT system.* Proceedings - 2014 IEEE International Conference on Internet of Things, iThings 2014, 2014 IEEE International Conference on Green Computing and Communications, GreenCom 2014 and 2014 IEEE International Conference on Cyber-Physical-Social Computing, CPS 20, no. iThings, pages 201–206, 2014. (Cité en pages 13 et 15.)

[Zanella 2014] A Zanella, N. Bui, A Castellani, L. Vangelista et M. Zorzi. *Internet of Things for Smart Cities.* IEEE Internet of Things Journal, vol. 1, no. 1, pages 22–32, 2014. (Cité en page 2.)

[Zhao 2012] Bingyang Zhao, Yongwang Zhao et Dianfu Ma. *A Constraint Mechanism for Dynamic Evolution of Service Oriented Systems.* Dans 2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, pages 103–110. IEEE, 4 2012. (Cité en page 28.)

[Zhovtobryukh 2007] Dmytro Zhovtobryukh. A Petri Net-based Approach for Automated Goal-Driven Web Service Composition, volume 83. 2007. (Cité en pages 13, 15 et 116.)

Résumé : L'Internet des Objets (IoT) est déjà omniprésent aujourd'hui : domotique, bâtiments connectés ou ville intelligente, beaucoup d'initiatives et d'innovations sont en cours et à venir. Le nombre d'objets connectés ne cesse de croître à tel point que des milliards d'objets sont attendus dans un futur proche. L'approche de cette thèse met en place un système de gestion autonomique pour des systèmes à base d'objets connectés, en les combinant avec d'autres services comme par exemple des services météo accessibles sur internet. Les modèles proposés permettent une prise de décision autonome basée sur l'analyse d'évènements et la planification d'actions exécutées automatiquement. Des paramètres comme le temps d'exécution ou l'énergie consommée sont aussi considérés afin d'optimiser les choix d'actions à effectuer et de services utilisés. Un prototype concret a été réalisé dans un scénario de ville intelligente et de bus connectés dans le projet investissement d'avenir S2C2.

Mots-clés : Internet des objets, Approche orientée service, Informatique autonomique, Web sémantique, Grammaire de graphes, Qualité de service

Abstract : The Internet of Things (IoT) is already everywhere today : home automation, connected buildings or smart city, many initiatives and innovations are ongoing and yet to come. The number of connected objects continues to grow to the point that billions of objects are expected in the near future. The approach of this thesis sets up an autonomic management architecture for systems based on connected objects, combining them with other services such as weather services accessible on the Internet. The proposed models enable an autonomous decision making based on the analysis of events and the planning of actions executed automatically. Parameters such as execution time or consumed energy are also considered in order to optimize the choices of actions to be performed and of services used. A concrete prototype was realized in a smart city scenario with connected buses in the S2C2 project.

Keywords : Internet of Things, Service Oriented Approach, Autonomic Computing, Semantic Web, Graph Grammar, Quality of Service