Sélection contextuelle de services continus pour la robotique ambiante

Benjamin Cogrel

UNIVERSITÉ PARIS-EST

École Doctorale Mathématiques et STIC

Laboratoire Images Signaux et Systèmes Intelligents
LISSI (EA 3956)

Thèse de doctorat
Spécialité Informatique

présentée par

Benjamin Cogrel

Sélection contextuelle de services continus pour la robotique ambiante

Soutenue publiquement le 18 novembre 2013 devant le jury composé de :

Amar Ramdane-Cherif	Professeur à l'Université de Versailles Saint-Quentin	Rapporteur
Olivier Simonin	Professeur à l'INSA de Lyon	Rapporteur
Nicole Levy	Professeur au CNAM de Paris	Présidente du jury
Patrick Reignier	Professeur à l'ENSIMAG de l'INP de Grenoble	Examinateur
Abdelghani Chibani	Maître de conférences à l'Université Paris-Est Créteil	Examinateur
Yacine Amirat	Professeur à l'Université Paris-Est Créteil	Directeur de thèse
Boubaker Daachi	Maître de conférences HDR à l'Université Paris-Est Créteil	Co-directeur de thèse

Résumé

La robotique ambiante s'intéresse à l'introduction de robots mobiles au sein d'environnements actifs où ces derniers fournissent des fonctionnalités alternatives ou complémentaires à celles embarquées par les robots mobiles. Cette thèse étudie la mise en concurrence des fonctionnalités internes et externes aux robots, qu'elle pose comme un problème de sélection de services logiciels. La sélection de services consiste à choisir un service ou une combinaison de services parmi un ensemble de candidats capables de réaliser une tâche requise. Pour cela, elle doit prédire et évaluer la performance des candidats. Ces performances reposent sur des critères non-fonctionnels comme la durée d'exécution, le coût ou le bruit.

Ce domaine applicatif a pour particularité de nécessiter une coordination étroite entre certaines de ses fonctionnalités. Cette coordination se traduit par l'échange de flots de données entre les fonctionnalités durant leurs exécutions. Les fonctionnalités productrices de ces flots sont modélisées comme des services continus. Cette coordination étroite impose que les compositions de services soient hiérarchiques et introduit des contraintes supplémentaires pour la sélection de services.

Cette thèse met en évidence la présence d'un important couplage non-fonctionnel entre les performances des instances de services de différents niveaux, même lorsque les flots de données sont unidirectionnels. L'approche proposée se concentre sur la prédiction de la performance d'un organigramme. Un organigramme regroupe l'ensemble des instances de services sollicitées pour réaliser une tâche de haut-niveau. L'étude s'appuie sur un scénario impliquant la sélection d'un service de positionnement en vue de permettre le déplacement d'un robot vers une destination requise.

Pour un organigramme considéré, la prédiction de performance doit, dans ce scénario, répondre aux exigences suivantes : elle doit (i) être contextuelle en tenant compte, par exemple, du chemin suivi pour atteindre la destination requise, (ii) prendre en charge le remplacement d'une instance de sous-service suite à un échec ou, par extension, de façon opportuniste. En conséquence, cette sélection de services est posée comme un problème de prise de décision séquentielle formalisé à l'aide de processus de décision markoviens à horizon fini. La dimensionnalité importante du contexte en comparaison à la fréquence des déplacements du robot rend inadaptées les méthodes consistant à apprendre directement une fonction de valeur ou une fonction de transition. L'approche proposée repose sur des modèles de dynamique locaux et exploite le chemin de déplacement calculé par un sous-service pour estimer en ligne les valeurs des organigrammes disponibles dans l'état courant. Cette estimation est effectuée par l'intermédiaire d'une méthode de fouille stochastique d'arbre, *Upper Confidence bounds applied to Trees*.

Mots-clés : sélection de services, robotique ambiante, apprentissage par renforcement, sensibilité au contexte, coordination étroite, processus de décision markovien, composition de services.

Abstract

Ambient robotics aims at introducing mobile robots in active environments where the latter provide new or alternative functionalities to those shipped by mobile robots. This thesis studies the competition between robot and external functionalities, which is set as a service selection problem. Service selection consists in choosing a service or a combination of services among a set of candidates able to fulfil a given request. To do this, it has to predict and evaluate candidate performances. These performances are based on non-functional requirements such as execution time, cost or noise.

This application domain requires tight coordination between some of its functionalities. Tight coordination involves setting data streams between functionalities during their executions. In this proposal, functionalities producing data streams are modelled as continuous services. Tight coordination requires hierarchical service composition and adds some constraints to the service selection problem.

This thesis shows that an important non-functional coupling appears between service instances at different levels, even when data streams are unidirectional. The proposed approach focuses on performance prediction of an organigram. This organigram gathers service instances involved in the high-level task processing. The scenario included in this study is the selection of a positioning service involved in a robot navigation high-level service.

For a given organigram, performance prediction in this scenario has to: (i) be contextual by, for instance, considering moving path towards the target destination, (ii) support service instance replacement after a failure or in an opportunist manner. Consequently, this service selection is set as a sequential decision problem and is formalized as a finite-horizon Markov decision process. Its high contextual dimensionality with respect to robot moving frequency makes direct learning of Q-value functions or transition functions inadequate. The proposed approach relies on local dynamic models and uses the planned moving path to estimate Q-values of organigrams available in the initial state. This estimation is done using a Monte-Carlo tree search method, *Upper Confidence bounds applied to Trees*.

Keywords: service selection, ambient robotics, reinforcement learning, context awareness, tight coordination, Markov decision process, service composition.

Remerciements

À mes encadrants, Boubaker Daachi et Yacine Amirat, pour m'avoir proposé ce sujet de thèse original et m'avoir permis de l'approfondir au cours de ces années.

À Olivier Simonin et Amar Randame-Cherif, rapporteurs, pour leur lecture attentive et critique de ce manuscrit et l'intérêt porté à ces travaux.

À Patrick Reignier, examinateur, pour sa lecture assidue et ses nombreuses questions sur la place de l'utilisateur dans cette approche.

À Nicole Levy, pour ses remarques critiques et s'est prêtée à l'exercice de présidente du jury.

À Abdelghani Chibani pour m'avoir intégré dans l'équipe de son projet et avoir accepté de faire partie de mon jury de thèse.

À Julien Perez, pour ses conseils et sa bonne connaissance de la communauté de l'apprentissage artificiel.

À Benjamin A. et Bertrand M. pour s'être prêtés bénévolement à l'exercice des relectures des premières versions de ce manuscrit.

Au département Réseaux et Télécoms de l'IUT de Créteil-Vitry pour m'avoir recruté comme ATER à mi-temps.

À toutes les personnes que j'ai pu rencontrer au cours de ces années au LISSI, pour leur convivialité, leur diversité et pour les riches discussions qu'elles ont engendrées.

À mes amis venus et parfois restés à Paris, pour avoir fait de cette seconde expérience parisienne un moment mémorable.

À mes parents et ma sœur, pour leur soutien sans faille durant ces longues années d'étude.

Table des matières

Introduction

Avec la numérisation du monde qui caractérise notre époque, les fonctionnalités d'observation et d'action sur nos environnements quotidiens se multiplient et sont désormais accessibles via des réseaux informatiques. Des exemples de telles fonctionnalités sont la mesure de la température, la reconnaissance d'activités humaines, la localisation et le déplacement d'un objet ou encore la diffusion d'un flot multimédia. L'étude des systèmes intégrant ces fonctionnalités est une thématique de recherche multidisciplinaire appelée, selon les auteurs, informatique ubiquitaire ou intelligence ambiante. Le domaine d'application retenu dans cette thèse se situe à l'intersection de cette thématique et de la robotique mobile : il s'agit de la robotique ambiante.

L'introduction de l'intelligence ambiante au sein de la robotique mobile est à l'origine de deux nouvelles considérations. Tout d'abord, elle remet en cause le paradigme du robot autonome : l'environnement n'est plus considéré comme étant passif car il est en capacité de fournir certaines des fonctionnalités dont le robot mobile a besoin. Le robot n'est alors plus contraint d'embarquer l'ensemble de ces fonctionnalités et peut désormais les mettre en concurrence. Enfin, la robotique ambiante accorde un rôle central aux utilisateurs et à leurs préférences dans les choix effectués au sein de ces systèmes.

Cette thèse s'intéresse à cette mise en concurrence entre les fonctionnalités robotiques disponibles dans un environnement actif. En modélisant ces fonctionnalités sous forme de services logiciels, nous posons cette mise en concurrence comme un problème de sélection de services. Ce problème consiste à choisir un service ou une combinaison de services parmi un ensemble de candidats capables de réaliser une tâche requise. Ces candidats devant répondre aux exigences fonctionnelles requises, la sélection effectue son choix à partir de critères non-fonctionnels. Dans ce manuscrit, les critères non-fonctionnels considérés sont tous associés à la performance du traitement d'une requête de service. Par exemple, ces critères peuvent être la durée d'exécution et le coût. Suivant les différences de performance des services et suivant la disponibilité de métriques pertinentes permettant d'en rendre compte, la sélection de services peut avoir un impact crucial sur l'appréciation par l'utilisateur de la tâche réalisée. Notre propos consiste à valoriser cet impact dans ce cadre de robotique ambiante.

La mise en œuvre de cette sélection de services nécessite l'étude de trois points importants : (i) la modélisation du domaine d'application sous forme de services, (ii) le choix des critères d'évaluation de la performance d'un traitement de requête et (iii) la prédiction de cette performance durant chaque phase de sélection.

0.1 Modélisation des services du domaine d'application

Les services robotiques actuels sont principalement atomiques et de haut-niveau. L'essentiel des fonctionnalités sous-jacentes n'est accessible qu'au niveau des couches logicielles inférieures, généralement sous la forme de composants logiciels. Si une mise en concurrence peut avoir lieu directement entre des composants logiciels distribués, nous pensons qu'il est préférable qu'elle se fasse entre services, afin de tirer partie de leur moindre couplage fonctionnel. En effet, les services logiciels offrent une plus grande indépendance technologique que les approches à base de composants car ces dernières impliquent un ensemble d'hypothèses restrictives comme l'utilisation d'une plateforme logicielle commune.

La modélisation de ces fonctionnalités sous forme de services logiciels vise donc à augmenter le nombre et la variété des alternatives disponibles. Cependant, comme l'indique la littérature sur les systèmes multi-robots, une part importante de ces fonctionnalités nécessite une coordination étroite. Cette exigence va à contre-courant des approches orientées service dominantes, étant donné que celles-ci supposent une coordination faible. Dans ces approches, les services sont en effet majoritairement modélisés comme réalisant des actions. L'intérêt fonctionnel se concentre alors sur le résultat final auquel abouti le traitement et non sur le processus qui y a mené.

Pour satisfaire cette exigence de coordination étroite, nous adaptons la proposition de [Lundh et al., 2008] aux architectures orientées service en modélisant ces fonctionnalités comme des services continus. Ceux-ci produisent des flots de données durant leurs exécutions. Le type de composition résultant est hiérarchique afin que les dépendances fonctionnelles induites par la consommation de ces flots n'apparaissent pas dans les interfaces des services.

0.2 Critères d'évaluation

Conformément aux principes de l'intelligence ambiante, la sélection de services est ici centrée sur l'utilisateur : celui-ci est responsable du choix des critères d'évaluation d'une instance de service. Cette responsabilité peut toutefois être déléguée à un représentant logiciel. L'utilisateur, ou son représentant, doit donc désigner les métriques à considérer, la façon de les normaliser ainsi que la fonction d'agrégation qui calculera un score à partir de ces valeurs normalisées [1]. Ces choix forment un modèle d'évaluation.

Les métriques fréquemment utilisées pour la sélection de services sont le temps de réponse, la disponibilité, le coût, la bande passante ou encore la fiabilité. Nous proposons d'introduire de nouvelles métriques complémentaires tenant compte de la nature physique des processus de ces services robotiques. Aussi, nous prendrons comme exemples le bruit moyen que génère le déplacement d'un robot, en complément du coût moyen et de la durée d'exécution. De plus, afin qu'un utilisateur puisse introduire de nouvelles métriques non prévues par les concepteurs d'un robot, nous suggérons que ces mesures puissent être effectuées par des services externes dédiés.

Nous proposons que le choix du modèle d'évaluation ne soit pas prédéfini mais effectué tardivement, de façon à ce qu'il puisse être intégré à la requête de service de haut-niveau. Cette approche a pour autre avantage d'obtenir une fonction de score choisie en fonction

1. Ces valeurs normalisées sont généralement appelées paramètres de Qualité de Service (QdS).

du contexte perçu par l'utilisateur, ou son représentant, au moment de l'élaboration de la requête. Enfin, nous supposons que l'utilisateur, ou son représentant, ne s'intéresse qu'à l'instance de service de haut-niveau. Il n'a alors pas besoin de spécifier de modèle d'évaluation pour les instances des sous-services dont les performances sont comprises dans celle de l'instance de haut-niveau.

0.3 Prédiction de performance

Une fois le modèle d'évaluation fourni, la sélection de services peut prédire la performance d'un ensemble de candidats considérés. L'étude de cette prédiction comporte traditionnellement deux axes : (i) les aspects combinatoires liés au nombre de candidats à comparer et (ii) la prédiction de performance d'un candidat.

0.3.1 Complexité combinatoire des candidats

L'utilisateur, ou son représentant, ne fournissant un modèle d'évaluation que pour l'instance de service de haut-niveau, les sous-services ne peuvent pas être sélectionnés séparément, sur la base de leur performance locale. Il s'agit d'une sélection de services globale et est en cela confrontée à un problème combinatoire important.

Ce problème combinatoire a été traité à de multiples reprises dans le cas de compositions de services réalisant des actions. Cependant, ce type de composition ne se situe souvent pas au même niveau qu'une composition à base de services continus : cette dernière vise généralement à réaliser une seule action tandis que le premier type sollicite un plan d'actions. Cette limitation de portée de ces compositions à base de services continus favorise la régularité de leur structure et l'introduction de connaissances *a priori* sur l'importance de certains sous-services dans la performance globale. Cela nous permet de traiter *a minima* cette problématique combinatoire en ne considérant pas les alternatives de certains types de sous-services. Ceux-ci sont jugés non prioritaires.

L'approche proposée se concentre sur la prédiction de la performance d'une instance de haut-niveau sachant son organigramme à l'issue de la sélection. Un organigramme regroupe l'ensemble des instances de services sollicitées pour réaliser une tâche de haut-niveau. Cette approche a été proposée suite au constat d'un important couplage non-fonctionnel pouvant apparaître entre les performances des instances de services de différents niveaux, même lorsque les flots de données sont unidirectionnels. Ainsi, elle évite la modélisation des relations complexes entre les performances de ces différentes instances qui représenterait un pré-requis contraignant.

0.3.2 Prédiction de performance dans le cadre d'une prise de décision séquentielle

Notre étude consacrée à la prédiction de performance d'un organigramme lors du traitement d'une requête de haut-niveau s'appuie sur l'exemple d'un service de déplacement de robot. Ce service est composite : son organigramme inclut principalement des services de positionnement et de planification de chemin. Comme de nombreuses fonctionnalités robotiques, ce service est soumis à des contraintes environnementales qui impactent directement sa performance. Cet impact est très marqué pour son sous-service

de positionnement dont la portée peut être limitée et exposée à des zones d'ombre de part la présence d'obstacles. De plus, le robot étant mobile, ces contraintes changent au fur et à mesure que ce dernier se déplace. Elles peuvent entraîner l'échec de l'instance de service de positionnement et nécessiter son remplacement.

Compte tenu de l'importance de ces contraintes environnementales, nous proposons d'effectuer une prédiction de performance contextuelle d'une instance de service de haut-niveau de ce type. Par ailleurs, le caractère fréquent et prévisible des échecs de sous-services en cours d'exécution oblige le sélectionneur à en tenir compte lors de sa prédiction de performance. Cette prédiction de performance est donc directement associée à une prise de décision séquentielle. Deux types de remplacement des sous-services sont considérés : le simple remplacement d'un sous-service ayant échoué et le remplacement opportuniste.

Ces deux problèmes de sélection avec remplacement sont formalisés sous forme de processus de décision markoviens (*Markov Decision Processes*, MDPs) à horizon fini. Dans chacun de ces MDPs, la fonction de récompense est définie à partir du modèle d'évaluation assigné par l'utilisateur. Cependant, la fonction de transition doit être apprise. Cet apprentissage est ici confronté à une dimensionnalité du contexte importante en comparaison à la fréquence des déplacements du robot. En effet, les exécutions des instances d'un service de déplacement du robot sont longues, ne surviennent que de façon occasionnelle et changent de positions initiales et de destinations. Dans cette condition, l'espace $\mathbb{R}^4$ des positions initiales et des destinations possibles offre à lui-seul [2] une dimensionnalité suffisamment importante pour que les méthodes habituellement employées pour prendre en charge les problèmes de dimensionnalité, l'apprentissage supervisé de la fonction de transition ou d'une fonction de valeur Q, soient inadaptées.

Nous proposons d'aborder cette difficulté en raisonnant à partir du contexte courant et en exploitant des modèles exposés à une représentation réduite du contexte. Ces projections facilitent le partage d'expérience entre les exécutions et par conséquent l'apprentissage de ces modèles. Notre proposition repose sur une segmentation des transitions effectuée à partir de la trajectoire fournie par un sous-service de planification de chemin. Chaque transition est ensuite calculée à partir de deux sous-modèles de dynamique associés à la phase de préparation et au déplacement du robot sur un segment. Chacune de ces transitions désigne un ensemble fini d'états successeurs ainsi que leurs probabilités d'être atteints. Enfin, les valeurs des actions disponibles dans l'état courant sont estimées à l'aide d'une méthode de fouille stochastique d'arbre : *Upper Confidence bounds applied to Trees* (UCT).

0.4 Organisation du manuscrit

Chapitre 1 : Vers la robotique ambiante Ce chapitre introduit le domaine d'application de cette thèse, la robotique ambiante, qui est à la croisée de l'intelligence ambiante et de la robotique mobile. À l'occasion de la présentation de ces deux composantes, les éléments-clés retenus dans nos propositions sont mis en avant. L'intelligence ambiante place l'utilisateur au centre, ce qui nous invitera à l'impliquer dans le choix du modèle d'évaluation. La robotique mobile introduit la distinction entre coordination

2. L'espace contextuel, qui équivaut à l'espace des états dans les MDPs, inclut également d'autres dimensions liées aux métriques considérées et aux sous-services disponibles ou en cours d'utilisation.

faible et coordination étroite de tâches et apporte en cela un élément déterminant dans le positionnement de nos travaux vis-à-vis des architectures orientées service classiques.

Chapitre 2 : Architectures pour les services réalisant des actions Le second chapitre d'état de l'art présente les architectures orientées service sous leur angle le plus courant où les services réalisent des actions. Il présente les technologies développées par différentes communautés scientifiques et d'ingénierie ainsi que les problématiques allant de la description sémantique à la sélection en passant par la composition de services.

Chapitre 3 : Sélection contextuelle de services robotiques : enjeux et architecture La coordination étroite exigée par certaines tâches robotiques ne peut pas être réalisée à partir de plans d'actions. Pour cela, ce troisième chapitre propose une seconde catégorie de services, les services continus, et montre que leur composition est hiérarchique. L'impact de la coordination étroite sur la prédiction de performance est ensuite étudiée et de nouveaux enjeux sont dégagés. Dans un troisième temps, ce chapitre effectue un ensemble de recommandations architecturales pour permettre la sélection de services au sein de compositions de services continus. Il se termine en détaillant le processus d'évaluation de l'exécution d'une instance de service.

Chapitre 4 : Prédiction de performance dans le cadre du possible remplacement de sous-services continus Ce dernier chapitre est consacré à la prédiction de la performance d'une composition de services continus autorisant le remplacement de certains de ses sous-services. L'exemple de composition ici étudié est celle d'un service de déplacement de robot. Deux types de remplacement sont considérés et donnent chacun lieu à une formalisation basée sur des processus de décision markoviens à horizon fini. Face au problème de la dimensionnalité du contexte de ce service de haut-niveau, ce chapitre propose une modélisation de la fonction de transition à base de sous-modèles de dynamique. Ensuite, il motive et décrit l'usage d'une méthode de fouille stochastique d'arbre pour estimer les valeurs des actions disponibles dans l'état courant. Enfin, ces propositions sont évaluées au travers de simulations.

Chapitre 1

Vers la robotique ambiante

Sommaire

Le domaine d'application de cette thèse est la robotique ambiante, qui est à la croisée de l'intelligence ambiante et de la robotique mobile. Ce chapitre dresse une liste des enjeux qui seront retenus en étudiant la robotique ambiante sous le prisme de ses deux domaines d'origine. Dans un premier temps, il présente l'intelligence ambiante comme une variante particulière de l'informatique ubiquitaire et positionne la problématique de ce manuscrit vis-à-vis d'applications qui lui sont proches. Enfin, la seconde section met l'accent sur la composition de fonctionnalités en robotique mobile et les besoins de coordination qu'elle suscite, avant de se consacrer à la tâche de navigation robotique qui sera reprise comme exemple d'application.

1.1 L'informatique ubiquitaire et ses variantes

L'informatique ubiquitaire (*ubiquitous computing* ou *ubicomp*) est une communauté de recherche consacrée à l'étude et au développement d'une troisième génération d'ordinateurs. Cette section présente une brève description de ses concepts et des principaux débats suscités puis se focalise sur une variante particulière, l'intelligence ambiante.

1.1.1 Informatique ubiquitaire

La notion d'informatique ubiquitaire est née de la vision de Mark Weiser, alors chercheur au Xerox Parc à Palo Alto en Californie. À la fin des années 80, il a anti-

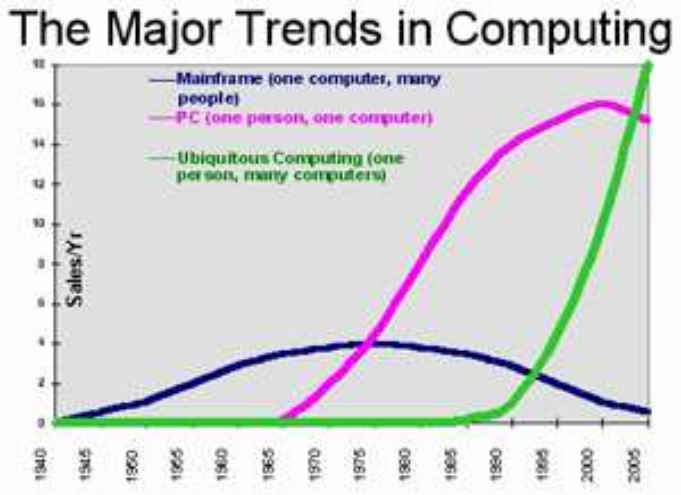


FIGURE 1.1 – Tendances des trois générations d'ordinateurs pronostiquées par Mark Weiser. Anciennement accessible à l'adresse `http://sandbox.xerox.com/ubicomp/`.

cipé l'émergence d'une troisième génération d'ordinateurs qui ne seraient plus centraux *(mainframes)* ni personnels (*Personal Computer, PC*) mais seraient de formes variées, répartis dans l'environnement et dans un nombre excédant grandement le nombre d'utilisateurs (voir la figure 1.1). L'informatique deviendrait alors ubiquitaire, omniprésente (*everyware* [Greenfield 2006]), diffuse *(pervasive)* ou encore ambiante.

1.1.1.1 Vision initiale

À partir de cette anticipation, Mark Weiser a proposé plusieurs réflexions visionnaires sur les orientations à donner aux développements à venir. Elles ont eu un impact considérable sur les réalisations de ces vingt dernières années.

Informatique invisible et calme En analysant les technologies réussies comme l'écriture, Mark Weiser constate que :

> «The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.» [Weiser 1991]

Il juge l'interaction avec un PC insatisfaisante [Weiser 1994] car elle demande trop d'attention de la part de l'utilisateur et ne peut donc en aucun cas être généralisée au nombre croissant d'ordinateurs à venir. Il est donc nécessaire que l'informatique soit mieux intégrée à la vie quotidienne, plus discrète voire si possible invisible afin de ne pas encombrer l'esprit. Il précisera sa pensée en parlant de technologie calme [Weiser & Brown 1996], où l'essentiel des informations doivent être laissées en périphérie de l'attention pour ne revenir au centre qu'en cas de nécessité, suite à un changement inhabituel. De là en découle le but implicite d'assister la vie quotidienne sans la surcharger. Par ailleurs, il est important de noter que dès son origine, l'informatique ubiquitaire revendique une approche centrée sur l'utilisateur.

1.1.1.2 Autres concepts dominants

Prise en considération du contexte Afin que ces ordinateurs puissent s'intégrer à cette vie quotidienne, ils doivent tirer partie du fait qu'ils sont situés dans un environnement. Dès lors qu'ils sont en mesure de communiquer et que certains d'entre eux disposent de capteurs physiques, il leur devient possible d'accéder à une grande source d'informations sur cet environnement et l'utilisateur qui y évolue. On parle alors de prise en compte du contexte *(context-awareness)*. De multiples définitions ont été données au contexte ; pour notre part, nous retenons celle proposée par [Dey 2001] où

> «Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.»

Le contexte est alors vu comme une source d'information implicite que l'utilisateur n'a désormais plus le besoin d'expliciter [Lieberman & Selker 2000] [1]. Il permet la représentation de situations vécues par l'utilisateur ou par n'importe quelle autre entité.

Sans-couture Face à la multiplicité et la diversité de dispositifs fixes et mobiles, l'objectif d'invisibilité est souvent interprété comme l'offre d'une expérience aussi uniforme que possible où le passage d'un dispositif à un autre est transparent pour l'utilisateur. On parle alors de sans-couture *(seamless)*. Parmi les travaux emblématiques de cette problématique, citons le projet *Oxygen* initié par le MIT [Dertouzos 1999] qui a proposé des appareils génériques destinés à de nombreux utilisateurs ainsi que le serveur personnel [Want et al. 2002] permettant à l'utilisateur de transporter ses données personnelles sur lui et de les rendre accessibles aux dispositifs environnants.

1.1.1.3 Autres perspectives

Avec-couture [Chalmers et al. 2003] critiquent l'interprétation alors hégémonique du sans-couture, que Mark Weiser avait lui-même considéré comme un concept susceptible d'induire en erreur. En effet, en matière d'interaction, le sans-couture tend à uniformiser et à sacrifier la richesse d'un dispositif au prix de la compatibilité. Ces auteurs montrent qu'il peut au contraire être intéressant d'exploiter les particularités des dispositifs afin d'offrir de belles coutures. À titre d'exemple, ils s'intéressent à la visite d'un musée qui peut être virtuelle ou physique et où le visiteur a à sa disposition une carte interactive indiquant sa position courante. Dans le premier cas, la position du visiteur virtuel est parfaitement connue tandis que dans le second cas, elle peut être très incertaine. Les auteurs proposent donc de ne pas nier cette différence de nature et de ne pas prendre le plus petit dénominateur commun mais d'afficher une distribution de probabilité lorsque le dispositif est imprécis. À cet égard, il convient donc d'avoir un rapport modéré au sans-couture.

1. Certains champs d'un formulaire peuvent par exemple être remplis automatiquement à partir d'informations contextuelles. L'utilisateur n'a alors besoin de se concentrer que sur les éventuels champs vides restants.

Acceptation de l'existant De nombreux papiers en informatique ubiquitaire font référence à un futur très proche qui est sans cesse repoussé à demain. Or en 2006, [Bell & Dourish 2006] observent que de nombreux dispositifs sont déjà utilisés de façon pragmatique à grande échelle, comme à Singapour ou en Corée du Sud. Cependant leur nature désordonnée et profondément hétérogène entre encore une fois en contradiction avec l'idéal du sans-couture où le futur est toujours vu comme homogène et ordonné.

Engagement Au-delà du sans-couture, [Rogers 2006] interroge l'idée même d'informatique calme formulée par Weiser. Selon l'auteur, elle a pour idéal un utilisateur passif et un environnement pro-actif se disant capable de répondre aux attentes implicites de celui-ci. Or dans de nombreux cas, les utilisateurs ont un comportement complexe et des attentes subtiles, si bien que de nombreuses solutions proposées apparaissent très réductrices et peu respectueuses de cette complexité. Cette limite est, selon lui, une des raisons principales de la concentration des travaux sur l'analyse du comportement de l'utilisateur vers la surveillance et l'assistance de personnes en situation de dépendance, qu'elles soient âgées ou sujettes à un handicap (voir par exemple [Abowd et al. 2002, Beckwith & Lederer 2003] mais aussi plus récemment [Hong et al. 2009, Zouba et al. 2009]). Le degré d'intrusion induit par ces techniques rend leur acceptation difficile en dehors de ces situations jugées extrêmes. Pour remédier à ces limitations, il convient selon cet auteur de passer la pro-activité davantage du côté de l'utilisateur et de miser sur l'offre de fonctionnalités engageantes. La question cruciale du contrôle revient davantage à l'utilisateur, qui est désormais plus à même de développer ses propres usages et de mieux encadrer l'utilisation faite de ses données sensibles *(privacy)*.

1.1.1.4 Communauté académique

Au niveau académique, l'informatique ubiquitaire est animée par une communauté foncièrement multi-disciplinaire, où se mêlent les sciences sociales comme l'ethnologie, la conception de nouveaux dispositifs matériels, les intergiciels, la reconnaissance d'activités, la multi-modalité [Dourlens et al. 2013] ou encore la protection des données privées. La culture *hacker*[2], du *do it yourself*, y est très prégnante. À ce titre, Abowd a proposé le slogan « *Your noise is my signal* » [Abowd 2012] pour décrire la démarche qui a conduit à l'élaboration de systèmes de localisation ré-exploitant les données disponibles comme le WiFi [Bahl & Padmanabhan 2000] mais aussi les propriétés physiques d'infrastructures comme le réseau électrique [Gupta et al. 2010] ou la ventilation [Patel et al. 2008].

Néanmoins, selon [Abowd 2012], l'informatique contemporaine est désormais suffisamment imprégnée des idées de l'informatique ubiquitaire pour que la motivation de l'existence de cette dernière comme sujet-niche de recherche soit révolue. Pour s'en convaincre, il n'y a qu'à constater la désormais grande banalité des applications géolocalisées sur téléphone mobile ou encore des plateformes électroniques de prototypage comme Arduino.

1.1.2 Intelligence ambiante

Le terme «intelligence ambiante» a été introduit une décennie après celui d'informatique ubiquitaire. Cette sous-section recense plusieurs interprétations de ce terme puis

2. Traduisible littéralement par bidouilleur.

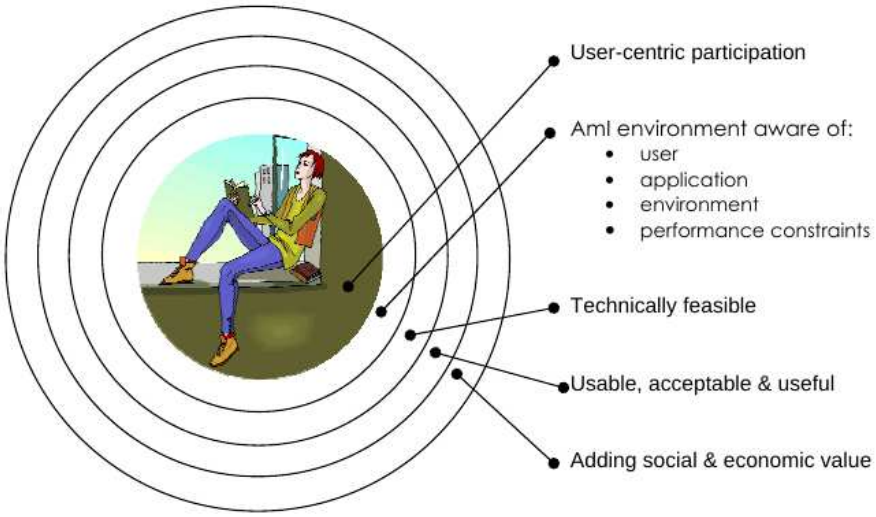


FIGURE 1.2 – Intelligence ambiante participative centrée sur l'utilisateur vue par l'ISTAG [ISTAG 2003]

présente une des applications proche de la problématique de cette thèse : le déclenchement pro-actif d'actions.

1.1.2.1 Interprétations

À partir d'une dynamique initiée par Philips, le groupe de travail ISTAG *(Information Societies Technology Advisory Group)* de la commission européenne a introduit la notion d'intelligence ambiante *(Ambient Intelligence, AmI)* et l'a proposée comme thème de recherche pour le programme européen IST *(Information Societies Technology)* [ISTAG 2001; 2003]. Elle en donne une définition très vague, en parlant d'intelligence ambiante comme «d'un ensemble de propriétés d'un environnement que nous sommes en train de créer» [ISTAG 2003]. Conscient du scepticisme de nombreux citoyens européens à l'égard du concept de technologie intelligente, l'ISTAG invite à concevoir l'intelligence dans un environnement ambiant comme «ce qui est fourni au travers de l'interaction ou de la participation, qui peut être apprécié comme une fonctionnalité d'assistance de la part du système ou de l'environnement et qui s'adresse aux besoins réels et aux désirs de l'utilisateur» [ISTAG 2003]. Tout comme en informatique ubiquitaire, l'ISTAG conçoit l'intelligence ambiante comme étant centrée sur l'utilisateur, à l'image de la figure 1.2. Par ailleurs, ce groupe insiste sur la nécessité de rendre ces systèmes contrôlables par l'utilisateur.

De nombreux auteurs distinguent l'intelligence ambiante de l'informatique ubiquitaire par l'introduction explicite de la notion d'intelligence [Augusto 2007, Ramos et al. 2008, Zaidenberg & Reignier 2011]. Selon Dey et al., l'intelligence consiste ici à «effectuer des inférences à partir d'informations captées dans l'environnement» [Dey et al. 2004]. Zaidenberg et Reignier en donnent une définition plus précise comme capacité d'analyse et de réaction au contexte courant [Zaidenberg & Reignier 2011]. L'ascendance de l'intelligence artificielle sur l'intelligence ambiante est souvent évoquée [Augusto 2007, Ramos et al. 2008, Leake & Gary 2008, Zaidenberg & Reignier 2011], comme en témoignent les figures 1.3 et 1.4. Enfin, des rapporteurs français du CNRS [Coutaz & Crowley 2008]

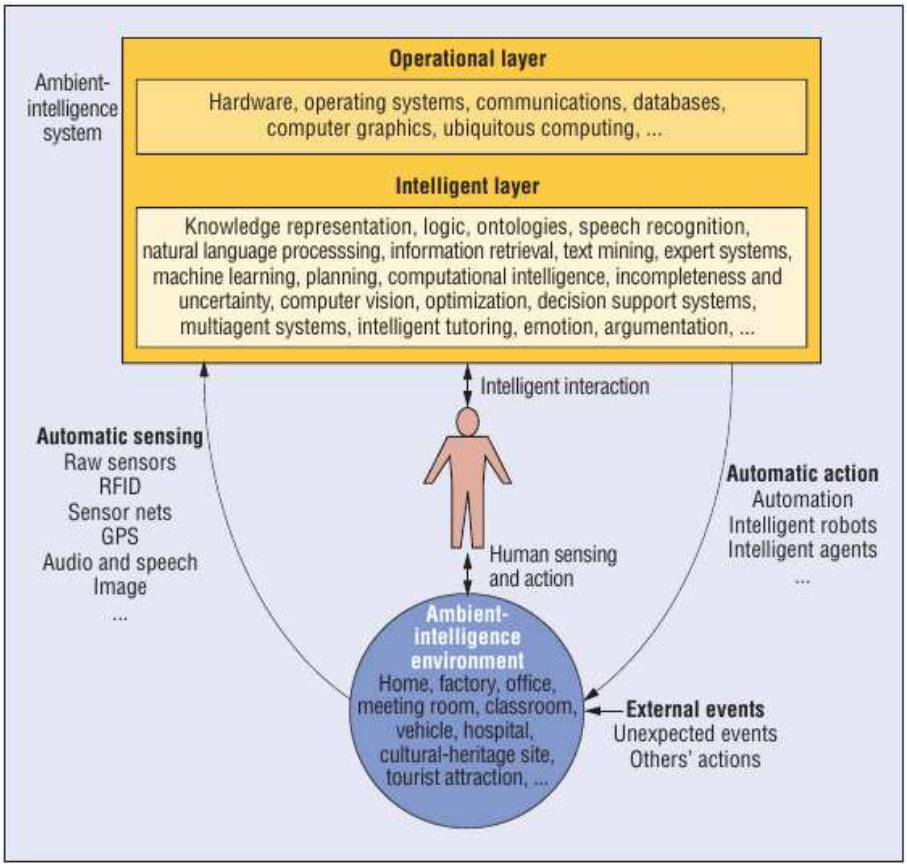


FIGURE 1.3 – L'intelligence ambiante dans une perspective d'intelligence artificielle [Ramos et al. 2008]

en retiennent une acception œcuménique aussi large que possible qui inclut les communautés de l'informatique ubiquitaire et diffuse[3] mais aussi celles où l'humain n'est pas dans la boucle comme le machine-à-machine *(Machine-to-Machine, M2M)* et l'internet des objets *(Internet of Things, IoT)*.

À partir de la précision apportée par l'ISTAG, nous proposons une interprétation écologique de la notion d'intelligence ambiante : l'intelligence prêtée aux dispositifs l'est dans la mesure où ceux-ci sont capables de «vivre en bonne intelligence» avec les utilisateurs et leurs attentes. L'intelligence ambiante étant centrée sur l'utilisateur, il revient à ce dernier le droit de juger si le comportement de certains dispositifs est adapté au contexte actuel d'utilisation. Plus précisément, la question de l'intelligence s'apparente ici, dans la mesure de ses restrictions de portée, à celle du savoir, c'est-à-dire à celle de savoir comment se comporter. Or, selon certains philosophes, la question du savoir est elle-même indissociable de celle de la bêtise [Stiegler 2012]. Dans le cadre de l'intelligence ambiante, les environnements dans lesquels évoluent les utilisateurs, comme les lieux publics ou les maisons, sont généralement suffisamment dynamiques pour que de nouveaux contextes d'utilisation inédits y surviennent. Les dispositifs n'y étant pas préparés, ils ne savent pas quels comportements sont appropriés et risquent donc d'entrer en dissonance avec les attentes de l'utilisateur et de faire preuve par la même occasion

3. Signe que la distinction entre informatique ubiquitaire et informatique diffuse est devenue ambiguë, deux de leurs principales conférences respectives *Ubicomp* et *Pervasive* vont fusionner en 2013 sous le nom d'*Ubicomp*.

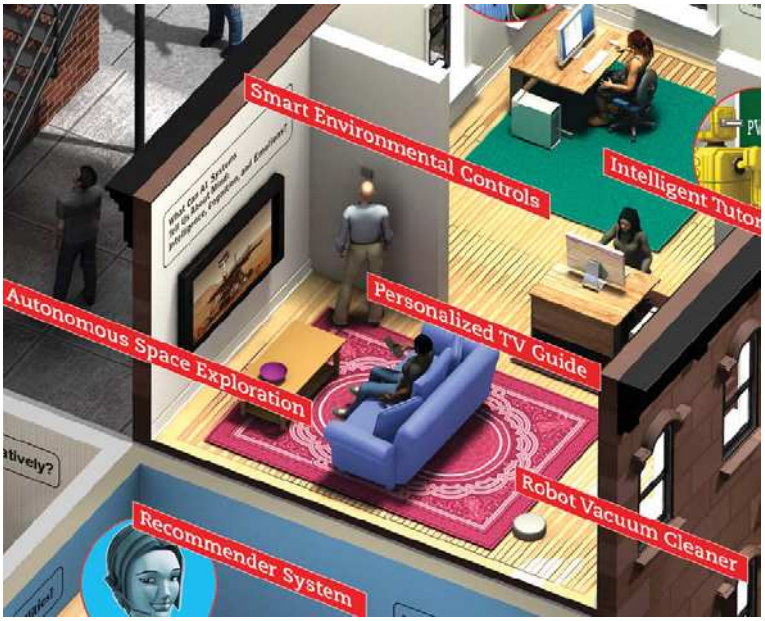


FIGURE 1.4 – Extrait du panorama de l'intelligence artificielle vu par l'Association for the Advancement of Artificial Intelligence (AAAI) [Leake & Gary 2008]

de bêtise. Néanmoins la face positive de la bêtise réside dans la capacité de constater a posteriori un comportement désajusté et de le corriger ; elle offre donc une possibilité d'acquisition de nouveaux savoirs. Dans cette perspective, la capacité d'apprentissage apparaît donc comme primordiale.

1.1.2.2 Déclenchement pro-actif d'actions

Nous présentons ici une catégorie d'applications d'intelligence ambiante proche mais distincte de notre problématique de sélection de services. Ces applications sont en effet complémentaires de nos travaux et leur association sera laissée en perspective. Elles portent sur le déclenchement pro-actif d'une ou de plusieurs actions en fonction du contexte selon les préférences de l'utilisateur et demandent la participation de ce dernier durant la phase d'apprentissage.

Le projet *a CAPella* [Dey et al. 2004] incite l'utilisateur final à effectuer des démonstrations de situations qu'il souhaite identifier. Pendant cette démonstration, de nombreuses sources d'informations contextuelles sont enregistrées, comme par exemple le nombre de personnes dans une pièce, le niveau de bruit ou encore la présence d'une conversation téléphonique. Ensuite, ces sources d'information sont présentées sous forme de pistes dans une interface graphique (dont la figure 1.5 en présente une capture d'écran) et l'utilisateur peut sélectionner les sources pertinentes et segmenter le passage intéressant. Une fois la situation identifiée, il peut lui associer une ou plusieurs actions. Cette démarche participative de l'utilisateur final génère des exemples qui sont utilisés pour effectuer l'apprentissage semi-supervisé d'un réseau bayésien dynamique mais dont les détails ne sont pas donnés.

L'approche introduite par [Brdiczka 2007] n'exige pas de démonstration de la part de l'utilisateur mais sa réaction immédiate suite à un mauvais choix d'action. Le cas échéant, il doit corriger ce choix en indiquant l'action qu'il aurait souhaité. Une absence

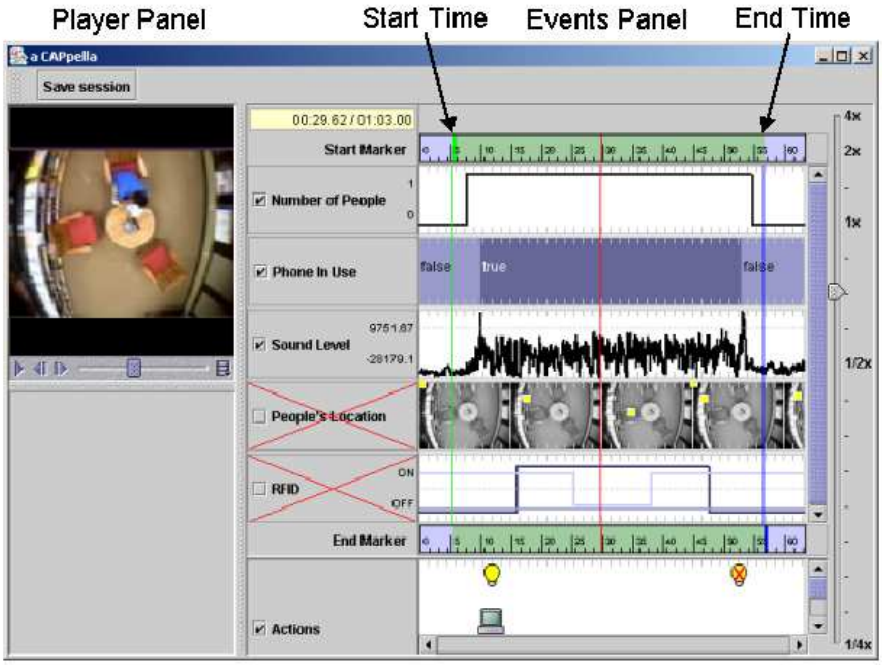


FIGURE 1.5 – Sélection des sources d'information et segmentation dans le logiciel *a CAPella* [Dey et al. 2004]

de réaction est interprétée comme une validation tacite de l'action. L'auteur effectue ici une hypothèse forte : les retours de l'utilisateur sont supposés être toujours cohérents. Il utilise cette hypothèse de cohérence des exemples pour généraliser les situations dans lesquelles peuvent être appliquées les actions ; une situation trop générale sera alors immédiatement divisée en deux situations plus spécifiques suite à la correction l'utilisateur. Son problème est formalisé comme une tâche de classification et il obtient un arbre de décision en effectuant un apprentissage supervisé.

Ce type de dispositif visant à déplacer le contrôle des actions de l'utilisateur final vers un agent dédié, [Zaidenberg 2009, Zaidenberg & Reignier 2011] identifient plusieurs propriétés importantes quant à l'établissement d'une relation de confiance nécessaire à la légitimité du dispositif. Parmi celles-ci, on trouve la nécessité d'un comportement initial cohérent ainsi que d'un droit de regard à tout instant sur une prise de décision de l'agent (on parle alors de scrutabilité). Cette dernière propriété repose ici sur une représentation intelligible d'une situation sous forme de prédicats logiques du premier ordre dont certaines valeurs sont laissées libres. Par ailleurs, cette proposition relâche l'hypothèse forte de cohérence de [Brdiczka 2007] : les retours de l'utilisateur sont supposés peu fréquents (ils n'interviennent pas après chaque action), ambigus et les préférences de l'utilisateur peuvent évoluer dans le temps. L'utilisateur évalue ici une action et l'état dans lequel elle a mené par une valeur numérique comprise dans l'ensemble $\{-1; -0.5; 0; 0.5; 1\}$. Comme précédemment dans [Brdiczka 2007], les auteurs sont confrontés au problème d'assignation du retour utilisateur à une représentation de la situation courante suffisamment générale. Néanmoins, ne pouvant bénéficier de l'hypothèse de cohérence des retours, seuls les triplets (situation initiale, action, situation suivante) prédéfinis dans le comportement initial peuvent avoir une représentation générale ; tout nouveau couple se limite à une représentation concrète faute de capacité de généralisation. Enfin, le formalisme du processus de décision markovien est employé (il sera également utilisé dans le

chapitre 4) en vue d'effectuer une prise de décision séquentielle. La motivation de cette dernière aurait mérité d'être davantage discutée.

1.1.3 Points retenus

Notre problématique de sélection de services s'inscrit dans le courant de l'intelligence ambiante et l'interprétation que nous en avons donnée orientera nos choix. Aussi notre sélectionneur de services sera un dispositif apprenant en boucle fermée et nous veillerons à impliquer les attentes de l'utilisateur dans sa boucle de rétro-action. Il accordera une place centrale au contexte. Les fournisseurs de services considérés seront hétérogènes et situés dans l'environnement de l'utilisateur.

À la différence des applications que nous venons de présenter, notre sélection de services ne sera pas pro-active mais se concentrera sur la réponse à une requête de service de haut-niveau. Une telle requête peut être vue comme la réalisation technique d'une action déclenchée par un dispositif pro-actif. Les préférences de l'utilisateur seront véhiculées à travers cette requête de haut-niveau par le biais d'un modèle d'évaluation de la performance de la réalisation de la tâche requise (ces aspects seront détaillés dans la section 3.3.2).

1.2 Robotique ambiante

La seconde composante applicative de cette thèse est la robotique mobile qui, en rejoignant les principales problématiques évoquées dans la section précédente, forme la robotique ambiante. Cette dernière est aussi connue sous les noms de robotique ubiquitaire, d'écologie de robots ou de systèmes robotiques en réseau.

Dans un premier temps, cette section retrace l'évolution de la robotique mobile autonome à la robotique ambiante. Ensuite, elle s'intéresse à l'assemblage de fonctionnalités robotiques à l'aide de techniques de planification puis se focalise sur la navigation robotique en intérieur. Enfin, cette section se clôture par une discussion sur le positionnement de nos travaux vis-à-vis de ce domaine.

1.2.1 De la robotique mobile autonome à la robotique ambiante

Un robot mobile est une machine capable de se déplacer dans un environnement donné. Nous présentons différents courants de la robotique mobile dans lesquels les tâches sont effectuées soit de façon autonome, soit en collaboration avec d'autres robots.

1.2.1.1 Robotique mobile autonome

La robotique mobile autonome est une des thématiques les plus étudiées en robotique. Elle recouvre plusieurs problématiques comme la navigation permettant au robot de se déplacer dans son environnement, la manipulation d'objets qui elles-mêmes nécessitent la perception de l'environnement et de l'état du robot. Parmi les exemples de robots autonomes, citons les voitures sans conducteur [Thrun et al. 2007], les hélicoptères de voltige [Abbeel et al. 2007] ou encore les robots domestiques comme le STAIR (*STanford Artificial Intelligence Robot*) capables d'ouvrir des portes [Petrovskaya & Ng 2007]. De

(a) Trois *foot-bots* de transport et un *hand-bot* escaladeur

(b) Robot volant *eye-bot*

FIGURE 1.6 – Trois types de robots *Swarmanoid* [Dorigo et al. 2012]. Démonstration consultable sur `http://www.youtube.com/watch?v=M2nn1X9Xlps`.

part leur autonomie, ces robots embarquent toutes les fonctionnalités dont ils dépendent, qu'il s'agisse de capteurs, d'actionneurs ou de modules logiciels. Ils peuvent donc agir seuls dans un environnement passif.

1.2.1.2 Systèmes multi-robots

Les systèmes multi-robots sont d'abord apparus pour étendre des tâches jusque-là effectuées par des robots individuels comme l'exploration [Burgard et al. 2005] et la patrouille [Ahmadi & Stone 2006, Glad et al. 2008] en apportant robustesse et diminution du temps d'exécution. Ces systèmes multi-robots introduisent la problématique de la collaboration entre robots, dont l'une des principales questions est leur degré de coordination, sur lequel nous reviendrons dans la sous-section 1.2.2. Au sein de la thématique des systèmes multi-robots, deux approches peuvent être distinguées : la robotique en essaim et la robotique ambiante.

La robotique en essaim *(swarm robotics)* est consacrée à des flottes de robots relativement simples, homogènes et présents en grand nombre. Elle a pour source principale d'inspiration le comportement des insectes sociaux. Le projet *Swarmanoid* [Dorigo et al. 2012] est un représentant. Il consiste en trois types de robots, représentés par la figure 1.6, aux fonctionnalités différentes : des robots escaladeurs capables de grimper sur des meubles comme des étagères, des robots roulants pouvant transporter à plusieurs ces robots escaladeurs et enfin des robots volants dotés de caméras et capables de se fixer au plafond.

1.2.1.3 Robotique ambiante

La robotique ambiante est connue sous diverses appellations : robotique ubiquitaire [Ha et al. 2005], systèmes robotiques en réseau *(network robot systems)* [Sanfeliu et al. 2008, Kanda et al. 2008] ou encore écologies de robots [Saffiotti & Broxvall 2005, Saffiotti et al. 2008, Gritti 2008]. Le terme de robot a ici une acception très générale : il s'agit

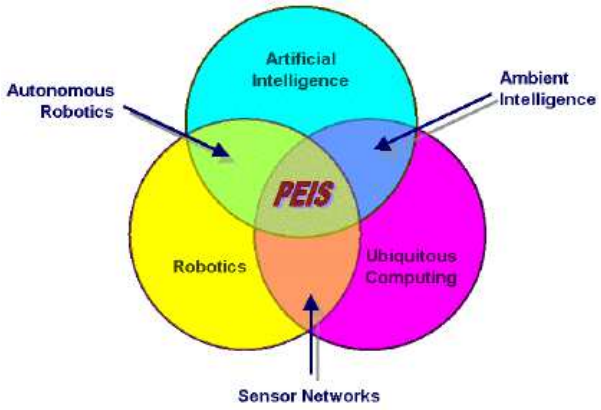


FIGURE 1.7 – Positionnement de la robotique ambiante représentée par le projet *PEIS-Ecology* [Saffiotti & Broxvall 2005]

d'un «système informatisé interagissant avec l'environnement au travers de capteurs et d'actionneurs» [Saffiotti & Broxvall 2005]. En ce sens, un robot extrêmement compétent évoluant dans un environnement passif est remplacé par un système robotique en réseau où les fonctionnalités sollicitées sont plus simples et sont réparties dans un environnement actif [Saffiotti & Broxvall 2005, Sanfeliu et al. 2008]. La robotique ambiante offre une hétérogénéité plus importante que la robotique en essaim en intégrant notamment les dispositifs domotiques. Cette hétérogénéité apporte de la flexibilité et de la robustesse [Lundh et al. 2008], à condition que les compositions de fonctionnalités soient effectuées automatiquement à partir des fonctionnalités découvertes au lieu d'être préprogrammées [Ha et al. 2005]. Par ailleurs, ces systèmes robotiques sont destinés à des activités en relation avec des humains. De ce fait, la robotique ambiante est à la croisée de la robotique et de l'intelligence ambiante, comme l'illustre la figure 1.7 [Saffiotti & Broxvall 2005].

PEIS-Ecology Le projet *PEIS-Ecology (Physically Embedded Intelligent Systems Ecology)* [Saffiotti & Broxvall 2005, Saffiotti et al. 2008] est une des illustrations les plus significatives de ce paradigme. Il a consisté en la mise en place d'un système de robotique ambiante au sein d'un appartement étudiant (cf. figure 1.8). L'environnement est principalement équipé de :

1. Plusieurs robots roulants diversement équipés en capteurs comme les sonars, les caméras et parfois les télémètres laser et les nez électroniques (permettant la classification d'odeur).
2. Un réfrigérateur disposant d'un bras manipulateur capable de saisir des objets, d'une porte motorisée, d'un lecteur RFID *(Radio-Frequency IDentification)* et de capteurs d'odeur.
3. Un ensemble de caméras fixes rotatives dans le salon couplé à un dispositif de reconnaissance d'objets.
4. Un interrupteur et des capteurs pour l'éclairage.

Ce projet a donné lieu à des contributions de différentes natures, allant de la réalisation d'un intergiciel [Broxvall et al. 2007, Rashid et al. 2012] au partage de connaissance [LeBlanc & Saffiotti 2008] en passant par l'assemblage automatique de fonctionnalités sous forme de configurations de composants logiciels [Lundh et al. 2008, Gritti 2008].

FIGURE 1.8 – Appartement expérimental du projet *PEIS-Ecology* [Saffiotti et al. 2008]

Parmi les scénarios expérimentés se trouvent le transport d'une barre par deux robots, la classification d'une odeur suspecte dans le réfrigérateur grâce au nez électronique d'un robot et l'apport d'une boisson prise dans le réfrigérateur.

1.2.2 Planification de tâches robotiques

Bon nombre des tâches effectuées par les robots mobiles sont composées de sous-tâches qui, selon le degré d'abstraction, peuvent être elles-mêmes composites ou bien correspondre à des fonctionnalités sensorimotrices de bas-niveau. Les techniques de planification classique et hiérarchique [4], apportent en partie la flexibilité et la robustesse dont ces compositions ont besoin pour faire face à l'ouverture et la dynamicité des environnements dans lesquels les robots mobiles évoluent. Elles peuvent par ailleurs être complétées par des mécanismes d'apprentissage.

Cette sous-section dresse un succinct aperçu de ces techniques de planification en les distinguant principalement selon leur degré de coordination. À cette occasion, elle met en avant les infrastructures logicielles sous-jacentes. Enfin, elle présente brièvement quelques approches de sélection au sein de ces compositions.

1.2.2.1 Coordination faible

La coordination est dite faible lorsque les tâches peuvent être divisées en sous-tâches indépendantes. Comme suggéré dans [Lundh et al. 2008], ces sous-tâches indépendantes seront désignées dans la suite de ce manuscrit sous le terme d'actions. Ce type de coordination est évidemment un cas favorable qui ne concerne cependant qu'un ensemble limité de tâches robotiques.

La planification classique, dite indépendante du domaine, est consacrée à la construction de plans où les actions ont exécutées en séquence ou en parallèle à partir simplement des descriptions de ces actions ; nous y reviendrons plus en détail dans la sous-sous-section 2.3.2.1 du chapitre suivant. Elle peut être employée avec des services logiciels ordinaires et par des composants logiciels ne traitant pas de flot de données. L'usage de ce

4. Les planifications de chemin, de mouvement, de navigation sont ici vues comme des fonctionnalités sensorimotrices.

type de planification pour les robots individuels est cependant essentiellement restreint à la planification de missions comme la séquence `déplacement vers une position puis saisie d'un objet` [Ghallab 2004]. La planification hiérarchique peut également être utilisée pour générer des plans d'actions [Ha et al. 2005] (sous-sous-section 2.3.2.2).

Au sein des systèmes multi-robots, l'allocation de tâches [Gerkey & Matarić 2004, Cao et al. 2012], aussi appelée assignation de rôles, a également recours à une décomposition en sous-tâches indépendantes qui sont menées par des robots différents. Enfin, la fusion entre plans d'actions [Alami et al. 1995] aux buts indépendants ayant un conflit sur des ressources est un cas intermédiaire nécessitant une coordination autour de l'usage de ces ressources. L'allocation de tâches et la fusion de plans d'actions ne seront pas davantage étudiés dans ce manuscrit.

1.2.2.2 Coordination étroite

Les fonctionnalités sensorimotrices, au cœur de la robotique mobile [Ghallab 2004], ne sont souvent pas reliées par des relations causales ou temporelles comme les actions mais par des flots d'informations [Lundh et al. 2008]. En conséquence, les planificateurs ne reposent plus sur des pré-conditions et des effets comme en planification classique mais requièrent la mise à disposition de connaissances spécifiques pour effectuer ces liaisons entre fonctionnalités. De telles compositions sont appelées *configurations* [Parker et al. 2005, Lundh et al. 2008, Gritti 2008]. La coordination entre sous-tâches y est étroite car les sous-tâches peuvent être amenées à interagir entre elles durant leurs exécutions. L'exemple-type de cette problématique est le déplacement d'un objet rigide, comme un carton [Chaimowicz et al. 2004] (voir la figure 1.9) ou une barre [Lundh et al. 2008], par deux robots mobiles. Les deux robots mobiles doivent en effet sans cesse coordonner leurs déplacements respectifs afin de respecter les contraintes physiques induites par leur tâche. Une solution courante est d'assigner un rôle de meneur à un robot et de suiveur à l'autre : le meneur envoie sa position et sa vitesse et le suiveur s'y adapte. Or dans certaines situations, le suiveur ne peut pas effectuer un déplacement compatible avec le déplacement du meneur de part la nature non-holonome [5] de son déplacement. Il est alors nécessaire d'inverser les rôles de meneur et de suiveur et donc de mettre en place un mécanisme de négociation de rôles [Chaimowicz et al. 2004].

Sur le plan logiciel, la coordination étroite est généralement effectuée au sein d'architectures à base de composants [Nahrstedt & Balke 2004, Kim et al. 2006, Gritti 2008, Lundh et al. 2008]. L'intégration de ce type de coordination dans une approche orientée service est également possible ; ce point sera discuté dans la section 3.1. La coordination étroite se traduit principalement par la mise en place de canaux de communications entre les composants devant être coordonnés [Parker et al. 2005, Lundh et al. 2008, Gritti 2008]. Ainsi ceux-ci reçoivent des flots continus d'information en provenance des autres composants durant leur exécution. Chaque flot d'entrée peut être vu comme une dépendance fonctionnelle. La configuration est alors un graphe dirigé où chaque arc représente un canal de communication unidirectionnel et où des boucles peuvent apparaître. Ces boucles peuvent avoir un impact très important sur le couplage fonctionnel car elles permettent d'implémenter par exemple le mécanisme de négociation de rôles et l'échange d'informations entre les deux robots transportant un objet rigide, comme

5. Bon nombre d'objets roulants ont un déplacement non-holonome : il est par exemple impossible pour une voiture ordinaire de se déplacer latéralement. Il est alors nécessaire d'effectuer une manœuvre comme un créneau pour atteindre la position désirée.

FIGURE 1.9 – Exemple de tâche nécessitant une coordination étroite : transport d'un carton par deux robots [Chaimowicz et al. 2004].

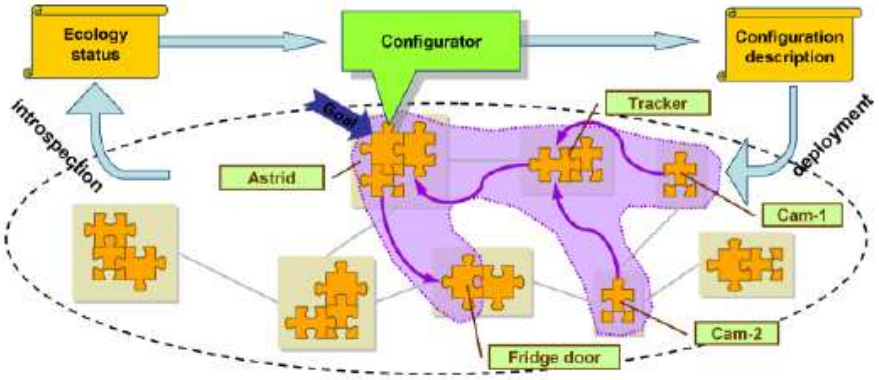


FIGURE 1.10 – Exemple de configuration dans *PEIS-Ecology* [Saffiotti et al. 2008]

proposé par [Lundh et al. 2008]. Nous reviendrons sur la question du couplage fonctionnel dans la sous-section 2.1.1 du chapitre suivant et nous verrons qu'elle représente un enjeu fondamental dans les approches orientées service et est un des principaux critères de distinction entre composants et services logiciels.

Le formalisme de réseaux de tâches hiérarchiques *(Hierarchical Task Network, HTN)* a été employé par plusieurs auteurs pour représenter ces configurations partiellement [Parker et al. 2005, Lundh et al. 2008, Gritti 2008] ou intégralement [Morisset & Ghallab 2008]. Les décompositions sont exprimées sous forme de sous-tâches connectées avec canaux de communications et non sous forme de séquences comme dans le cas de la coordination faible. Dans le cas où ces décompositions ne représentent que des configurations partielles, la configuration complète est construite grâce à un planificateur hiérarchique [Lundh et al. 2008] ou à l'aide d'une approche réactive [Gritti 2008]. Une configuration partielle permet par exemple d'obtenir le flot de la position globale d'un robot à partir de caméras fixes et d'un système de vision [Lundh et al. 2008]. Dans le cadre de la robotique ambiante, les fonctionnalités intégrées à la configuration proviennent de robots multiples, comme l'illustre la figure 1.10.

Si ses constituants ne peuvent pas être décrits en termes de pré-conditions et d'effets,

la configuration complète peut quant à elle être interprétée comme une action et de ce fait être intégrée au sein d'un plan d'actions de niveau supérieur, comme proposé dans [Lundh et al. 2008]. Ainsi l'apport d'une boisson depuis le réfrigérateur vers un utilisateur présent dans une autre pièce consiste dans cette dernière proposition en la génération d'un plan d'actions puis en l'exécution de chacune de ces actions sous forme de configuration. Le déplacement du robot vers la cuisine y est à la fois représenté comme une action et comme une configuration.

1.2.2.3 Sélection au sein de configurations de composants

La sélection au sein de plans d'actions s'apparente aux formes dominantes de la sélection de services ; la section 2.4 leur sera consacrée. Certains des travaux que nous venons d'évoquer au sujet de la coordination étroite effectuent une sélection sur des critères non-fonctionnels. Ces sélections peuvent être distinguées selon qu'elles surviennent durant la construction d'une configuration [Tang & Parker 2005, Lundh et al. 2008] ou consistent en la sélection d'une configuration complète [Morisset & Ghallab 2008].

[Lundh et al. 2008] effectuent des sélections durant la construction d'une configuration à partir d'un critère de coût. Ce coût est pour l'essentiel associé au nombre de composants utilisés et de canaux de communication et va donc en croissant au fur et mesure du développement de la configuration. Ainsi, il adopte une approche *branch and bound* (séparation et évaluation) pour élaguer les solutions dépassant la borne inférieure de coût et ce en amont, sans nécessiter que la configuration élaguée soit complète. Sa recherche est de type *best-first*[6] en explorant en priorité le nœud le plus prometteur. La borne inférieure de coût est obtenue à partir de la meilleure configuration complète connue.

[Tang & Parker 2005] s'intéressent au partage d'informations entre robots d'une même équipe devant se déplacer et effectuent une sélection robot par robot au sein d'une configuration à l'échelle de l'équipe. Cette sélection s'appuie sur une fonction d'utilité associant la probabilité de succès aux coûts des capteurs sollicités.

Robel (RObot BEhavior Learning) [Ghallab 2004, Morisset & Ghallab 2008] sélectionne une configuration complète prédéfinie et la remplace potentiellement à plusieurs reprises durant l'exécution complète d'une tâche. La sélection apparaît donc ici comme une prise de décision séquentielle et s'appuie sur un formalisme de processus de décision de Markov *(Markov Decision Process, MDP)*. La fonction de transition de ce MDP n'étant pas connue à l'avance, elle donne lieu à l'estimation de ses paramètres au fil des exécutions. Cette approche est consacrée à la navigation en intérieur et est en cela très proche de nos travaux. Nous allons donc revenir sur cette proposition dans la sous-section suivante qui est dédiée à cette tâche de navigation.

1.2.3 Navigation robotique en intérieur

La navigation d'un robot consiste à déplacer un robot d'une position à une autre. Cette sous-section présente succinctement les deux principales composantes de cette tâche : le mouvement et la localisation du robot.

6. Littéralement, le meilleur en premier

1.2.3.1 Mouvement

Diverses techniques peuvent être employées pour permettre le mouvement du robot vers sa destination. En règle générale, elles supposent la mise à disposition d'une information de localisation régulièrement mise à jour.

Planification de chemin Dans la majorité des cas, la trajectoire entre la position initiale et la destination ne s'effectue pas en ligne droite car il est nécessaire de changer de pièce ou d'éviter de volumineux obstacles. Dans de tels cas, il est souvent recommandé de planifier un chemin. Le lecteur intéressé par les principes de planification de chemin en trouvera une présentation succincte dans [Ghallab 2004]. Notons à cette occasion que cette littérature fait également usage du terme de configuration dans un sens très différent de celui qui a été retenu dans ce manuscrit : une configuration y représente la position, l'orientation et l'articulation du système robotique.

Approche réactive Une méthode réactive de mouvement n'utilise, par définition, que les valeurs courantes des capteurs et non les données provenant d'un modèle interne pour décider de l'action à effectuer [Filliat 2011]. La méthode des champs de potentiel est une méthode réactive très courante. Le robot y est vu comme une particule se déplaçant sous l'influence d'un champ de potentiel dépendant d'obstacles et de la direction souhaitée vers une destination intermédiaire. Le gradient de ce champ de potentiel donne la direction de déplacement du robot à partir de la position courante. De façon générale, les méthodes réactives sont utiles pour éviter les obstacles et atteindre une destination intermédiaire à moyenne distance. Elles peuvent donc être couplées à un chemin qui détermine les points intermédiaires à atteindre. Le principal défaut de ces méthodes est qu'elles risquent d'être bloquées dans des minima locaux comme des voies sans issue.

Ces approches permettent également de s'approcher d'un objet visible sans avoir à en connaître la position. Néanmoins, les destinations des robots sont généralement situées hors du champ visible par le robot depuis sa position initiale et n'ont souvent aucune marque visuelle particulière. En conséquence, les méthodes réactives ont ici besoin de connaître leur localisation.

Bande élastique La méthode de la bande élastique fournie une trajectoire flexible vers la destination finale pouvant être déformée en fonction des obstacles rencontrés. Cette méthode est robuste pour la navigation à longue distance mais est susceptible de connaître des minima locaux lorsqu'un obstacle bloque la bande contre un autre [Ghallab 2004]. Par ailleurs, c'est une méthode coûteuse qui doit donc veiller à ce que la largeur de sa bande ne soit pas trop importante et qui a une réactivité limitée peu efficace dans des environnements dynamiques et très obstrués.

1.2.3.2 Localisation

Le problème de la localisation consiste à estimer la position et l'orientation d'un robot vis-à-vis de son environnement. La localisation est généralement considérée comme une tâche complexe de part la dynamicité des environnements dont lesquels elle est effectuée. Cette dynamicité peut être par exemple due au passage d'humains aux alentours du robot qui forment alors des obstacles non anticipés.

Cette tâche repose sur l'utilisation de capteurs, qu'ils soient :

1. *proprioceptifs :* ils mesurent directement le déplacement du robot. C'est le cas par exemple de l'odométrie.
2. *ou extéroceptifs (perceptifs)* : ils mesurent directement la position du robot par observation de points de repères ou à partir de balises dans l'environnement. On retrouve ici la plupart des dispositifs évoqués ci-dessous, comme la télémétrie.

En robotique ambiante et plus généralement en intelligence ambiante, il est courant d'avoir recours à des systèmes de positionnement fixes installés dans l'environnement et constitués de capteurs sans fil ou de caméras. Grâce à la robotique autonome, les robots mobiles embarquent régulièrement des dispositifs de localisation extéroceptifs en plus de l'odométrie.

Dispositifs embarqués

Odométrie L'odométrie est le système de localisation le plus élémentaire et le plus répandu. Dans le cas des robots roulants, elle consiste principalement à mesurer la rotation des roues motrices à l'aide d'encodeurs. En tant que système proprioceptif, elle a pour principal défaut d'accumuler les erreurs. Elle ne mesure que les déplacements et ne peut donc pas fournir une position et une orientation en absolu sans connaître celles de son repère d'origine. Par ailleurs, elle devient fortement bruitée lorsque les roues glissent, ce qui peut être causé par le revêtement du sol, la tâche effectuée par le robot ou encore la trajectoire suivie. En conséquence, elle est rarement utilisée seule et vient généralement en complément d'un système extéroceptif. Elle entre en concurrence dans certaines méthodes avec le modèle cinématique associé à la commande demandée qui est lui utilisé en boucle ouverte. Son bruit est généralement modélisé comme gaussien [Thrun et al. 2001].

Boussole électronique Ce dispositif donne la direction du Nord géographique à partir du champ magnétique terrestre lorsqu'il n'est pas perturbé par les champs magnétiques ambiants produits par les fils et dispositifs électriques. Ce dispositif ne fournit qu'une information d'orientation. Ce dispositif est couramment employé dans *PEIS-Ecology* [Lundh et al. 2008].

Télémétrie Il existe trois grands types de télémètres : ultrasons, infrarouge et lasers à balayage (cf. figure 1.11a). Ces télémètres sont complémentaires pour la détection d'obstacles ou la prévention de chutes : par exemple, les télémètres lasers et infrarouge ne sont pas capables de détecter les vitres ni les surfaces chromées tandis que le télémètre à ultrasons l'est. Le télémètre laser est très utilisé pour la localisation à partir de cartes métriques de l'environnement comme par exemple avec les méthodes de localisation Monte-Carlo basées sur des filtres particulaires [Thrun et al. 2001] [7]. Ces techniques associent le modèle de mouvement représenté soit par la commande exécutée et son modèle cinématique, soit par l'odométrie avec le modèle de perception du capteur pour estimer les nouvelles position et orientation. Dans le cas du télémètre laser, le modèle de bruit n'est pas gaussien car il peut être perturbé par des obstacles inconnus comme

7. Les auteurs proposent également d'utiliser cette technique avec un télémètre à ultrasons et des caméras.

(a) Télémètre laser à balayage

(b) Capteur Cricket

FIGURE 1.11 – Télémètre laser à un axe de balayage et capteur Cricket

les humains et a une portée limitée [Thrun et al. 2001]. Ces techniques ont besoin de trouver des invariants dans l'environnement, ce qui pose problème dans des zones trop uniformes comme les couloirs où un télémètre laser n'obtient que très peu d'information dans l'axe de son déplacement [Morisset & Ghallab 2008]. La carte utilisée est souvent générée de façon autonome par le robot via la technique du *Simultaneous Localization And Mapping (SLAM)* [Montemerlo et al. 2002] que nous ne traiterons pas ici.

Vision à partir de caméras Comme mentionné précédemment, les méthodes de localisation Monte-Carlo peuvent être utilisées avec des caméras lorsque celles-ci sont embarquées sur des robots [Thrun et al. 2001]. Par ailleurs, une seule caméra peut être utilisée pour repérer des quadrangles caractéristiques comme des posters ou des portes lorsque le robot est à proximité [Hayet et al. 2007]. Cela permet de recaler l'odométrie de façon régulière, à partir par exemple des affiches présentes sur chaque porte d'un couloir [Hayet et al. 2007, Morisset & Ghallab 2008]. La mono-vision embarquée sur un robot est également utilisée de façon coopérative pour localiser d'autres robots [Parker et al. 2005, Lundh et al. 2008]. Cette approche consiste pour un robot à mesurer la position relative d'un autre robot par l'intermédiaire d'un système de vision et d'exploiter la position absolue de l'un pour inférer celle de l'autre en s'appuyant également sur l'orientation absolue du robot visionneur. Cette approche peut nécessiter une proximité entre les robots [8]. Un autre cas d'application similaire est celui des systèmes multi-caméras fixes installées au plafond, également utilisé dans le projet *PEIS-Ecology* [Muñoz-Salinas et al. 2007].

Dispositifs fixes

Balises ultrasons Cricket La technologie Cricket [Priyantha 2005] est un réseau de capteurs spécialisé dans la localisation permettant de mesurer les distances relatives entre chaque nœud ainsi que leurs orientations à partir des temps de vols de signaux ultrasons. Pour cela, un capteur Cricket dispose de deux récepteurs ultrasons (voir figure 1.11b). La position d'un nœud situé sur l'objet est obtenue par trilatération à partir des distances d'au moins trois nœuds fixes visibles et de la connaissance de leurs positions. La précision de ce dispositif dépend de la distance et du nombre de balises visibles. D'après [Priyantha 2005], ce dispositif peut atteindre une précision de positionnement de 10 cm et une erreur d'orientation de 3°. Toutefois de telles performances sont rarement vérifiées car ce système est facilement perturbé par les rebonds des signaux sur les obstacles et par le chevauchement des émissions de plusieurs balises. Néanmoins, ce dispositif présente

8. [Lundh et al. 2008] évoquent une distance maximale de 2 mètres.

l'avantage d'avoir un chemin de vision vertical, ce qui le rend moins sensible à la présence de personnes que les télémètres laser.

Empreintes digitales de puissances de signal Cette approche, issue de l'informatique ubiquitaire [Bahl & Padmanabhan 2000], consiste à exploiter les signaux de puissance *Received Signal Strength Indication* (RSSI) déjà disponibles dans l'environnement de part la présence de points d'accès WiFi ou de réseaux de capteurs sans fil. Cette information de mauvaise qualité peut être utilisée pour calculer une distance. Cependant son erreur croît proportionnellement au carré de la distance [Mao et al. 2007], est fortement perturbée et effectue généralement plusieurs rebonds, ce qui rend cette approche difficile à utiliser. Les méthodes les plus couramment employées se basent sur la reconnaissance d'empreintes digitales où une empreinte est le vecteur des signaux de puissance reçus [Bahl & Padmanabhan 2000, Sun et al. 2005]. Ces méthodes sont assez peu précises et ne fournissent pas d'information d'orientation.

1.2.3.3 Robel

Comme évoqué précédemment, Robel [Morisset & Ghallab 2008] est consacré à la sélection d'une configuration pour la navigation. L'exécution complète de la tâche de navigation consiste en une prise de décision séquentielle où la configuration courante peut être remplacée à chaque instant. Cette prise de décision séquentielle repose sur un processus de décision de Markov (MDP) dont nous allons détailler ici les principales particularités de façon informelle (le formalisme sera introduit dans la sous-section 4.1.1). Avant cela, les configurations utilisées sont présentées.

Configurations Les cinq configurations proposées sont composées des fonctionnalités suivantes :

1. *Planification de chemin, bande élastique et localisation avec le télémètre laser.*
2. *Planification de chemin, navigation réactive vers des points intermédiaires et localisation avec le télémètre laser.* La réactivité aux obstacles et l'attraction de la destination intermédiaire peuvent conduire à des oscillations qui perturbe la fonction de localisation et ce tout particulièrement dans les longs couloirs.
3. *Navigation réactive sans point intermédiaire, localisation avec le télémètre laser et vitesse réduite pour l'évitement d'obstacle.* Cette configuration est utile dans les environnements très encombrés et étroits où la planification de chemin pourrait échouer.
4. *Navigation réactive sans point intermédiaire, localisation par odométrie et par vision d'amers quadrangulaires.* Cette configuration est utile pour traverser de longs couloirs à condition que des amers y soient régulièrement espacés.
5. *Navigation réactive sans point intermédiaire et localisation par un dispositif extérieur fixe.* La nature du dispositif de localisation n'est pas précisée.

Contrôle Le processus de décision de Markov utilisé pour la sélection de configurations a plusieurs particularités.

État Cette approche utilise une représentation topologique de l'environnement : l'environnement est partitionné en cellules et celles-ci forment un graphe topologique à partir de leurs relations de voisinage. De plus, elle extrait des attributs spécifiques à la tâche de navigation qui sont directement observables par le robot. Plus précisément, l'état du MDP est caractérisé par les attributs suivants :

1. *Encombrement de l'environnement.* Sa valeur dépend de la distance des obstacles les plus proches. Ceux étant proches de l'axe de déplacement du robot ont un poids plus important.
2. *Variation angulaire.* La variation angulaire est faible près d'un mur alors que la valeur d'encombrement y est importante. Dans un environnement ouvert, l'encombrement est faible mais la variation angulaire peut être importante.
3. *Imprécision de la position actuelle du robot.* Cette valeur est estimée par la matrice de covariance du modèle de la fonctionnalité de localisation.
4. *Confiance dans l'estimation de la position du robot.* Cette valeur est estimée de façon heuristique et vient compléter la valeur d'imprécision.
5. *Couleur de la zone courante.* Chaque couleur correspond à une cellule étiquetée du graphe topologique de l'environnement. Ces cellules représentent un couloir ordinaire, un couloir avec des amers, une porte étroite ou encore un espace confiné.
6. *Configuration en cours d'utilisation.*

Chaque variable continue de cet espace est discrétisée en 3 ou 4 intervalles définis de façon *ad hoc*. De ce fait, l'espace des états est de taille restreinte : il ne contient que 5040 états. Ces valeurs sont mises à jour à la fréquence de 10 Hz.

Apprentissage par renforcement indirect La destination n'étant pas la même d'une exécution à une autre et n'apparaissant pas dans l'espace des états, les auteurs ont dû employer une approche indirecte de l'apprentissage par renforcement au lieu d'apprendre directement une fonction de valeur associée aux états ou aux paires état-action. La fonction de transition et une composante de la fonction de renforcement immédiat sont estimées au fur et à mesure des exécutions. Cette composante est la fonction de coût : ses paramètres sont mis à jour à partir du temps de traversée. La seconde composante de la fonction de renforcement est spécifique à la route dans le graphe topologique et est, par conséquent, définie à partir de la tâche de haut-niveau. Elle assigne une valeur négative à toute transition vers un état dont la couleur ne correspond pas à la couleur courante ni à la suivante dans la route topologique. Les autres transitions ont soit une valeur nulle, soit une valeur positive pour celles effectuant un changement de zone correspondant à la zone terminale. Enfin, à partir de ces fonctions accessibles pour un état courant et une destination donnée, la fonction d'utilité associée à chaque paire état-action est calculée à l'aide de l'algorithme d'itération de valeur [Bellman 1957].

Remarques Les auteurs [Morisset & Ghallab 2008] prétendent que le contrôleur ainsi appris est indépendant de l'environnement des spécificités d'un environnement particulier. Si cette propriété apparaît effectivement au niveau de la conception de ce contrôleur, il en va différemment de son apprentissage. En effet, certaines des fonctions de transition apprises concernent par exemple le passage d'une zone à une autre ; or le graphe topologique est spécifique à un environnement donné. Il en va de même pour les paramètres de la fonction de coût qui dépendent du temps de traversée. L'intérêt principal de cette représentation de l'état est l'importante factorisation qu'elle apporte. Néanmoins, comme

le reconnaissent les auteurs [Morisset & Ghallab 2008], l'élaboration de cette représentation a nécessité une connaissance fine de la tâche de navigation et de ses fonctionnalités sensorimotrices.

1.2.4 Positionnement

La robotique ambiante est le domaine d'application de nos travaux. Elle fait le lien entre la robotique mobile et l'intelligence ambiante ; les problématiques que nous avons dégagées précédemment (sous-section 1.1.3) concernent donc directement ce domaine.

Notre sélection de services devant effectuer ses choix suivant la performance des services candidats, nous nous intéresserons prioritairement aux compositions liées aux enjeux de la coordination étroite où des flots d'information continus sont communément employés. Alors que l'essentiel des travaux existants de ce courant s'appuient sur des composants logiciels, nous étudierons leur modélisation sous forme de services logiciels continus (section 3.1). La distinction entre composants et services sera discutée dans la sous-section 2.1.1.

Par la suite, nous prendrons comme exemple d'application la navigation d'un robot en intérieur et nous nous focaliserons principalement sur l'impact de la localisation. Dans ce cadre, nous effectuerons comme [Morisset & Ghallab 2008] un apprentissage par renforcement indirect (chapitre 4). Toutefois, notre approche se distingue de la leur par les considérations suivantes :

1. Leur fonction objectif se base sur une seule métrique, le temps de traversée. Dans notre approche, nous avons généralement plusieurs métriques, qui sont normalisées sur l'ensemble de l'exécution de façon non-linéaire avant d'être agrégées pour calculer un score. Par ailleurs, tous ces choix sont externalisés : ils sont effectués par l'utilisateur ou par son représentant au moment de la requête. Ces exigences ne peuvent pas être aussi aisément introduites dans la fonction de renforcement immédiat. Comme nous le verrons, ces exigences sont très liées à l'approche orientée service qui n'était pas le paradigme logiciel employé par ces auteurs.
2. Notre espace d'états de contrôle exploite directement une représentation métrique de l'environnement : il s'intéresse aux coordonnées cartésiennes de la position courante et la destination tandis que celui de [Morisset & Ghallab 2008] considère la cellule d'un graphe topologique ainsi qu'un ensemble d'attributs spécifiques à la tâche de navigation. Contrairement à ces auteurs, nous visons l'apprentissage d'idiosyncrasies de l'environnement. L'observation de ces particularités dépend des métriques de performance choisies par l'utilisateur et nous ne supposons pas la disposition d'une connaissance fine permettant de dégager des attributs pertinents pour chacune d'entre elles. Ces attributs fournissent une représentation de l'état centrée sur la perception du robot qui convient bien à des services de positionnement embarqués dans le robot. Cependant, au contraire de ces travaux, la majorité des services de positionnement utilisés dans notre scénario sont fixes et externes au robot. Notre approche semble plus adéquate pour rendre compte des angles morts de ces services de positionnement qui dépendent de leurs positions d'installation et celle du robot à positionner.
3. Nous nous focalisons sur le positionnement et nous ne proposons qu'une seule alternative pour la gestion du mouvement du robot.

4. Notre approche est capable d'intégrer de nouveaux services et de gérer leurs indisponibilités. Ce point n'a à notre connaissance pas été évoqué par ces auteurs.

1.3 Conclusion

Ce chapitre a présenté la robotique ambiante à partir de ses deux domaines d'origine, l'intelligence ambiante et la robotique mobile. Divers points en ont été retenus et listés dans les sous-sections 1.1.3 et 1.2.4. En résumé, l'intelligence ambiante s'intéresse aux environnements actifs offrant de multiples fonctionnalités et place l'utilisateur au centre du système. La robotique mobile, quant à elle, introduit la problématique de la coordination étroite et fournit la tâche de navigation qui constituera notre scénario.

Le chapitre suivant présente un état de l'art sur les services logiciels et les architectures qui les utilisent. Il se concentre sur les services réalisant des actions et qui, par conséquent, sont faiblement coordonnés. L'adaptation des services et de ces architectures au cas de la coordination étroite sera étudiée dans le chapitre 3.

Chapitre 2

Architectures pour les services réalisant des actions

Sommaire

Le paradigme logiciel porté par les services est consacré à la mise en relation et à l'interaction de systèmes communicants indépendants. Dans leur usage le plus courant, les services fournis par ces systèmes indépendants réalisent des actions qui, comme nous l'avons vu dans le chapitre précédent (sous-section 1.2.2), ne nécessitent qu'une coordination faible. Ce second chapitre d'état de l'art est consacré consacre une place centrale à ce type de service ; les cas d'utilisation nécessitant une coordination étroite entre services seront étudiés dans le chapitre suivant.

Ce chapitre est organisé de la façon suivante. Dans un premier temps, il introduit la notion de service et la distingue des approches à base de composants puis présente les enjeux dégagés et les technologies employées par les communautés des systèmes d'information d'entreprise, du World Wide Web, de l'intelligence ambiante et de la robotique (section 2.1). Ensuite, il expose les autres problématiques générales de l'informatique

orientée service : la description sémantique, la découverte (section 2.2), la composition (2.3) et la sélection de services (2.4). Enfin, il conclut en positionnant notre problématique vis-à-vis de ces travaux (section 2.5).

2.1 Services Web

La notion de service est une abstraction logicielle visant à mettre des fonctionnalités à disposition de tiers tout en garantissant à ces derniers un certain niveau d'indépendance. À ce titre, elle partage plusieurs points communs avec la notion de composant, points que la prochaine sous-section élucidera. Les services visent à être accessibles à un large public d'applications, ce qui leur impose de reposer sur des standards aussi répandus que possibles. Aussi, les travaux autour des services se sont intéressés très tôt aux standards du World Wide Web, au point de les qualifier de services Web. En ce sens, IBM [IBM 2000] a défini en 2000 les services Web comme

> «des applications modulaires auto-contenues *(self-contained)* pouvant être décrites, publiées, localisées et invoquées via un réseau ou plus généralement au travers du World Wide Web.»

Les services Web réutilisent de nombreux standards Internet et Web pré-existants, normalisés par leurs institutions respectives, l'*Internet Engineering Task Force* (IETF) et le *World Wide Web Consortium* (W3C). D'autres institutions comme l'*Organization for the Advancement of Structured Information Standards* (OASIS) ou l'*International Organization for Standardization* (ISO) viennent compléter l'ensemble de ces protocoles.

Dans un premier temps, cette section introduit la notion de service en la comparant à celle de composant logiciel et en insistant sur la minimisation du couplage fonctionnel. Ensuite, elle s'intéresse aux développements et aux enjeux dégagés par les communautés des systèmes d'information d'entreprise, du Web, de l'intelligence ambiante et de la robotique.

2.1.1 Notion de service

Comme nous l'avons vu dans le chapitre précédent, les architectures logicielles utilisées pour la réalisation peuvent être à base de composants ou de services. Ces deux notions logicielles proches, souvent amalgamées, méritent d'être distinguées. La notion de service peut être vue comme une évolution du concept de composant de la même façon que ce dernier est issu de la programmation orientée objet [Hock-koon & Oussalah 2011]. Ce parallèle est intéressant à plusieurs titres :

1. Le service représente un troisième stade dans la *séparation entre l'interface et l'implémentation* après celles introduites par les composants et les objets. Les informations liées au déploiement telles que les dépendances fonctionnelles ou l'usage d'une plateforme spécifique disparaissent des interfaces des services : elles sont reléguées du côté de l'implémentation. Ainsi il est courant d'employer une plateforme à base de composants pour implémenter un service et ce sans que ce choix n'ait de conséquence sur l'interface. Cette plateforme agit comme un conteneur auprès des composants en assurant leur exécution et en configurant leurs relations afin de gérer leurs dépendances fonctionnelles. Le service a quant à lui une relation inverse avec une telle plateforme : il a autorité sur elle et est en cela *auto-contenu*. Libérée des contraintes de déploiement, l'interface du service est

susceptible de devenir directement accessible à un public de clients beaucoup plus large en se basant sur des standards. Ces standards sont majoritairement issus de l'IETF et du W3C si bien que la diversité de modèles de services est bien plus restreinte que celle des modèles de composants [Crnkovic et al. 2011]. Les clients d'un service peuvent alors se concentrer sur l'envoi de requêtes et l'interprétation des réponses. Ils n'ont pas accès à l'implémentation du service : son exécution est en conséquence assumée par le fournisseur de services. Ce cas de figure n'est pas systématique dans les approches à base de composants car les composants dits *off-the-shelf*[1] doivent être déployés sur la plateforme du client avant exécution [Di Nitto et al. 2008]. Néanmoins, l'exécution à distance est une pratique courante des composants distribués. Cet aspect représente un changement radical dans des domaines comme le jeu vidéo en ligne [Hock-koon & Oussalah 2011] où la responsabilité de la performance est transférée au fournisseur et peut désormais faire l'objet d'un contrat.

2. Ces deux évolutions entraînent à chaque fois une perte d'expressivité de part les contraintes qu'elles introduisent [Hock-koon & Oussalah 2011]. Cette perte est due à une augmentation de l'exigence de *minimisation du couplage fonctionnel*. Cette grande exigence est une caractéristique essentielle des approches orientées service sur laquelle nous reviendrons à de nombreuses reprises.

Ces deux évolutions ont également connues des influences extra-logicielles, respectivement la construction de systèmes à partir de composants matériels dans d'autres domaines de l'ingénierie et l'externalisation de certaines activités d'une entreprise auprès de sous-traitants [Hock-koon & Oussalah 2011].

Couplage fonctionnel Les services logiciels se caractérisent par une grande exigence en matière de couplage fonctionnel. En ingénierie logicielle, le couplage mesure l'interdépendance entre deux modules. Appliqué aux services, le couplage fonctionnel concerne *l'ensemble des hypothèses devant être partagées par le client et le fournisseur de services pour pouvoir interagir*. La restriction de cet ensemble est cruciale pour garantir un bon niveau d'indépendance entre les clients et les fournisseurs en abaissant la barrière à l'entrée et en limitant autant que possible l'impact des changements [Hohpe & Woolf 2003, Pautasso & Wilde 2009]. Elle peut cependant entraîner une complexification de l'implémentation et une baisse de performance par rapport à des solutions plus couplées.

Cette notion est régulièrement sollicitée dans les débats autours des architectures orientées service de façon vague. Toutefois, quelques travaux [Kaye 2003, Krafzig et al. 2005, Pautasso & Wilde 2009] ont tenté de la préciser. L'annexe A.1 reproduit la classification proposée par Pautasso et Wilde [Pautasso & Wilde 2009] qui s'inscrit dans le contexte du débat entre les approches REST-HTTP (sous-section 2.1.3) et WS-* (sous-section 2.1.2) et vise à orienter les choix techniques effectués en interne de ces deux approches. Elle repose sur douze facettes comme l'origine de la description les messages échangés (auto-décrite ou reposant sur un modèle partagé) et l'interaction avec ou sans état.

Vocabulaire Avant d'étudier les différentes technologies employées dans l'industrie et la littérature académique, nous présentons ici les principaux termes employés dans les approches orientées service sous un angle technologiquement neutre.

1. Littéralement : «sur l'étagère».

Client *Émetteur d'une requête de service.*

Fournisseur de services *Infrastructure se chargeant de l'implémentation d'un ou de plusieurs services.* Un fournisseur de services peut correspondre à une machine virtuelle tout comme à un robot mobile.

Service *Entité logicielle auto-contenue exposant une ou plusieurs fonctionnalités au travers d'opérations.* Pour les besoins de la sélection de services, les opérations d'un même service ne doivent pas être concurrentes pour réaliser une même tâche (cf. section 2.4). Son indépendance est permise par sa gestion en interne de ses dépendances fonctionnelles : un service est auto-contenu au lieu d'être configuré par un conteneur de niveau supérieur.

Opération *Fonctionnalité adressable pouvant être désignée par une requête de service.*

Requête de service *Message à destination d'une opération d'un service donné décrivant une tâche à réaliser.* Une requête est un message concret qui est directement traitable par un service. Lorsque le service n'a pas encore été sélectionné, la description de la tâche demandée peut être incluse au sein d'une requête de découverte ou de sélection de services ; elle sera ensuite traduite en une requête de service à l'issue de la sélection. La réalisation de cette tâche se matérialise par le traitement d'une requête.

Instance de service *Instance consacrée au traitement d'une requête de service.* Chaque traitement de requête implique donc une instance de service. Lorsqu'une instance de service est adressable, elle peut exposer de nouvelles opérations qui lui sont spécifiques (sous-figure 2.1b). Un grand nombre de traitements de requêtes ne donnent pas lieu à la réification [2] d'une instance de service si bien que ce terme est peu employé dans la littérature. C'est notamment le cas des interactions synchrones où l'unique message reçu en réponse à une requête contient le résultat ; aucune adresse sur le traitement ayant abouti à ce résultat n'a été communiquée durant son exécution (sous-figure 2.1a). Néanmoins, l'approche qui sera proposée dans le chapitre 3 s'appuiera sur cette notion (cf. sous-section 3.1.1).

Service atomique *Service ne faisant pas appel à d'autres services dans ses traitements de requêtes.*

Service composite *Service faisant appel à d'autres services durant le traitement de requêtes adressées à certaines de ses opérations.*

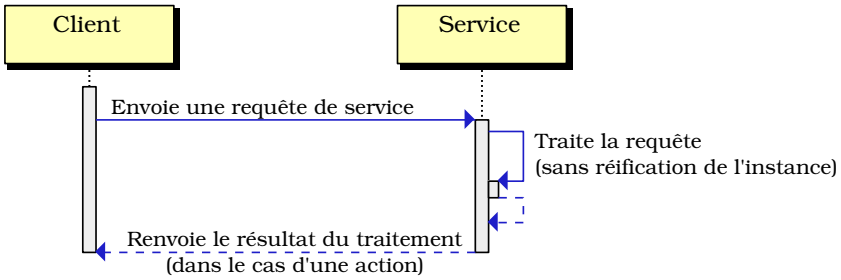
Instance de service composite *Instance de service faisant appel à d'autres instances de sous-services.* Ces sous-services peuvent être sélectionnés lors de l'exécution de l'instance de service composite.

Description de service *Description de l'interface d'un service.* Cette description détaille les opérations et publie certaines propriétés non-fonctionnelles du service.

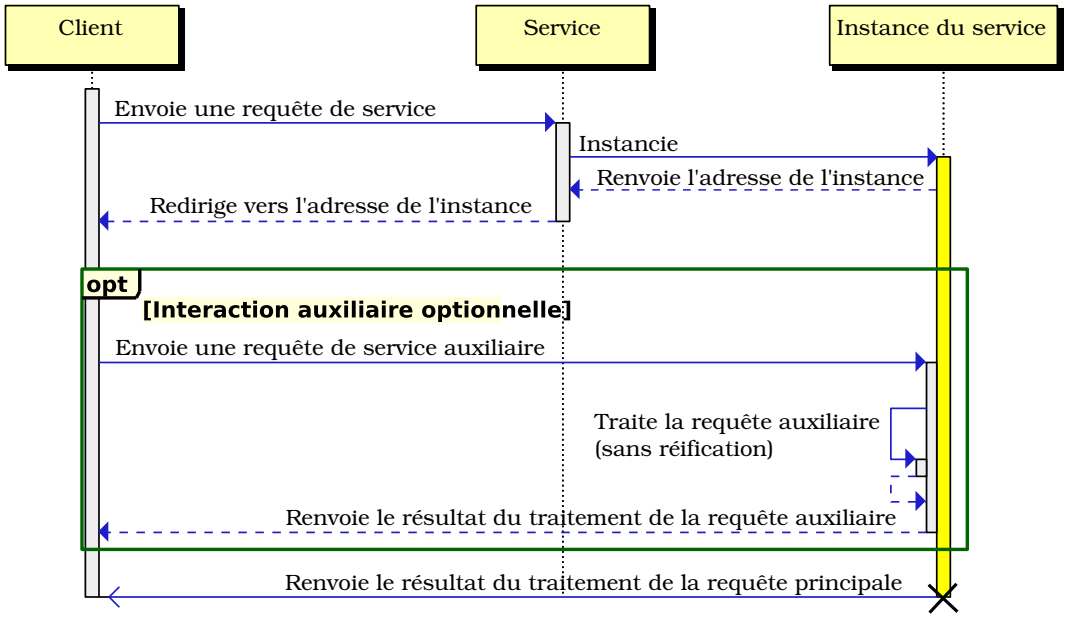
Nous enrichirons ce vocabulaire dans le chapitre suivant avec les notions de *service continu* (section 3.1), *d'organigramme* (sous-section 3.1.2) et de *couplage non-fonctionnel* (sous-sous-section 3.2.2.2).

Après cette présentation générale de la notion de service, voyons comment elle est utilisée par diverses communautés.

2. La réification d'une instance de service consiste ici à lui donner une adresse et une interface. Elle devient dès lors un «objet» manipulable.



(a) Interaction synchrone sans réification de l'instance



(b) Interaction asynchrone avec réification de l'instance

FIGURE 2.1 – *Interaction avec ou sans réification de l'instance.* Ces sous-figures sont données à titre illustratif et devraient être précisées pour être mises en œuvre dans les protocoles SOAP (sous-section 2.1.2) ou REST-HTTP (2.1.3). La boucle de traitement n'est pas explicitée dans l'instance de service (sous-figure 2.1b) car l'instance elle-même est dédiée au traitement de la requête principale. Cette instance peut par ailleurs recevoir de nouvelles requêtes auxiliaires. La réification de ces nouvelles requêtes forme des instances d'instance de service.

2.1.2 Systèmes d'information d'entreprise

Les services Web ont été introduits dans les systèmes d'information d'entreprise de façon importante depuis une décennie. Ces services, soutenus par les industriels du domaine comme Microsoft et IBM, reposent dans leur majorité sur le protocole SOAP [W3C 2007b] et ses extensions WS-*. Ces standards apportent une couche d'interopérabilité face à l'hétérogénéité technologique des multiples solutions d'éditeurs composant le système d'information de bon nombre d'entreprises. Un des principaux objectifs est la réutilisation de fonctionnalités métiers.

Cette sous-section commence par brièvement décrire les briques fondamentales de cette approche : le protocole SOAP et la description de service WSDL. Elle présente ensuite succinctement quelques extensions WS-* ainsi qu'une architecture intergicielle qui est souvent mise en place entre ces services. Enfin, elle résume les particularités de cette approche.

2.1.2.1 SOAP

SOAP est, selon sa spécification, un protocole «conçu pour échanger des informations structurées au sein d'un environnement décentralisé ou distribué»[W3C 2007b]. À l'origine, SOAP signifiait *Simple Object Access Protocol*, mais n'est désormais plus un acronyme depuis sa version 1.2, suite à la disparition de cette notion d'objet. SOAP repose sur les technologie XML *(eXtensible Markup Language)* et en particulier sur des schémas XSD *(XML Schema Definition)* pour décrire ses messages.

Structure Un message SOAP est une enveloppe XML composée d'un élément XML entête (`<soap:Header>`) et d'un élément XML corps (`<soap:Body>`) (voir figure 2.2). Comme d'usage dans de nombreux protocoles tels qu'HTTP et SMTP [3], la partie entête contient les métadonnées du message et le corps son contenu. À la différence de ces derniers, SOAP n'est pas un protocole de transport mais un protocole *wire*, c'est-à-dire un format [Papazoglou 2012]. Il est conçu pour être indépendant du protocole de transport, ce qui permet à un même message de transiter par plusieurs protocoles comme HTTP, SMTP ou JMS *(Java Message Service)* sans avoir à convertir les métadonnées du message d'un protocole à l'autre : elles sont uniquement contenues dans l'élément d'entête de l'enveloppe SOAP.

Styles de communication On trouve principalement deux styles de communication en SOAP : l'appel de procédures distantes et l'envoi de document. L'appel de procédures distantes (*Remote Procedure Call* ou RPC) est un style de communication emprunté aux architectures à base de composants. Il permet à un client de mimer un appel de procédure locale en cachant le fait que celle-ci soit distante par l'intermédiaire d'un mécanisme de souche *(stub)*. Cela se traduit en SOAP par l'envoi d'un élément XML au nom de la méthode appelée contenant la liste des arguments passés puis par l'attente bloquante d'une réponse. Cette réponse peut contenir plusieurs paramètres ou retourner une exception. Ce style est réputé fortement couplé [Pautasso & Wilde 2009] principalement parce qu'il suppose un modèle partagé pour connaître les paramètres à envoyer et est très sensible à son changement.

3. Acronymes respectifs de *HyperText Transfer Protocol* et *Simple Mail Transfer Protocol*

```
POST /magasin HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <a:passeCommande xmlns:a="http://www.example.org/achat" date="01-07-2012" >
      <a:acheteur>Benjamin</a:acheteur>
      <a:article>Produit A</a:article>
    </a:passeCommande>
  </soap:Body>
</soap:Envelope>
```

FIGURE 2.2 – Exemple de message SOAP transporté dans une requête HTTP

L'usage conseillé est d'envoyer un document décrit par un schéma XSD. Ce document peut représenter par exemple un ordre d'achat (voir figure 2.2). Grâce à la description XSD de son schéma, il peut être validé et converti dans d'autres formats à partir de règles de transformation XSLT *(eXtensible Stylesheet Language Transformations)*. La granularité grossière de cette description du message offre une stabilité suffisante pour certains messages, leur garantissant ainsi une certaine pérennité dans le temps. Ce style encourage l'asynchronicité : le patron de base le plus courant est l'envoi de message à sens unique en fournissant une adresse de rappel pour recevoir une réponse.

2.1.2.2 Web Service Description Language (WSDL)

Afin de pouvoir communiquer en SOAP avec un service, le client a besoin de connaître l'interface de celui-ci. Pour cela, l'approche courante est de fournir une description à partir du standard *Web Service Description Language* (WSDL). Ce standard décrit la localisation d'un service, les opérations qu'il fournit ainsi que les détails nécessaires à son invocation. Chaque opération du service fournit une fonctionnalité. Lorsque celui-ci est accessible dans le style RPC, une opération peut être vue comme une méthode distante à appeler.

Une description WSDL est strictement syntaxique ; seule la structure de ses fonctionnalités est décrite par l'intermédiaire des schémas XSD impliqués. Ceci est suffisant pour générer automatiquement du code client, mais cette pratique est contestée de part l'important couplage qu'elle entraîne [4] [Pautasso & Wilde 2009]. Des extensions sémantiques seront présentées dans la section 2.2. Le lecteur intéressé par le standard WSDL peut consulter l'annexe A.2.1 pour avoir plus de détails.

4. Ce point est l'objet de la facette *génération de code* de l'annexe A.1

2.1.2.3 Extensions WS-*

Les standards WS-* viennent compléter SOAP et WSDL par des fonctionnalités d'adressage, de communication événementielle, de description de politiques, de sécurité, de gestion de transactions ainsi que de la fiabilité. Nous ne donnons ici qu'un bref aperçu de certaines de ses extensions. La problématique complexe de la sécurité ne sera pas traitée dans cette thèse. Des descriptions plus complètes sont disponibles en annexe A.2.2.

Adressage des messages et des extrémités communicantes : WS-Addressing SOAP ne fournit pas en lui-même l'adressage des extrémités communicantes mais le relègue à la couche de transport (par exemple aux entêtes HTTP ou SMTP), ce qui limite le nombre de patrons de communication possibles. La spécification WS-Addressing [Box et al. 2004] vient ajouter cette fonctionnalité, appelée *EndPoint Reference* (EPR), en l'intégrant aux entêtes SOAP et aux descriptions WSDL. Cette fonctionnalité est rendue indépendante de la couche de transport [Papazoglou 2012]. Il devient possible de désigner l'émetteur, la destination mais aussi les futurs destinataires de la réponse, d'une éventuelle erreur ou d'un accusé de réception. Au niveau de WSDL, la description des extrémités communicantes est augmentée en permettant d'adresser dynamiquement les instances de services via la propriété `resourceID`. Cet identifiant serait nécessaire à une implémentation en SOAP de notre approche (cf. sous-section 3.1.1).

Communication événementielle : WS-Notification et WS-Eventing Un autre mode de communication utilisé dans certaines architectures distribuées est la communication événementielle entre services de type publication-souscription *(publish-subscribe)* [Carzaniga et al. 2000]. Un service intéressé par les événements sur un sujet ou émanant d'un service particulier peut être notifié lorsqu'un nouveau message (appelé événement) est publié moyennant l'inscription de son EPR auprès du service émetteur ou d'un intermédiaire que ce dernier a désigné. Pour cela, l'OASIS et le W3C ont publié deux spécifications concurrentes et non-compatibles : WS-Notification [OASIS 2006] et WS-Eventing [W3C 2011]. La notion de service continu, qui sera introduite dans la section 3.1 du chapitre suivant, accorde une place centrale à la communication événementielle.

Politiques : WS-Policy La communication entre services exige régulièrement l'échange de propriétés supplémentaires à celles décrites dans les descriptions WSDL. Ces propriétés prennent la forme d'assertions qui expriment des préférences, des exigences ou des capacités et sont regroupées en ensembles cohérents, appelés politiques [Papazoglou 2012]. Ces politiques couvrent divers domaines comme la performance, la sécurité, la fiabilité des messages et la gestion de transactions et s'appuient sur un framework standardisé par le W3C : WS-Policy [W3C 2007c]. Chaari et al. [Chaari et al. 2008] en ont proposé une extension permettant la confrontation entre les exigences d'un client et les capacités d'un fournisseur en termes de paramètres de Qualité de Service (QdS).

2.1.2.4 Bus de services d'entreprise

Jusqu'à présent, les spécifications que nous venons de voir pourraient permettre à des services de communiquer entre eux de façon distribuée, sans intermédiaire. Toutefois,

cela supposerait qu'ils soient mis en relation, soient capables de gérer leur probable hétérogénéité et supportent eux-mêmes ces standards. Au lieu d'aborder cette problématique sous cet angle, il est courant dans l'approche WS-* de mettre en place une *architecture intergicielle entre ces services*[5]. Un intergiciel a pour objectifs d'aider la mise en relation et de cacher l'hétérogénéité des éléments communicants [Krakowiak 2006]. Nous présentons ici l'architecture d'intergiciel orienté service la plus courante : l'*Enterprise Service Bus* (ESB) ou bus de services d'entreprise [Chappell 2004].

Un ESB joue le rôle de médiateur entre les consommateurs et les fournisseurs de services et assure de cette façon leur découplage. En conséquence, il est responsable du routage des messages (qu'ils s'agissent de requêtes ou d'événements) mais aussi de leur transformation lorsque les services interagissant utilisent des formats et des protocoles de communication différents. Profitant de sa position d'intermédiaire, il peut aider à la mise en œuvre de certaines spécifications WS-*, comme la gestion de la fiabilité des messages, la communication événementielle et la sécurité. La transformation de message d'un format à un autre est son activité qui consomme le plus de ressources [Papazoglou 2012].

Cette zone de médiation est généralement centralisée, elle peut être fédérée [Papazoglou 2012] mais elle ne peut être déployée que dans un environnement contrôlé et ne convient donc pas aux environnements ouverts [Georgantas et al. 2010]. Du fait du découplage des services et de sa main-mise sur le routage des messages, elle représente l'endroit de prédilection pour la supervision des services mais aussi de leur composition [Zeng et al. 2007]. On y retrouve les fonctionnalités de découverte, de sélection mais également de composition de services [Leymann 2005] que nous étudierons au cours des prochaines sections.

2.1.2.5 Bilan

Cette approche de services WS-* repose sur des standards et est en mesure d'offrir une nette séparation entre l'interface et l'implémentation de ces services. Sa principale particularité est d'être centrée sur l'échange de documents qui peuvent être routés entre services grâce à l'extension WS-Addressing ou par l'intermédiaire d'un ESB. Cette logique est familière des processus métiers sur laquelle nous reviendrons dans la section 2.3 consacrée à la composition de services. La réutilisation des fonctionnalités métiers permises par ces processus nécessite régulièrement la conversion de messages, qui est facilitée par l'usage systématique d'XML.

L'exposition des services WS-* se limite traditionnellement aux extrémités de communication qui ne disposent généralement que d'un nombre limité d'opérations. Toutefois, leurs instances peuvent être rendues adressables via l'extension WS-Addressing. Dans son usage en entreprise, cette approche est peu confrontée aux problèmes de traduction d'adresse (*Network Address Translation*, NAT), ce qui facilite la mise en place de communication asynchrone et événementielle. Sur le plan des pratiques de développement, les services sont souvent développés pour un usage interne à l'entreprise et il est fréquent que leurs interfaces soient conçues en ayant la possibilité de modifier la partie

5. Notons qu'une part importante des intergiciels employés dans ces systèmes doivent être placés du côté de l'implémentation des services. C'est le cas par exemple des intergiciels liés à une plateforme de composants ainsi que de la plupart des intergiciels orientés messages. Les intergiciels entre services sont moins courants et ne doivent pas perdre de vue les enjeux de minimisation du couplage fonctionnel portés par les services.

cliente. Le contrôle simultané des implémentations du client et du service rend possible la mise en place d'un couplage fonctionnel important entre eux. Celui-ci pénalisera à long terme l'évolution et la réutilisation de ces services. Par ailleurs, ce type de service est minoritaire sur le Web [6] de part ses difficultés de passage à l'échelle. Ceci est en partie dû au fait qu'il ne peut pas bénéficier des mécanismes de cache HTTP, contrairement à l'approche REST-HTTP.

2.1.3 World Wide Web

Les services disponibles sur le Web adoptent dans leur majorité [7] la plupart des principes du style d'architecture REST *(REpresentational State Transfert)*. Cette sous-section présente les aboutissants de ce style d'architecture ainsi que son application à HTTP qui est le protocole applicatif communément employé par ces services. Enfin, elle discute des principales contributions apportées par ce style et ses pratiques.

2.1.3.1 REpresentational State Transfer (REST)

Le Web a été conçu par ses principaux concepteurs comme un système hypermédia distribué à large échelle, dépassant les frontières des organisations et devant offrir une faible barrière à l'entrée [Fielding 2007]. L'un d'entre eux, Roy Fielding [8], a proposé dans sa thèse de doctorat une formalisation a posteriori des principes architecturaux fondateurs de ce système, intitulée *REpresentational State Transfert* (REST) [Fielding 2000]. Cette formalisation prend la forme d'un style d'architecture, c'est-à-dire d'un «ensemble coordonné de contraintes qui restreignent les rôles et les fonctionnalités des éléments architecturaux ainsi que les relations autorisées entre ces éléments» [Fielding 2007]. REST définit les six contraintes suivantes :

1. *Les préoccupations d'une application doivent être séparées en deux rôles : client et serveur.* L'état de l'application est gardé par le client tandis que tous les états partagés, c'est-à-dire ceux des ressources, sont gérés par le serveur. Cette séparation des rôles, couplée aux interactions sans état, améliore la résilience en cas de défaillance d'un des rôles. Ainsi, un échec du serveur ne met pas à mal l'état du client et un état partagé est récupérable en tant que ressource.
2. *Les interactions doivent être sans état.* Ceci simplifie le serveur puisqu'il ne doit plus gérer de session et aide par conséquent son passage à l'échelle. En contrepartie, les messages échangés ne peuvent pas faire référence à un état précédent et sont donc plus volumineux.
3. *Des mécanismes de cache doivent être mis en place* pour réduire la charge du serveur et améliorer son passage à l'échelle.
4. *Les ressources doivent être accessibles via une interface uniforme* afin d'améliorer la visibilité et de découpler les implémentations. Tout d'abord, l'interface uniforme exige que les ressources importantes aient un identifiant unique et que les méthodes d'accès aient une sémantique bien définie pour toutes ces ressources. Par ailleurs, les ressources sont manipulées par l'échange de représentations, qui doivent être des messages auto-descriptifs. Enfin, les représentations renvoyées

6. 17% en mai 2011 des services inscrits sur le site `programmableweb.com` [Musser 2011].
7. 73% des services répertoriés par le site `programmableweb.com` en mai 2011 [Musser 2011].
8. Premier auteur de la spécification *HyperText Transfer Protocol* (HTTP) 1.1 [IETF 1999a].

par le serveur doivent fournir des indications sur les prochaines actions possibles à la manière d'un hypermédia.

5. *L'architecture doit être en couches* afin de permettre la balance de charge, le partage de cache sans que le client n'ait à s'en soucier.
6. *L'architecture peut offrir du code client à la demande*, mais cela reste optionnel.

Hypermédia À partir des identifiants universels des ressources et en effectuant des liens entre elles, l'architecture a une structure d'hypermédia, qui devient alors le «moteur de l'état de l'application» (HATEOAS [9]) [Fielding 2000]. Les représentations reçues par le client doivent alors contenir des liens vers d'autres ressources, c'est-à-dire vers de possibles prochains états de l'application. Certains de ces liens ne se contentent pas de faire référence à une ressource mais peuvent également indiquer comment lui transférer un état : c'est le cas par exemple des formulaires HTML *(HyperText Media Language)*. Ces informations sont transmises à l'exécution, ce qui réduit la part de connaissances préalables nécessaires au client et par là-même le couplage fonctionnel [10]. Ces enjeux importants mais exigeants portés par l'hypermédia sont négligés par la majorité des services actuels [Fielding 2008b] qui se regroupent eux-mêmes sous l'appellation «REST pragmatique». Le lecteur intéressé par les enjeux de cette problématique est invité à consulter l'ouvrage de Mike Amundsen [Amundsen 2011] ainsi que sa description des facteurs hypermédia *(H-Factors)* [Amundsen 2010] [11].

2.1.3.2 REST-HTTP

Bien qu'inspiré par le Web, REST en est son abstraction en un style d'architecture. Son application au Web traditionnel implique directement le protocole HTTP [12], les URI *(Uniform Resource Identifier)* et les *media-type* Internet.

Représentations de ressources Une application devient fortement adressable lorsqu'elle expose les aspects intéressants de ses données comme des ressources. Une ressource peut correspondre aussi bien à un objet physique qu'à des éléments plus conventionnels tels qu'un document, une entrée dans une base de données ou le résultat d'un algorithme. Celle-ci doit cependant avoir une ou plusieurs représentations. Elle est référencée par au moins une URI. Une représentation est n'importe quelle information utile à propos de l'état d'une ressource. Lorsque plusieurs représentations sont disponibles, le protocole HTTP permet d'effectuer une négociation de contenu. L'exemple type de négociation concerne les préférences de langues ; cependant, on peut également négocier le format (HTML, XML, JSON [13], etc.). Conformément au principe HATEOAS, ces représen-

9. *Hypermedia As The Engine Of Application State*
10. L'évolution du service est par ailleurs plus aisée puisque le client redécouvre ces transitions à chaque interaction sans avoir besoin de générer du code client.
11. L'ouvrage successeur de [Richardson & Ruby 2008] et [Amundsen 2011] est paru au moment de l'envoi de ce mémoire aux rapporteurs [Richardson & Amundsen 2013]. Il serait intéressant de relire cette sous-section à la lumière des avancées proposées dans ce dernier opus.
12. D'autres protocoles Web liés à HTTP comme WebDAV *(Web Distributed Authoring and Versioning)* et CoAP *(Constrained Application Protocol)* revendiquent le style REST. WebDAV est une extension d'HTTP venant enrichir l'interface uniforme des verbes `PROPFIND`, `PROPPATCH`, `MKCOL`, `COPY`, `MOVE`, `LOCK` et `UNLOCK`. CoAP se veut être facilement traduisible en HTTP et adopte la même interface uniforme.
13. *JavaScript Object Notation*

tations doivent contenir autant que possible les URI d'autres ressources en tant que candidates pour le prochain état de l'application.

Interface uniforme d'HTTP L'interface uniforme repose au niveau du protocole HTTP sur ses verbes ainsi que ses codes de retour [Alarcón et al. 2011]. La description des verbes HTTP est disponible en annexe A.3.1. Les principaux sont `GET`, `POST`, `PUT` et `DELETE`. Certains d'entre eux réalisent des transferts d'états à partir de représentations. À l'exception de `POST`, tous les verbes ont la propriété d'idempotence, ce qui permet d'envoyer plusieurs fois le même message tout en garantissant d'arriver au même résultat. La méthode `GET` et d'autres méthodes secondaires sont mêmes sûres car elles n'entraînent pas de changement d'état côté serveur. Ces propriétés sont très utiles pour l'infrastructure car elles lui permettent d'effectuer des optimisations comme la mise en cache. De plus, elles apportent de la robustesse à un réseau non fiable comme Internet : lorsqu'une requête n'a pas reçu de réponse, il suffit de la réitérer sans crainte d'effet de bord. La méthode `POST` est la seule à être délicate. Elle est toutefois systématiquement utilisée par HTTP RPC et SOAP-HTTP.

Interaction asynchrone et communication événementielle Une interaction asynchrone diminue le couplage entre le client et le serveur lorsque celui-ci est très sollicité ou effectue des traitements coûteux. Ce type d'interaction est géré par HTTP d'ordinaire de la façon suivante. HTTP étant un protocole synchrone, le client attend de façon bloquante une réponse à sa requête. Si cette réponse est une réponse intermédiaire (de type `202 Accepted`), le client effectue une nouvelle requête attendant cette fois-ci la génération d'un nouvel état pour recevoir une réponse. Il s'agit de la technique du *HTTP long polling*. De façon générale, un client peut souscrire aux changements d'états d'une ressource avec cette technique qui est toutefois coûteuse car elle nécessite de maintenir une connexion TCP *(Transmission Control Protocol)* de façon permanente. Néanmoins, étant effectuée à l'initiative du client, celui-ci n'a pas besoin de disposer d'un serveur Web ni d'une adresse publique, exigences qui sont au contraire requises par la communication événementielle.

La communication événementielle ne fait pas partie du style d'architecture REST car elle induit une inversion des responsabilités entre le client et le serveur et passe difficilement à l'échelle [Fielding 2008a]. Pour être viable, il est selon cet auteur nécessaire de responsabiliser le client afin qu'il ne souscrive pas à trop de ressources et s'en désinscrive. Elle nécessite donc des mécanismes de gestion des clients et de filtrage des événements qui sont trop spécifiques au domaine d'application pour être intégrés dans le style d'architecture [Fielding 2008a]. En conséquence, il n'existe pas de méthode dans l'interface uniforme d'HTTP pour gérer ce type de communication. La communication événementielle est donc généralement réalisée de façon ad-hoc ou à l'aide de protocoles supplémentaires comme PubSubHubbub [Fitzpatrick et al. 2010] impliquant l'usage du format Atom [IETF 2005] côté serveur pour décrire les événements. Comme évoqué précédemment, la communication événementielle est très liée à la notion de service continu qui sera introduite dans le chapitre suivant.

2.1.3.3 Description syntaxique de services

La fourniture de descriptions compréhensibles par une machine est encore une pratique minoritaire pour les services se référant à l'approche REST. En effet, la plupart d'entre eux ne fournissent qu'une documentation de leur interface de programmation à destination des développeurs. Les URI des requêtes doivent être construites par le programme client à partir d'une structure documentée. Il en résulte des implémentations clientes fortement couplées à la documentation [Pautasso & Wilde 2009]. Par ailleurs, la structure des adresses des ressources étant connue des clients, les représentations ne contiennent que très rarement des liens vers d'autres ressources. De fait, cette pratique viole le principe hypermédia (HATEOAS) de REST [Fielding 2008b].

Il existe aussi plusieurs formats de description syntaxique des services : WADL *(Web Application Description Language)* [Hadley 2006] à base de XML, SPORE *(Specification to a POrtable Rest Environment)* [Cuny 2011] et Swagger [Wordnik 2012] qui sont quant à eux décrits en JSON. De même que l'approche documentaire, ces descriptions servent généralement à la génération de code client. WADL est présenté en annexe A.3.2.

WSDL a donné un sens très précis à ses opérations de services. L'équivalent en REST-HTTP d'une opération WSDL associe un verbe HTTP à un sous-ensemble de ressources. Dans certains cas, ce sous-ensemble de ressources peut être représenté par un patron d'URI (*URI Templates* [IETF 2012]) où certains champs qui étaient des paramètres de l'opération WSDL apparaissent à des endroits spécifiques de l'URI. Ce mécanisme est intensément utilisé dans les descriptions mentionnées ci-dessus. Néanmoins, REST-HTTP permet l'opacité des URI ; dès lors, l'association entre une ressource et certains champs doit être découverte à titre individuel.

2.1.3.4 Résumé

Le style REST, en étant centré sur les ressources, encourage à son extrême l'exposition des services où tout élément jugé utile peut être représenté par une ressource. Les possibilités de réutilisation se trouvent alors considérablement augmentées [Vinoski 2008]. Par ailleurs, son application à HTTP fournit une architecture très performante, capable de répondre aux fortes exigences de passage à l'échelle qui caractérisent le Web. À cet égard, bon nombre de ces services sont exposés à un public très large et très varié de clients. Le choix des hypothèses devant être partagées par les clients et les fournisseurs de services y est très sensible. La séparation entre l'interface et l'implémentation y est généralement très nette : par exemple, les infrastructures sous-jacentes des principaux acteurs sont très sophistiquées pour pouvoir supporter la charge tandis que leurs interfaces restent simples et soignées.

Les principes hypermédia sont encore généralement négligés mais leurs enjeux devraient croître au fur et à mesure que les clients interagiront avec une multitude de services comme c'est le cas depuis longtemps pour les navigateurs Web. La mise en relation entre clients et serveurs sur le Web est encore majoritairement manuelle. Enfin, REST-HTTP ne propose pas de mécanisme standard pour la communication événementielle.

2.1.4 Intelligence ambiante

L'intelligence ambiante s'intéresse à des services situés dans un environnement particulier. Elle introduit deux nouveaux enjeux au sein des architectures orientées service : l'automatisation de mise en relation entre clients et services ainsi que l'adaptation des compositions ou des applications au contexte.

2.1.4.1 Automatisation de la mise en relation

Les environnements augmentés par des dispositifs d'intelligence ambiante sont en règle générale des environnements ouverts, où les services sont hétérogènes et dont certains ne sont présents que de façon intermittente. La mise en relation entre clients [14] et services doit être dynamique, opportuniste et autant que possible automatique. Plus précisément, elle se concrétise par la mise en place des mesures suivantes :

1. *Connaissance des services présents dans l'environnement local.* Les multiples mécanismes de découverte dynamique offrent des mécanismes de publication et de requêtage permettant aux services d'informer de leur arrivée ou de leur départ ainsi que de rechercher les services susceptibles de satisfaire une tâche requise. La sous-section 2.2.3 est consacrée à cette problématique.
2. *Test de correspondance entre une description et une tâche requise.* Les solutions industrielles comme UPnP *(Universal Plug and Play)* [ISO 2008] reposent sur le respect de descriptions de services standardisées comme par exemple celle d'un service d'impression. L'interaction qui en suit est immédiate puisque les rôles du client et du service sont définis par le standard. Néanmoins, ce type d'approche est limitée à un sous-ensemble restreint de services bien définis [15] et ne peut pas prendre en charge l'hétérogénéité des autres services disponibles. La sous-section 2.2.2 s'intéresse aux tests de correspondance s'appuyant sur des descriptions sémantiques par opposition aux approches syntaxiques comme UPnP ou WSDL.

UPnP et *Device Profile Web Service* (DPWS) *Universal Plug and Play* [ISO 2008] est une architecture orientée services basée sur le protocole SOAP se présentant comme une extension du mécanisme du *Plug and Play* (dédié aux périphériques connectés directement à l'ordinateur) aux dispositifs disponibles sur le réseau. Il décrit les interactions entre les périphériques *(devices)* UPnP et les points de contrôle *(control points)* jouent le rôle de consommateurs. UPnP est composé de plusieurs protocoles qui lui sont spécifiques pour la gestion des événements (*General Event Notification Architecture*, GENA), la découverte dynamique locale (*Simple Service Discovery Protocol*, SSDP) et la description des fournisseurs et des services. Par ailleurs, un certain nombre de profils ont été standardisés avec pour horizon la certification d'appareils grand public.

Device Profile Web Service (DPWS) [OASIS 2009a, Jammes et al. 2005] est le successeur d'UPnP [16] se basant cette fois-ci sur les extensions standardisées WS-*. Il emploie notamment les standards WSDL, WS-Addressing, WS-Eventing, WS-Policy ainsi que WS-Discovery [OASIS 2009b] pour la découverte dynamique de services.

14. Les services composites sont eux-mêmes les clients de leurs sous-services.
15. Comme les services multimédia *UPnP Audio and Video* certifiés par la *Digital Living Network Alliance* (DLNA).
16. Le changement de nom est motivé par des raisons légales et l'absence de rétro-compatibilité.

2.1.4.2 Adaptation au contexte

Comme évoqué dans le chapitre précédent (section 1.1), la prise en compte du contexte est un aspect essentiel de l'intelligence ambiante. Appliquée à l'informatique orientée service, elle concerne principalement la localité et les préférences de l'utilisateur. Enfin, elle peut exiger une réaction lors de changements significatifs en cours d'exécution.

Localité Dans certains cas d'usage, l'utilisateur est en situation de mobilité. En fonction des lieux dans lesquels il se trouve, les réseaux et les services disponibles diffèrent. Les compositions de services doivent alors s'y adapter ; nous y reviendrons dans la sous-sous-section 2.3.2.3. Pour notre part, notre domaine d'application se place dans le cadre favorable d'un environnement unique, domestique, disposant d'une infrastructure réseau locale. De plus, le service de haut-niveau considéré dans notre scénario est exécuté de façon ponctuelle, à la demande, en réponse à une requête de service de haut-niveau émanant de l'utilisateur ou de son représentant. Ainsi son exposition aux changements de contexte en cours d'exécution est moins importante que dans le cas des services fournis à un utilisateur en situation de mobilité. Nous pourrons ainsi faire l'hypothèse que les attentes de l'utilisateur ne changent pas durant une exécution.

Par ailleurs, certains services locaux ont une portée limitée. Par exemple, une caméra rotative installée au plafond a un champ de vision restreint à une pièce où à une zone de cette pièce. Ces restrictions, lorsqu'elles sont explicitées, peuvent alors être prises en compte par le module de découverte de services lors des tests de correspondance entre la tâche requise et les descriptions de services.

Préférences de l'utilisateur Les préférences de l'utilisateur, dépendantes du contexte, peuvent également être prises en considération lors de l'adaptation dynamique des compositions (cf. sous-sous-section 2.3.2.3) ou lors de la sélection de services (section 2.4 et chapitre 3).

Réaction aux changements de contexte La réaction aux changements de contexte suppose en règle générale une communication événementielle entre ces services. Ce type de communication est courant auprès des services Web SOAP disponibles dans le réseau local, ce qui justifie leur plus grande popularité en intelligence ambiante. Néanmoins, les services HTTP-REST, qui sont eux destinés en premier lieu au Web, peuvent faire usage de la technique du *HTTP long polling* de façon modérée ou être augmentés d'un mécanisme de publication et de souscription. Les conséquences de la souscription d'un service à un flot d'événements seront étudiées dans la section 3.1 consacrée aux services continus.

2.1.5 Robotique

L'emploi de services logiciels en robotique est une pratique moins développée que dans les domaines précédemment évoqués. L'externalisation des fonctionnalités n'y est pas encore une problématique centrale, si bien que les solutions reposant sur une homogénéité entre le client et le serveur sont encore fréquentes [Blake et al. 2011]. Néanmoins, cette sous-section recense quelques propositions employant des services pour réaliser des actions. Elles permettent le contrôle à distance de robots mobiles ou s'inscrivent dans

la lignée des domaines précédents. Par ailleurs, cette sous-section évoque les deux principales plateformes logicielles robotiques et désambiguïse leur rapport à l'informatique orientée service.

2.1.5.1 Applications similaires aux domaines précédents

Les robots mobiles en intérieur étant désormais connectés aux réseaux sans fil informatiques, ils peuvent solliciter des services locaux ou sur le Web en utilisant les solutions logicielles décrites dans les trois sous-sections précédentes. Nous en dressons ici une liste représentative succincte. [Ha et al. 2005] utilisent des services SOAP pour requêter les capteurs de température et les divers actionneurs situés dans un appartement comme les stores, l'éclairage, l'air conditionné ou un robot mobile. Ces actions sont issues d'un plan généré automatiquement de façon hiérarchique (cf. sous-sous-section 2.3.2.2). [Kim et al. 2005] utilisent un service SOAP distant pour obtenir des informations sur un objet à partir de l'URI stockée dans son tag RFID. [Ambroszkiewicz et al. 2010] sollicitent quant à eux des services atomiques de déplacement de robots, de reconnaissance visuelle d'objets et d'ouverture de portes. Enfin, le projet *RoboEarth* [Waibel et al. 2011] permet le partage d'expérience entre robots à partir d'une base de données accessible via un service REST-HTTP. Ce service permet à un robot de tirer partie de l'expérience de précédents robots pour trouver son chemin dans un labyrinthe, apprendre la trajectoire de la porte d'un placard ou encore reconnaître et saisir une boisson afin de la livrer à un usager.

2.1.5.2 Contrôle à distance

Une autre application courante dans la littérature robotique faisant appel à des services est le contrôle à distance de robots mobiles. Ces robots mobiles ne fournissent pas un service autonome de déplacement entre deux points distants mais simplement un service dont les opérations consistent à changer un paramètre de l'algorithme de contrôle de ce déplacement [Majedi et al. 2008, Chen et al. 2009, Cardozo et al. 2010]. De telles opérations peuvent alors être modélisées comme des actions mais aucune d'entre elles ne représente le processus sous-jacent de déplacement : celui-ci *évolue en dehors du traitement d'une requête bien définie*. Les approches ordinaires de sélection de services (cf. section 2.4) ne peuvent alors pas considérer la performance associée à ce déplacement. Dans le cas où la vitesse de déplacement est négligeable, le déplacement d'un robot peut être décomposé en de courts déplacements dont les paramètres sont fixes et déterminés par rectification de la trajectoire précédente [Prüter et al. 2009] ; le processus de déplacement est alors directement associé à l'opération sollicitée. Ces propositions emploient des services HTTP-REST [Prüter et al. 2009, Cardozo et al. 2010] et DPWS [Prüter et al. 2009].

2.1.5.3 Plateformes robotiques

Les deux principales plateformes pour la conception d'applications robotiques, *Robot Operating System* (ROS) [Quigley et al. 2009] et *Microsoft Robotics Developer Studio* (MRDS) [Jackson 2007, Frystyk Nielsen & Chrysanthakopoulos 2007] qualifient de services certains de leurs composants distribués bien que ceux-ci ne prennent pas en charge certains aspects essentiels des approches orientées service.

Comme expliqué dans la sous-section 2.1.1, une architecture orientée service relègue la problématique du déploiement du côté de l'implémentation afin d'en décharger le client. Suivant cette séparation, le client se concentre sur l'envoi d'une requête à un service d'un fournisseur comme un robot mobile. Les éventuels déploiements et paramétrages comme par exemple la connexion entre robot mobile et un capteur distant particulier font partie intégrante du traitement de la requête effectué par le service et ne nécessitent aucune action préalable de la part du client. Or MRDS demande au client de s'assurer que les composants nécessaires soient déployés et configurés [17] avant de déclencher l'exécution d'une action comme le déplacement du robot vers une destination en changeant l'état d'un composant particulier de la configuration concernée [18]. MRDS ne peut donc pas traiter directement une requête de service de déplacement ; une couche logicielle supplémentaire doit être ajoutée pour compléter l'implémentation d'un tel service. Cette plateforme offre toutefois une interface HTTP-REST permettant d'accéder et parfois de manipuler l'état de ses composants. Cette pratique vise en premier lieu à faciliter l'observation du comportement des applications.

De même, la plateforme ROS ne prend pas directement en charge le déploiement automatique d'une configuration nécessité par le traitement d'une requête de service. Par ailleurs, ses interfaces fonctionnelles, qu'elles soient de type synchrone (RPC) ou événementiel, sont simplement décrites avec des tableaux et des structures de types primitifs (par exemple des entiers représentés sur 32 bits) et n'offrent en conséquence qu'un degré d'abstraction très limité.

2.1.5.4 Bilan

Les actions en robotique peuvent aisément être réalisées à partir de services. Néanmoins, la modélisation sous forme d'action ne se prête pas convenablement à toutes les fonctionnalités robotiques. C'est le cas du contrôle à distance où le processus sous-jacent est alors mal intégré à l'architecture dès lors que celui-ci est suffisamment dynamique. Les actions sont destinées à des tâches autonomes, qu'elles soient ponctuelles (obtention d'une information par exemple) ou de plus longue durée (déplacement entre deux points distants). Comme évoqué dans la sous-section 1.2.2, la décomposition des tâches autonomes propres à la robotique nécessite une approche permettant une coordination étroite. Les plateformes robotiques à base de composants distribués ROS et MRDS sont principalement destinées à l'approche de type configuration et c'est pourquoi elles font un usage important des mécanismes de communication événementielle. La section 3.1 du chapitre suivant reviendra sur l'équivalent de ces configurations de composants sous forme de services.

Cette section a mis l'accent sur les approches orientées service industrielles employées dans les communautés des systèmes d'information d'entreprise, du World Wide Web, de l'intelligence ambiante et de la robotique. Les seules descriptions de services présentées étaient de nature syntaxique ; l'apport de leur enrichissement sémantique pour la découverte et de la composition de services est l'un des principaux objets des deux prochaines sections.

17. Ces entités logicielles ne sont donc pas auto-contenues.
18. Une configuration est ici la résultante d'un ensemble de partenariats spécifiés dans les descriptions des composants [Frystyk Nielsen & Chrysanthakopoulos 2007].

2.2 Description sémantique et découverte de services

Nous avons vu que la minimisation du couplage fonctionnel entre un client et les services qu'il consomme est la motivation première de l'approche orientée service. Dans ce cadre, la découverte de services peut s'avérer être une étape essentielle dans la réalisation d'une tâche requise par un client. Elle s'appuie *a minima* sur la description des fonctionnalités de ces services, description qui pourra être enrichie de propriétés non-fonctionnelles. Les services considérés dans cette section sont supposés réaliser des actions.

Dans cette section, nous verrons tout d'abord pourquoi les descriptions syntaxiques évoquées dans la section précédente présentent des insuffisances qui ont motivées de nombreuses extensions sémantiques. Enfin nous présenterons différents mécanismes permettant de récupérer ces descriptions.

2.2.1 Descriptions sémantiques de services

WSDL, dans son utilisation préconisée, encourage l'échange de documents XML décrits par des schémas XSD. Ces schémas sont interprétés comme des contrats. Dans de nombreux cas, les interlocuteurs utilisent des formats différents, ce qui nécessite une conversion à partir de règles XSLT. À défaut de description sémantique de ces schémas, la production des règles de transformation n'est pas entièrement automatisable et nécessite l'intervention d'un humain reconnaissant la compatibilité de ces formats. Cette approche ne peut être viable que dans un cadre où l'hétérogénéité des interlocuteurs est limitée et maîtrisée. Par ailleurs, la fonctionnalité d'une opération n'est décrite qu'indirectement par les schémas des paramètres d'entrée et de sortie. Leurs schémas ne peuvent pas être génériques (par exemple une liste de personnes décrites par leurs noms) car sinon leur fonctionnalité serait alors inconnue.

L'ajout d'un niveau sémantique introduit des capacités de raisonnement logique sur ces descriptions. Il vise à mettre en relation les services capables de réaliser une tâche requise ainsi qu'à automatiser l'éventuelle conversion de format, dans le cas où des technologies XML seraient utilisées au niveau inférieur. Pour cela, il s'appuie sur les technologies du Web sémantique : ontologies légères (RDFS [19] [W3C 2004b]) ou lourdes (OWL [20] [W3C 2009]), règles logiques (SWRL [21] [Horrocks et al. 2004] , N3Logic [Berners-Lee et al. 2008], RIF [22] [W3C 2010]) et graphe RDF [23] [W3C 2004a].

2.2.1.1 Descriptions de services WS-*

Les premiers travaux autour des services Web dits sémantiques [24] se sont focalisés sur l'ajout de modèles sémantiques aux descriptions WSDL.

19. RDF Schema
20. Web Ontology Language
21. Semantic Web Rule Language
22. Rule Interchange Format
23. Resource Description Framework
24. Le terme *Semantic Web Services* a été introduit par [McIlraith et al. 2001]

OWL-S OWL-S est une ontologie de description des services Web proposée par [Martin et al. 2005]. Elle est composée de trois parties : le profil, le modèle et l'ancrage *(grounding)*.

1. Le profil, destiné à la découverte de services, décrit les opérations du service en termes de paramètres d'entrées-sorties, de pré-conditions et d'effets. Les préconditions et les effets peuvent être exprimés dans des langages de règles logiques (comme SWRL et N3 Logic) dont la diversité pose des problèmes en terme d'intégration des différents interpréteurs nécessaires [Verborgh et al. 2011]. De plus, lorsque les pré-conditions ne sont pas supportées, elles sont ignorées [Verborgh et al. 2011].
2. Les opérations sont modélisées comme des processus de trois types : atomiques, simples ou composites. Les processus atomiques et simples sont vus comme ayant une seule étape d'interaction avec le client ; la différence est que le processus simple est abstrait, pouvant être atomique ou composite. Un processus composite est décrit comme l'agencement de sous-processus à l'aide de structures de contrôle [Sirin et al. 2004]. Ces processus n'émanent pas forcément du même fournisseur de services, qui peut alors être mis en concurrence avec des fournisseurs ayant des profils équivalents.
3. L'ancrage est effectué en WSDL.

SAWSDL *Semantic Annotations for WSDL and XML Schema* (SAWSDL) est une recommandation du W3C [W3C 2007a] intégrant des annotations au niveau des éléments des descriptions WSDL. Les éléments concernés sont les opérations et les schémas XSD associés aux paramètres d'entrée-sortie. SAWSDL permet le passage des entrées et des sorties d'un format XML à RDF et inversement via respectivement les attributs `liftingSchemaMapping` et `loweringSchemaMapping`. Le cas montant implique des règles de transformations XSLT tandis que le cas descendant s'effectue à l'aide d'une requête SPARQL [25] [W3C 2008]. Ainsi il est possible d'automatiser la conversion de données XML d'un schéma à un autre en passant par un graphe RDF intermédiaire. Toutefois, la spécification de ces règles reste délicate [Pedrinaci & Domingue 2010]. À la différence d'OWL-S et d'autres approches similaires, SAWSDL ne fournit pas d'ontologie.

WSMO-Lite WSMO-Lite [Vitvar et al. 2008] se présente à la fois comme une extension des annotations SAWSDL ainsi qu'une ontologie légère RDFS. L'ontologie, inspirée du modèle WSMO [26], permet de classifier les fonctionnalités du service, d'exprimer des propriétés non-fonctionnelles comme des valeurs constantes de paramètres de qualité de service ainsi que des pré-conditions et des effets. Ces derniers sont exprimés dans le format de règles RIF préconisé par le W3C.

2.2.1.2 Descriptions de services REST

Avec les services REST, les opérations ne sont plus directement liées aux services mais aux ressources. Leurs descriptions sémantiques décrivent donc les opérations de chacune des ressources ; du fait du grand nombre de ces dernières, les descriptions doivent être factorisées. Par ailleurs, le respect de la structure hypermédia d'un service Web invite à

25. *SPARQL Query Language for RDF*
26. *Web Services Modelling Ontology*

l'annotation des liens hypertextes entre ressources. hRESTS/MicroWSMO et SA-REST ne considèrent pas le principe HATEOAS contrairement aux descriptions ReLL et RESTdesc.

hRESTS/MicroWSMO et SA-REST hRESTS [Kopecky et al. 2008] permet l'annotation sémantique des services à partir de la page HTML de documentation de leur interface de programmation, accessible de la racine du service. Elle annote les opérations, leurs entrées et leurs sorties à l'aide de microformats [27] ou d'annotations RDFa. Son extension MicroWSMO [Kopecky et al. 2008] fournit des fonctionnalités similaires à SAWSDL et s'appuie sur l'ontologie WSMO-Lite. L'approche SA-REST [Lathem et al. 2007] est similaire à hRESTS/MicroWSMO mis à part qu'elle n'utilise pas une ontologie particulière.

ReLL *Resource Linking Language* (ReLL) [Alarcón & Wilde 2010a] est une description permettant de filtrer ou de créer des liens depuis une représentation d'une ressource donnée puis de les nommer. Le nommage de ces liens pourra faire l'objet d'une transformation GRDDL vers une représentation RDF. Ainsi, il est possible de naviguer entre les ressources d'un service, bien que celui-ci n'ait pas initialement inclus de liens dans les représentations de ses ressources. Par ailleurs, les descriptions peuvent être spécialisées pour un type de navigation en sélectionnant certains liens. La sélection des liens se fait à partir d'expressions régulières tandis que leur construction exploite le langage XPath. De façon similaire, les descriptions sont liées à des ressources via des expressions régulières.

RESTdesc RESTdesc [Verborgh et al. 2011] est une description fonctionnelle de services s'appuyant sur Notation 3 Logic [28] [Berners-Lee et al. 2008]. La figure 2.3 présente les descriptions de deux ressources de téléversement et de téléchargement d'images. Une description est construite comme un triplet d'implication, où un graphe de pré-condition implique un graphe de post-condition. Si la pré-condition est vraie, la post-condition peut être exécutée ; le cas échéant elle effectuera une requête HTTP et ajoutera des triplets RDF à la base de connaissance du client. Par ailleurs, des liens vers des ressources avoisinantes sont fournis sous forme d'entête HTTP Link [IETF 2010] et ce indépendamment de la représentation choisie. Ainsi par exemple, une requête GET auprès d'une image peut fournir un lien vers son auteur avec l'entête suivante :

```
Link: /authors/benjamin; rel="http://dbpedia.org/ontology/author"
```

2.2.2 Test de correspondance

Une fois qu'un service est décrit, il devient possible de tester la correspondance entre ses fonctionnalités et la tâche requise par le client. La tâche requise est généralement supposée être décrite dans un format proche de celui des opérations de service. En fonction du niveau de description d'un service, le test de correspondance peut soit être syntaxique, soit impliquer un raisonnement sur les entrées-sorties ou sur les pré-conditions et les effets.

27. Dans ce cas, il est nécessaire d'effectuer une opération GRDDL *(Gleaning Resource Descriptions from Dialects of Languages)* pour transformer les annotations en triplets RDF.

28. Notation 3 (N3) est une représentation et une extension de RDF ajoutant la quantification et les graphes imbriqués. N3Logic est sa logique.

```
@prefix : <http://restdesc.no.de/ontology#>.
@prefix http: <http://www.w3.org/2006/http#>.
@prefix tmpl: <http://purl.org/restdesc/http-template#>.
@prefix foaf: <http://xmlns.com/foaf/>.

# Téléversement d'une image vers le service
{
    ?photo a foaf:Image.
} => {
    _:request http:methodName "POST";
        http:requestURI "/photos";
        http:body [ tmpl:formData ("photo=" ?photo) ];
        http:resp [ tmpl:location ("/photos/" ?photoId) ].
        ?photo :photoId _:photoId.
}.

# Téléchargement d'une image depuis le service
{
    ?photo :photoId ?id.
} => {
    _:request http:methodName "GET";
        tmpl:requestURI ("/photos/" ?photoId);
        http:resp [ tmpl:represents ?photo ].
}.
```

FIGURE 2.3 – Description RESTdesc d'une ressource de téléchargement d'image. D'après [Verborgh et al. 2011]

Correspondance syntaxique Dans le cas de WSDL, la correspondance syntaxique est établie dès lors que les entrées de la tâche requise peuvent être converties suivant le schéma XSD de l'opération du service et inversement pour les sorties. Le schéma XSD induit la catégorie du traitement réalisé par l'opération.

Raisonnement hiérarchique sur les entrées et les sorties [Paolucci et al. 2002b] ont émis l'idée de tester les relations de subsomption entre les entrées et les sorties de deux descriptions lorsque celles-ci sont décrites par des ontologies. Cette idée a été depuis reprise dans de multiples travaux [Klusch et al. 2005, Ben Mokhtar et al. 2008].

Une classe A subsume une classe B lorsque B hérite [29] directement ou indirectement de A ; la classe A est alors considérée comme plus générale que B. Outre le cas où les concepts des entrées-sorties seraient identiques entre les descriptions de la tâche requise et de l'opération du service, le test de subsomption met en évidence un autre cas valide : il faut que les sorties requises soient identiques ou subsument les sorties de la description du service et que l'opération inverse soit vraie pour toutes les entrées. Par exemple, un client souhaitant louer un véhicule à faible encombrement et disposant de son permis de conduire sur lui pourra ainsi trouver satisfaction auprès d'un loueur de vélos acceptant une pièce d'identité officielle.

Toutefois, les types des sorties n'induisant généralement pas un type de fonctionnalité précis, il convient d'effectuer un test supplémentaire de subsomption sur la catégorie de l'opération du service [Martin et al. 2005, Ben Mokhtar et al. 2008, Vitvar et al. 2008]. Prenons pour exemple une opération de service prenant en entrée le matricule d'une personne et retournant en sortie le nom et le prénom d'une personne. Sans catégorie, il n'est pas possible d'inférer qu'il s'agit du nom et du prénom de la personne identifiée et non celle par exemple d'un parent, d'un ami ou d'une autre relation. La catégorie apparaît ici comme essentielle pour faire le lien entre les entrées et les sorties. Elle peut être associée à des règles permettant la conversion de la description en pré-conditions et en effets.

Raisonnement sur les préconditions et les effets Lorsque les deux descriptions disposent de pré-conditions et d'effets, le raisonnement peut s'effectuer en termes de transformation d'états [Stollberg et al. 2007, Junghans et al. 2010]. Les entrées et sorties sont alors considérées comme des variables libres et leurs types sont intégrés aux pré-conditions et aux effets. Le raisonnement s'opère également à l'aide de relations de subsomption qui s'intéressent cette fois aux ensembles d'états désignés par les pré-conditions et les effets.

Par ailleurs, des exigences non-fonctionnelles, liées par exemple à la sécurité ou à des niveaux de performance garantis, peuvent être intégrées sous forme d'effets requis [Junghans & Agarwal 2010]. Une extension de OWL-S a également été proposée pour modéliser les propriétés non-fonctionnelles [Jean et al. 2012].

2.2.3 Découverte de services

Lorsqu'un client souhaite interagir avec un service qu'il ne connaît pas encore, il émet une requête auprès d'un dispositif de découverte de services. Ce dispositif recherche parmi

29. Notons que l'héritage multiple est une pratique courante des ontologies.

tous les services auxquels il a accès ceux dont la description correspond à la requête. Nous allons commencer par le dispositif d'annuaire, puis nous évoquerons la problématique de la découverte dynamique.

2.2.3.1 Annuaire

Nous présentons ici l'annuaire le plus courant ainsi que les optimisations pouvant être réalisées auprès des services répertoriés.

UDDI *Universal Description, Discovery and Integration* (UDDI) [OASIS 2002] est un standard industriel d'annuaire central de services. À l'origine, il devait être utilisé comme courtier entre les clients et les services. Toutefois, les services n'étant répertoriés sur le plan technique que par leurs descriptions syntaxiques WSDL et par un ensemble de mot-clés, les capacités d'interrogations par UDDI sont trop limitées pour offrir un tel niveau d'automatisation [Junghans et al. 2010, Kourtesis & Paraskakis 2008]. Pour y pallier, des extensions sémantiques ont été proposées [Paolucci et al. 2002a, Kourtesis & Paraskakis 2008]. Ce standard n'a pas eu l'écho espéré si bien que les principaux annuaires UDDI publics ont été fermés en 2005 [Krill 2005]. Toutefois, on le trouve régulièrement dans des réseaux internes d'entreprises.

Pré-traitements Le temps de traitement d'une requête de découverte peut être considérable car il implique des inférences sur des ontologies et d'éventuelles pré-conditions et effets. Le regroupement de descriptions de services dans un ou plusieurs annuaires offre l'occasion d'effectuer des opérations hors-ligne pour réduire ce temps de traitement des requêtes. Les optimisations sont d'autant plus appréciables que le but d'une découverte n'est généralement pas de découvrir un service mais l'ensemble des services correspondant à la tâche requise.

Patrons de buts [Stollberg et al. 2007] créent un graphe de relations de subsomption entre les descriptions de services répertoriées et des patrons de buts. La tâche requise, représentée comme une instance de but, est d'abord testée sur ces patrons puis, grâce au graphe de relations, des correspondances peuvent être inférées avec des services sans vérification supplémentaire. De plus, dans le cas d'une correspondance partielle, cette approche connaît les tests supplémentaires à effectuer.

Encodage numérique des relations hiérarchiques [Ben Mokhtar et al. 2008] ont proposé d'optimiser le test de subsomption en encodant les hiérarchies à partir de nombres premiers et en permettant ainsi de réduire ce test à une simple opération modulo. Pour cela, chaque concept a deux valeurs : son gène personnel et le produit de son gène personnel avec ceux de tous ses ancêtres. Tous ces gènes personnels correspondent à un nombre premier et sont uniques. Ainsi un concept est dérivé d'un autre si sa seconde valeur est divisible par le gène personnel de ce dernier concept. Cette technique permet de reléguer l'encodage et la lourde opération de classification des ontologies en mode hors-ligne.

2.2.3.2 Découverte dynamique

Dans des cas à petite échelle, les services peuvent être inscrits manuellement auprès d'un annuaire à une adresse pré-déterminée. Toutefois, pour des systèmes dynamiques comme les dispositifs d'intelligence ambiante ou des systèmes à grande échelle comme le Web, cette hypothèse est impraticable.

Il existe de nombreux standards pour la découverte dynamique dans les réseaux locaux comme *Service Location Protocol* (SLP) [IETF 1999b], *Simple Service Discovery Protocol* (SSDP) d'UPnP, WS-Discovery [OASIS 2009b] ou *DNS-Based Service Discovery* (DNS-SD) [IETF 2013a] couplé à *multicast DNS* (mDNS) [IETF 2013b]. Leurs fonctionnalités communes sont l'annonce de l'arrivée ou de la désactivation d'un service, la découverte à la demande et parfois l'intégration d'annuaires locaux pour optimiser le trafic réseau. [Zhu et al. 2005] ont proposé une taxonomie de plusieurs de ces protocoles.

Sur le Web, les services peuvent être découverts par des robots d'indexation ou tout simplement par le client à l'exécution, à partir d'un lien hypertexte. Les descriptions de services REST sont toutes prévues pour être indexées par un agent externe. Ceci est effectué pour hRESTS et SA-REST en analysant de la page documentation tandis que ReLL et RESTdesc, conformément à la contrainte hypermédia, invitent l'agent à naviguer entre leurs ressources. RESTler [Alarcón & Wilde 2010b] est un robot indexateur utilisant les descriptions ReLL dont il prédispose pour extraire les liens nécessaires à son exploration à partir des représentations des ressources. Les descriptions RESTdesc sont, quant à elles, accessibles au travers d'une requête HTTP `OPTIONS` sur la ressource d'intérêt. La navigation entre ressources s'effectue à l'aide des entêtes HTTP Link.

Nous venons de voir comment la sémantique des services était décrite et découverte. La section suivante montrera comment elle est utilisée pour construire des compositions de services.

2.3 Composition de services

Les requêtes de services de haut-niveau sont généralement suffisamment complexes pour nécessiter l'agencement de plusieurs services : on parle alors de composition de services. Selon que ces compositions soient préconçues ou construites à la demande, la littérature consacrée aux services réalisant des actions se réfère aux processus métiers ou aux techniques de composition automatique. Cette section se propose d'étudier ces deux cas de figure.

2.3.1 Processus métiers

Les services destinés aux entreprises sont généralement conçus pour réaliser une action métier précise décomposable en un ensemble d'actions de niveau inférieur. Ces contextes d'utilisation prennent la forme de compositions appelées processus métiers. Ces compositions de services sont généralement définies comme des workflows.

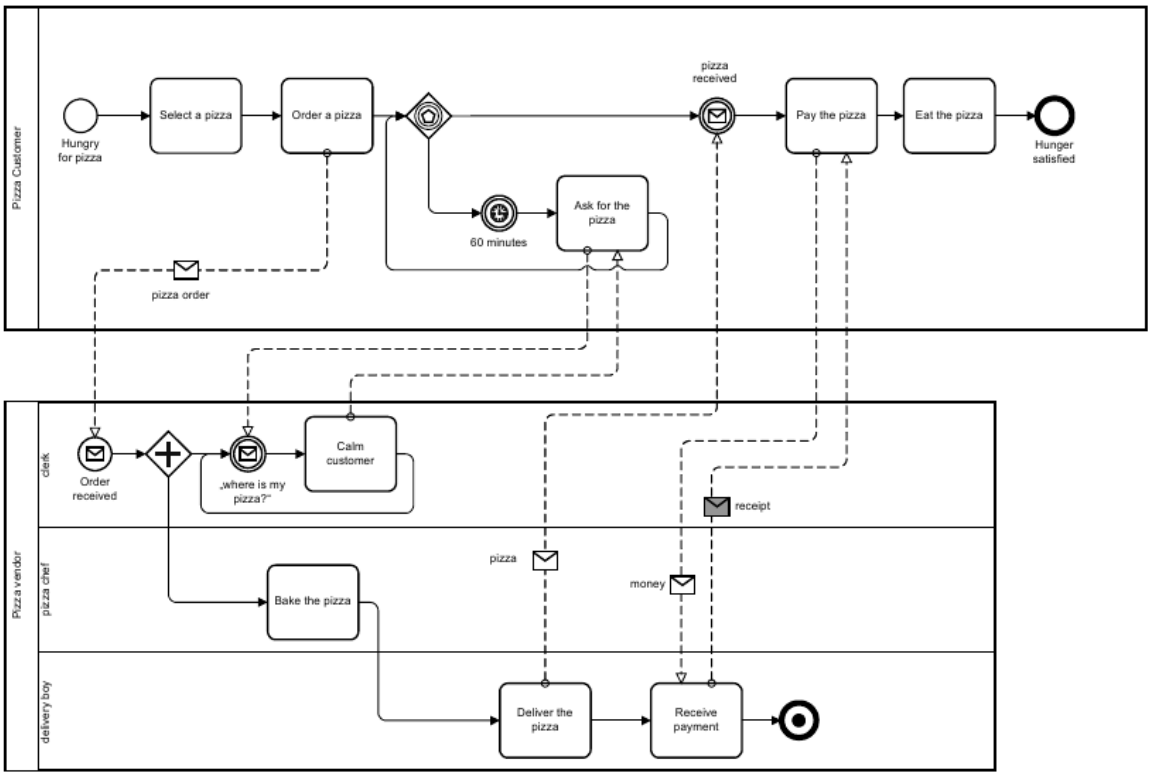


FIGURE 2.4 – Collaboration BPMN 2 autour d'une pizza, extraite de [OMG 2010]

2.3.1.1 Workflow

Un workflow est défini comme «l'automatisation d'un processus métier, en intégralité ou en partie, durant lequel les documents, les informations ou les tâches sont passés d'un participant à un autre pour être traités, selon un ensemble de règles procédurales»[WfMC 1996, Papazoglou 2012].

Les étapes d'un workflow sont réalisées par des services logiciels ou des humains et sont agencées à l'aide de structures de contrôle. *Business Process Management Notation 2* (BPMN 2) [OMG 2011] est le formalisme graphique permettant de spécifier ces workflows. La figure 2.4 présente l'ensemble du processus lié au cycle de vie d'une pizza, de la commande à sa consommation en passant par sa préparation. Ce processus fait intervenir trois participants et décrit leur comportement normal ainsi que leur réaction face à une livraison anormalement trop longue. Lorsque les processus sont fortement automatisables, les humains n'y interviennent généralement que pour gérer les cas exceptionnels, souvent dus à une erreur.

Outre la spécification graphique d'un workflow, la gestion des processus métiers (*Business Process Management*, BPM) s'occupe de son exécution et du recueillement de diverses métriques métiers, appelées indicateurs-clés de performance (*Key Performance Indicator*, KPI) [Papazoglou 2012]. Concernant son exécution, il convient de distinguer deux approches : l'orchestration et la chorégraphie.

2.3.1.2 Orchestration et chorégraphie

Lorsqu'un processus métier fait intervenir plusieurs participants, il peut soit décider qu'un des participants orchestre les interactions en se présentant comme l'intermédiaire responsable du processus ou soit répartir le flot de contrôle entre les participants. Dans le premier cas on parle d'orchestration ; dans le second, il s'agit d'une chorégraphie.

Plus précisément, selon Papazoglou [Papazoglou 2012],

> «une orchestration décrit comment des services peuvent interagir entre eux au niveau des messages, incluant la logique métier et l'ordre d'exécution des interactions depuis la perspective et sous le contrôle d'un participant spécifique. L'intention est de construire un flot de contrôle autour des interactions de ce participant spécifique avec tous ses partenaires. [...]
>
> Une chorégraphie est la description d'un protocole global d'échange de messages qui gouverne la façon dont interagissent deux ou plusieurs participants. Contrairement à une orchestration, une chorégraphie n'implique pas un mécanisme de contrôle centralisé. Au lieu de cela, chaque service impliqué dans la chorégraphie sait exactement quand exécuter ses opérations et avec qui interagir. Il suppose que le contrôle est équitablement partagé entre les participants interagissant, où chaque parti impliqué dans la chorégraphie est parfaitement conscient du processus métier, des opérations à exécuter, des messages à échanger et de leur ordre.»

Chorégraphie Les modèles de chorégraphie se répartissent en deux types [Kopp et al. 2008] : les modèles d'interconnexion (comme les collaborations BPMN 2 ou BPEL4Chor [Decker et al. 2007]) et les modèles d'interaction (WS-CDL [W3C 2006] et les chorégraphies BPMN 2). Un modèle d'interconnexion capture l'ensemble observable du comportement des participants tandis qu'un modèle d'interaction se concentre sur l'interaction globale entre les participants [Kopp et al. 2008].

Une chorégraphie n'est pas directement exécutable, elle doit être convertie en orchestrations locales qui seront exécutées par chacun des participants. L'éventuelle supervision du modèle d'interaction nécessite de disposer d'un intergiciel adapté, comme un ESB ayant conscience de la chorégraphie [Kopp et al. 2008].

Orchestration BPEL *Business Process Execution Language* (BPEL) [OASIS 2007a] est le standard de l'orchestration. Il repose, dans sa version 2.0, sur des descriptions WSDL 1.1 mais une extension a été proposée pour supporter la dimension orientée ressource des services REST-HTTP [Pautasso 2009]. BPEL gère les flots de contrôle, de messages, de données ainsi que les exceptions ; il dispose de variables et peut manipuler les données. Enfin, les sous-services impliqués dans la composition font l'objet de partenariats assignant au sous-service et au service orchestrateur un rôle bien précis défini par un type de port WSDL. Ce partenariat étant abstrait, les sous-services peuvent être assignés tardivement à l'exécution [Papazoglou 2012].

La plupart des concepts BPMN 2 peuvent être convertis en BPEL mais il semble que la tendance soit de développer des moteurs spécifiques comme Bonitasoft [Bonitasoft 2012].

Remarques Dans les chorégraphies et les orchestrations, un service est un intervenant qui converse avec ses partenaires. Ces conversations invitent les services à disposer de plusieurs opérations dont certaines sont dédiées à la réception des résultats de requêtes asynchrones. Les services exposant toutes les opérations d'un rôle peuvent être regroupés en un service abstrait.

2.3.2 Composition automatique

Une des principales perspectives offertes par la description sémantique des services est la construction automatique de compositions de services réalisant des actions. Pour cela, l'intelligence artificielle propose plusieurs approches comme les planifications classique (2.3.2.1) et hiérarchique (2.3.2.2) et l'inférence logique (2.3.2.4). Les compositions peuvent également être construites à l'aide de règles d'assemblage contextuelles (2.3.2.3). Ces techniques centralisées s'apparentent à l'orchestration et non à la chorégraphie.

2.3.2.1 Planification classique

La planification est à la croisée de deux grands sous-domaines de l'intelligence artificielle : l'exploration et la logique. Dans leur version de base, les algorithmes standards de planification classique supposent des informations complètes et correctes, ainsi que des environnements déterministes et observables. Par ailleurs, ils ne quantifient pas les ressources sollicitées comme le temps [Russell & Norvig 2009].

Un problème de planification est composé d'un état initial et d'un but, lui-même défini comme une pré-condition, c'est-à-dire comme un sous-ensemble d'états. Sa résolution produira un plan constitué d'actions qui seront, dans notre cas de figure, délivrées par des services.

Schémas d'actions La planification classique, aussi appelée planification indépendante du domaine, factorise les actions possibles sous forme de schémas d'actions pouvant être décrits dans le langage PDDL [30] [McDermott et al. 1998].

Chaque état est représenté par une conjonction de fluents [31] et d'atomes logiques sans variable ni symbole de fonction. Ainsi cette représentation supporte l'inférence logique et les opérations ensemblistes en étant à la fois conjonction et ensemble de fluents [Russell & Norvig 2009]. Elle fait l'hypothèse de l'unicité des noms et du monde clos (tous les fluents non énoncés sont faux).

Un schéma d'action (voir figure 2.5) est représenté par une conjonction de pré-conditions et par une conjonction d'effets. Ses pré-conditions et effets s'exercent sur des variables quantifiées universellement. Lorsque toutes les variables ont été substituées par des fluents, on parle d'action close (cf. figure 2.5). Face au problème de la persistance, un schéma d'action ne décrit que ce qui change. Ainsi les effets positifs sont de nouveaux fluents et les effets négatifs désignent les fluents devant être supprimés. Ces schémas forment la base de connaissance du domaine.

30. *Planning Domain Definition Language*
31. Un fluent désigne un aspect du monde qui change [Russell & Norvig 2009].

Schéma d'action :

```
Action(
       Déplacer(robot,départ,destination),
       Précondition : À(robot,départ) ∧ RobotMobile(robot)
       Effet :  ¬À(robot,départ) ∧ À(robot,destination)
      )
```

Action close :

```
Action(
       Déplacer(D2R2,couloir,bureau),
       Précondition : À(D2R2,couloir) ∧ RobotMobile(D2R2)
       Effet :  ¬À(D2R2,couloir) ∧ À(D2R2,bureau)
      )
```

FIGURE 2.5 – Schéma d'action et action close de déplacement d'un robot mobile. Notation tirée de [Russell & Norvig 2009]

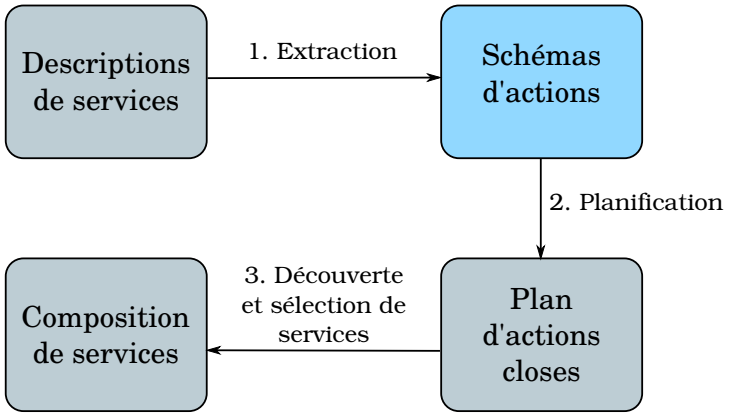


FIGURE 2.6 – *Étapes de la construction d'une composition de services par planification indépendante du domaine.*

Extraction des schémas d'actions à partir des descriptions de services Les schémas d'actions peuvent être extraits des descriptions sémantiques de services lorsqu'elles disposent de pré-conditions et d'effets. Cette extraction doit veiller à factoriser le nombre de schémas d'actions afin qu'ils regroupent les opérations similaires des différents services. Une telle factorisation réduit la taille du graphe de recherche lors de la planification et permet également d'obtenir des actions closes réalisables par plusieurs services candidats. La phase de planification est alors suivie d'une phase de découverte et de sélection de services (cf. figure 2.6). L'extraction de schémas d'actions à partir de descriptions OWL-S a été traitée par [Sirin et al. 2004].

Exploration dans l'espace d'états Il existe diverses méthodes pour résoudre un problème de planification classique : l'exploration dans un espace d'états, l'exploitation directe d'un graphe de planification (algorithme Graphplan), la déduction en premier ordre sur le calcul situationnel, l'encodage comme un problème de satisfiabilité ou un problème de satisfaction de contraintes et enfin l'exploration de l'ensemble des plans

partiellement ordonnés [Russell & Norvig 2009]. Nous n'évoquerons ici que l'exploration dans l'espace d'états.

La planification est avant tout un exercice de contrôle de l'explosion combinatoire. L'objectif n'est pas d'énumérer tous les plans valides mais d'en trouver un petit nombre jugés quasi-optimaux. L'exploration dans l'espace d'états doit être informée, c'est-à-dire qu'elle doit utiliser des heuristiques. Cette exploration peut être effectuée en avant ou en arrière : on parle respectivement de progression et de régression. La progression s'intéresse aux actions applicables dans un état donné, tandis que la régression recherche des actions pertinentes, qui pourraient être le dernier pas vers le but courant. La régression offre un facteur de branchement plus faible que la recherche en avant ; toutefois le fait d'utiliser des ensembles d'états plutôt que des états seulement complique l'élaboration d'heuristiques. C'est pourquoi la recherche avant est majoritairement préférée depuis la fin des années 1990 grâce à de puissantes heuristiques indépendantes du domaine [Russell & Norvig 2009].

Dans de nombreux cas, le calcul d'heuristiques consiste à résoudre une forme relaxée du problème de la planification en effaçant par exemple certaines ou toutes les préconditions ou en omettant les listes de suppression. D'autres heuristiques souvent plus performantes s'appuient sur un graphe de planification dont la taille est polynomiale. À leur façon, ces heuristiques visent à décomposer le problème en sous-problèmes indépendants qui dans le meilleur des cas mènerait à une réduction de la complexité de façon exponentielle. Cependant cette décomposition est entravée par les interactions négatives entre actions mais comme ces méthodes sont heuristiques, elles ne requièrent pas que les sous-problèmes soient complètement indépendants [Russell & Norvig 2009].

Extensions Les hypothèses d'informations complètes et correctes, de déterminisme et d'observabilité ne sont bien entendu pas toutes soutenables dans de nombreux problèmes de composition de services. Divers travaux ont tenté de les relever. En comparaison à BPEL, [Srivastava & Koehler 2003] ont soulevé plusieurs défis pour les plans comme la gestion du non-déterminisme et de structures de contrôle complexes comme les boucles, les sauts conditionnels et les exécutions parallèles. Les sauts conditionnels [Peer 2004, Pistore et al. 2005] permettent de gérer un nombre limité de résultats non-déterministes [Kaldeli et al. 2011].

[Au et al. 2005] ont proposé une extension permettant aux informations fournies par les services d'être volatiles, c'est-à-dire de pouvoir évoluer au cours de l'exécution du plan. [Kaldeli et al. 2011] ont présenté une approche de planification continue permettant de fixer des contraintes non-fonctionnelles sur des variables (comme la distance d'un concert) dont la valeur ne sera obtenue qu'en cours d'exécution. En cas de violation, une adaptation du plan est effectuée.

2.3.2.2 Planification hiérarchique

À la différence de la planification classique, la planification hiérarchique utilise plusieurs niveaux d'abstraction pour construire son plan et peut envisager des plans de plus grande taille [Russell & Norvig 2009]. Chaque niveau dispose de son propre sous-domaine et peut éventuellement utiliser un planificateur spécifique. Les réseaux hiérarchisés de tâches (HTN) ne peuvent pas se contenter des schémas d'actions PDDL mais nécessitent des schémas supplémentaires appelés méthodes, ou affinements, indiquant comment dé-

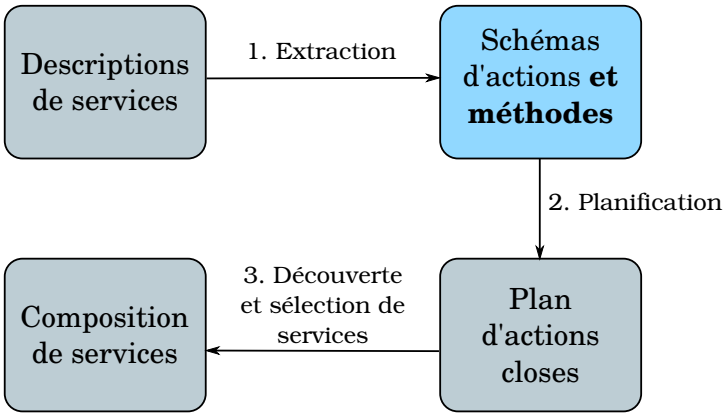


FIGURE 2.7 – *Étapes de la construction d'une composition de services par planification hiérarchique.* Ce processus se distingue de celui de la figure 2.6 par l'extraction et l'utilisation de méthodes HTN. Ces méthodes peuvent être également obtenues par d'autres moyens que les descriptions de services.

composer une tâche en sous-tâches. Les actions PDDL correspondent ici à des tâches abstraites ou à des actions primitives. Les méthodes peuvent avoir une structure arbitrairement complexe et possèdent un pouvoir d'expression à mi-chemin entre la programmation et la planification classique [Ghallab 2004]. Ces méthodes peuvent représenter des plans partiels allant de plans d'actions sous forme de workflows aux compositions de type configuration [32].

Un plan est résolu de façon récursive jusqu'aux actions primitives. Certaines tâches peuvent rester abstraites jusqu'au moment de leur exécution si l'incertitude engendrée reste acceptable [Russell & Norvig 2009].

Application aux services réalisant des actions En plus des schémas d'actions, les méthodes peuvent elles aussi être extraites des descriptions de services (cf. figure 2.7), comme l'ont proposé [Sirin et al. 2004] pour OWL-S. Ces méthodes ont une structure de workflow pouvant être complexe. Au dire de ces auteurs, «la génération de service composite doit davantage être vue comme la spécialisation de processus composites prédécrits que comme l'écriture d'un programme complexe nouveau» [Sirin et al. 2004]. Ces schémas ont ensuite été transmis au planificateur SHOP2 [Nau et al. 2003].

Notons que si certains services prennent en charge la réalisation d'une tâche et se comportent comme des méthodes en sollicitant eux-mêmes des sous-services, il s'agit alors d'une hiérarchie d'orchestrations de workflows prédéfinis [33]. Cette pratique donne lieu à une série de sélections de services à différents niveaux.

2.3.2.3 Règles d'assemblage contextuelles

Un service composite n'est pas forcément uniquement composé de sous-services : il peut également faire appel à des composants logiciels. Le modèle *Service Lightweight Component Architecture* (SLCA) [Hourdin et al. 2008, Tigli et al. 2009] permet de

32. Les compositions de services de type configuration feront l'objet du chapitre suivant.
33. La hiérarchie est permise en BPEL par les services abstraits représentant les partenaires.

construire de tels services composites sous forme d'assemblages de composants où certains composants sont les mandataires *(proxy)* de sous-services. Cet assemblage peut ensuite être tissé automatiquement à l'aide d'aspects d'assemblage [Tigli et al. 2009]. Ces aspects d'assemblage peuvent être obtenus à partir de règles ORCA *(lOw Response time and dynAmic Context Adaptation rule system)* [Tigli et al. 2012] dépendantes du contexte. La gestion d'éventuels conflits entre les règles est déterminée par une politique de composition.

En réaction à un changement de contexte important, le tissage peut être adapté en remplaçant les aspects d'assemblage invalidés et les composants indisponibles tout en exploitant les éventuelles nouvelles opportunités. En se basant sur des composants logiciels et en introduisant des mécanismes de communication événementielle, SLCA peut faire appel à des sous-services réalisant des actions tout comme permettre la réalisation de configurations. Ces modèles ont donné lieu à la réalisation de l'intergiciel WComp [Tigli et al. 2009].

2.3.2.4 Déduction logique

Il est également possible de construire des compositions de moindre complexité à partir de règles logiques. [Verborgh et al. 2011] ont proposé de convertir en règles N3Logic des descriptions OWL-S dont les pré-conditions et les effets sont exprimés en N3.

N3Logic a pour principal objet de gérer de nouveaux triplets RDF. Toutefois, comme vu dans le paragraphe 2.2.1.2, une règle peut également déclencher une action. En revanche, la forme originale de N3Logic se conforme aux hypothèses du Web sémantique, à savoir celles de monde ouvert, de non-unicité des noms et de monotonicité [Berners-Lee et al. 2008]. Toutefois, les moteurs de raisonnement N3Logic rajoutent des prédicats internes pour fournir quelques fonctionnalités du monde fermé [Vanel 2010]. Parmi celles-ci, on trouve des opérations non-monotones comme le retrait de prédicats.

Ces moteurs de raisonnement peuvent fournir l'ensemble des solutions possibles mais il convient pour cela que cet ensemble soit fortement limité en taille. Cela donne lieu à la sélection de la meilleure combinaison de services.

2.4 Sélection de services

Les sections précédentes ont montré que la découverte et la composition de services s'intéressent avant tout aux aspects fonctionnels d'une tâche requise. Elles gèrent aussi parfois des propriétés non-fonctionnelles leur permettant de rejeter certaines propositions comme des sous-plans ou des services. Comme nous l'avons souligné à plusieurs reprises, leurs résultats fournissent régulièrement plusieurs alternatives qui doivent être arbitrées sur des critères non-fonctionnels ; cette problématique est celle de la sélection de services.

L'atomicité d'une sélection de services correspond soit à :

1. *Une tâche isolée.*
2. *Un ensemble de tâches devant être réalisé par le même fournisseur de services.* Ce cas de figure apparaît dans les workflows où les partenaires conversent mais aussi en planification classique et en déduction logique où une variable se référant à un

fournisseur de services [34] ou sa substitution apparaissent dans une série d'actions. Cet ensemble de tâches concerne parfois une même instance de service ; cette pratique est courante dans les processus métiers.

Afin que ce choix puisse se réduire à celui d'un service [35], il est nécessaire *qu'un service n'offre qu'une seule opération pour réaliser une tâche donnée*. Les services REST-HTTP peuvent être identifiés par un préfixe d'URI, ce qui permet par exemple de diviser en plusieurs services un ensemble de robots mobiles homogènes rendus accessibles au travers d'une même passerelle. De même, lorsque des services WS-* fournissent plusieurs extrémités communicantes *(endpoints)* fonctionnellement équivalentes ayant des niveaux de qualité ou des protocoles différents, celles-ci sont considérées comme des services distincts par le module de sélection de services. À l'issue de la sélection de services, la description de la tâche requise est traduite en une requête de service (cf. sous-section 2.1.1).

Dans cette section, nous allons tout d'abord présenter les divers critères non-fonctionnels de sélection présents dans la littérature. Ensuite, nous nous focaliserons sur le critère de performance et nous verrons comment les services sont évalués. Enfin nous examinerons la complexité et les hypothèses sous-jacentes aux approches de sélection locale et globale.

2.4.1 Critères non-fonctionnels

Les méthodes de sélection de services existantes distinguent principalement trois types de critères non-fonctionnels : la performance, la réputation et la confiance. La performance est dominante dans la littérature et sera le critère exclusif de nos travaux. Bien que ces critères peuvent être combinés de diverses manières, nous accorderons dans cette section un rôle secondaire à la réputation et à la confiance et nous verrons comment ils peuvent s'insérer dans la perspective d'une sélection basée sur la performance. Enfin, il est à noter que certaines sélections ne s'effectuent pas sur ces critères : elles peuvent par exemple évaluer les préférences entre plusieurs services dans un contexte donné en observant le comportement d'un utilisateur [Hossain et al. 2008, Zaidenberg & Reignier 2011].

Performance Le critère de performance se base sur des mesures objectives du comportement d'un service. Ces métriques sont ensuite normalisées et deviennent des paramètres de Qualité de Service (QdS). Ces paramètres de QdS sont généralement multiples, ce qui permet d'évaluer un service selon plusieurs dimensions de son comportement.

Les paramètres de QdS les plus courants sont le temps de réponse, le débit, la fiabilité, la disponibilité, la capacité à passer à l'échelle *(scalability)* et le coût [W3C 2003, Maximilien & Singh 2004, Ben Mabrouk et al. 2009]. Ces quelques paramètres s'inscrivent principalement dans une perspective d'administration système et réseaux mais d'autres paramètres de QdS spécifiques à une application peuvent également être introduits [Maximilien & Singh 2005]. Par exemple, l'utilisateur pourrait être intéressé par la distance du relais auquel un colis serait livré.

34. Une variable se référant à un service entre aussi dans ce cadre car elle désigne indirectement un fournisseur de services. Nous plaçons le critère au niveau du fournisseur car celui-ci peut fournir plusieurs services liés. Par exemple un robot mobile peut fournir un service de déplacement ainsi qu'un service de caméra. Si le plan consiste à déplacer le robot pour que la caméra puisse accéder au champ de vision souhaité, alors ces deux services sont liés par leur fournisseur.

35. ou, plus rarement, d'une instance de service

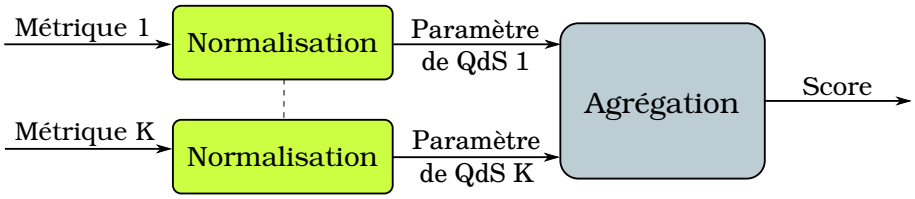


FIGURE 2.8 – Évaluation d'un service

Réputation et confiance Les critères de réputation et de confiance interviennent lorsqu'on souhaite utiliser des valeurs émanant d'entités tierces comme les fournisseurs de services et leurs précédents utilisateurs. Diverses définitions contradictoires de ces critères existent [Maximilien & Singh 2005, McNamara et al. 2006, Wang & Vassileva 2007]. Dans notre perspective centrée sur la performance, nous dirons que la réputation s'intéresse aux valeurs des paramètres de QdS estimées par les précédents utilisateurs et tient compte de leur fraîcheur [Maximilien & Singh 2005, Dehousse et al. 2009] tandis que la confiance vise à quantifier la crédibilité de ces informations [McNamara et al. 2006].

2.4.2 Évaluation d'un service

Le processus d'évaluation d'un service selon le critère de performance consiste généralement en deux étapes : l'obtention de paramètres de QdS et leur agrégation en un score (voir figure 2.8). Ce score permettra ensuite à la sélection de choisir l'alternative maximisant le score.

2.4.2.1 Obtention des paramètres de QdS

Pour simplifier notre exposé, nous supposons dans cette sous-sous-section que les compositions n'ont qu'un seul niveau. Suivant que le traitement effectué par le service évalué soit atomique ou composite, la sélection est respectivement dite locale ou globale.

Niveau local La sélection d'un service atomique ayant lieu avant son exécution, les paramètres de QdS associés à sa réalisation de la tâche sont soit prédits, soit supposés connus à l'avance ; il peut s'agir d'une distribution de probabilité [Hwang et al. 2007, Rosario et al. 2009], d'une valeur simple ou d'un accord de niveau de performance [36] [Rosario et al. 2009]. Les valeurs les plus vraisemblables de ces paramètres de QdS peuvent être :

1. *Publiées dans les descriptions de services.* Une telle approche ne peut convenir qu'à un nombre limité de métriques et de services.

36. Un *Service Level Agreement* (SLA) est un accord contractuel garantissant un niveau de performance selon plusieurs paramètres de QdS [Papazoglou 2012]. Ce contrat prévoit des pénalités de compensation en cas de violation. Rosario et al. [Rosario et al. 2009] utilisent directement la valeur de cette contrainte comme estimation de la performance des traitements effectués par les sous-services et supervisent le respect de cette contrainte durant l'exécution. De plus, il convient de s'assurer qu'une performance supérieure au niveau de garantie n'ait pas un effet négatif sur le reste de la composition. Ce phénomène de non-monotonie peut être évité en prenant quelques précautions lors de la modélisation d'une orchestration [Bouillard et al. 2009].

Métriques	Structure de contrôle			
	Séquence	ET	OU exclusif	Boucle
Temps de réponse *(RT)*	$\sum_{i=1}^{n} rt_i$	$max(rt_i)$	$max(rt_i)$	$rt * k$
Fiabilité *(RE)*	$\prod_{i=1}^{n} re_i$	$\prod_{i=1}^{n} re_i$	$min(re_i)$	re^k
Disponibilité *(AV)*	$\prod_{i=1}^{n} av_i$	$\prod_{i=1}^{n} av_i$	$min(re_i)$	av^k
Débit *(TH)*	$min(th_i)$	$min(th_i)$	$min(th_i)$	$min(th_i)$
Prix *(P)*	$\sum_{i=1}^{n} p_i$	$\sum_{i=1}^{n} p_i$	$max(p_i)$	$rt * k$

TABLE 2.1 – Règles d'agrégation proposées par [Ben Mabrouk et al. 2009]

2. *Estimées à partir de leur réputation.* Les valeurs communiquées par d'autres utilisateurs sont alors pondérées selon leur fraîcheur et la confiance accordée à l'utilisateur puis sont agrégées.
3. *Inférées à l'aide d'opérations logiques et d'algorithmes déterministes.* Par exemple, la distance du relais de livraison peut être inférée à partir du réseau de relais partenaire du service.
4. *Estimées ou prédites statistiquement.* Nous mettrons l'accent sur ce point dans le chapitre suivant et nous présenterons plusieurs cas motivant une prédiction statistique dépendant du contexte.

Niveau global Face à un service composite, l'approche la plus fréquente consiste à calculer la valeur des métriques à partir de celles associées aux traitements effectués par ses sous-services [37]. Cardoso et al. [Cardoso et al. 2004] ont proposé des règles algébriques pour des paramètres de QdS comme la durée, le temps de réponse et le coût adapté aux structures de contrôle présentes dans les workflows. Cette algèbre a été reprise et complétée [Zeng et al. 2004, Yu et al. 2007, Ben Mabrouk et al. 2009] (voir le tableau 2.1). La plupart des propositions recherchent une expression analytique des paramètres de QdS en supposant généralement un comportement déterministe de la composition [Zeng et al. 2004, Canfora et al. 2005, Yu et al. 2007, Vanrompay et al. 2008, Ben Mabrouk et al. 2009]. Tan et al. [Tan et al. 2003] ont proposé d'utiliser une autre approche analytique basée sur les réseaux de Petri stochastiques pour prédire le débit d'une composition en prenant en compte le caractère stochastique des performances des sous-services. Rosario [Rosario 2009] indique que, dans ce cadre stochastique, une approche analytique est restreinte, pour des questions combinatoires, à des orchestrations sans saut conditionnel dépendant de données ou de dates d'expiration. Pour lever cette limitation, l'auteur propose de réaliser des simulations de type Monte-Carlo.

Ce type d'approche s'appuie sur les performances individuelles des instances de sous-service. Certaines métriques ne concernent qu'un nombre limité de ces instances [38]. Ce type d'approche effectue l'hypothèse suivante : *toutes les instances de sous-service prises isolément ont un comportement prédictible pour chacune des métriques les concernant.* Or dans le cas d'une coordination étroite (chapitre 3), la performance d'une instance de sous-service peut être dépendante de celle du service composite car celle-ci n'a par exemple pas toujours le contrôle sur sa durée d'exécution. Aussi certaines métriques

37. Les valeurs de ces métriques doivent rester sous leur forme physique et non sous leur forme normalisée pour pouvoir être agrégées en une métrique de plus haut niveau.

38. Par exemple la distance du relais ne concerne que l'instance de livraison.

d'un service composite ne sont pas décomposables et doivent être prédites au niveau composite à partir de l'ensemble des instances de sous-service impliquées, comme nous le verrons dans la sous-section 3.2.2.

2.4.2.2 Agrégation des paramètres de QdS en un score

La fonction d'agrégation établit un compromis entre les paramètres de QdS. Ce compromis est généralement constant : la fonction consiste alors en la somme pondérée de ces valeurs. Toutefois, ce compromis peut être rendu dépendant des valeurs ; la fonction est alors non-linéaire. Les détails sur l'origine des poids sont souvent parcellaires : elles sont généralement présumées avoir été fournies par l'utilisateur sous une forme explicite [Canfora et al. 2005, Alrifai et al. 2008, Ben Mabrouk et al. 2009] ou par le biais d'un processus analytique hiérarchique [39] [Dehousse et al. 2009]. Ces poids sont généralement constants mais sont parfois rendus dépendants du contexte [Tari et al. 2010]. Lorsque la sélection d'un service implique plusieurs tâches, le score retenu est la moyenne du score associé à chacun de la réalisation de chacune d'entre elles.

2.4.3 Complexité

Examinons maintenant la complexité en nombre d'évaluations des sélections locales et globales.

Sélection locale La sélection locale a une complexité linéaire puisqu'elle ne s'intéresse qu'à une seule tâche de bas-niveau à la fois. Toutefois, cette approche suppose la disponibilité d'une fonction d'agrégation locale en lieu et place d'une fonction d'agrégation globale sur le service de plus haut niveau. En effet, cette fonction ne peut pas être la même à ces deux niveaux car cela supposerait que la sélection globale puisse se réduire à une suite de sélections locales. S'il choisit une approche de sélection locale, l'utilisateur doit fournir une fonction d'agrégation pour chaque type de sous-services possible sans garantie sur leur cohérence au niveau de la composition. Il semble donc plus intéressant de spécifier une fonction d'agrégation pour le service de plus haut niveau, d'autant plus qu'il peut y avoir plusieurs niveaux de composition.

Sélection globale La sélection globale semble donc plus appropriée pour sélectionner les sous-services d'une composition. Toutefois, en tant que problème d'optimisation multi-critères, elle est exposée à une complexité NP-complète en nombre d'évaluations [Yu et al. 2007, Ben Mabrouk et al. 2009]. Dans le cas de la coordination faible, les méthodes de résolution exacte comme l'optimisation linéaire en nombres entiers [Zeng et al. 2004, Yu et al. 2007] peuvent être utilisées pour les problèmes de petite taille mais le passage à l'échelle exige l'usage de méthodes approximatives comme l'exploration informée à base d'heuristiques [Ben Mabrouk et al. 2009] et les algorithmes génétiques [Canfora et al. 2005, Vanrompay et al. 2008].

39. Cette technique permet le calcul de poids à partir de préférences exprimées entre chaque paire de métriques [Saaty 1980].

2.5 Conclusion

Ce chapitre d'état de l'art a effectué une présentation d'ensemble des principaux enjeux, problématiques et technologies des architectures orientées service. Il s'est concentré sur le cas majoritaire des services modélisés comme réalisant des actions ; le chapitre suivant lèvera cette restriction de portée en intégrant un autre type de service permettant la mise en place d'une coordination étroite au sein de compositions de services.

Après avoir introduit la problématique de la sélection de services dans ce chapitre et l'avoir positionnée par rapport à celles de la composition et de la découverte de services, nous allons également revenir en détail dans le prochain chapitre sur les étapes de prédiction de performance et d'évaluation des exécutions.

Chapitre 3

Sélection contextuelle de services robotiques : enjeux et architecture

Sommaire

Dans le cadre de cette thèse, la sélection de services est destinée en premier lieu aux services robotiques. Elle a pour critère non-fonctionnel la performance et nécessite la prédiction de cette dernière. Une telle prédiction peut s'avérer être une étape difficile nécessitant dans certains cas l'apprentissage de fonctions contextuelles.

Pour cette étape puisse être valorisée, il est primordial de disposer d'alternatives offrant des performances différentes pour une même tâche. Or, comme nous avons pu le constater dans la sous-section 2.1.5, les compositions de services de type action ont un intérêt limité en robotique si bien que la plupart des services fournissant des fonctionnalités spécifiquement robotiques sont atomiques. De plus, ces services sont souvent seuls à pouvoir réaliser une tâche donnée. Faute de choix, une sélection exploitant des services issus des approches existantes ne peut donc pas valoriser les considérations non-fonctionnelles. Paradoxalement, les enjeux portés par ces considérations sont importants car ces services peuvent être coûteux, bruyants et sont soumis à de nombreuses sources de perturbation physiques.

Pour remédier à cette limitation, nous proposons de considérer un autre type de service : les services continus. À la différence des actions, les services continus ne s'intéressent pas à l'état final mais au processus engendré par le traitement de la requête.

Certains de ces services fournissent des flots d'information qui, comme nous l'avons vu dans la partie 1.2.2.2, forment le mécanisme de base nécessaire à la coordination étroite. En modélisant certaines fonctionnalités robotiques comme des services continus, il devient possible de créer des compositions de services dédiées à la robotique en lieu et place de services atomiques. Ainsi, de nouvelles alternatives sont mises à disposition de la sélection de services ; celle-ci est alors en mesure de valoriser la fonction objectif assignée par l'utilisateur ou son représentant.

La plus forte coordination permise par ces compositions de services continus a cependant un sérieux impact sur la prédiction de performance car les performances des instances de leurs sous-services ne sont plus indépendantes mais couplées. La prise en charge de ce couplage non-fonctionnel nécessite des changements procéduraux significatifs pour la sélection de services.

Plan du chapitre L'objectif principal de ce chapitre est de dégager le problème de sélection de services qui sera ensuite formalisé et résolu dans le chapitre suivant. Il est organisé de la façon suivante.

La première section (3.1) est consacrée à l'étude des services continus. Elle décrit leurs aspects fonctionnels (3.1.1) et montre comment les fonctionnalités robotiques peuvent être composées avec des services de ce type tout en garantissant une coordination étroite (3.1.2). Enfin, divers exemples de services continus, robotiques ou multimédia, sont présentés (3.1.3).

Dans le cadre d'une sélection basée sur la performance, il est fréquent que la performance doive être prédite. La section 3.2 relate les approches présentes dans la littérature (3.2.1) puis montre qu'une nouvelle approche est nécessaire pour répondre aux enjeux soulevés par la robotique ambiante (3.2.2).

La section 3.3 s'intéresse aux aspects architecturaux nécessaires à la mise en œuvre des solutions venant d'être décrite. Tout d'abord, elle expose les exigences architecturales imposés par notre problématique de sélection de services puis effectue des propositions afin d'y répondre (3.3.1). Enfin, elle détaille le processus d'évaluation de l'exécution d'un service de haut-niveau (3.3.2).

Ce chapitre se clôture en effectuant un bilan de nos contributions.

3.1 Services continus

Le chapitre précédent s'est concentré sur les services dont les traitements de requêtes étaient formalisables comme des actions. Dans une telle approche, la partie fonctionnelle d'une opération s'intéresse à l'état final et non au processus qui y a mené. Le processus est relégué au plan non-fonctionnel où il peut être considéré pour sa performance.

Cette section distingue une autre catégorie de services où l'intérêt fonctionnel porte sur le processus et non sur son état final. Ces services sont qualifiés de *services continus*. Cette section est consacrée à l'étude de cette distinction qui semble toutefois méconnue dans la littérature consacrée aux services logiciels. Dans un premier temps, elle en présente les aspects fonctionnels (sous-section 3.1.1). En s'appuyant sur les configurations de composants destinées à la coordination étroite, elle met ensuite en évidence la nature hiérarchique des compositions de services résultantes (3.1.2). Enfin, elle donne quelques

exemples représentatifs de ce type de services dont les services robotiques qui seront employés dans nos scénarios (sous-section 3.1.3).

3.1.1 Description fonctionnelle

Les descriptions des services continus partagent de nombreux points communs avec celles de services réalisant des actions. Cette sous-section présente la nature des informations dont doit disposer un client pour pouvoir générer une requête et traiter les retours d'un service continu.

Génération de requêtes de services Pour pouvoir générer une requête, le client doit disposer d'informations sur le service ainsi que sur l'objectif du traitement souhaité. Ces informations sont les suivantes :

1. *Catégories des opérations du service.* Les services continus peuvent bénéficier du raisonnement sur les catégories d'opérations au même titre que les services de type action lors du test de correspondance (cf. sous-section 2.2.2).
2. *Restrictions de portée.* Tout comme les services de type action, les services continus peuvent exprimer les restrictions de portée de leurs opérations sous forme de pré-conditions.
3. *Cible de la requête.* Toute requête de service doit définir la cible du traitement souhaité de façon complète. Comme pour tout type de service, ces informations représentent les entrées de l'opération de service et peuvent être formalisées comme des fluents associés à certaines pré-conditions.
4. *Durée.* Il est fréquent que les instances des services continus aient une durée d'exécution longue qui est parfois même indéterminée. En effet, il est souvent intéressant pour un client de ne pas spécifier de durée fixe ni de condition d'arrêt normal pour un service continu. Il doit cependant alors prendre en charge son arrêt lorsque ce service n'est plus nécessaire [1]. Lorsque la condition d'arrêt est déterminée, elle doit être transmise dans la requête de service.
5. *Démarrage.* Le traitement de la requête peut démarrer immédiatement tout comme être mis en attente après l'allocation des ressources.

Traitement des retours Les informations fournies à l'exécution par une instance de service continu sont principalement relatives :

1. *À l'état de l'exécution.* Le client doit être informé de l'état de l'instance de service : celle-ci est généralement soit en cours d'exécution, soit échouée ou s'est terminée de façon normale. Dans certains cas de figure, elle peut être en attente de démarrage lorsque celui-ci est dissocié de son allocation et elle peut également parfois être mise en pause.
2. *Aux flots d'information.* La finalité de nombreux services continus est de fournir un ou plusieurs flots d'information. La description de ces services doit spécifier la structure des événements qui composent ces flots. Cette structure est généralement similaire à celle des résultats renvoyés par un service de type action. L'obtention de ces événements s'effectue via un mécanisme de souscription aux

1. Des mécanismes d'expiration peuvent être mis en place pour assurer la robustesse du système face aux possibles manquements du client.

événements ou par des techniques de scrutation longue *(long polling)* ou active. L'adresse de souscription ou de scrutation est en règle générale obtenue lors de l'allocation de l'instance de service.

Remarque sur les instances de services Comme nous l'avions souligné dans la sous-section 2.1.1, la notion d'instance de service est plus fréquemment sollicitée dans le cas des services continus que dans celui des services de type action. Ceci est une conséquence de l'importance que les services continus accordent à leurs processus et de la possibilité d'effectuer des interactions auxiliaires durant les traitements de requêtes. Dès lors, il devient nécessaire de réifier ces instances en les rendant adressables [2]. Des opérations supplémentaires peuvent apparaître et disparaître avec des instances de services. Cette réification est également utile pour des services de type action comme le déplacement d'un robot afin de permettre à des services indépendants de mesurer leurs performances (cf. sous-section 3.3.2).

3.1.2 Compositions de services continus

Les décompositions de tâches de type configuration permettent une coordination étroite entre les processus des sous-tâches. Cette coordination exige, en général, l'exécution simultanée de ces processus et l'échange de flots d'information entre ceux-ci. Dans les approches à base de composants évoquées dans la sous-sous-section 1.2.2.2, chaque sous-tâche atomique est représentée par un composant logiciel [Nahrstedt & Balke 2004, Kim et al. 2006, Gritti 2008, Lundh et al. 2008]. La configuration résultante consiste en un plan horizontal [3] où les composants sont reliés entre eux par des canaux de communication, et ce même lorsque celui-ci est construit à l'aide de techniques de planification hiérarchique [Lundh et al. 2008].

La modélisation équivalente aux configurations de composants à partir de services logiciels est appelée *composition de services continus* [4]. Les services continus sont en effet capables de fournir les flots d'information dont ont besoin les sous-tâches pour se coordonner. Similairement à [Lundh et al. 2008], le service produit par cette composition peut parfois être modélisé comme une action. Le service de déplacement d'un robot vers une destination en est un exemple.

Structure hiérarchique À la différence des configurations de composants, ce type de composition de services a une structure *hiérarchique* pour les raisons suivantes. De façon générale, une décomposition en services implique une décomposition en requêtes pouvant être traitées de façon autonome sur le plan fonctionnel par un ou plusieurs fournisseurs de services. Conformément au principe de séparation entre l'interface et l'implémentation d'un service (sous-section 2.1.1), la gestion des dépendances fonctionnelles est à la charge exclusive du fournisseur de services. Ces dépendances se traduisent par des requêtes de sous-services. Or la souscription à un flot d'information généré par un service est une

2. Une instance de service est identifiée par une URI dans le cas d'une approche REST-HTTP ou est désignée par un `resourceID` via l'extension WS-Addressing dans le cas d'un service SOAP.

3. à un seul niveau de hiérarchie

4. Nous réservons le terme *configuration* aux assemblages de composants car il témoigne d'une logique de prise en charge des composants par une autorité supérieure, la plateforme, se chargeant ici de mettre en place des canaux entre les composants. Les services sont au contraire auto-contenus et ne sont donc pas soumis à une telle autorité (cf. sous-section 2.1.1).

dépendance fonctionnelle car l'ensemble d'information véhiculé par ce flot ne peut pas être contenu à l'avance dans la requête de service. En conséquence, l'instance de service souscripteur doit donc faire appel à un sous-service (sous-figure 3.1b) ; deux cas de figure peuvent alors se présenter :

1. *Le service générant les événements est déjà instancié et a été requis par une autre entité.* L'instance de service souscripteur se contente alors de faire appel à un simple service de diffusion des événements générés.
2. *L'instance de service souscripteur est à l'initiative de la requête de service générant les événements.* Il est alors le responsable hiérarchique de cette instance de sous-service. À cette occasion, en tant que cliente, elle peut avoir le contrôle du cycle de vie de l'instance de sous-service (démarrage synchronisé, arrêt prématuré voire mise en pause).

De part cette structure hiérarchique, le rassemblement des instances de services impliquées dans l'exécution d'une instance de service de haut-niveau forme un *organigramme* [5].

Modélisation des sous-tâches Les sous-tâches atomiques n'ayant pas de dépendance fonctionnelle peuvent donc être modélisées comme des services atomiques. Une sous-tâche atomique ayant une dépendance fonctionnelle doit être englobée dans un service composite où elle pourra avoir le rôle de composant logiciel [6] (sous-figure 3.1a).

Notons que des flots d'information bidirectionnels entre un service composite et son sous-service peuvent être mis en œuvre via un sous-sous-service (voir la sous-figure 3.1c) [7]. Il convient toutefois d'en faire un usage modéré en veillant à ne pas utiliser ces flux d'information bidirectionnels pour ré-implémenter un nouveau protocole de type requête-réponse. Une telle pratique créerait en effet un couplage fonctionnel conséquent [8].

Liens avec la planification hiérarchique Les compositions de services continus peuvent être vues comme une variante de la planification hiérarchique dès lors que les sous-services sont sélectionnés dans une perspective si possible globale. En effet, ce parallèle invitera à intéresser la sélection des sous-services d'une instance de service composite comme un affinement (sous-sous-section 3.2.2.3). Pour que la sélection soit globale, le sélectionneur doit pouvoir reconstruire l'organigramme ; pour cela, nous opterons pour une centralisation des sélections auprès d'un même sélectionneur (recommandation RAF1), dont la ressemblance avec un planificateur hiérarchique central ne sera pas fortuite.

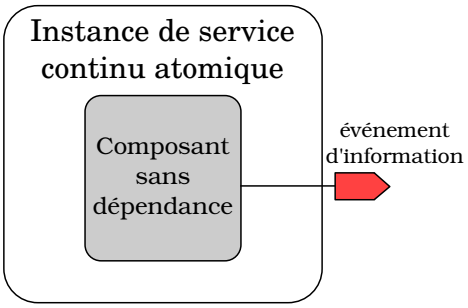
Ces compositions de services continus ont pour particularité de devoir garder leur structure hiérarchique de part la présence de composants ayant une dépendance envers des flots d'informations et ne pouvant pas par conséquent être représentés en dehors de services composites. Cette contrainte est une différence avec les plans d'actions ou les

5. Toute instance de service étant liée à un service, il est également possible de dégager un ensemble de services d'un organigramme.

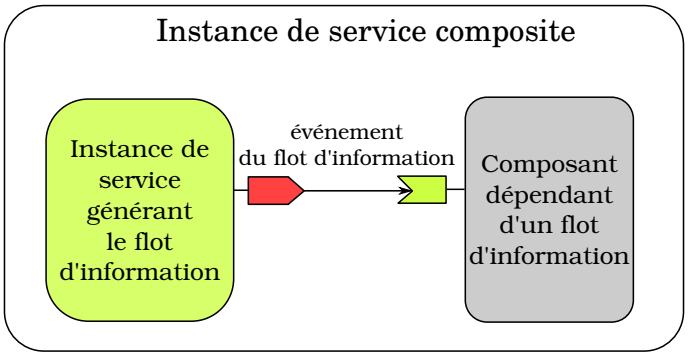
6. L'usage de composants logiciels dans nos représentations de compositions de services continus permet de réifier des traitements et d'expliciter leurs dépendances fonctionnelles en matière de flot d'information. Il se peut cependant qu'en pratique ces traitements ne soient pas réifiés dans l'implémentation du service composite ; cette abstraction logicielle permet néanmoins de faciliter notre exposé.

7. L'adresse du flot généré par le service composite est transmis dans la requête de sous-service.

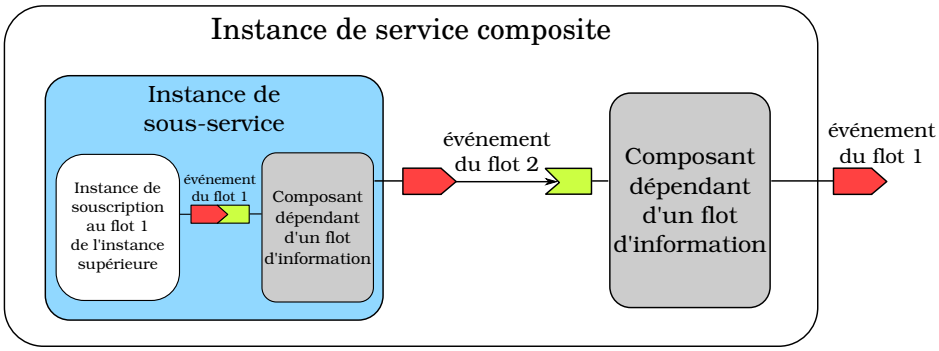
8. Le mécanisme de négociation de rôles et d'échange d'informations entre les deux robots transportant un objet rigide, évoqué dans la sous-sous-section 1.2.2.2 et proposé par [Lundh et al. 2008], pourrait être implémenté de la sorte. Néanmoins, ce mécanisme semble de façon inhérente fonctionnellement fortement couplé et s'éloigne par conséquent des enjeux principaux des approches orientées service.



(a) Instance de service continu atomique générant un flot d'information



(b) Instance de service continu avec un flot d'information unidirectionnel



(c) Instance de service continu avec un flot d'information bidirectionnel

FIGURE 3.1 – *Dépendances fonctionnelles dues aux flots d'informations.* Toutes les représentations graphiques des compositions de services continus de ce mémoire n'employant pas de flot d'information bidirectionnels ne distinguent pas une instance générant le flot d'information d'une instance dédiée à l'obtention de ce flot. Seule la sous-figure 3.1c est contrainte de faire cette distinction. Ces représentations mettent en évidence que ces compositions de services continus pourraient être implémentées à partir de composants communiquant par événements de façon similaire à l'architecture SLCA [Hourdin et al. 2008] (cf. sous-sous-section 2.3.2.3). Les sous-services externes seraient représentés en interne par un composant mandataire *(proxy)*. Lorsque ces sous-services généreraient des flots d'information, le mandataire prendrait alors en charge les requêtes de génération du flot et d'obtention de celui-ci. Si l'usage de composants logiciels dans nos représentations de compositions de services continus permet de réifier des traitements et d'expliciter leurs dépendances fonctionnelles en matière de flot d'information, la réification de tels composants n'est cependant en aucun cas obligatoire. Elle vise seulement à faciliter notre exposé. La représentation graphique des connecteurs d'envoi et de réception d'événements est inspirée du modèle de composant CORBA *(Corba Component Model)* [Merle 2003].

configurations à base de composants qui peuvent être aplatis même après une planification hiérarchique. Par ailleurs, ces composants présents dans les services composites compliquent le recours à des orchestrateurs génériques voire rendent cette délocalisation très difficile lorsque ces composants sont directement liés à des périphériques matériels. Les services composites fournissent un ancrage aux affinements.

Avantages pour la génération et la diffusion de flots d'information L'emploi d'un service continu pour générer et diffuser un flot d'information n'est pas une obligation : le client peut très bien demander en séquence la réalisation d'actions générant à chaque fois une seule information. Néanmoins, les services continus facilitent la tâche du client mais surtout celle du fournisseur de services en effectuant une distinction entre la génération d'un flot d'information et sa diffusion. En effet, cette distinction leur permet de mettre en place les mesures suivantes :

1. *Côté client : communication événementielle et ré-exploitation d'instances existantes.* Le client génère une seule requête pour la génération des événements et peut ensuite se concentrer sur l'obtention des informations liées à ces événements. Pour cela, il peut utiliser un mécanisme de publication-souscription lorsque celui-ci est disponible ou, à défaut, une méthode de scrutation longue ou active [9]. Par ailleurs, si une instance de service en cours d'exécution générant les événements conformément à la requête est découverte [10], ce client peut alors ré-exploiter cette instance et n'exercer sa responsabilité que sur sa propre instance du service de diffusion du flot d'information.
2. *Côté fournisseur de services : allocation des ressources.* Une requête de service continu aide le fournisseur de services à anticiper les besoins en ressources et lui permet ainsi d'éviter de nombreux cycles d'allocation et de désallocation. Cet aspect est essentiel lorsque ces allocations sont coûteuses et que des ressources rares doivent être réservées.

Le tableau 3.1 dresse une comparaison entre une composition de services continus et une compositions de services réalisant des actions dans un cas de figure impliquant l'échange régulier d'informations des sous-services vers le service composite. Pour ces deux types de compositions, le service consommateur de ces informations doit respecter la structure hiérarchique car ce besoin forme une dépendance fonctionnelle. Ce service est par conséquent composite. En vue de limiter le nombre de sélections locales, on suppose que la composition de services réalisant des actions n'effectue qu'une seule sélection locale par sous-service. Chacun de ses sous-services reçoit cependant de nombreuses requêtes au cours d'une exécution du service composite. Le nombre de ces requêtes dépend généralement du temps d'exécution du service composite. Dans le cas d'une composition de services continus, chaque sélection locale n'est suivie que d'une seule requête de sous-service. Le principal avantage d'une composition de services réalisant des actions est la réutilisation de services existants prévus pour fournir une information à la demande. Les services réalisant des actions n'ayant pas été directement conçus pour la production de flot d'événements, il faut s'attendre à ce que leur performance soit moins bonne que celle des services continus qui ont été élaborés principalement pour ce type d'utilisation.

9. La scrutation longue est généralement préférée à la scrutation active sauf lorsque les événements ont une fréquence faible et régulière : le maintien d'une connexion n'est alors plus justifié.
10. Cette instance de service est sous la responsabilité d'un autre client.

Comparaisons	Compositions de services continus	Compositions de services réalisant des actions
Structure hiérarchique	Obligatoire	Obligatoire
Granularité d'une sélection locale	**Une requête de service continu**	Nombre indéterminé, potentiellement important, de requêtes de service similaires
Gestion des ressources	**Alloue des ressources jusqu'à la réception d'un ordre d'arrêt**	Doit anticiper les requêtes suivantes afin d'éviter des cycles d'allocation et de libération des ressources
Charge du côté du service composite	**Génération d'un nombre réduit de requêtes et gestion du mécanisme d'obtention des événements**	Génération d'un nombre important de requêtes
Fréquence maximale de réception d'événements	**Généralement limitée par le mécanisme d'obtention des événements**	Limitée par le temps de génération d'une nouvelle requête et le temps de réponse de bout-en-bout
Services atomiques actuellement disponibles	Rares	**Nombreux**

TABLE 3.1 – *Tableau de comparaison des compositions de services continus et des compositions de services réalisant des actions en situation de coordination étroite.* Les compositions considérées impliquent l'échange régulier d'informations des sous-services vers le service composite. Les propriétés avantageuses sont mises en gras.

Il ressort de cette comparaison que l'impact le plus important de la coordination étroite sur les compositions de services est la structure hiérarchique rendue nécessaire par la consommation de flot de données. Par ailleurs, les services continus permettent une production et une consommation de ces données plus efficace que les services réalisant des actions. De plus, ils évitent la remise en cause de l'équivalence habituelle entre une sélection locale et une instanciation de service. Aussi, dans la suite de ce manuscrit, nous emploierons systématiquement des compositions de services continus en situation de coordination étroite.

3.1.3 Exemples

Cette sous-section présente divers exemples de services continus et de compositions de services continus dont certains types seront utilisés dans nos scénarios.

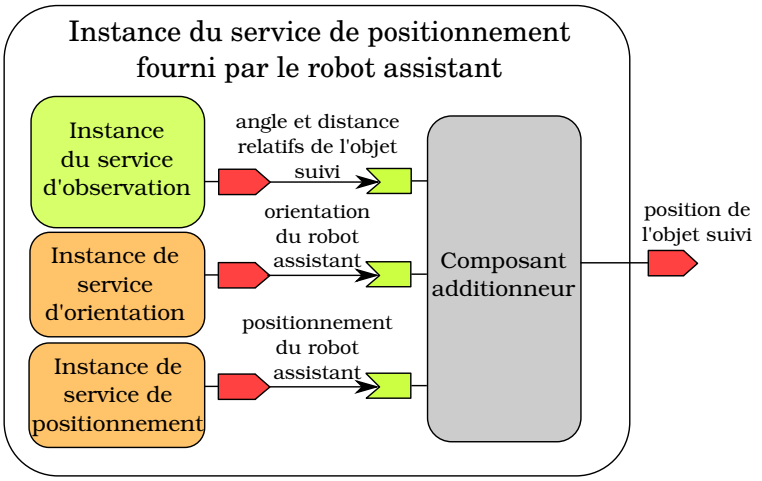
Transformation de flot audio-vidéo Cette activité consiste traditionnellement à connecter en série des composants logiciels de façon à ce que chacun d'entre eux transforme le flot audio-vidéo émanant du composant précédent [Nahrstedt & Balke 2004, Derdour et al. 2010]. Elle peut être également modélisée à l'aide de services en créant un service composite par composant prenant un flot audio-vidéo en entrée et des services atomiques pour les composants en début de chaîne. Comme dans la sous-section précédente 3.1.2, ces services composites requêtent un sous-service pour obtenir le flot généré par le service précédent dans la chaîne de traitement. L'adresse de ce flot leur est transmise dans leur requête de service.

Service de nettoyage Ce service robotique effectue une patrouille de nettoyage dans l'étage d'un bâtiment durant une certaine tranche horaire ou pendant un nombre prédéfini de cycles de rotation. Il peut être noté que ce service ne génère pas de flot d'information particulier mais n'en reste pas moins utile.

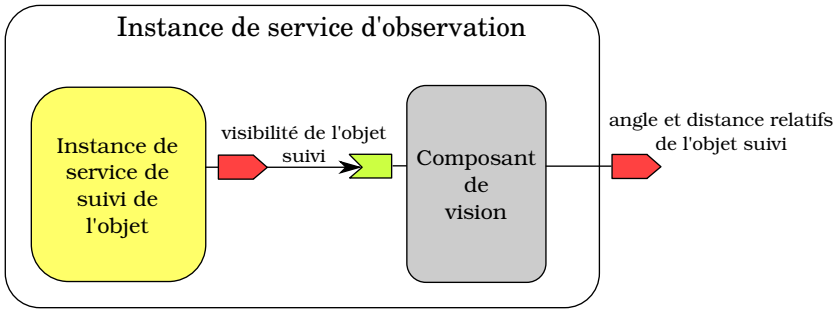
Mesure instantanée d'une métrique sur un service en cours d'exécution Les services de mesure instantanée, mesurant par exemple le bruit acoustique causé par un service robotique, sont également des services continus. Ce type de service fait partie des services employés pour évaluer la performance d'un service ; la sous-section 3.3.2 reviendra sur cette étape.

Localisation d'objets terrestres en intérieur Les services de localisation d'objets mobiles terrestres en intérieur fournissent principalement des flots d'information d'un ou des deux types suivants : l'orientation absolue et la position cartésienne absolue dans le plan horizontal. Ces informations peuvent parfois être complétées d'une estimation de l'incertitude qui est modélisée, dans le meilleur des cas, comme une distribution de probabilités [11]. La sous-sous-section 1.2.3.2 en a présenté différentes technologies de dispositifs proprioceptifs et extéroceptifs. La plupart de celles-ci peut être implémentée sous forme de services atomiques, hormis la solution fournie par un robot assistant qui est elle composite. En effet, comme le montre la figure 3.2, elle fait intervenir trois

11. Dans le cas gaussien, il suffit de fournir la variance. Dans le cas d'une distribution multimodale, des échantillons pondérés peuvent être communiqués.



(a) Instance du service continu de positionnement fourni par le robot assistant



(b) Instance du service continu d'observation

FIGURE 3.2 – *Instances composites du service de positionnement fourni par un robot assistant et de son sous-service d'observation.* Seuls les instances des sous-services continus et les composants sont représentés.

sous-services. Le premier sous-service est un service d'observation estimant l'angle et la distance d'un objet suivi en s'appuyant sur une caméra et un système de vision et en faisant appel à un sous-service de déplacement de type suivi pour s'approcher puis se maintenir à proximité de l'objet cible. Enfin, il requiert un ou deux sous-services pour obtenir l'orientation et la position du robot assistant à partir de laquelle il pourra déduire celle de l'objet suivi. Ces sous-services peuvent être partagés avec le service de suivi de l'objet cible qui a lui aussi besoin de ces informations pour son propre déplacement.

D'autres services composites peuvent être obtenus en fusionnant des services proprioceptifs et extéroceptifs. Il est en effet possible d'utiliser un filtre bayésien dynamique pour estimer la distribution postérieure de la localisation du robot à partir d'un service proprioceptif ou de la commande exécutée (dans le cas où l'objet cible est déplacé par un service) et d'un service extéroceptif. Cette approche nécessite de disposer de modèles de bruit pour chacun de ces services.

Déplacements de robots mobiles Bien que le déplacement d'un robot vers une destination soit formalisable comme une action, son implémentation peut consister en la composition de services continus. En effet, comme représenté dans la figure 3.3a, il peut être composé d'un sous-service de navigation en charge de calculer des petits déplacements relatifs (par exemple une rotation et une translation) qui sont ensuite exécutés par

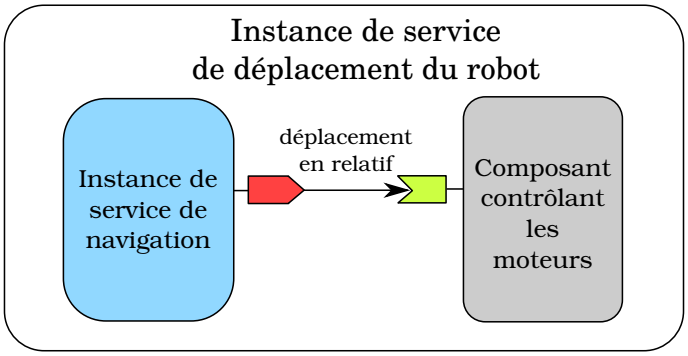
un actionneur (comme un système de roues différentielles d'un robot mobile). Cet actionneur est modélisé comme un composant logiciel et rend cette composition de services spécifique au fournisseur de services, qui est ici le robot mobile déplacé.

Le service de navigation (figure 3.3b) est quant à lui générique et de type continu, tout comme son sous-service de planification de chemin. Ce service de navigation fait appel à des sous-services pour connaître sa localisation ; il sépare volontairement l'obtention de l'orientation et celle de la position afin de permettre l'usage, si nécessaire, de deux sous-services différents. Il reste néanmoins tout à fait possible qu'une même instance de service de localisation génère ces deux types d'information. Le service de navigation peut faire appel à un service de type action pour connaître le modèle cinématique d'un nouveau robot mobile pour lequel il doit générer des commandes. Son sous-service continu de planification de chemin indique le prochain point intermédiaire vers lequel le robot doit se diriger. Ce sous-service se tient informé de la position courante du robot : il fait donc lui aussi appel à un service de positionnement (figure 3.3c). Il génère également un événement en cas de changement de chemin. Les algorithmes de navigation réactive et de planification de chemin de ces deux services composites sont implémentés dans des composants logiciels qui reçoivent divers flots d'information en entrée. Une autre version du service de planification de chemin existe sous forme d'action. Elle est cette fois-ci atomique car la position courante de l'objet à déplacer est contenue dans la requête de service au même titre que la destination.

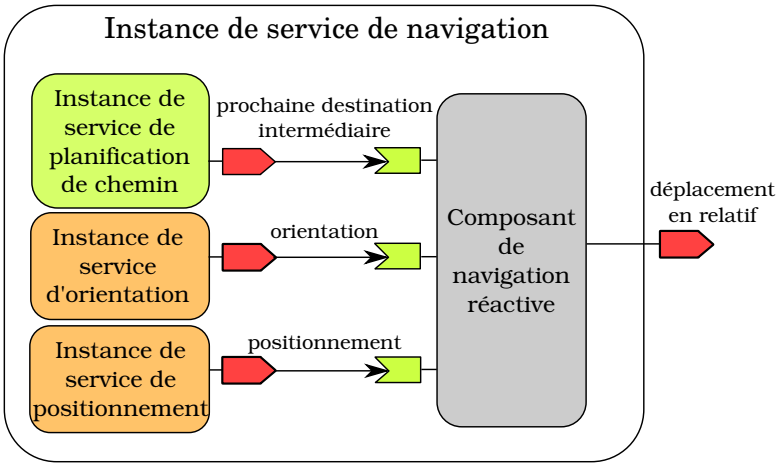
La décomposition du service de déplacement présentée dans la figure 3.3 est radicale puisque même la fonctionnalité de navigation réactive est externalisée. En pratique, il convient d'effectuer un compromis entre les besoins de performance et les opportunités qu'offre une externalisation sous forme de services. Il pourrait par exemple être jugé que seuls les services de localisation méritent d'être externalisés et que les autres fonctionnalités pourraient être modélisées directement comme des composants locaux, économisant ainsi un trafic réseau et l'ajout de couches protocolaires supplémentaires. Une approche intermédiaire consisterait à faire en sorte que certains services soient fournis par le même fournisseur. Il convient toutefois de rester prudent quant à de possibles optimisations car elles peuvent rapidement éloigner les dits «services» des objectifs des architectures orientées service. Toutefois, les faibles vitesses des robots circulant en intérieur et la présence d'une bonne infrastructure réseau semblent fournir des conditions suffisantes pour un usage raisonnable des compositions de services continus de la figure 3.3.

Services utilisés dans notre scénario Dans le scénario à venir, nous supposerons l'absence de robots-remorqueurs et d'alternatives aux services de navigation et de planification de chemin. En conséquence, les robots mobiles devront se déplacer eux-mêmes sans pouvoir faire appel au service d'un remorqueur et utiliseront toujours les mêmes algorithmes de navigation et de planification de chemin. De même, seules les boussoles fourniront les informations d'orientation afin que les implémentations des autres services de localisation ne se concentrent que sur le positionnement. Ainsi, seuls ces derniers services de positionnement fourniront des alternatives à la sélection de services lors de la résolution d'une requête de déplacement d'un robot.

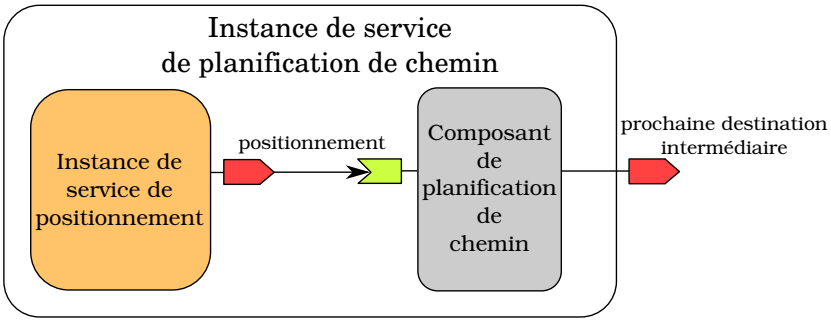
Une telle restriction permet de mettre en place le cadrage décrit dans la sous-sous-section 3.2.2.3. Cette restriction permet de focaliser notre étude sur la prédiction de performance tenant compte de possibles remplacements qui sera traitée par le chapitre suivant. Elle invite par ailleurs à concentrer les efforts de modélisation, quoique modestes, sur les services de positionnement. Notons que cette restriction ne s'exerce que sur l'offre



(a) Instance du service de déplacement (spécifique au robot)



(b) Instance du service continu de navigation (générique)



(c) Instance du service continu de planification de chemin (générique)

FIGURE 3.3 – *Instances de compositions de services continus impliquées dans le déplacement d'un robot mobile.* De même, seuls les instances des sous-services continus et les composants sont représentés.

de services et n'intervient pas sur les aspects architecturaux décrits dans la sous-section 3.3.1.

Bilan L'intégration de services continus permet de décomposer des services jusqu'alors atomiques en sous-services en introduisant la problématique de la coordination étroite au sein de compositions de services logiciels. Les services robotiques comme le déplacement d'un robot sont les premiers bénéficiaires de cette décomposition et sont désormais exposés à de nouvelles opportunités en matière de sélection de services.

Par ailleurs, les services continus accordent une importance de premier ordre au processus engendré par le traitement d'une requête : celui-ci n'est plus cantonné au plan non-fonctionnel associé à la performance mais est aussi désormais pleinement considéré sur le plan fonctionnel. A contrario, les requêtes d'actions ne considèrent pas le processus dans des termes fonctionnels. Elles s'intéressent toutefois à la performance dans une perspective de passage à l'échelle ou lorsque les actions nécessitent un traitement long et coûteux. La considération de la performance est essentielle car il s'agit du critère objectif guidant la sélection de services, telle que cette tâche est traitée dans ce manuscrit. Sa prédiction est donc une étape cruciale et fait l'objet de la section suivante.

3.2 Prédiction de performance

Lorsque les valeurs des métriques ne sont pas fixes, connues à l'avance, estimables par réputation ni même inférables de façon déterministe, elles doivent être obtenues à partir de méthodes statistiques.

La sous-section 3.2.1 présente les méthodes statistiques existantes employées dans les approches orientées service. Celles-ci sont ensuite confrontées dans la sous-section suivante aux exigences de la robotique ambiante. L'objectif de cette sous-section 3.2.2 est alors de dégager une nouvelle forme de sélection adaptée aux compositions de services continus étant basée sur la prédiction contextuelle de performance.

3.2.1 Approches existantes

Nous regroupons les approches proposées en quatre catégories : les approches moyennantes, les séries temporelles, le cas des accords de niveau de performance et la prédiction contextuelle. Sauf mention contraire, les travaux cités dans cette sous-section se réfèrent à une métrique exprimée en unité physique et non à sa forme normalisée que nous appelons paramètre de QdS.

3.2.1.1 Approches moyennantes

L'estimation la plus simple de ces métriques est la moyenne des mesures. En tant que moment du premier ordre, la moyenne suit les évolutions de ces métriques en lissant fortement ses variations. Elle est associée à une opération d'un service donné (par exemple [Tari et al. 2010]). Cette moyenne devant être mise à jour en ligne à chaque nouvelle mesure, elle peut être calculée de façon incrémentale :

$$\mu_{t+1} = (1 - \alpha) \times \mu_t + \alpha \times m_{t+1} \tag{3.1}$$

où μ est la moyenne, α le pas de mise à jour *(step-size)* et m_{t+1} la nouvelle mesure.

Cas des processus de décision markoviens Les moyennes incrémentales sont également couramment utilisées pour estimer les paramètres des fonctions de transition ou de renforcement immédiat de processus de décision markoviens *(Markov Decision Processes, MDP)* à états discrets. Ce formalisme sera présenté dans la sous-section 4.1.1 du chapitre suivant. De tels MDPs ont été proposés pour construire des compositions en sélectionnant leurs sous-services de type action [Doshi et al. 2004, Dehousse et al. 2009]. Toutefois, pour que l'emploi d'un MDP soit justifié, le problème traité doit nécessiter une prise de décision séquentielle qui ne peut pas être réduite en une séquence de décisions locales indépendantes. Dès lors, pour rester dans le cadre non-fonctionnel de la sélection de services, la sélection d'un sous-service doit être rendue dépendante des performances des autres sous-services sélectionnés durant le même épisode. Les sélections précédentes doivent donc être représentées de façon directe ou indirecte (via la performance accumulée) dans l'état pour pouvoir effectuer une prise de décision en respectant l'hypothèse de Markov [12]. Or les états des MDPs des travaux de [Doshi et al. 2004] et de [Dehousse et al. 2009] ne tiennent pas compte de ces éléments : la légitimité de leurs formalisations repose sur la combinaison des considérations fonctionnelle et non-fonctionnelle dans leurs prises de décisions. Ainsi leurs MDPs ne se cantonnent pas à la sélection de sous-services mais aussi à celle de tâches précises.

La proposition de [Doshi et al. 2004] est principalement orientée vers le choix d'actions mais permet également d'effectuer un choix entre des services fonctionnellement équivalents. Leur approche consiste à prédire le résultat de la réalisation d'une action par l'intermédiaire de la fonction de transition de leur MDP. L'état du MDP représente le niveau de connaissance du client qui peut choisir d'effectuer des actions caractérisées par les couples tâche-service. Les résultats considérés doivent avoir un nombre de valeurs possibles très limités et sont généralement binaires. Les auteurs donnent comme exemples la disponibilité d'un produit auprès d'un fournisseur et le succès de l'expédition d'une commande. La fonction de renforcement immédiat fournit le coût associé à chaque réalisation de tâche par un service donné qui est fixe et connu à l'avance.

Au sein du MDP proposé par [Dehousse et al. 2009], les états correspondent à des tâches abstraites, partiellement définies, dont la réalisation passe par le choix d'une tâche précise couplée à un service. Les réalisations de deux tâches proches mais distinctes peuvent conduire vers des tâches abstraites suivantes différentes. Comme précédemment, les actions du MDP sont formalisées comme l'association d'une tâche précise à un service. Ces auteurs estiment chaque paramètre de la fonction de renforcement immédiat comme la moyenne du score local associé à l'exécution d'une action dans un état donné. Cette moyenne est mise à jour après chaque exécution à partir de l'agrégation des paramètres de QdS venant d'être mesurés et normalisés. Lorsque la relation entre les tâches abstraites et précises d'une composition est bijective, cette approche est strictement équivalente à une sélection locale.

12. Cette hypothèse stipule que la connaissance de l'état courant est suffisante pour la prise en compte du passé dans le choix de l'action courante.

3.2.1.2 Séries temporelles

Les séries temporelles s'intéressent à la modélisation de l'évolution d'une valeur dans le temps. Elles permettent de prédire une valeur d'une métrique à partir d'un nombre fini de ses valeurs lors des exécutions précédentes. La modélisation d'une série temporelle d'une variable aléatoire suppose qu'elle soit stationnaire : une métrique non-stationnaire doit d'abord être transformée en une variable aléatoire stationnaire.

Divers travaux se sont intéressés à cette forme de prédiction de performance pour des services accessibles sur le Web [Cavallo et al. 2010, Godse et al. 2010, Amin et al. 2012]. Bien que le modèle linéaire ARIMA (*AutoRegressive Integrated Moving Average*) ait été utilisé par divers auteurs [Cavallo et al. 2010, Godse et al. 2010] avec des résultats mitigés, les expérimentations menées par [Amin et al. 2012] ont montrées qu'un modèle non-linéaire SETARMA (*Self Exciting Threshold ARMA*) serait plus intéressant dans près de 70% des cas pour prédire le temps de réponse. Par ailleurs, selon ces auteurs, 95% des 800 services observés se prêteraient à une modélisation à base de séries temporelles pour cette métrique.

3.2.1.3 Accords de niveau de service

Une part considérable des travaux consacrés au suivi du comportement de services est destinée à l'application d'accords de niveau de performance. Ces accords ne visent pas simplement à offrir une bonne performance mais aussi à garantir une stabilité. Cette exigence de stabilité est exacerbée lorsqu'un fournisseur de services a établi des accords à la fois avec ses clients et avec les sous-services externes qu'il sollicite : il apparaît alors des dépendances dans les engagements des différents fournisseurs de service. En conséquence, la vérification de ces accords ne peut se contenter de valeurs moyennes : elle doit tenir compte de l'ampleur des variations possibles et de leur fréquence.

Devant le comportement variable de ces métriques, il convient de distinguer les contraintes souples des contraintes rigides de SLA [Rosario et al. 2009, Amin et al. 2012]. Une contrainte souple spécifie une borne devant être respectée avec une certaine probabilité et non dans tous les cas. À cette occasion, elle encourage une estimation de la distribution des valeurs d'une métrique. Comme nous l'avons vu précédemment dans la sous-sous-section 2.4.2.1, ces distributions sont également utiles pour étudier le comportement de compositions complexes [Rosario et al. 2009]. Elles peuvent être estimées à l'aide d'histogrammes ou suivant des familles de distributions leur ayant été assignées à l'avance [Cardoso et al. 2004].

Ces distributions ne sont pas toujours stationnaires et leur évolution peut causer une violation des accords de SLA. Au lieu de dégrader les bornes de ces accords de façon à tolérer certaines évolutions, il est parfois intéressant d'offrir des bornes plus exigeantes et les remettre en cause en cas d'évolution. Le nouvel enjeu est alors de détecter au plus tôt les violations d'accord entraînées par ce changement afin de permettre au système de prendre des contre-mesures aussitôt que possible. Pour cela, [Amin et al. 2012] ont utilisé une carte de contrôle CUSUM (*CUmulative SUM control chart*), technique issue de la maîtrise statistique des procédés.

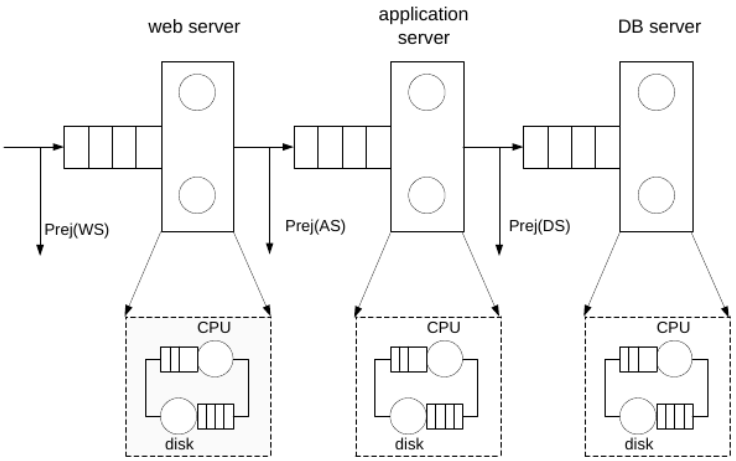


FIGURE 3.4 – Files d'attente associées à un service Web de commerce en ligne [Menascé et al. 2001]

3.2.1.4 Prédiction contextuelle

Au lieu de se concentrer sur des séquences de mesures et de suivre leurs évolutions, on peut s'intéresser au contexte qui conditionne le comportement d'un service. S'il peut être représenté par les sources d'informations à disposition du sélectionneur, alors la prédiction de performance devient envisageable.

Files d'attente Lorsque les services sont soumis à une forte charge en matière de requêtes, leur performance est alors très liée au niveau de concurrence pour l'accès aux ressources matérielles (processeur, disque dur, réseau) et logicielles (bases de données). La théorie des files d'attente, communément utilisée pour modéliser le comportement des routeurs dans les réseaux informatiques, offre un cadre analytique permettant ici de prédire le temps de traitement d'une requête à partir de l'état des files d'attentes (voir la figure 3.4) [Menascé et al. 2001, Chandra et al. 2003, Garlan et al. 2004]. La charge calculatoire d'une requête est souvent considérée comme fixe pour une opération donnée [Menascé et al. 2001] mais est parfois estimée [Chandra et al. 2003] et pourrait être prédite, tandis que l'état des files d'attente est mesuré continuellement. Les files d'attente étant de taille finie, cette approche peut également anticiper les rejets [Menascé et al. 2001]. Les prédictions sont effectuées du côté du fournisseur de services et sont souvent utilisées pour l'allocation dynamique de ressources, au moyen d'une prévision de la charge sur la période à venir [Chandra et al. 2003]. La prédiction de performance rejoint ainsi les problématiques de l'informatique autonomique *(autonomic computing)*.

Fouille de données Au-delà de cette forme analytique, divers auteurs [Castellanos et al. 2004, Zeng et al. 2008, Leitner et al. 2009] ont émis l'idée d'effectuer une fouille de données pour dégager les sources d'information pertinentes pour la prédiction des paramètres de QdS. Ces informations, couplées aux valeurs des paramètres de QdS formeraient alors des exemples permettant d'effectuer l'apprentissage d'un modèle de prédiction. [Leitner et al. 2009] s'intéressent au temps de réponse tandis que [Zeng et al. 2008] ainsi que [Castellanos et al. 2004] évoquent le taux de succès. Malheureusement, leurs propositions n'ont pas davantage étudié ce problème de fouille de données, reléguant

la sélection des sources d'informations ainsi que l'apprentissage à une entité externe, en supposant que celle-ci puisse traiter ces tâches automatiquement.

3.2.2 Prédiction contextuelle pour la robotique ambiante

La prédiction de performance de services robotiques peut s'avérer être une tâche complexe. Ces services sont soumis à d'autres contraintes que dans les approches venant d'être évoquées ; cette sous-section s'en fait l'écho (3.2.2.1). La robotique ambiante pouvant désormais proposer des services composites grâce aux sous-services continus, nous verrons qu'il est devenu particulièrement difficile d'évaluer et d'utiliser la performance individuelle des instances de sous-services alors que ceux-ci peuvent être en étroite coordination avec leur instance de service composite supérieure (3.2.2.2). Au lieu de cela, nous proposerons de prédire directement la performance d'une instance de service de haut-niveau lors de chaque évaluation d'une combinaison de services candidats (3.2.2.3). Enfin, nous présenterons le cas d'un service de déplacement de robot mobile nécessitant des prédictions de performance spatio-contextuelles (3.2.2.4) avant de conclure (3.2.2.5).

3.2.2.1 Contraintes physiques

Les services robotiques employés en robotique ambiante sont soumis aux contraintes de leur environnement physique, ce qui induit des variations parfois importantes dans la performance d'une même opération de service. Cette forte variation entre en contradiction avec l'objectif de stabilité porté par les accords de niveau performance. Tandis que ces derniers visent à rendre la performance d'une instance de service la plus indépendante possible d'un contexte subit en encourageant les fournisseurs à allouer suffisamment de ressources informatiques et à basculer vers d'autres sous-services distants en cas de nécessité, la robotique ambiante ne peut s'abstraire aussi aisément du contexte de son environnement physique. En effet, elle n'a à sa disposition qu'un ensemble limité de services robotiques disponibles dans son environnement immédiat [13] et elle doit composer avec les limites physiques de leurs capteurs et de leurs actionneurs.

Nouvelles hypothèses La coordination étroite exigée par certains services composites rend difficile l'usage de mécanismes de files d'attente pour plusieurs raisons. Tout d'abord, certaines exécutions doivent être synchronisées : c'est le cas par exemple des exécutions d'un service de mesure et du service mesuré. De plus, certaines exécutions modifient le contexte des exécutions suivantes (par exemple lorsqu'un robot se déplace), ce qui complexifie le processus de sélection en exigeant de celui-ci qu'il tienne compte du contenu des requêtes stockées dans une file d'attente. Aussi pour des raisons de simplicité, nous effectuerons les hypothèses suivantes :

1. *Nos services robotiques et nos autres services continus ne disposent pas de file d'attente.* Si une opération de service ne peut pas traiter plusieurs requêtes à la fois ou que son maximum est déjà atteint, alors les requêtes excédentaires seront rejetées.
2. *Le traitement simultané de plusieurs requêtes par une même opération de service ne disposant pas de file d'attente n'impacte pas mutuellement leurs performances*

13. À la différence d'un nombre considérable de services logiciels accessibles via Internet.

individuelles. Pour cette raison, les opérations de service particulièrement sensibles à cette concurrence sont restreintes à une seule exécution à la fois.

3. *Tous les services de l'environnement sont immédiatement disponibles pour traiter une requête de haut-niveau.* Les conflits n'interviennent qu'entre requêtes de sous-services.

3.2.2.2 Quelles performances doivent être prédites ?

La sélection de services considérée dans cette sous-section se place dans une perspective de sélection globale. En effet, nous considérons que seule la performance associée à la réalisation d'une tâche de haut-niveau requise par l'utilisateur intéresse ce dernier : il n'a pas besoin de connaître les constituants d'une instance de service de haut-niveau composite ni à savoir comment les départager. Comme nous l'avons évoqué dans la sous-sous-section 2.4.2.1, les métriques d'une instance de service composite à base de sous-services de type action peuvent être estimées à partir des métriques individuelles de ses instances de sous-services. Comme nous allons maintenant le voir, la coordination étroite induite par les compositions de services continus rend cette procédure difficilement praticable.

Dépendances entre les performances Dans le cadre d'une coordination étroite, il est légitime de s'attendre à ce que les performances des instances de services composites et de leurs instances de sous-services soient liées. Les dépendances entre ces performances peuvent être à double sens, même lorsque les dépendances fonctionnelles sont à sens unique :

1. *Dépendance non-fonctionnelle des traitements internes d'une instance de service composite envers ses instances de sous-services.* Lorsqu'un composant d'une instance de service composite s'abonne à un flot d'information généré par une instance de sous-service, il est fortement probable que la performance du traitement effectué par le composant devienne dépendante de celle de l'instance de sous-service. C'est le cas par exemple du composant de navigation réactive du service de navigation présenté dans la sous-section 3.1.3 et illustré par la figure 3.3b dont la performance est dépendante de l'instance du sous-service de localisation mais aussi de la direction indiquée par l'instance du sous-service de planification de chemin.
2. *Dépendance non-fonctionnelle d'une instance de sous-service continu envers son client.* Les sous-services ne sont pas toujours autonomes sur le plan non-fonctionnel. Tout d'abord, de nombreuses instances de sous-services continus sont exécutées à durée indéterminée : leur durée est alors souvent surdéterminée par celle de l'instance de service composite. De plus, il arrive dans certains cas de figure que l'instance de sous-service sollicite un flot d'information généré par l'instance de service composite (pour rappel, voir la sous-figure 3.1b). Il est par ailleurs fréquent en robotique mobile que la tâche effectuée par l'instance de sous-service soit influencée par l'instance de service composite sans qu'aucun flot d'information ne soit pourtant échangé en ce sens. Ceci a pour conséquence qu'une métrique comme la précision moyenne [14] d'une instance de service de localisation est elle

14. Cette métrique est par ailleurs difficilement mesurable car elle nécessite de comparer les valeurs de l'instance de service observée à celles d'un dispositif plus précis faisant office de référence. Lors d'une exécution normale, il semblerait alors plus judicieux d'utiliser directement ce dispositif de référence.

aussi dépendante de la navigation du robot car si ce dernier éprouve des difficultés et reste longtemps dans une zone où cette instance de service de localisation est peu précise, la valeur moyenne de cette métrique sera alors dégradée. Il en va de même pour le bruit acoustique qu'émet une instance de service de localisation comme celle fournie par un robot assistant. Bien qu'elle ne puisse souvent pas être prédite prise isolément, nous verrons dans la sous-section 3.3.2 que la performance individuelle d'une instance de sous-service contribue directement aux métriques cumulatives de son instance de service composite supérieure comme le coût et le bruit acoustique.

Ce phénomène de dépendance peut être qualifié de *couplage non-fonctionnel*.

Performances considérées Face au constat de l'existence de relations de dépendance en matière de performance entre certains composants et les instances de sous-services, deux stratégies peuvent être distinguées pour prédire la performance de l'instance du service de haut-niveau :

1. *Modéliser les relations entre composants et les instances de sous-services.* Dans sa version simple, cette approche modélise les relations d'indépendance entre les performances de certains groupes d'instances de sous-services et de composants, ce qui lui permet d'évaluer leur performance séparément puis de les agréger pour obtenir des métriques de haut-niveau. Néanmoins, dans le cas du service de déplacement et de ses sous-services composites, les composants et les instances de services subordonnées sont liés et ne peuvent pas en bénéficier. Par la suite, cette approche pourrait décrire les relations de dépendance entre ces éléments mais ceci nécessite une étude approfondie qui ne sera pas réalisée dans le cadre de cette thèse ; elle fera néanmoins partie des principales perspectives.
2. *Considérer la performance d'une instance de service composite comme non-décomposable.* Face à la difficulté de prédire les performances individuelles des composants et des instances de sous-services puis d'en déduire à l'aide de règles algébriques la performance de l'instance de service composite, la seconde approche consiste à prédire directement la performance de l'instance de service composite. Lorsque ces instances de services composites sont elles-mêmes les instances de sous-service d'une autre instance de service composite, cette approche ne prédit pas leur performance mais se concentre sur celle *in fine* de l'instance de service de haut-niveau au sommet de la hiérarchie de compositions de services continus. Cette instance de service de haut-niveau est d'ailleurs la seule à disposer d'un modèle d'évaluation, lui-même constitué d'une fonction de score et d'un choix de paramètres de QdS (cf. sous-section 3.3.2).

   En tant que première contribution à cette problématique de sélection des services d'un organigramme sous étroite coordination, l'approche retenue dans ce manuscrit adopte cette seconde stratégie impliquant un moindre effort de modélisation.

3.2.2.3 Sélections à partir de prédictions de la performance du service de haut-niveau

Structure De part leur structure hiérarchique, les services de l'organigramme ne sont pas sélectionnés en une seule étape mais suivant une relation d'ordre partiel. Nous proposons que chaque instance de service composite obtienne l'ensemble de ses sous-services en réponse à une seule requête de sélection : cette sélection est dès lors assimilable à un

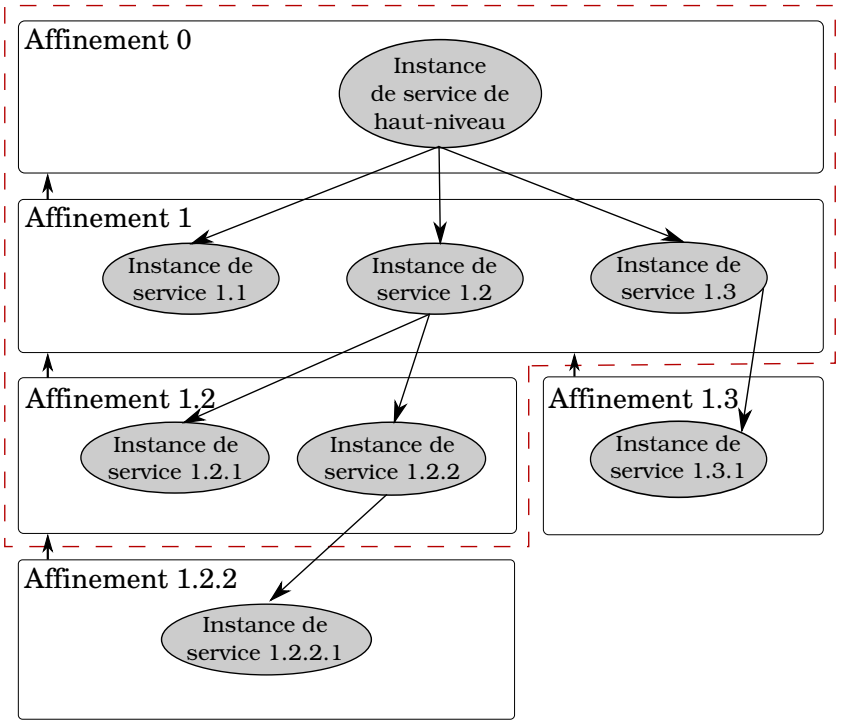


FIGURE 3.5 – *Hiérarchie d'affinements.* À titre d'exemple, l'affinement 1.2.2 a pour chaîne hiérarchique les affinements regroupés dans le cadre rouge en tirets.

affinement (refinement), tel qu'utilisé en planification hiérarchique (cf. sous-sous-section 2.3.2.2). La figure 3.5 présente les affinements d'un organigramme. Ces affinements ont eux aussi une structure hiérarchique et sont donc représentés suivant un arbre. Suivant cette représentation, les ancêtres d'un affinement forment une chaîne hiérarchique.

Prises de décisions Le choix des sous-services d'un affinement n'a pas forcément lieu lors du traitement de la requête de sélection de services correspondante mais peut avoir été décidé en amont, lors de la sélection d'un affinement de sa chaîne hiérarchique. L'utilisation d'un même service de sélection par l'ensemble des services composites de l'organigramme rend en effet possible cette *dissociation entre les traitements des requêtes de sélection et les prises de décisions*. Trois grandes questions se posent alors :

1. *Portée d'une prise de décision.* Doit-on décider de l'ensemble des services de l'organigramme en une seule fois ou peut-on ne sélectionner qu'un sous-ensemble de nœuds de l'organigramme ? Ce sous-ensemble peut-il correspondre à :
   (a) un sous-arbre complet de l'organigramme ou limité en profondeur,
   (b) une chaîne d'affinements,
   (c) un affinement,
   (d) une sous-tâche au sein d'un affinement,
   (e) un sous-ensemble où certaines tâches sont écartées par manque d'influence sur la performance globale ?

   L'impact des prises de décision effectuées dans d'autres branches de l'organigramme que ce sous-ensemble est probablement un élément-clé à considérer afin de répondre à ces questions.

2. *Combinaisons évaluées pour chaque prise de décision.* Sur quels critères peut-on guider l'évaluation des combinaisons et éviter ainsi l'énumération ?
3. *Modèles de performance employés.* Un modèle de performance est-il spécifique à un modèle d'évaluation donné ou doit-il alors le considérer en entrée ? De plus, un modèle de performance est-il :
   (a) *Global ?* Cette solution n'est possible que si à l'issue de la prise de décision tous les services de l'organigramme sont connus. Dans ce cas, le modèle de performance peut être commun à l'ensemble des combinaisons possibles au sein de l'organigramme. La structure de l'organigramme fait alors partie prenante de l'espace d'entrée de ce type de modèle de performance.
   (b) *Commun à l'ensemble des combinaisons considérées mais spécifique à un sous-ensemble des autres services de l'organigramme ?* On ne s'intéresse alors plus à la structure de l'organigramme mais à celle regroupant les combinaisons considérées. Il faut déterminer à quelle structure le sous-ensemble des autres services de l'organigramme peut être réduit. Ce point semble être connexe de la question de la portée des prises de décision. Il représente un enjeu de taille car les services de ce sous-ensemble doivent avoir été sélectionnés avant que cette évaluation puisse avoir lieu.
   (c) *Spécifique à chaque combinaison ainsi qu'à un sous-ensemble des autres services de l'organigramme ?* Cette forte réduction de la dimensionnalité de son espace d'entrée se fait au détriment d'une plus grande rareté en matière d'exemples. En conséquence, le nombre de modèles de performance subit la croissance exponentielle du nombre de combinaisons et est également sensible au nombre des modèles d'évaluation possibles. L'expérience nécessaire à l'apprentissage se fait donc plus rare.

Ces questions cruciales ne seront pas davantage examinées dans le cadre de cette thèse de part l'importance de l'étude qu'elles exigent. Elles feront cependant naturellement partie des principales perspectives. Le chapitre suivant se concentrera sur un exemple particulier qui repose sur le cadrage suivant.

Notre restriction de cadre Nous proposons de ne traiter qu'*a minima* les questions précédentes, en nous restreignant au cadre suivant :

1. *Portée de la prise de décision : organigramme.* La portée est donc maximale. Cela suppose que le sélectionneur soit capable de prédire quelles requêtes de sous-services seront émises par les services composites d'une combinaison afin que le sélectionneur puisse découvrir les services correspondants et s'assurer de leur disponibilité. Dans le cadre du service de déplacement de robot, cette exigence n'entraîne pas de difficulté car seule la requête de haut-niveau varie d'une exécution à une autre pour un même organigramme.
2. *Évaluation des combinaisons : par énumération, après que les restrictions du sélectionneur aient été appliquées.* Comme la sous-sous-section 3.1.3 l'a évoqué, seules les alternatives du service de positionnement sont considérées. Par ailleurs, le même service de positionnement est utilisé par le service de navigation et son sous-service de planification de chemin. Si des alternatives se présentent pour les autres services de l'organigramme, elles seront arbitrairement ignorées par le sélectionneur. Néanmoins, de telles alternatives n'apparaîtront pas dans notre scénario, ce qui permet de minimiser l'effort de modélisation de services robotiques. La disparition de services ne sera par ailleurs pas étudiée.

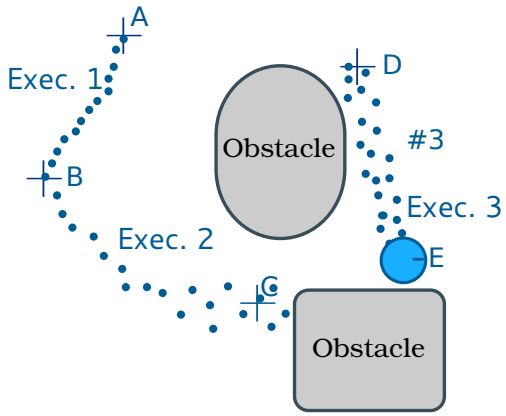


FIGURE 3.6 – *Trois exécutions successives d'organigrammes impliquant les mêmes services pour le déplacement d'un robot mobile.* Le robot (en bleu) va de A à B pour sa première exécution puis de B à C pour sa seconde exécution. Bien que ces deux exécutions partagent un point (B), les trajectoires sont très différentes et ne sont pas perturbées de la même manière. Ensuite, un autre organigramme dont certains services diffèrent provoque le déplacement de C à D. Enfin un organigramme similaire au deux premiers est exécuté par aller de D en E. Du fait des obstacles, sa trajectoire est très perturbée.

3. *Modèles de performance : spécifiques aux combinaisons évaluées.* La combinaison ne fait dès lors pas partie de l'espace d'entrée de ces modèles.

3.2.2.4 Modèles de performance spatio-contextuels du déplacement

Suivant la proposition qui vient d'être effectuée, les modèles de performance employés pour évaluer des affinements dans le cadre de compositions de services continus robotiques effectuent leurs prédictions à partir d'informations contextuelles. Comme nous allons le voir, le choix de ces informations est pour l'essentiel spécifique à la nature du service de haut-niveau. Aussi, nous nous concentrons sur le cas particulier du service de déplacement fourni par un robot mobile.

Pourquoi une prédiction contextuelle ? Dans le cas du déplacement d'un robot, le traitement de la requête réalise une trajectoire entre le point de départ et la destination. Comme le montre la figure 3.6, la corrélation entre les exécutions successives d'organigrammes impliquant les mêmes services semble moins évidente car deux destinations consécutives sont *a priori* indépendantes [15]. Il convient donc de prédire les valeurs des paramètres de QdS à partir du contexte spatial plutôt que d'estimer ces valeurs à l'aide de séries temporelles dont la stationnarité serait difficile à dégager.

Informations contextuelles La performance du service de déplacement est intimement liée à l'environnement dans lequel le robot mobile évolue. Intuitivement, il semble pertinent de considérer la trajectoire qui, dans sa forme la plus simple, est décrite par une estimation de la position initiale du robot et par sa destination. Par ailleurs, certains sous-services peuvent demander des informations contextuelles supplémentaires.

15. Leur dépendance serait alors due à une dépendance entre les requêtes émises par l'utilisateur.

Par exemple, le service de localisation fourni par un robot assistant nécessite que ce dernier se déplace à proximité du robot à localiser : il semble donc judicieux de considérer sa position actuelle. Dans une version précédente de ce scénario, seule la distance du robot assistant était considérée [Cogrel et al. 2011].

Apprentissage Les modèles de prédiction de performance considérés pour le service de déplacement sont obtenus par apprentissage et feront l'objet du chapitre suivant. De part l'utilisation d'informations contextuelles spatiales sous une forme métrique, ces modèles apprennent les idiosyncrasies de l'environnement. Étant donné que l'observation de ces particularités dépend des paramètres de QdS choisis par l'utilisateur, cette approche métrique a pour principal avantage sur l'approche de [Morisset & Ghallab 2008] de ne pas exiger de connaissance préalable fine nécessaire à l'extraction d'attributs complémentaires. De tels attributs devraient en effet capturer des caractéristiques spécifiques à la tâche de navigation qui permettraient la prédiction directe ou indirecte des valeurs de chacun des paramètres de QdS [16]. Ces informations contextuelles peuvent donc être utilisées pour divers modèles d'évaluations sollicitant des paramètres de QdS différents.

3.2.2.5 Résumé

L'introduction de coordination étroite au sein de compositions de services exigée par la robotique ambiante nécessite de reconsidérer significativement la sélection des sous-services. En effet, les performances des instances de services de l'organigramme sont étroitement couplées, ce qui rend particulièrement difficile la prédiction de leurs performances individuelles et de leurs effets. En conséquence, nous avons opté pour une approche basée sur des modèles de performance prédisant directement la performance de l'instance de service de haut-niveau en fonction du contexte, du modèle d'évaluation et d'un organigramme candidat. En tant que première contribution à cette problématique, nous avons choisi de nous concentrer sur la prédiction de performance et de ne traiter qu'*a minima* le problème combinatoire posé par l'organigramme. Ce contournement est effectué par l'intermédiaire des restrictions imposées par notre scénario.

Le contexte considéré pour l'apprentissage des modèles de performances étant pour l'essentiel spécifique à la catégorie du service de haut-niveau, l'étude qui vient d'être débutée se consacre au cas exclusif du service de déplacement d'un robot mobile. Son contexte caractérise les extrémités de la trajectoire que doit effectuer ce dernier. Cette étude sera poursuivie plus en détail dans le chapitre suivant. Avant cela, la section suivante va présenter les aspects architecturaux nécessaires à la mise en œuvre d'une sélection dans le cadre qui vient d'être dégagé.

3.3 Besoins architecturaux

Après la reformulation de la problématique de la sélection de services imposée par la modélisation des fonctionnalités robotiques sous forme de compositions de services continus, cette section étudie les aspects architecturaux nécessaires à la mise en œuvre de cette nouvelle approche. Dans un premier temps, la sous-section 3.3.1 dresse une

16. La prédiction indirecte consiste ici à prédire les valeurs de leurs métriques associées puis de les normaliser.

liste d'exigences architecturales puis effectue des recommandations afin d'y répondre. Enfin, la sous-section 3.3.2 détaille le processus d'évaluation de l'exécution d'un service de haut-niveau à la lumière des exigences portées par la robotique ambiante.

3.3.1 Exigences, recommandations et propositions architecturales

Les Exigences Architecturales (EA) spécifient les contraintes imposées par la problématique de sélection de services dégagée dans la sous-section 3.2.2. Les Recommandations Architecturales Fortes (RAF) répondent aux exigences demandant d'être précisées et facilitent la mise en œuvre des services participant à l'évaluation de l'exécution d'une instance de service de haut-niveau. Le rassemblement des recommandations fortes et des exigences architecturales se veut être un ensemble cohérent. Une recommandation optionnelle est par ailleurs proposée.

Ces exigences et recommandations s'adressent aux compositions de services continus souhaitant donner lieu à une sélection de services basée sur la performance. Certains services de type action peuvent cependant être intégrés à l'organigramme. Néanmoins, la plupart d'entre eux est exclue de l'organigramme ; le cas le plus représentatif de cette exclusion est celui des services de sélections bien que ceux-ci soient sollicités par tous les services composites à base de services continus. Dans notre scénario s'appuyant sur le service de déplacement, le service de haut-niveau sera l'unique service de type action de l'organigramme.

Cette sous-section s'appuie sur l'expérience obtenue lors de la réalisation d'un prototype architectural. Cette expérience est succinctement présentée en annexe B.1.

3.3.1.1 Exigences architecturales

EA1 *Le sélectionneur doit avoir connaissance du modèle d'évaluation assigné au service de haut-niveau.* Cette information est en effet indispensable puisque ce modèle d'évaluation dirige les choix de la sélection de services. Ce modèle d'évaluation indique quels paramètres de QdS et quelle fonction d'agrégation doivent être considérés pour évaluer l'instance de service de haut-niveau traitant la requête fonctionnelle. Ces choix seront discutés dans la sous-section suivante 3.3.2.

EA2 *Une instance de service composite doit fournir les informations nécessaires à son positionnement dans l'organigramme lorsqu'elle requête une sélection de services.* Pour pouvoir la positionner dans l'organigramme, le sélectionneur a besoin de trois connaissances : la requête de services que cette instance est en train de traiter, son adresse ainsi que l'adresse de l'instance de service supérieure directe [17] qu'il doit savoir positionner. L'adresse d'une instance peut être une URI lui étant dédiée dans le cas d'un service REST-HTTP ou une EPR couplée à un `resourceID` dans le cas d'un service SOAP muni de l'extension WS-Addressing (cf. sous-sous-section 2.1.2.3). En fonction des choix architecturaux effectués, le sélectionneur peut disposer à l'avance d'une partie de ces informations que le requêteur n'aura pas besoin de préciser dans sa requête. Ces informations peuvent s'avérer insuffisantes si l'instance de service supérieure directe n'a pas été sélectionnée par le même sélectionneur.

17. Il s'agit de son client, qui est lui-même un service composite.

EA3 *Le sélectionneur doit avoir connaissance de l'adresse de l'instance du service de haut-niveau et de la requête qu'elle traite.* Cette exigence est très proche de la précédente (EA2) mais s'en distingue par le fait qu'il n'est pas strictement obligatoire que le service de haut-niveau fasse appel à un service de sélection ; cela est cependant fortement recommandé (RAF4). Cette exigence permet au sélectionneur d'instancier des services de mesures sur l'exécution de ce service afin d'obtenir des retours nécessaires à l'apprentissage de ses modèles de performance. Les valeurs de ces mesures peuvent bien entendu être ensuite normalisées et agrégées en un score, comme nous le verrons dans la sous-section suivante 3.3.2.

EA4 *Les informations contextuelles n'étant pas fournies dans une requête de sélection doivent être rendues accessibles par un service.* Par exemple, la position initiale d'un robot n'est en règle générale pas transmise dans la requête de sélection de service contrairement à la destination finale qui fait partie de la description de la tâche requise. Le sélectionneur doit donc émettre une requête afin de l'obtenir.

3.3.1.2 Recommandations architecturales

Recommandations fortes

RAF1 *Toutes les sélections des instances de services d'un organigramme sont effectuées par le même service de sélection.* L'éventualité d'une prise de décision unique pour l'ensemble de l'organigramme et donc d'une dissociation entre prise de décision et traitement de requête de sélection rend cette approche quasi-incontournable. Par ailleurs, même si une prise de décision est effectuée à chaque requête de service, il n'en reste pas moins nécessaire de connaître l'organigramme en cours de constitution. Si plusieurs services de sélection différents étaient utilisés au sein d'un même organigramme, la reconstitution de ce dernier pourrait exiger la collaboration active de ces services de sélection. Si une instance de service composite ne sollicite pas le même service de sélection que sa supérieure directe (son client), elle risque alors d'être considérée par le sélectionneur comme étant atomique. Au-delà de la reconstitution de l'organigramme, la sollicitation du même service de sélection au fil des exécutions du service de haut-niveau lui permet de *centraliser l'expérience* nécessaire à l'apprentissage de ses modèles de performance.

RAF2 *La sélection des sous-services continus d'une instance de service composite requis au même instant doit donner lieu à une seule requête.* Cette sélection correspond à la sélection d'un affinement, telle que décrite dans la sous-sous-section 3.2.2.3.

RAF3 *L'utilisateur ou son représentant indique un modèle d'évaluation en complément de la description fonctionnelle de la tâche de haut-niveau.* Ce modèle d'évaluation vient en complément d'une description fonctionnelle de la tâche requise exprimant le but à atteindre dans le cas d'une action (comme par exemple `déplacer tel objet à telle destination`) ou le traitement à effectuer pour un service continu (par exemple `localiser tel objet de façon régulière durant une période indéterminée`). Comme [Junghans & Agarwal 2010], nous supposons que les propriétés non-fonctionnelles sont spécifiées dans cette description fonctionnelle. Leur vérification a lieu en amont de la sélection de services et elles agissent donc comme des filtres. Par exemple, il peut s'agir de vérifier le support

de fonctionnalités de sécurité comme le chiffrement TLS [18] [Chollet et al. 2010] ou la disponibilité du fournisseur de service pour la requête courante. Ce modèle d'évaluation peut ensuite être indiqué au service de sélection de services pour satisfaire l'exigence EA1.

RAF4 *Toutes les instances de services de l'organigramme sont sélectionnées en réponse à des requêtes de sélection de services.* Les services composites n'ayant a priori pas connaissance de l'organigramme ni du modèle d'évaluation, ils seraient susceptibles d'effectuer des choix sous-optimaux. Par ailleurs, si ces choix venaient à varier d'une exécution à une autre, cela apporterait de la variance à la performance globale.

RAF5 *Tous les services composites indiquent leur adresse dans leur requête de sélection de services.* Cette information répond partiellement à l'exigence EA2. Cette recommandation, combinée à RAF4, implique donc que les instances des services composites sont adressables.

RAF6 *Tous les services sélectionnés sont instanciés par le sélectionneur.* Cette recommandation est motivée en premier lieu par la satisfaction de l'exigence EA3 voulant que le sélectionneur connaisse l'adresse de l'instance du service de haut-niveau et sa requête. Grâce aux recommandations RAF4 et RAF1, on sait que l'instance de service de haut-niveau sera sélectionnée par le même sélectionneur que les instances de ses sous-services : l'unique sélectionneur a donc connaissance de sa requête de haut-niveau [19]. En envoyant lui-même la requête de service au service sélectionné, il obtient directement l'adresse de cette instance, que le service de haut-niveau se doit de lui fournir en vertu de l'exigence EA3. La figure 3.7 illustre cette séquence d'interactions. Dès lors, l'obtention de cette adresse ne repose plus sur sa transmission par des requêtes de services effectuées par ses éventuelles instances de sous-services composites. Cette approche est robuste face à la possible atomicité du service de haut-niveau. Après avoir obtenu cette adresse, le sélectionneur la redirige vers son client. Cette recommandation est peu contraignante pour le service de haut-niveau car pour obtenir une adresse, il faut que le traitement de la requête de haut-niveau soit en mode asynchrone (cf. sous-section 2.1.1). L'envoi et l'acceptation de la requête peut donc être distinguée du suivi de l'exécution de l'instance de service et de l'attente de sa terminaison.

Cette recommandation généralise cette pratique à l'ensemble des instances de services de l'organigramme (voir la figure 3.8). Combinée à la recommandation RAF5, elle permet de satisfaire l'exigence EA2. Elle a aussi pour conséquence que *toutes les instances de services apparaissant dans l'organigramme sont adressables* car le sélectionneur doit communiquer une adresse au client pour chaque instance de service sélectionné. Cette recommandation convient bien aux services continus habitués aux interactions asynchrones et à l'adressage de leurs instances.

RAF7 *Les instances permettent le suivi des moments-clés de leur exécution comme la terminaison et le démarrage (lorsqu'il n'est pas immédiat).* Selon les technologies disponibles, des mécanismes de type publication-souscription ou de scrutation longue ou active peuvent être employés. Dans le cas d'un service REST-HTTP, une solution simple consiste à effectuer une scrutation longue auprès d'une ressource dédiée à l'état de l'instance. Cette recommandation est très utile aux

18. *Transport Layer Security*

19. C'est d'ailleurs le sélectionneur qui traduit la description fonctionnelle de la tâche en une requête de service suite à la sélection du service.

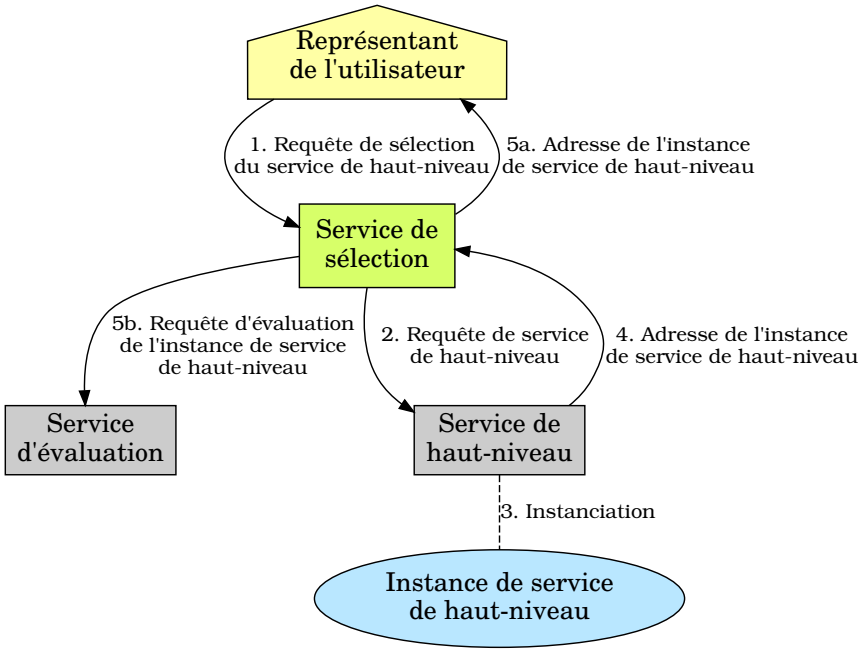


FIGURE 3.7 – *Instanciation du service de haut-niveau.* Le service de sélection sélectionne puis demande l'instanciation du service de haut-niveau (phases 2 à 4). À partir du modèle d'évaluation que lui a transmis le représentant de l'utilisateur dans sa requête (phase 1) et de l'adresse de l'instance de haut-niveau (phase 4), il fait appel à un service d'évaluation (5b). En parallèle, il renvoie cette adresse au client de haut-niveau (phase 5a). La suite de son interaction avec le service d'évaluation n'a pas besoin d'être davantage spécifiée.

services de mesure indépendants qui ont besoin de suivre l'évolution des services qu'ils observent. De même, les services composites utilisent ces mécanismes pour détecter les terminaisons de leurs sous-services.

RAF8 *Toutes les instances de services composites mettent à disposition une opération indiquant leurs instances de sous-services en cours d'exécution.* Cette fonctionnalité est utilisée par certains services de mesure (cf. sous-sous-section 3.3.2.1).

Recommandation optionnelle

RAO1 *Les services disponibles dans l'environnement peuvent être découverts dynamiquement.* Cette fonctionnalité est avant tout utile aux services de sélection car les clients et les autres services composites n'ont besoin de ne découvrir que des services de sélection ou de méta-sélection [20]. Il est important que le test de correspondance entre les descriptions de services et la description de la tâche considère d'éventuelles restrictions sur la cible requise (cf. sous-section 3.1.1). Il doit également s'assurer du support des propriétés non-fonctionnelles demandées.

20. Un service de méta-sélection sélectionne un service de sélection adaptée à une requête de service donnée. Il peut ainsi gérer la répartition des requêtes de services entre sélectionneurs en fonction des catégories de services de haut-niveau considérés.

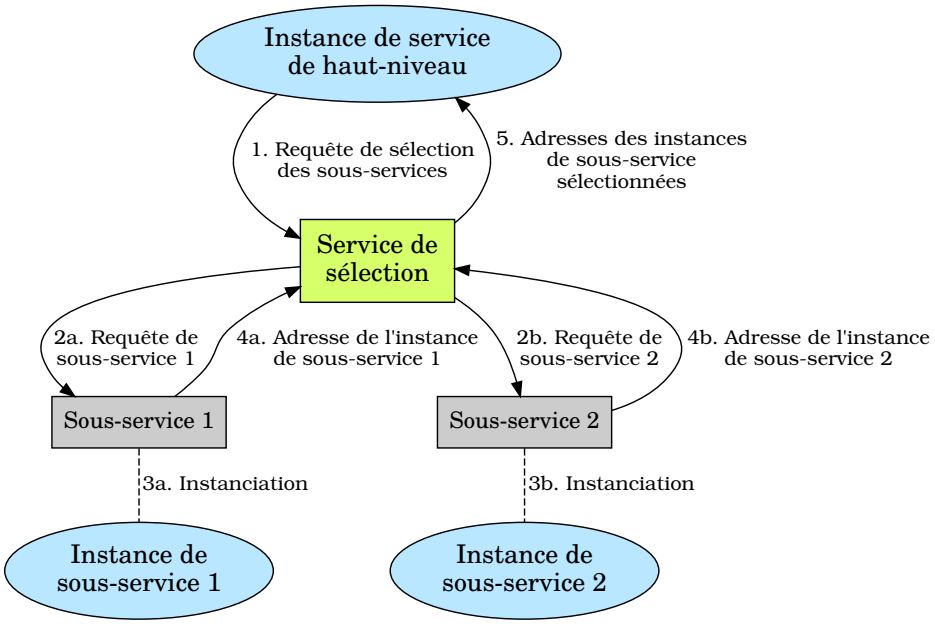


FIGURE 3.8 – *Instanciation des sous-services de l'instance de haut-niveau.* Les instanciations des deux sous-services se déroulent en parallèle des phases 2 à 4.

3.3.1.3 Propositions complémentaires

Cette avant-dernière sous-sous-section fournit quelques propositions utiles à l'implémentation de certaines recommandations et exigences architecturales.

Entêtes L'introduction de nouvelles entêtes HTTP ou SOAP dans les requêtes permet d'implémenter certaines recommandations. Nous proposons les entêtes suivantes :

`evaluation` Sa valeur désigne l'URI du modèle d'évaluation. Les détails de ce modèle sont alors disponibles en résolvant cette adresse. Cette entête est transmise par l'utilisateur ou son représentant lors de leur requête de sélection du service de haut-niveau. Les recommandations RAF1 et RAF6 rendent non nécessaire sa transmission aux services de l'organigramme.

`selection-service` Cette entête indique l'adresse d'un service de sélection [21] que devra utiliser l'instance de service créée si celle-ci s'avère être composite, conformément à la recommandation RAF1. Suivant la recommandation RAF6, cette entête est introduite par le sélectionneur.

`client-instance` Cette entête est une implémentation directe de la recommandation RAF5 exigeant des instances de services composites de l'organigramme qu'elles indiquent leur adresse lors d'une requête de sélection de services.

Observation du contexte Les observateurs de contexte ont pour mission de collecter les informations en lien avec un domaine d'intérêt précis. Dans notre exemple de déplacement de robot, l'observateur s'intéresse au contexte spatial : il se tient informé des dernières positions et orientations des objets physiques en s'abonnant aux flots des

21. Il s'agit de l'adresse d'un service et non de son instance. En effet, les instances des services des sélections ne sont généralement pas réifiées et n'ont donc pas d'adresse.

services de localisation. Pour se tenir informé des instanciations de services de localisation, il peut faire appel alternativement à des services fournis par, suivant les approches, les services de localisation, un sélectionneur ou un éventuel bus de service routant les messages.

3.3.1.4 Bilan

Au vu des recommandations fortes et des exigences architecturales qui viennent d'être établies, on constate que cette problématique de sélection de services exige la participation des services de l'organigramme. Cette participation est toutefois relativement limitée et peu intrusive. Le comportement attendu de la part des services se résume essentiellement aux mesures suivantes. Ces services doivent toujours s'adresser au même service de sélection, qui leur est communiqué par l'intermédiaire d'une entête dans le cas où nos propositions complémentaires seraient également adoptées. Ils font par ailleurs systématiquement appel à un service de sélection pour sélectionner leurs sous-services en indiquant leur adresse. Enfin, il leur est demandé de faciliter la tâche des services de mesure en exposant les détails de leur éventuelle composition. Comme nous allons maintenant le voir, cette dernière collaboration peut s'avérer très intéressante.

3.3.2 Évaluation des exécutions des instances de services

L'évaluation d'une exécution [22] consiste en trois étapes : la mesure de métriques durant l'exécution (sous-sous-section 3.3.2.1), leur normalisation en paramètres de QdS (3.3.2.2) et enfin leur agrégation en un score (3.3.2.3). Avant d'étudier chacune de ces étapes, le modèle spécifiant l'évaluation ainsi que le service qui lui est dédié sont présentés.

Modèle d'évaluation Comme mentionné lors de la proposition de l'entête `evaluation` (sous-sous-section 3.3.1.3), nous proposons que les modèles d'évaluation soient accessibles via des URI. Une représentation d'un tel modèle doit contenir les informations suivantes :

1. *Paramètres de Qualité de Service (QdS) considérés.* Un paramètre de QdS désigne une métrique et ainsi qu'une fonction permettant sa normalisation. Dans la mesure du possible, il est souhaitable que la formule analytique de cette fonction soit fournie dans ce document. À défaut, un service de normalisation ou une catégorie de service peut être désigné en remplacement. Par ailleurs, comme nous le verrons dans la sous-sous-section 3.3.2.2, il est fréquent d'avoir recours à des services pour déduire une valeur de référence à partir de la requête traitée par l'instance de service observée.
2. *Fonction d'agrégation en un score.* La fonction d'agrégation calcule un score à partir des valeurs des paramètres de QdS et parfois d'informations contextuelles complémentaires. Pour les raisons qui seront présentées dans la sous-sous-section 3.3.2.3, les fonctions d'agrégations considérées dans le cadre de cette thèse ne prendront pas en paramètre d'informations contextuelles complémentaires. La formulation analytique de cette fonction pourrait par ailleurs s'avérer très utile

22. À ne pas confondre avec la phase d'évaluation d'un service ou d'une combinaison de services lors de la sélection, qui a lieu avant l'exécution. Ces deux phases partagent néanmoins le même modèle d'évaluation.

dans le cas où le modèle d'évaluation serait considéré comme une entrée du modèle de prédiction de performance (cf. sous-sous-section 3.2.2.3). Ce cas de figure n'est cependant pas celui retenu dans notre approche qui pourra dès lors intégrer en toute transparence des fonctions de type boîte noire accessibles sous forme de services.

Service d'évaluation Un service d'évaluation a pour objet principal de traiter des requêtes d'évaluation indiquant l'adresse d'une instance de service en cours d'exécution ainsi que le modèle d'évaluation à considérer. Ce traitement instancie tout d'abord des services de mesure correspondant aux paramètres de QdS requis. Il se charge ensuite de leur normalisation et de leur agrégation en un score.

3.3.2.1 Mesures

La mesure des métriques associées aux paramètres de QdS est une pratique fréquente au sein des architectures orientées service. Cependant, les travaux existants ne considèrent qu'un nombre limité de métriques. Aussi nous mettrons en évidence les enjeux portés par de nouvelles métriques comme le bruit puis nous décrirons le processus de mesure de deux des quatre métriques que nous considérerons à l'avenir.

Approches existantes Les mesures réalisées dans la littérature de l'informatique orientée service sont dans l'essentiel effectuées par le client, le serveur ou un intermédiaire et portent sur les messages échangés. Les intermédiaires sont généralement des mandataires couplés à un moteur d'orchestration BPEL [Baresi & Guinea 2005] ou des services associés à un médiateur qui leur redirige des événements [Zeng et al. 2007]. Le serveur s'intéresse principalement à la charge de messages qu'il reçoit et qui viennent s'accumuler dans ses files d'attente [Garlan et al. 2004, Michlmayr et al. 2009]. Le client ou l'intermédiaire mesure le temps de réponse entre l'envoi d'une requête et la réception de la réponse [Michlmayr et al. 2009] ainsi que le débit exprimé comme le nombre de requêtes traitées par minute [Zeng et al. 2007]. Ils peuvent également analyser le contenu de ces messages pour vérifier des assertions [Baresi & Guinea 2005] à l'aide d'un analyseur comme CLiX [Nentwich et al. 2002] ; la métrique résultante est alors binaire.

Outre ces mesures effectuées sur des messages, les sondes clientes fournies par le framework Rainbow [Garlan et al. 2004] peuvent également mesurer l'état de sa liaison réseau avec un service de façon régulière en termes de bande passante et de latence. Il peut être noté que Rainbow est l'une des rares approches fournissant des sondes aussi bien aux clients qu'aux serveurs et qui les rendent accessibles sous forme de services.

Les mesures effectuées par un intermédiaire ou par le client sont généralement utilisées pour surveiller des accords de niveau de performance (SLA) décrits en WSLA [Keller & Ludwig 2003] ou en WS-Agreement [Andrieux et al. 2007]. Les mesures côté serveur peuvent servir d'informations contextuelles pour prédire le comportement du service ; nous verrons qu'elles peuvent également participer à l'obtention de métriques de haut-niveau.

Pourquoi de nouvelles métriques ? Ces précédentes approches, conçues initialement dans des perspectives d'accord de niveau de performance et d'administration de

systèmes d'information, ne répondent que partiellement aux enjeux des services robotiques. Ces approches sont suffisantes pour des *métriques mesurables à partir de messages* comme le temps de réponse d'un flux de valeurs ou la durée d'exécution [23] ou même le coût dans sa version simple. Cependant, d'autres métriques comme le bruit acoustique nécessitent l'utilisation de capteurs et portent leur attention sur le processus physique.

Coût Dans les SLA, le coût est fixé à l'avance par l'accord ; il est alors de la responsabilité du fournisseur du service de minimiser le coût de ses sous-services externes sans possibilité de répercuter un surcoût sur son client. Nous pensons qu'une métrique de coût ne doit pas être interprétée de la même façon en robotique ambiante. Le coût ne doit pas être compris dans des termes monétaires mais devrait être fonction de la rareté et de la consommation énergétique de certains dispositifs. Une façon simple de le mesurer consiste à calculer le coût en fonction du temps d'exécution [24] de chacune des instances de services de l'organigramme. Cependant, l'architecture nécessitée par des métriques telles que le bruit incite à concevoir la mesure de coût d'une instance de service composite elle-même comme une composition de services de mesure adressés aux instances de sous-service de l'instance de service observée [25]. Chaque service atomique de mesure du coût est alors libre de choisir une formulation plus complexe, incluant par exemple le niveau de la batterie. Néanmoins, nos services se limiteront à la première formulation afin de faciliter la prédiction de la valeur du coût.

Bruit acoustique Le bruit acoustique est une externalité négative survenant durant l'exécution de services robotiques et n'ayant pas d'influence sur les messages émis par ceux-ci. Sa mesure nécessite l'exécution synchronisée d'une instance de service de mesure utilisant des capteurs internes ou externes.

Métriques arbitraires D'une façon plus générale, de nombreuses autres métriques peuvent être envisagées pour ces services robotiques. Elles peuvent être conçues de toute pièce par un utilisateur expérimenté qui observe au quotidien le comportement de certains services et peut constater des difficultés ou des nuisances dont il suppose qu'elles pourraient être diminuées en effectuant une meilleure sélection de services. De telles métriques peuvent introduites en respectant les deux conditions suivantes : une métrique de service doit être mesurable et pouvoir être prédite au moment de la sélection.

Mise en œuvre Dans notre exemple du déplacement d'un robot, les métriques considérées sont le bruit moyen, la durée d'exécution et le coût total [Cogrel et al. 2011]. Pour reprendre la terminologie utilisée dans le projet Rainbow [Garlan et al. 2004], les mesures de performance d'une instance de service sont décomposées en sondes *(probes)* et en jauges *(gauges)*. Les sondes sont des captures de la performance à un instant donné

23. Bien qu'habituellement temps de réponse et durée d'exécution soient des mesures quasi-identiques (au temps de latence du réseau près), le cas des exécutions de services de localisation exigent que nous les distinguions.

24. Il pourrait être étendu au temps d'allocation mais il conviendrait alors de distinguer le temps d'attente du temps d'exécution. Par simplicité, nous nous limitons au temps d'exécution.

25. Dans un soucis de simplicité, nous ne comptabiliserons pas dans le coût les instances dédiées à la souscription à un flot d'information mais celles générant ce flot (cf. sous-section 3.1.2). Le coût engendré par la première souscription peut être considérée comme compris dans le coût de génération tandis que les coûts des souscriptions supplémentaires occasionnelles sont négligés.

tandis que les jauges sont leur agrégation sur l'ensemble de l'exécution. Soit Σ l'ensemble des instances de service et $\varsigma \in \Sigma$ une instance de service. Soit $s_k : \Sigma \times \mathbb{R}^+ \to \mathbb{R}$ et $g_k : \Sigma \to \mathbb{R}$ les fonctions respectivement associées aux sondes et aux jauges de la métrique k.

Les sondes d'une instance de service composite sont internes ou externes. Une sonde interne considère les composants et les instances de sous-service de l'instance de service mesurée tandis qu'une sonde externe observe uniquement le traitement réalisé. Par exemple, le temps de réponse est une mesure externe et la métrique de coût est interne. Une sonde interne agrège les mesures sur les instances de sous-service de l'instance mesurée avec les mesures locales sur cette dernière. Par exemple, la sonde de coût agrège de la façon suivante :

$$s_{co\hat{u}t}(\varsigma, t) = l_{co\hat{u}t}(\varsigma, t) + \sum_{i \in \mathrm{sub}(\varsigma, t)} s_{co\hat{u}t}(i, t) \tag{3.2}$$

$$l_{co\hat{u}t}(\varsigma, t) = c(\varsigma) \times \Delta t \tag{3.3}$$

où $s_{co\hat{u}t}(\varsigma, t)$ est la valeur du coût de l'instance de service ς associée à l'intervalle $[t-\Delta t, t]$, $l_{co\hat{u}t}(\varsigma, t)$ son coût local, $\mathrm{sub}(\varsigma, t)$ son sous-ensemble d'instances de sous-service, $c(\varsigma)$ son coût instantané et Δt la période entre deux mesures. La sonde de bruit est externe ou interne. Une sonde externe mesure le bruit résultant à partir de microphones tandis la sonde interne agrège les estimations locales des composants et des instances de sous-service bruyantes. Dans notre scénario, nous utilisons une sonde de bruit interne impliquant des estimateurs locaux pour les robots mobiles dont les valeurs sont proportionnelles aux estimations odométriques de translation et de rotation. Les autres périphériques, comme les réseaux de capteurs sans fil et les télémètres laser sont supposés ne pas émettre de bruit acoustique et ne fournissent donc pas d'estimateur. La sonde de bruit agrège les mesures locales à celles à des instances de sous-service comme suit :

$$s_{bruit}(\varsigma, t) = 10 \log_{10} \left(\exp\left(\frac{\ln 10}{10} \times l_{bruit}(\varsigma, t) \right) + \sum_{i \in sub(\varsigma, t)} \exp\left(\frac{\ln 10}{10} \times s_{bruit}(i, t) \right) \right) \tag{3.4}$$

où $s_{bruit}(\varsigma, t)$ est la valeur de la sonde de bruit de ς sur l'intervalle $[t - \Delta t, t]$ et $l_{bruit}(\varsigma, t)$ sa valeur locale.

En terme d'extension, de nouvelles sondes externes peuvent être fournies facilement en introduisant de nouveaux capteurs extérieurs. Pour les sondes internes, cette démarche est plus difficile à mettre en place par un utilisateur car elles nécessitent des capteurs locaux ou des estimateurs spécifiques aux services [26].

Les mesures issues des sondes sont collectées par les jauges qui les agrègent à la fin de l'exécution de l'instance de service ou à la demande. Les bruits et coûts moyens finaux de ces jauges, respectivement $g_{bruit}(\varsigma)$ et $g_{co\hat{u}t}(\varsigma)$ sont calculés de la façon suivante :

$$g_{bruit}(\varsigma) = \frac{1}{M} \sum_t s_{bruit}(\varsigma, t) \tag{3.5}$$

$$g_{co\hat{u}t}(\varsigma) = \frac{\sum_t s_{co\hat{u}t}(\varsigma, t)}{M * \Delta t} \tag{3.6}$$

26. Bon nombre d'estimateurs ont une relation étroite avec l'implémentation du service et doivent être déployés sur le même fournisseur de service.

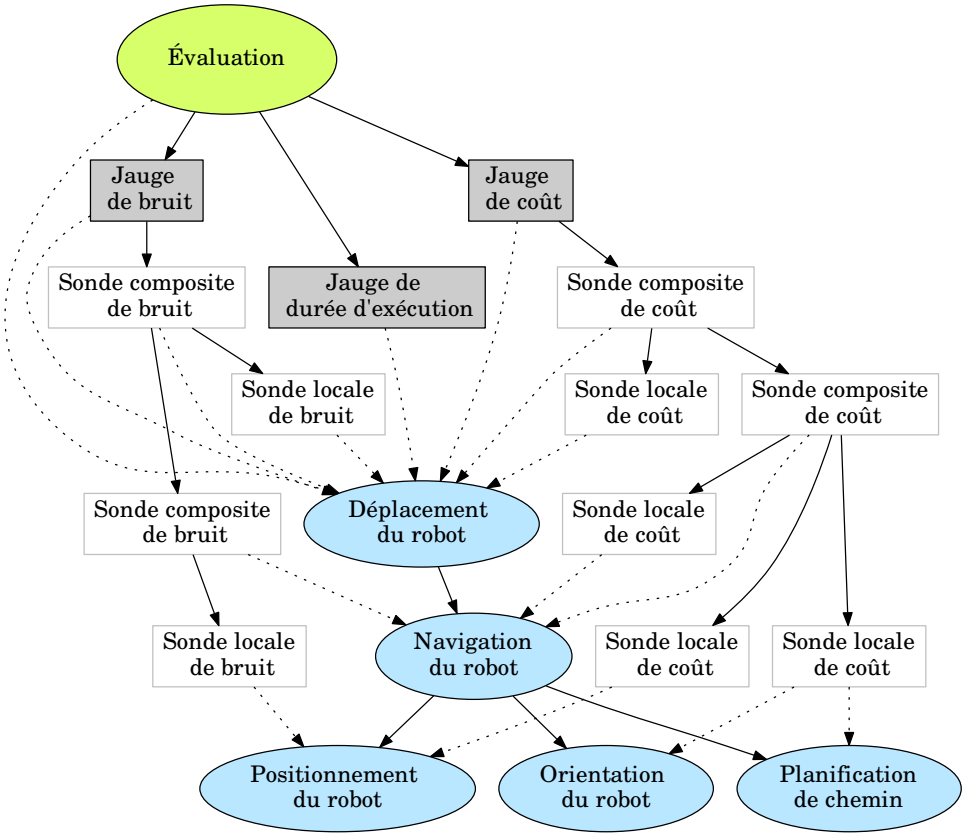


FIGURE 3.9 – *Services de mesure instanciés pour évaluer l'instance du service de déplacement.* Tous les nœuds de ce graphe sont des instances de service. Les instances de type souscription à un flot d'information ne sont pas représentées car elles ne sont pas considérées pour le coût ni pour le bruit acoustique. En conséquence, l'instance de service de planification n'apparaît pas comme composite dans cette figure.

où M est le nombre de mesures issues de la sonde ; ces mesures sont supposées avoir été effectuées à intervalle régulier.

Les sondes et les jauges sont implémentées comme des services de mesure. La figure 3.9 montre les services instanciés pour un service de déplacement. Certaines jauges requièrent les services de sondes sur l'instance de service de haut-niveau ; d'autres plus triviales comme la durée d'exécution n'en ont pas besoin. Pour chaque instance de service composite mesurée, une sonde composite est instanciée par métrique requérante et est en charge d'agréger les mesures issues des sondes des sous-services selon les formules 3.2 et 3.4. Si les instances de service de type jauge s'intéressent en premier lieu à l'ensemble de l'exécution, elles peuvent offrir également une nouvelle opération pour obtenir des valeurs sur des segments de l'exécution [27]. Cette seconde fonctionnalité sera également très utilisée dans le chapitre suivant pour augmenter le nombre d'exemples générés (voir la sous-sous-section 4.2.2.2).

27. Cette opération est rendue accessible non par l'interface du service mais par celle de l'instance de service.

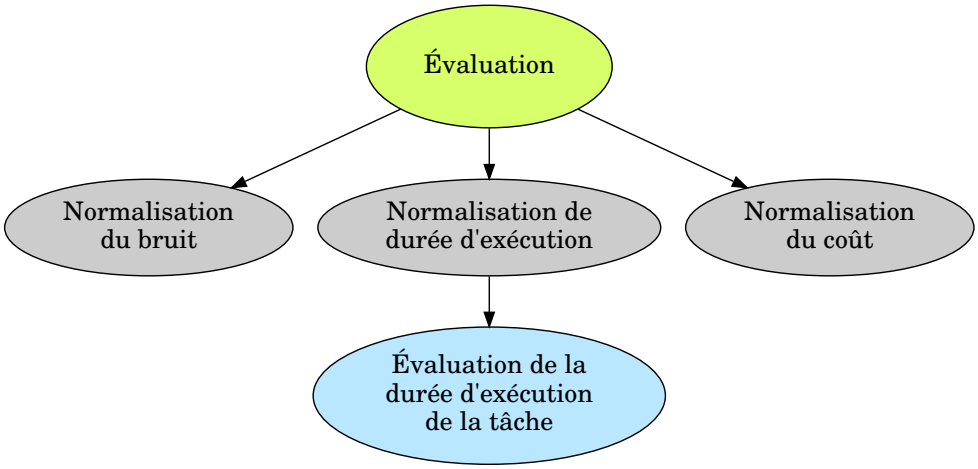


FIGURE 3.10 – Services de normalisation instanciés pour évaluer l'instance du service de déplacement

3.3.2.2 Normalisation

La normalisation des métriques en paramètres de QdS est une étape préalable à l'agrégation en un score ; elle n'est pas nécessaire dans une stricte perspective d'accord de niveau de performance où les valeurs doivent rester dans leur unité physique. Dans les propositions existantes, cette étape s'effectue à partir des valeurs extrêmes [Zeng et al. 2004, Ben Mabrouk et al. 2009] ou de l'écart-type et de la moyenne [Yu et al. 2007, Ben Mokhtar et al. 2008] d'un ensemble de valeurs connues des paramètres de QdS d'un type de service donné.

Notre approche se distingue des précédentes par les deux points suivants. Tout d'abord, les valeurs des métriques d'une même opération peuvent varier fortement selon la tâche requise : il est donc parfois nécessaire de prendre en compte cette tâche pour normaliser certaines métriques. À titre d'exemple, on ne peut pas attribuer la même valeur à deux exécutions de trente secondes où le robot parcourt deux mètres lors la première exécution et vingt lors de la deuxième. Enfin, nous souhaitons faire intervenir l'utilisateur dans le choix des fonctions de normalisation et des valeurs de référence.

Mise en œuvre Notre proposition de normalisation consiste en la projection de la valeur dans l'intervalle $[0, 1]$ en comparant la mesure à une valeur de référence. Cette valeur de référence est, selon la métrique, constante ou évaluée en fonction de la tâche requise. Tous les paramètres de QdS doivent être maximisés ; en conséquence, les mesures représentant des effets négatifs sont inversées. Soit ψ la tâche de haut-niveau. La valeur de référence $r_{durée}(\psi)$ du paramètre de QdS de durée d'exécution dépend de la tâche de déplacement ψ et est calculée de la façon suivante :

$$r_{durée}(\psi) = d_{min}(\psi) \times v_{max} \tag{3.7}$$

où $d_{min}(\psi)$ est le chemin le plus court déduit à partir de la destination de la tâche requise et de la position initiale de l'objet déplacé et v_{max} est la vitesse maximale pour ce type de service. Les valeurs de référence du coût moyen $r_{coût}$ et du bruit moyen r_{bruit} sont constantes pour ce type de service.

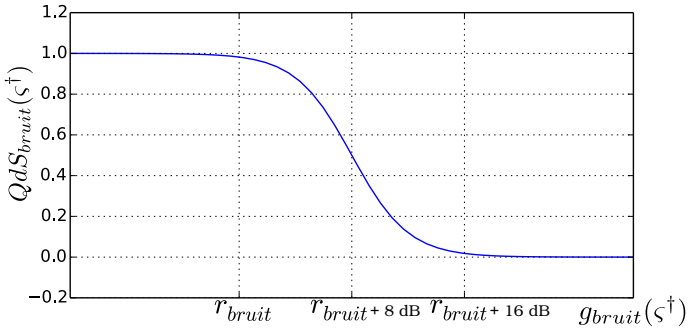


FIGURE 3.11 – Fonction de normalisation du bruit moyen $QdS_{bruit}(\varsigma^\dagger)$

Soit $\Sigma^\dagger \subseteq \Sigma$ l'ensemble des instances de service de haut-niveau et $QdS_k^{(\psi)} : \Sigma^\dagger \to [0,1]$ la fonction de normalisation de la métrique k à partir de la tâche de haut-niveau ψ. Les paramètres de QdS considérés dans notre scénario sont normalisés en utilisant diverses fonctions à base d'exponentielles :

$$\forall \alpha \in [0;1], f_1(\alpha) = \frac{e^\alpha - 1}{e - 1} \tag{3.8}$$

$$\forall \alpha \in \mathbb{R}, f_2(\alpha) = \frac{1}{1 + e^\alpha} \tag{3.9}$$

$$QdS_{coût}^{(\psi)}(\varsigma^\dagger) = f_1\left(\frac{r_{coût}}{g_{coût}(\varsigma^\dagger)}\right) \tag{3.10}$$

$$QdS_{durée}^{(\psi)}(\varsigma^\dagger) = f_1\left(\frac{r_{durée}}{g_{durée}(\varsigma^\dagger)}\right) \tag{3.11}$$

$$QdS_{bruit}(\varsigma^\dagger) = f_2\left(\frac{g_{bruit}(\varsigma^\dagger) - r_{bruit}}{2} - 4.0\right) \tag{3.12}$$

Comme le montre la figure 3.11, la normalisation de la métrique de bruit est effectuée à partir d'une sigmoïde, la fonction logistique. Les bruits moyens supérieurs de 16 dB par rapport à la valeur de référence ont une valeur normalisée proche de 0.

En choisissant une classe de service de normalisation, l'utilisateur assigne implicitement une fonction de normalisation et une procédure pour obtenir la valeur de référence. Lorsqu'elle est constante, la valeur de référence est intégrée dans le service de normalisation. Dans le cas contraire, un sous-service d'évaluation de la tâche requise est appelé. La figure 3.10 montre les services de normalisation invoqués par le service d'évaluation. Du point de vue de leur description, les services de normalisation ne sont pas spécifiques au service de haut-niveau : ils sont génériques. Ce sont leurs éventuels sous-services d'évaluation de la tâche requise qui en dépendent.

Notons que lorsque, comme ici, la fonction de normalisation est monotone alors il est possible de retrouver la valeur de la métrique à partir de celle du paramètre de QdS et de la tâche requise.

3.3.2.3 Fonction d'agrégation en un score

Le rôle de la fonction d'agrégation a été présenté dans le paragraphe 2.4.2.2 ; nous nous intéressons ici à sa possible relation au contexte. L'agrégation devant représenter

les attentes des utilisateurs, il est légitime que celles-ci puissent varier d'un contexte d'utilisation à un autre. À ce titre, les utilisateurs n'ont par exemple pas la même tolérance au bruit causé par des robots mobiles lorsqu'ils sont en train de lire que lorsqu'ils discutent avec leurs amis. Cette relation entre la fonction d'agrégation et le contexte d'utilisation peut être considérée de deux façons différentes :

1. *En passant les informations contextuelles en paramètres de la fonction d'agrégation.* Cette approche peut être vue comme une source de difficulté. Nous avons vu dans la sous-sous-section 3.2.2.2 que les modèles de performance sont dépendants du modèle d'évaluation. Si la fonction d'agrégation est par ailleurs surdéterminée par un contexte d'utilisation, alors ces informations contextuelles supplémentaires doivent être ajoutées à celles déjà considérées par le modèle de performance. Les sources d'information caractérisant le contexte d'utilisation risquent d'être très variées et ne peuvent pas être anticipées par le concepteur du service. On peut en effet attendre légitimement du concepteur du service de déplacement qu'il modélise le temps d'exécution de son service comme étant dépendant de sa trajectoire mais on ne peut pas exiger qu'il tienne compte de l'agenda de ses potentiels utilisateurs ni de la météo. La prise en compte de ce contexte d'utilisation nécessiterait une étape supplémentaire de pré-traitement qu'il est préférable d'éviter en adoptant l'approche suivante.
2. *En exigeant de l'utilisateur qu'il choisisse le modèle d'évaluation et par conséquent la fonction d'agrégation en fonction du contexte au moment de la spécification de la tâche requise.* L'utilisateur peut déléguer ce choix à son représentant. Les entrées de la fonction d'agrégation sont alors restreintes aux paramètres de QdS uniquement et il convient d'effectuer les hypothèses suivantes :
   (a) *Les informations contextuelles pertinentes pour la fonction d'agrégation sont supposées rester identiques entre l'instant de spécification de la tâche requise et la fin d'exécution de l'instance de service.* Ainsi, au lieu d'avoir une fonction d'agrégation générale capable de gérer de nombreux contextes, nous la divisons en des fonctions plus simples satisfaisant notre restriction.
   (b) *Le contexte a un impact discret et non continu sur la fonction d'agrégation.*

   Nous adoptons cette seconde approche.

Mise en œuvre La fonction d'agrégation $score_{\omega}^{(\psi)} : \Sigma^{\dagger} \times \{0,1\} \to [0,1]$, appelée aussi fonction de score, est traitée comme une boîte noire déterministe : nous ne faisons aucune hypothèse sur sa linéarité ou non-linéarité. Soit Ω l'ensemble des modèles d'évaluation et $\omega_1, \omega_2 \in \Omega$ les deux modèles considérés par notre scénario. Ces modèles d'évaluation utilisent les mêmes paramètres de QdS mais leurs fonctions d'agrégation se différencient par leur paramétrisation. Afin d'encourager le rejet ($\varpi = 1$) d'une exécution ayant une forte probabilité d'échouer ou d'avoir une très mauvaise performance, un score légèrement supérieur au bonus de succès lui est associé.

Le score est calculé comme la somme pondérée sur K paramètres de QdS en cas de succès :

$$score_{\omega}^{(\psi)}(\varsigma^{\dagger}, \varpi) = \begin{cases} \alpha_{suc} + (1 - \alpha_{suc}) \sum\limits_{k \leqslant K} w_k^{(\omega)} \times QdS_k^{(\psi)}(\varsigma^{\dagger}) & \text{si } \varpi = 0 \\ \alpha_{rej} & \text{si } \varpi = 1 \\ 0 & \text{sinon} \end{cases} \tag{3.13}$$

avec $\alpha_{suc} = 0.1$ et $\alpha_{rej} = 1.1\alpha_{suc}$.

Présence d'un humain	w_{bruit}	$w_{durée}$	$w_{coût}$
`Non` (ω_1)	0.1	0.5	0.4
`Oui` (ω_2)	0.6	0.3	0.1

TABLE 3.2 – Poids des paramètres de QdS

Nous distinguons deux contextes selon la présence ou l'absence d'humains dans les pièces se trouvant sur le chemin du déplacement de robot. La table 3.2 donne les poids des paramètres de QdS associés à chacun de ces contextes.

3.4 Conclusion

En portant son intérêt sur la modélisation de fonctionnalités robotiques sous forme de composition de services, ce chapitre a été amené à étudier trois axes sous des angles qui avaient été jusqu'alors peu investis dans la littérature consacrée aux services logiciels. Nos contributions et nos principaux choix dans chacun de ces axes sont les suivants :

1. *Les enjeux fonctionnels de la coordination étroite portés par les configurations de composants [Lundh et al. 2008] ont été adaptés aux architectures orientées service.* Cette adaptation a donné lieu à la distinction entre les services réalisant des actions et les services continus ainsi qu'à la caractérisation d'une nouvelle forme de composition de services.
2. *La procédure de sélection de services a été revue pour faire face aux défis posés par la prédiction de performance de ces services robotiques.* Plus précisément, nous avons mis en avant les points suivants :
   (a) *Constat du couplage non-fonctionnel entre services au sein de l'organigramme.* Face à ce constat, la modélisation des relations de dépendance entre les performances des sous-services et des composants représentant les traitements spécifiques effectués par l'instance de service composite apparaît comme une perspective importante pour affronter le problème combinatoire inhérent à la sélection globale. Cependant, nous avons décidé de traiter a minima cette problématique combinatoire pour nous concentrer sur la prédiction directe de performance du traitement de haut-niveau. En ce sens, notre scénario ne propose qu'un ensemble restreint d'organigrammes possibles permettant ainsi son énumération. Nos modèles de performance sont directement associés à ces combinaisons et l'organigramme est sélectionné en une seule étape. L'objet de cette thèse est de mettre en évidence les opportunités offertes par la sélection de services dans le cadre de la coordination étroite en matière de performance et de motiver ainsi une étude approfondie ultérieure de cette problématique combinatoire.
   (b) *Justification d'une approche spatio-contextuelle pour la prédiction de performance du déplacement d'un robot mobile et caractérisation du contexte considéré.* À la différence des approches existantes mentionnées, la prédiction de performance de ce type de service ne peut pas reposer sur des séries temporelles ni sur une modélisation à base de files d'attente ; nous avons donc pro-

posé d'obtenir ces modèles par apprentissage en leur donnant des informations spatio-contextuelles en entrée. Les informations contextuelles pertinentes, spécifiques à ce type de service de haut-niveau, sont en effet de nature métrique. L'approche métrique est préférée à une approche topologique comme celle de [Morisset & Ghallab 2008] car elle offre une meilleure généricité vis-à-vis des paramètres de QdS considérés et des idiosyncrasies de l'environnement.

3. *Nous avons dressé une liste d'exigences et de recommandations architecturales puis avons décrit l'intégralité du processus d'évaluation d'une exécution d'une instance de service de haut-niveau.* Ces deux points mettent en évidence les aspects architecturaux nécessaires à la mise en œuvre et à la valorisation d'une sélection de services robotiques composites.

Le chapitre suivant peut désormais se consacrer à la modélisation effective de la performance des services de déplacement de robots mobiles et à son intégration au cœur d'une problématique de remplacement de sous-services.

Chapitre 4

Prédiction de performance dans le cadre du possible remplacement de sous-services continus

Sommaire

L'étude menée par ce chapitre est consacrée à la prédiction de performance d'un organigramme lors du traitement d'une requête de haut-niveau. Pour cela, elle s'appuie sur l'exemple d'un service de déplacement de robot. Cet exemple présente un double intérêt pour l'étude des compositions à coordination étroite : (i) il exige une prédiction de performance contextuelle et (ii) nécessite régulièrement le remplacement de ses sous-services continus. Ces deux aspects sont fortement couplés car certaines instances de sous-services peuvent échouer en cours d'exécution ou plus généralement voir leurs performances évoluer.

Par manque de connaissance préalable, la modélisation de la performance d'un tel service de haut-niveau doit faire l'objet d'un apprentissage sur la base des métriques

collectées au fur et à mesure des exécutions et des contextes associés. Cet apprentissage est ici confronté à un problème de dimensionnalité. En effet, il est particulièrement difficile pour un robot mobile d'explorer de façon suffisante l'espace $\mathbb{R}^4$ des possibles positions initiales et destinations correspondant à une exécution complète, sans compter la position courante, les autres dimensions apportées par les spécificités de certains sous-services et la multiplicité des modèles d'évaluation. À cet égard, l'exécution du service de déplacement de robot doit être considérée comme une ressource rare et il convient d'étudier plus finement son comportement. En ce sens, nous proposons deux mesures pour dépasser cette difficulté :

1. *Solliciter le sous-service de planification de chemin avant l'évaluation de la combinaison afin de connaître le chemin suivi.*
2. *Segmenter les métriques pour permettre l'apprentissage de sous-modèles.*

Lorsque le contexte évolue en cours d'exécution, les performances de certains sous-services continus ainsi que leurs probabilités d'échec varient. Les services de positionnement en sont l'exemple-type de part leurs portées limitées et leurs performances très variables. Cette perspective rend nécessaire et opportune la prise en considération de leur remplacement. Elle invite ainsi à considérer la sélection de services comme une prise de décision séquentielle. Le remplacement suite à un échec ainsi que son extension opportuniste sont étudiés dans ce chapitre.

Un processus sélectionnant le service de déplacement ainsi que de l'ensemble de ses sous-services puis effectuant d'éventuels remplacements en cours d'exécution peut être formalisé comme un processus de décision markovien (*Markov Decision Process, MDP*). Nous en proposons deux variantes en distinguant deux problèmes de remplacement. Leurs résolutions s'appuient sur une approche d'apprentissage par renforcement modélisant la fonction de transition à partir de sous-modèles. Les valeurs sont ensuite obtenues en effectuant une fouille stochastique d'arbre *(Monte-Carlo Tree Search, MCTS).*

Plan du chapitre Ce chapitre est organisé de la façon suivante.

Dans un premier temps, la section 4.1 introduit la notion de processus de décision markovien à horizon fini (sous-section 4.1.1) puis l'emploie pour formaliser les problèmes associés aux deux types de remplacement considérés : suite à un échec (sous-section 4.1.2) ou de façon opportuniste (4.1.3). À cette occasion, elle motive la modélisation de la fonction de transition en sous-modèles afin de contourner le problème de dimensionnalité auquel sont directement confrontés un apprentissage direct de la fonction de transition et une approche d'apprentissage par renforcement sans modèle.

La section 4.2 est consacrée à la modélisation de la fonction de transition. Cette modélisation distingue deux phases : la préparation des nouveaux services sélectionnés (4.2.1) et le déplacement sur plusieurs segments (4.2.2). Elle renvoie un ensemble fini d'états successeurs auxquels elle associe des probabilités.

À partir du modèle de transition et de la fonction de renforcement immédiat qui est elle connue sous une forme analytique, la section 4.3 s'emploie à estimer la valeur associée à chaque action dans l'état courant afin que la sélection alors effectuée soit quasi-optimale. Cette estimation est réalisée par planification à partir d'un arbre de recherche. Après la caractérisation de la taille de cet arbre (4.3.1), sa fouille est accomplie de façon stochastique à partir de la méthode *Upper Confidence bounds applied to Trees* (UCT) (4.3.2).

Les diverses propositions de ce chapitre sont enfin évaluées au travers de simulations dans la section 4.4.

4.1 Formalisations des problèmes de remplacement

Dans ce chapitre, deux problèmes de remplacement sont considérés. Le premier se limite au remplacement de sous-services venant d'échouer. Le second dépasse cette limitation en pouvant remplacer des sous-services n'ayant pas échoué et en étant effectué de façon opportuniste.

De façon générale, le remplacement de sous-services invite à concevoir la sélection de services comme un processus de décision séquentiel. La théorie la plus couramment utilisée pour ce type de problème est celle des processus de décision markoviens. La sous-section 4.1.1 en présente un succinct aperçu puis les sections suivantes 4.1.2 et 4.1.3 l'emploient pour formaliser les problèmes correspondant à ces deux cas de remplacement. Ces formalisations contiennent à la fois des éléments génériques et spécifiques au cas du service de déplacement d'un robot.

4.1.1 Introduction aux processus de décision markoviens

Le processus de décision markovien *(Markov Decision Process, MDP)* [Bellman 1957] est un formalisme couramment utilisé pour résoudre des problèmes de décision dans l'incertain dans lesquels l'incertitude provient des résultats d'une action sélectionnée et des prises de décisions ultérieures. Ce formalisme n'est toutefois applicable que lorsque la propriété de Markov est respectée. Il en existe de multiples variantes ; nous nous focalisons ici sur le cas du MDP à horizon fini qui sera utilisé dans les formalisations à venir.

4.1.1.1 Formalisme

Un processus de décision markovien (MDP) à horizon fini est défini comme le quintuplet $\left(\mathcal{S}, \{\mathcal{A}_s\}, \{P_s^{(t)}\}, \{r_s^{(t)}\}, T\right)$ avec :

- $\mathcal{S}$. *Espace des états.* Celui-ci peut être composé d'états discrets ou continus.
- $\mathcal{A}_s$. *Ensemble des actions disponibles dans l'état s.* Les actions peuvent également être discrètes ou continues ; toutefois, seul le cas discret sera considéré dans la suite de ce mémoire. L'ensemble de toutes les actions est $\mathcal{A} = \bigcup_{s\in\mathcal{S}} \mathcal{A}_s$.
- $P_s^{(t)} : \mathcal{A}_s \times \mathcal{S} \to [0,1]$. *Fonction de transition d'un état s à un autre état s' en choisissant l'action a à l'instant t.* L'action a et l'état *successeur* s' sont passés en arguments. Cette fonction est ici exprimée sous une forme probabiliste. Elle caractérise la dynamique du système.
- $r_s^{(t)} : \mathcal{A}_s \times \mathcal{S} \to \mathbb{R}$. *Fonction de renforcement immédiat suite au choix de l'action a dans un état s et à la transition vers un autre état s' à l'instant t.* Comme précédemment, l'action a et l'état s' sont passés en arguments.
- $T \in \mathbb{N}$. *Horizon.* Dans ce type de MDP, le nombre de transitions est limité à T.

La propriété de Markov, dans le cadre d'un MDP, stipule que la connaissance de l'état courant et d'une action choisie au même instant est suffisante pour prédire les distributions de l'état suivant et du renforcement immédiatement reçu. Formellement, cette propriété s'exprime comme une indépendance conditionnelle :

$$p(s_{t+1}, r_{t+1}|s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, ..., s_0, a_0) = p(s_{t+1}, r_{t+1}|s_t, a_t) \tag{4.1}$$

où s_t, a_t et r_t représentent respectivement l'état, l'action et le renforcement à l'instant t. Grâce à cette propriété, une décision ne tient compte que de l'état courant et des actions disponibles ; elle n'a donc pas besoin de connaître davantage son passé. L'état est par ailleurs supposé être entièrement observable [1].

En fonction du problème considéré, les fonctions des transitions et de fonction de renforcement immédiat peuvent être stochastiques ou déterministes, ce qui entraîne à l'occasion quelques variations dans leurs formulations.

4.1.1.2 Politiques et fonctions de valeurs

L'objectif de ce type d'approche est d'obtenir un contrôleur qui sache choisir de façon optimale ou quasi-optimale une action dans chaque état auquel il pourrait être confronté. Le comportement du contrôleur peut être formalisé comme une politique non-stationnaire $\pi = [\pi^{(0)}, ..., \pi^{(T-1)}]$ qui se décompose en sous-politiques pour chaque pas de temps. Dans le cas d'une politique déterministe, une sous-politique est, à l'instant t, $\pi^{(t)} : \mathcal{S} \to \mathcal{A}$. Elle choisit toujours la même action $\pi^{(t)}(s)$ dans un état s donné à l'instant t et doit vérifier la contrainte

$$\forall s \in \mathcal{S}, \pi^{(t)}(s) \in \mathcal{A}_s. \tag{4.2}$$

Lorsqu'une telle politique est suivie, il est possible d'associer une fonction de valeur $V_\pi^{(t)} : \mathcal{S} \to \mathbb{R}$ associant une valeur à chacun des états. Cette valeur représente *l'espérance de gain à horizon T* :

$$V_\pi^{(t)}(s) = \mathbb{E}\left[R_t|s_t = s\right] = \mathbb{E}\left[\sum_{i=t}^{T-1} \hat{r}_{i+1}|s_t = s, \pi\right] \tag{4.3}$$

où $\hat{r}_{i+1} = r_{s_i}^{(i)}(\pi^{(i)}(s_i), s_{i+1})$ et le gain à l'horizon T depuis l'état s_t est

$$R_t = \sum_{i=t}^{T-1} \hat{r}_{i+1}. \tag{4.4}$$

Elle peut aussi être écrite de façon récursive :

$$V_\pi^{(t)}(s) = \sum_{s'} P_s^{(t)}(\pi^{(t)}(s), s') \left[r_s^{(t)}(\pi^{(t)}(s), s') + V_\pi^{(t+1)}(s')\right] \tag{4.5}$$

Il est également courant d'utiliser une fonction de valeur directement associée au couple état-action (s, a) pour chaque pas de temps t, $Q_s^{(t)} : \mathcal{A}_s \to \mathbb{R}$. Elle est proche de

1. Dans le cas contraire, il est courant d'avoir recours à une extension des MDP, les processus de décision markoviens partiellement observables *(Partially Observable MDP, POMDP)*.

la fonction $V^{(t)}$ et peut être formulée ainsi lorsqu'elle suit la politique π :

$$Q^{(t)}_{s,\pi}(a) = \sum_{s'} P^{(t)}_s(a,s') \left[r^{(t)}_s(a,s') + V^{(t+1)}_\pi(s') \right] \tag{4.6}$$

$$= \sum_{s'} P^{(t)}_s(a,s') \left[r^{(t)}_s(a,s') + Q^{(t+1)}_{s',\pi} \left(\pi^{(t+1)}(s') \right) \right] \tag{4.7}$$

Politique optimale La résolution du MDP vise une politique (quasi-)optimale $\pi^* = [\pi^{*(0)}, ..., \pi^{*(T-1)}]$ qui maximise l'espérance de gain à horizon T. Lorsque les ensembles d'états et d'actions du MDP sont discrets et que leurs tailles tout comme l'horizon T sont modérées, le MDP peut être résolu de façon exacte à partir de l'algorithme d'itération de valeur à horizon fini *(finite-horizon value iteration)*.

Tout d'abord, les valeurs $V^{*(t)}(s)$ optimales sont calculées de façon récursive.

Algorithme 1 : Itération de valeur pour un MDP à horizon fini

$\forall s \in S, V^{*(T)}(s) \leftarrow 0$;
$t \leftarrow T-1$;
tant que $t \geq 0$ **faire**
  $\forall s \in S, V^{*(t)}(s) = \max_{a \in \mathcal{A}_s} \sum_{s'} P^{(t)}_s(a,s') \left[r^{(t)}_s(a,s') + V^{*(t+1)}(s') \right]$;
  $t \leftarrow t-1$;

Enfin, on en déduit les sous-politiques optimales.

$$\forall t < T, \forall s \in \mathcal{S}, \pi^{*(t)}(s) = \arg\max_{a \in \mathcal{A}_s} \sum_{s'} P^{(t)}_s(a,s') \left[r^{(t)}_s(a,s') + V^{*(t+1)}(s') \right] \tag{4.8}$$

Lorsque les fonctions de renforcement immédiat et de transition sont entièrement définies, la résolution d'un MDP peut être effectuée de façon hors-ligne, sans besoin d'interaction avec l'environnement. Ce cas de figure est typique de la *programmation dynamique*. Lorsque les fonctions de transition ou de renforcement immédiat ne sont pas connues, l'interaction avec l'environnement devient indispensable : on parle alors d'*apprentissage par renforcement*. Le lecteur intéressé par une introduction à ce domaine est invité à consulter les ouvrages [Sutton & Barto 1998] et [Szepesvari 2010].

4.1.2 Remplacement suite à un échec

Dans un premier temps, on s'intéresse au cas de remplacements survenant en réaction aux échecs d'instances de service. Un tel remplacement est effectué dans la mesure où cet échec n'est pas fatal pour l'instance de service supérieure [2] et que l'horizon T du nombre maximal de sélection n'a pas été atteint. Dans ce cas de figure, l'instance de service supérieure se charge alors d'effectuer une nouvelle requête de sélection de services auprès du sélectionneur. Seule l'instance de sous-service ayant échoué est remplacée. Suivant l'exemple considéré dans ce chapitre, l'échec concerne en premier lieu les services de positionnement.

2. C'est généralement le cas pour des services de type continu. Ceux-ci ont souvent pour rôle de générer des flots réguliers d'information qui seront reçus pour l'instance de service supérieure avant leur échec. Si cela n'avait pas été le cas, la perte de ces informations associées à leur période d'activité aurait pu être fatale pour leurs instances de service supérieures.

Pour bien prendre la mesure des particularités de ce problème de sélection de services, cette sous-section commence par en proposer une première formalisation générale (4.1.2.1). Le MDP résultant est défini sur l'ensemble des situations possibles mais ses états ont une trop grande dimensionnalité pour qu'une fonction de valeur Q puisse être apprise directement à partir des exécutions. En tenant compte plus finement de sa structure, une seconde formalisation est ensuite proposée (4.1.2.2). La dimensionnalité de ses états est réduite mais le MDP est cependant devenu spécifique à la tâche de haut-niveau et au modèle d'évaluation.

4.1.2.1 Formalisme général

Processus de décision markovien général Le MDP à horizon fini associé à ce problème de sélection est le suivant :

1. *État à l'instant t.*
   $s_t = [x_t, y_t, x_{ini}, y_{ini}, x_{dest}, y_{dest}, x_{as,t}^{(1)}, y_{as,t}^{(1)}, \ldots, x_{as,t}^{(L)}, y_{as,t}^{(L)}, m_t^{(1)}, ..., m_t^{(K)}, b_t, \omega]^\intercal \in \mathcal{S}$
   (a) $x_t, y_t \in \mathbb{R}$. Coordonnées cartésiennes connues de la position courante.
   (b) $x_{ini}, y_{ini} \in \mathbb{R}$. Coordonnées cartésiennes connues de la position initiale.
   (c) $x_{dest}, y_{dest} \in \mathbb{R}$. Coordonnées cartésiennes de la destination.
   (d) $x_{as,t}^{(1)}, y_{as,t}^{(1)}, \ldots, x_{as,t}^{(L)}, y_{as,t}^{(L)} \in \mathbb{R}$. Positions courantes des L robots assistants fournissant un service de positionnement.
   (e) $m_t^{(1)}, ..., m_t^{(K)} \in \mathbb{R}^+$. Valeurs des métriques 1 à K à l'instant t. Les métriques ici considérées sont positives.
   (f) $b_t \in \mathcal{B}$. Combinaison de services en cours d'exécution. $b_0 = \varnothing$. Cette combinaison correspond à l'ensemble de sous-services précédemment sélectionnés n'ayant pas échoué.
   (g) $\omega \in \Omega$. Modèle d'évaluation. L'ensemble Ω est discret.
   Certaines composantes de l'état sont continues. Il existe aussi un état spécial $s_\varnothing$ d'échec prématuré correspondant à un rejet ou à un échec fatal.
2. *Espace des actions $\mathcal{A}_s$ disponibles dans l'état s.* À l'exception de l'action spéciale, une action représente une combinaison des services correspondant à un organigramme. Cet espace fini comporte également l'action spéciale $\varnothing$ qui dénote un rejet de sélection. Cette action n'est disponible que dans l'état initial. L'ensemble de toutes les actions est $\mathcal{A} = \bigcup_{s \in \mathcal{S}} \mathcal{A}_s$.
   Soit $\mathcal{Z}$ l'ensemble des paires associant un service à une position dans un organigramme ; par abus de langage, nous appelons $z \in \mathcal{Z}$ un service. Soit les fonctions d'appartenance $c^\dagger : \mathcal{A} \times \mathcal{Z} \to \{0; 1\}$ et $c^\ddagger : \mathcal{B} \times \mathcal{Z} \to \{0; 1\}$ entre les services et respectivement les actions et les combinaisons en cours d'exécution. L'ensemble des actions disponibles dans l'état s_t est :
   $$\mathcal{A}_{s_t} = \begin{cases} \mathcal{A} & \text{si } b_t = \varnothing \\ \left\{ a \in \mathcal{A} | \{ z \in \mathcal{Z} | c^\ddagger(b_t, z) = 1 \} \subset \{ z \in \mathcal{Z} | c^\dagger(a, z) = 1 \} \right\} & \text{sinon} \end{cases} \tag{4.9}$$
3. *Fonctions de renforcement immédiat.* $r_s : \mathcal{A}_s \times \mathcal{S} \to \mathbb{R}$
   $$r_s(a, s') = \begin{cases} score(s', 0) & \text{si } a \neq \varnothing,\ s' \text{ est terminal et } s \text{ ne l'est pas} \\ score(s', 1) & \text{si } a = \varnothing \text{ et } s \text{ n'est pas terminal} \\ 0 & \text{sinon} \end{cases} \tag{4.10}$$

La fonction de score $score : \mathcal{S} \times \{0,1\} \to [0,1]$ est ici une légère adaptation de la fonction définie dans l'équation (3.13) où l'état s' informe de la performance de l'instance de service $\varsigma^\dagger$, de la tâche de haut-niveau ψ et du modèle d'évaluation ω. Elle récompense également le rejet de la sélection lorsque l'échec de l'exécution est trop fortement probable quelque soit la combinaison de sous-services choisie. Ces fonctions de renforcement sont donc entièrement spécifiées ; elles n'ont donc pas besoin d'être apprises. Elles sont par ailleurs déterministes. Ces fonctions restent les mêmes pour tout $t < T$ durant le même épisode ; aussi leurs exposants ont été omis.

4. *Fonctions de transition.* $P_s : \mathcal{A}_s \times \mathcal{S} \to [0,1]$. Les transitions vont directement vers des états finaux en cas de succès ou vers des états intermédiaires en cas d'échec d'une instance de sous-service qui n'est pas fatal pour l'instance de service de haut-niveau. La fonction de transition est supposée stochastique et n'est pas connue a priori. La seule transition connue est le choix de l'action $\varnothing$ qui conduit vers un état terminal d'échec prématuré. De même, ces fonctions ne changent pas au cours du même épisode.
5. *Horizon.* $T = 10$. Cet horizon fournit une borne supérieure au nombre de sélections.

Le MDP est ici *épisodique* car il dispose d'états terminaux. Un épisode correspond à une exécution du service de haut-niveau et consiste en au plus T transitions.

Description de l'état Pour satisfaire la propriété de Markov, on constate que l'état a une dimensionnalité importante. Les composantes de l'état peuvent être regroupées de diverses façons :

1. *Composantes spécifiques au déplacement du robot.* Les coordonnées cartésiennes associées à la destination et aux positions initiale et courante lui sont spécifiques.
2. *Composantes spécifiques à certains sous-services de positionnement.* Pour chacun des services de positionnement impliquant un robot assistant, il est nécessaire de prendre la position courante de ce dernier. Dans un souci de réduction de la dimension de l'espace d'état, il aurait été intéressant de ne considérer que la position du robot le plus proche. Néanmoins, il n'est pas certain que ce robot soit le plus opportun car d'autres robots pourraient être plus rapides et avoir de meilleures performances.
3. *Composantes génériques.* L'apparition des métriques, de la combinaison de services en cours d'exécution et du modèle d'évaluation dans l'état est commune aux MDPs associés à des services de haut-niveau autorisant le remplacement de certains sous-services de leur organigramme en cas d'échec.
4. *Composantes dépendantes de la tâche de haut-niveau.* La position initiale et la destination sont liées à la tâche de haut-niveau et sont nécessaires à la fonction de renforcement immédiat. En effet, afin que celle-ci puisse calculer le score, elle doit pouvoir normaliser les métriques selon ces informations (cf. sous-sous-section 3.3.2.2).
5. *Composantes évoluant durant un épisode.* Seules les métriques, la combinaison de services en cours d'exécution et les positions courantes du robot déplacé et des robots assistants évoluent entre les étapes d'un épisode. Les autres ne peuvent changer que d'un épisode à un autre.

Rareté des épisodes relativement à la dimensionnalité de l'espace d'états Couplée à la dimensionnalité de l'espace d'états, la rareté des épisodes représente la principale difficulté dans la résolution de ce MDP. Cette rareté est due à la nature du service de haut-niveau ici considéré : une exécution est souvent longue comparée à celle de la plupart des services d'information (elle peut atteindre plusieurs minutes) et elle n'est qu'occasionnelle. En supposant que les positions initiales et les destinations soient générées uniformément de façon aléatoire, les exemples seront très dispersés selon ces quatre dimensions. De plus, de multiples autres dimensions viennent s'ajouter à la représentation de l'état, ce qui accentue encore cette dispersion. Cette faible densité en exemples rend très difficile deux approches courantes de l'apprentissage par renforcement :

1. *Approche sans modèle.* L'apprentissage direct (supervisé) d'une fonction de valeur Q à partir des exécutions est directement confronté à ce problème de dimensionnalité.
2. *Apprentissage direct des fonctions de transition P_s.* Cet apprentissage est tout aussi exposé à ce problème.

Nous proposons d'aborder ce problème autrement, en combinant les points suivants :

1. *En rendant le MDP spécifique à la tâche de haut-niveau et au modèle d'évaluation.* La dimensionnalité des états peut alors être réduite.
2. *En modélisant plus finement les fonctions de transition.* Pour cela, on s'intéresse aux spécificités du comportement du service de haut-niveau. Le chemin suivi par le robot durant une exécution est planifié par un des sous-services du service de navigation ; cette information peut être obtenue au moment de la sélection [3]. L'usage d'un chemin vers la destination avait également été adopté dans [Morisset & Ghallab 2008] afin de ne pas introduire la destination dans la représentation de l'espace d'états de leur MDP. Le couplage de cette information avec la segmentation des métriques rend possible l'apprentissage et la prédiction de dynamiques locales [4] sur les segments de ce chemin. Ces segments ont pour principal avantage de pouvoir être partagés entre de nombreuses exécutions : leurs dynamiques peuvent donc être apprises plus rapidement que celles des trajectoires complètes. D'autres aspects peuvent par ailleurs être modélisés séparément comme le temps d'arrivée d'un robot assistant à proximité du robot devant être positionné. Au côté des prédictions de dynamiques locales effectuées sur des segments, ils fournissent les éléments de base de la modélisation des fonctions de transition. Cette dernière fera l'objet de la section 4.2.

4.1.2.2 MDP spécifique à la requête de haut-niveau et au modèle d'évaluation

En partant du constat que seules les métriques et la position courante évoluent entre les étapes d'un épisode, nous proposons un second MDP dont l'état se restreint à ces

3. Cela est possible dès lors que chacun des services candidats de planification de chemin dispose d'une opération de type action fournissant le plan comme résultat. Ces requêtes ont lieu au moment de la sélection, avant l'évaluation de chacune des combinaisons. Les implémentations de ces services sont ensuite libres de garder ce résultat en cache et de l'utiliser pour délivrer un traitement de type continu.

4. La dynamique concerne le changement d'état et comporte donc entre autres des informations relatives aux métriques.

composantes. Dès lors, ce MDP devient spécifique à la tâche de haut-niveau $\psi \in \Psi$ et au modèle d'évaluation $\omega \in \Omega$ afin que la propriété de Markov puisse être respectée.

Ce nouveau processus de décision markovien est défini comme suit :

1. *État à l'instant t.* $s_t = [x_t, y_t, x_{as,t}^{(1)}, y_{as,t}^{(1)}, \ldots, x_{as,t}^{(L)}, y_{as,t}^{(L)}, m_t^{(1)}, ..., m_t^{(K)}]^\intercal \in \mathcal{S}$
   (a) $x_t, y_t \in \mathbb{R}$. Coordonnées cartésiennes connues de la position courante.
   (b) $x_{as,t}^{(1)}, y_{as,t}^{(1)}, \ldots, x_{as,t}^{(L)}, y_{as,t}^{(L)} \in \mathbb{R}$. Positions courantes des L robots assistants disponibles fournissant un service de positionnement.
   (c) $m_t^{(1)}, ..., m_t^{(K)} \in \mathbb{R}^+$. Valeurs des métriques 1 à K à l'instant t.
   (d) $b_t \in \mathcal{B}$. Combinaison de services en cours d'exécution. $b_0 = \varnothing$. Cette combinaison correspond à l'ensemble de sous-services précédemment sélectionnés n'ayant pas échoué.

   L'état spécial $s_\varnothing$ d'échec prématuré est aussi compris dans cet espace.
2. *Espace des actions $\mathcal{A}_s$ disponibles dans l'état s.* Comme précédemment, cet espace est construit suivant l'équation (4.9).
3. *Fonctions de renforcement immédiat.* $r_{\omega,s}^{(\psi)} : \mathcal{A}_s \times \mathcal{S} \to \mathbb{R}$

$$r_{\omega,s}^{(\psi)}(a, s') = \begin{cases} score_\omega^{(\psi)}(s', 0) & \text{si } a \neq \varnothing,\ s' \text{ est terminal selon } \psi \text{ et } s \text{ ne l'est pas} \\ score_\omega^{(\psi)}(s', 1) & \text{si } a = \varnothing \text{ et } s \text{ n'est pas terminal selon } \psi \\ 0 & \text{sinon} \end{cases} \tag{4.11}$$

   Les fonctions de renforcement immédiat et de score sont désormais spécifiques à la tâche ψ et au modèle d'évaluation ω. La fonction de score a le prototype $score_\omega^{(\psi)} : \mathcal{S} \times \{0, 1\} \to [0, 1]$ où l'ensemble des états $\mathcal{S}$ remplace l'ensemble des instances de service $\Sigma^\dagger$ dans l'équation (3.13) et les autres équations de la sous-section 3.3.2. Ces fonctions de renforcement ne changent pas durant le même épisode.
4. *Fonctions de transition.* $P_s^{(\psi)} : \mathcal{A}_s \times \mathcal{S} \to [0, 1]$. Les fonctions de transition sont spécifiques à la tâche ψ car le chemin suivi dépend de la destination. De même, le choix de l'action $\varnothing$ conduit vers un état terminal d'échec prématuré. Ces fonctions restent identiques pour tout $t < T$ durant le même épisode.
5. *Horizon.* $T = 10$. Cet horizon est laissé fixe par simplicité. Néanmoins, il pourrait être augmenté en fonction de la distance entre la position initiale et la destination.

Cette seconde formalisation acte le non-partage d'une fonction de valeur Q d'un épisode à l'autre car les fonctions de transition et de renforcement immédiat n'y sont plus les mêmes. De même, ce changement de fonctions de transition d'une exécution de haut-niveau à l'autre dissuade leur estimation directe à l'aide de tables ou leur apprentissage supervisé. Cette formalisation conforte donc l'apprentissage supervisé ou de l'estimation de paramètres de sous-modèles que nous proposons pour la modélisation des fonctions de transition.

4.1.3 Remplacement opportuniste

Cette sous-section s'intéresse à une extension du cas précédent appelée remplacement opportuniste. Les nouveautés sont les suivantes :

1. *La portée du remplacement concerne désormais l'ensemble des sous-services remplaçables.* Dans notre cas, le remplacement ne concerne que le service de positionnement. Néanmoins, tous les services ici considérés pourraient être remplacés si des alternatives étaient mises à disposition.
2. *L'opportunité d'un remplacement est examinée tous les $M_m \in \mathbb{N}$ segments de niveau 2 ainsi qu'en réaction à un échec d'un sous-service.* Cette évaluation s'appuie sur la segmentation de niveau 2 du chemin correspondant à l'exécution de l'instance de service de haut-niveau. La distinction entre les deux niveaux de segmentation est présentée dans la sous-sous-section 4.2.2.2.

Processus de décision markovien Comme le MDP retenu dans le cas de remplacement précédent, son MDP à horizon fini est également spécifique à la tâche de haut-niveau et au modèle d'évaluation. Les nouvelles considérations de ce problème invitent à changer la définition de l'espace des actions $\mathcal{A}_s$ disponibles dans l'état s. Les fonctions de transition tiennent également compte de ces nouvelles prises de décision régulières. Le processus de décision markovien dédié au remplacement opportuniste est donc le suivant :

1. *État à l'instant t.* $s_t = [x_t, y_t, x_{as,t}^{(1)}, y_{as,t}^{(1)}, \ldots, x_{as,t}^{(L)}, y_{as,t}^{(L)}, m_t^{(1)}, ..., m_t^{(k)}, b_t]^\intercal \in \mathcal{S}$. Cet espace d'états reste donc inchangé par rapport au MDP retenu pour le cas précédent.
2. *Espace des actions $\mathcal{A}_s$ disponibles dans l'état s.* Soit $\mathcal{Z}_{remp} \subset \mathcal{Z}$ l'ensemble des paires associant un service à une position dans un organigramme pour lesquelles le remplacement est autorisé. L'ensemble des actions disponibles dans l'état s_t est :

$$\mathcal{A}_{s_t} = \begin{cases} \mathcal{A} & \text{si } b_t = \varnothing \\ \left\{ a \in \mathcal{A} | \{ z \in \mathcal{Z} \setminus \mathcal{Z}_{remp} | c^\ddagger(b_t, z) = 1 \} \subseteq \{ z \in \mathcal{Z} | c^\dagger(a, z) = 1 \} \right\} & \text{sinon} \end{cases} \tag{4.12}$$

3. *Fonctions de renforcement immédiat.* $r_{\omega,s}^{(\psi)} : \mathcal{A}_s \times \mathcal{S} \to \mathbb{R}$. Ces fonctions sont identiques à celles du MDP retenu dans le cas de remplacement précédent. Elles sont donc définies par l'équation (4.11).
4. *Fonction de transition.* $P_s^{(\psi)} : \mathcal{A}_s \times \mathcal{S} \to [0, 1]$. Une transition concerne désormais au plus M_m segments de niveau 2. Les points intermédiaires atteints ne concernent plus seulement les échecs mais le parcours de M_m segments de niveau 2 depuis la dernière prise de décision. Ces fonctions restent spécifiques à la requête ψ et identiques durant le même épisode.
5. *Horizon.* $T = \left(10 + \frac{\Lambda_m^{(\psi)}}{M_m}\right)$ avec $\Lambda_m^{(\psi)} \in \mathbb{N}$ le nombre de segments de niveau 2 sur l'ensemble du chemin associé à la tâche ψ.

Ce processus de décision markovien sera résolu de la même manière que le MDP précédent de la sous-sous-section 4.1.2.2. La modélisation des fonctions de transition de ces deux processus fait l'objet de la section suivante.

4.2 Modélisation d'une transition

Une fonction de transition d'un MDP modélise la distribution de l'état s_{t+1} successeur de l'état s_t suite au choix de l'action a_t. Dans nos MDPs, de telles fonctions de

transition ne sont pas connues a priori sauf dans le cas d'une action $a_t = \varnothing$ conduisant de façon déterministe vers l'état terminal spécial dédié à l'échec prématuré. Par ailleurs, elles sont spécifiques à une tâche de haut-niveau ψ. Pour permettre un apprentissage qui puisse être partagé entre les exécutions, nous proposons d'effectuer une *décomposition* de ces transitions qui rend possible l'usage de sous-modèles indépendants de cette tâche ψ. Il est en effet possible de distinguer deux phases au sein d'une transition :

1. *Préparation.* La sélection d'un ou de plusieurs services ainsi que leur mise en place ont un coût et une durée dont il convient de tenir compte. Durant cette période, les instances de services non remplacées sont toujours actives et ont donc toujours un coût et génèrent parfois du bruit. La durée de cette phase peut être longue lorsqu'un robot assistant doit venir sur place.
2. *Déplacement sur plusieurs segments.* Cette segmentation du déplacement est rendue possible par la mise à disposition du chemin planifié par chacun des sous-services de planification de chemin candidats [5] au moment de la sélection. Ce chemin est supposé être représenté par le point initial, un ensemble de points intermédiaires et par la destination.

Plan Cette section est organisée de la façon suivante. En préambule, elle définit les fonctions, les ensembles employés tout au long des sous-sections ainsi que quelques combinaisons de services particulières (4.2.0.1). La sous-section 4.2.1 est dédiée à la phase de préparation. Elle propose un sous-modèle dédié à la performance associée à l'instanciation puis montre comment ce sous-modèle peut être utilisé pour en déduire de possibles états à l'issue de cette phase. Dans certains cas de figure, il est également nécessaire d'y additionner la phase de préparation de certaines instances de service nouvellement sélectionnées : l'exemple étudié est celui du service de positionnement fourni par un robot assistant.

La sous-section 4.2.2 est consacrée à la phase de déplacement du robot principal. Tout d'abord, elle propose un modèle de dynamique segmentée effectuant une estimation de paramètres en se basant sur une grille (4.2.2.1). Enfin cette sous-section se termine en montrant comment, à partir des états de la phase de préparation, les états successeurs de s_t et leurs probabilités associées sont déduits des déplacements sur les segments (4.2.2.2).

4.2.0.1 Définitions des fonctions et des ensembles employés

Ensemble des services nouvellement sélectionnés et combinaisons particulières Soit $\mathcal{Z}^{(i)}_{nouv,t}$ les services nouvellement sélectionnés par le choix de l'action a_i à partir de l'état s_t.

$$\mathcal{Z}^{(i)}_{nouv,t} = \left\{ z \in \mathcal{Z} | (c^{\dagger}(a_i, z) = 1) \wedge (c^{\ddagger}(b_t, z) = 0) \right\} \tag{4.13}$$

Soit $b^{(i)+}$ la combinaison de services actifs équivalente à celle de l'action évaluée a_i. $b^{(i)+}$ est défini par la relation suivante :

$$\left\{ z \in \mathcal{Z} | c^{\ddagger}(b^{(i)+}, z) = 1 \right\} = \left\{ z \in \mathcal{Z} | c^{\dagger}(a_i, z) = 1 \right\}. \tag{4.14}$$

5. Pour rappel, le service de navigation fait appel à trois sous-services : un service de positionnement, un service d'orientation et un service de planification de chemin (cf. sous-section 3.1.3). Néanmoins, de part les restrictions du cadre de notre étude, un seul service de planification est ici disponible.

En cas d'échec non-fatal, nous supposons ici, de part les restrictions de notre scénario, que le service échoué est un service de positionnement. Soit $\mathcal{Z}_{pos}$ l'ensemble des services de positionnement. La combinaison $b^{(i)-}$ de services actifs suite à un échec lors de l'exécution de l'action a_i est définie par la relation suivante :

$$\left\{z \in \mathcal{Z} | c^{\ddagger}(b^{(i)-}, z) = 1\right\} = \left\{z \in \mathcal{Z} \setminus \mathcal{Z}_{pos} | c^{\dagger}(a_i, z) = 1\right\}. \tag{4.15}$$

Modèle de performance lors de l'instanciation Pour chaque combinaison de services correspondant à une action $a_i \neq \varnothing$, la performance associée à la phase d'instanciation dépend des services nouvellement sélectionnés. Ceux-ci sont indirectement désignés par le passage en paramètre de la combinaison formée par les services déjà sélectionnés auprès d'une fonction $h_{ins}^{(i)}$ spécifique à l'action a_i. Si aucun nouveau service n'est sélectionné (lorsque $b_t = b^{(i)+}$), cette phase de préparation est ignorée.

Cette fonction est définie de la façon suivante :

$$\begin{array}{rccc} h_{ins}^{(i)} : & \mathcal{B} \setminus \{b^{(i)+}\} & \rightarrow & [0,1]^2 \times \mathbb{R}^{K'+2} \\ & b_t & \mapsto & \begin{bmatrix} p_{ins}^{+} \\ p_{ins}^{-} \\ \Delta m_{ins}^{(\delta)} \\ \Delta m_{ins}^{(\rho)} \\ \dot{m}_{b_t}^{(1)} \\ \vdots \\ \dot{m}_{b_t}^{(K')} \end{bmatrix}. \end{array} \tag{4.16}$$

Elle associe la combinaison des services déjà sélectionnés b_t aux :

1. *Probabilités de succès p_{ins}^{+} et d'échec non-fatal p_{ins}^{-} de la phase d'instanciation.*
2. *Durée $\Delta m_{ins}^{(\delta)}$ nécessaire à l'instanciation de tous les nouveaux services.*
3. *Coût $\Delta m_{ins}^{(\rho)}$ associé à ces instanciations.*
4. *Valeurs moyennes $\dot{m}_{b_t}^{(k)}$ par unité de temps de l'ensemble des services déjà sélectionnés b_t pour chaque métrique k autre que la durée.* Les valeurs des métriques sont ensuite calculées sur la base de la durée complète de la phase de préparation qui est parfois plus longue que la stricte instanciation.

Fonction de prédiction de la position d'observation d'un robot assistant Les positions des robots assistants susceptibles d'être sélectionnés pour leurs services de positionnement sont intégrées aux états des deux MDPs considérés. Il est donc nécessaire de prédire leur évolution. Nous faisons l'hypothèse que les L robots assistants sont réservés durant la durée de l'exécution du service de haut-niveau et ne se déplacent que dans le cadre de celle-ci.

Lorsque le service de positionnement d'un robot assistant ℓ est sélectionné, alors sa position d'observation avant le déplacement ou à l'issue de celui-ci doit être déterminé. La fonction $\hbar_{as}^{(\ell)}$ remplit ce rôle. Nous supposons qu'elle nous est fournie sous une forme analytique soit par le robot assistant soit par un service Web dédié. La spécification de

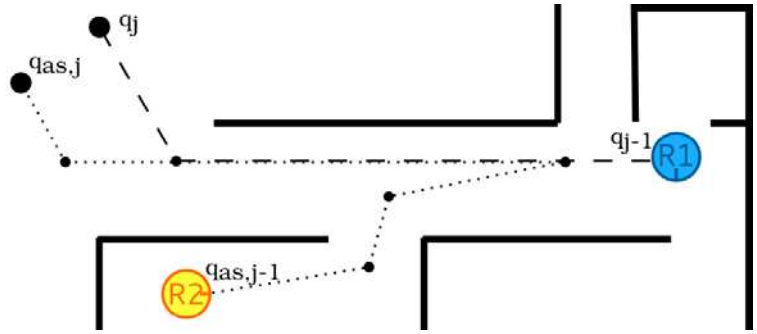


FIGURE 4.1 – Position d'observation $q^{(2)}_{as,j}$ du robot assistant $R2$ à l'issue du déplacement du robot principal $R1$ de q_{j-1} à q_j.

cette fonction suppose la connaissance d'une carte de l'environnement indiquant la présence d'obstacles fixes comme les murs et les meubles. Cette fonction a pour définition :

$$\begin{array}{rccc} \hbar^{(\ell)}_{as} : & \mathbb{R}^6 & \to & \mathbb{R}^2 \\ & \begin{bmatrix} x_{j-1} \\ y_{j-1} \\ x_j \\ y_j \\ x^{(\ell)}_{as,j-1} \\ y^{(\ell)}_{as,j-1} \end{bmatrix} & \mapsto & \begin{bmatrix} x^{(\ell)}_{as,j} \\ y^{(\ell)}_{as,j} \end{bmatrix}. \end{array} \tag{4.17}$$

Elle associe

1. *Les deux extrémités* $q_{j-1} = [x_{j-1}, y_{j-1}]^\intercal$ *et* $q_j = [x_j, y_j]^\intercal$ *du déplacement du robot principal.*
2. *La position courante* $q^{(\ell)}_{as,j-1} = [x^{(\ell)}_{as,j-1}, y^{(\ell)}_{as,j-1}]^\intercal$ *du robot assistant* ℓ.

à la position $q^{(\ell)}_{as,j} = [x^{(\ell)}_{as,j}, y^{(\ell)}_{as,j}]^\intercal$ du robot assistant à l'issue de ce déplacement. Ces différentes positions sont représentées dans la figure 4.1.

Modèle de dynamique segmentée Un modèle de dynamique segmentée, spécifique au choix de l'action $a_i \neq \varnothing$, est défini comme :

$$\begin{array}{rccc} f^{(i)}_{dyn} : & \mathbb{R}^4 & \to & (\mathbb{R} \cup \{\varnothing\})^4 \times [0,1]^2 \times \mathbb{R}^{2K} \\ & \begin{bmatrix} x_c \\ y_c \\ x_d \\ y_d \end{bmatrix} & \mapsto & \begin{bmatrix} x^+_f \\ y^+_f \\ x^-_f \\ y^-_f \\ p^+_{c\to d} \\ p^-_{c\to d} \\ \Delta m^{(1)+}_{c\to d} \\ \vdots \\ \Delta m^{(K)+}_{c\to d} \\ \Delta m^{(1)-}_{c\to d} \\ \vdots \\ \Delta m^{(K)-}_{c\to d} \end{bmatrix} \end{array} \tag{4.18}$$

Il associe aux extrémités d'un déplacement allant de $q_c = [x_c, y_c]^\intercal$ à $q_d = [x_d, y_d]^\intercal$ aux :

1. *Positions* $q_f^+ = [x_f^+, y_f^+]^\intercal$ *et* $q_f^- = [x_f^-, y_f^-]^\intercal$ *atteintes respectivement en cas de succès ou d'échec.*
2. *Probabilité de succès* $p_{c\to d}^+$ *de ce déplacement.*
3. *Probabilité d'un échec non-fatal* $p_{c\to d}^-$ *lors de ce déplacement.* Ce type d'échec peut donner lieu à un remplacement. Au contraire, un échec fatal n'est pas récupérable et conduit directement vers l'état d'échec prématuré. La probabilité d'échec fatal est $1 - p_{c\to d}^+ - p_{c\to d}^-$.
4. *Métriques* 1 *à* K *sur ce déplacement en cas de succès* $\Delta m_{c\to d}^{(k)+}$ *ou d'échec* $\Delta m_{c\to d}^{(k)-}$ avec $k \in \{1, ..., K\}$.

États successeurs et fonction de masse L'objet de cette modélisation de la fonction de transition est d'obtenir pour chaque action a_i disponible dans l'état s_t sachant la tâche ψ :

1. *Un ensemble* fini *d'états successeurs à* s_t *dans le MDP suite au choix de* $a_i \neq \varnothing$ *:* $\mathcal{C}_t^{(i,\psi)} \subset \mathcal{S}$.
2. *La fonction de masse* $\bar{P}_t^{(i,\psi)} : \mathcal{C}_t^{(i,\psi)} \to [0, 1]$ *associant une probabilité à chacun de ces états.*

L'ensemble $\mathcal{C}_t^{(i,\psi)}$ comporte au maximum un état correspondant à un succès et éventuellement plusieurs états d'échec. Dans le cas d'un remplacement non-opportuniste, l'état atteint après le succès de la transition est terminal, ce qui n'est pas forcément dans le cas opportuniste. L'unicité de cet état est due à l'uni-modalité du bruit modélisé comme étant gaussien.

Cet ensemble et cette fonction de masse sont obtenus à la suite de $N + 1$ itérations. Pour chacune de ces itérations, qu'il s'agisse de la phase préparatoire ou du déplacement sur un segment donné, un ensemble intermédiaire $\mathcal{C}_{t,j}^{(i,\psi)-}$ et une fonction de masse $\bar{P}_{t,j}^{(i,\psi)-}$ d'indice j et d'exposant $(-)$ sont ajoutés. Un tel ensemble intermédiaire ne contient que les états d'échecs connus. L'état de succès est ajouté à l'issue de la dernière itération (cf. sous-sous-section 4.2.2.2).

4.2.1 Performance de la phase préparatoire

La phase préparatoire regroupe l'instanciation des nouveaux services sélectionnés et l'éventuelle préparation de certains sous-services spéciaux eux-mêmes nouvellement sélectionnés. Elle est ignorée lorsqu'aucun nouveau service n'est sélectionné. Les seuls services spéciaux nécessite une préparation particulière de notre scénario sont les services de positionnement fournis par des robots assistants.

La prédiction de performance de cette phase permet d'obtenir les informations suivantes :

1. *Un sous-ensemble d'états successeurs d'échec* $\mathcal{C}_{t,0}^{(i,\psi)-} \subseteq \mathcal{C}_t^{(i,\psi)}$,
2. *Sa fonction de masse* $\bar{P}_{t,0}^{(i,\psi)-} : \mathcal{C}_{t,0}^{(i,\psi)-} \to [0, 1]$,
3. *L'état le plus probable* s_0^+ *en cas de succès de cette phase,*
4. *La probabilité* p_0^+ *qui lui est associée.*

Cette sous-section étudie tout d'abord la modélisation de la performance durant l'instanciation (4.2.1.1) puis s'intéresse à la déduction des états et de leurs probabilités à l'issue de cette phase de préparation (4.2.1.2).

4.2.1.1 Instanciation

L'instanciation des services nouvellement sélectionnés a une durée. Suivant la définition (4.16), cette sous-sous-section s'intéresse à la modélisation d'une fonction $h_{ins}^{(i)} : \mathcal{B} \setminus \{b^{(i)+}\} \to [0,1]^2 \times \mathbb{R}^{K'+2}$ pour chaque combinaison de services associée à une action a_i. Pour simplifier les notations, l'indice i est ignoré dans cette sous-sous-section 4.2.1.1.

Nous proposons un modèle composé de $K' + 2 + |\mathcal{B}|$ paramètres :

- *Un paramètre pour le taux de succès* $\mu_{ins}^{\sharp}$.
- *Un paramètre pour le taux de non-fatalité des échecs* $\mu_{ins}^{\triangledown}$.
- *Un paramètre* $\mu_{ins,b_t}^{(\delta)}$ *pour la durée d'instanciation associée à* $\mathcal{Z}_{nouv}$. Il estime le temps mis en moyenne entre l'émission par le sélectionneur d'une première requête de service et la réception de la dernière adresse d'instance de service.
- *Un paramètre* $\mu_{ins,b_t}^{(k)}$ *pour chaque* b_t *et par métrique* k *autre que la durée.* On s'intéresse ici aux services restés actifs durant cette phase d'instanciation car ils ont été instanciés dans une étape précédente. Chacun de ces paramètres correspond à la valeur moyenne de la métrique sur ces services par unité de temps.

Valeurs retournées Les correspondances entre les valeurs retournées $h_{ins}^{(i)}(b_t)$ (voir l'équation (4.16)) et les paramètres sont les suivantes :

$$p_{ins}^{+} = \mu_{ins}^{\sharp} \tag{4.19}$$
$$p_{ins}^{-} = (1 - \mu_{ins}^{\sharp})\mu_{ins}^{\triangledown} \tag{4.20}$$
$$\Delta m_{ins}^{(\delta)} = \mu_{ins,b_t}^{(\delta)} \tag{4.21}$$
$$\forall k \neq \delta, \dot{m}_{b_t}^{(k)} = \mu_{ins,b_t}^{(k)} \tag{4.22}$$

Mise à jour des paramètres après l'instanciation des nouveaux services Soit $\beta \in [0,1]$ le pas de mise à jour des paramètres.

1. *En cas d'échec fatal lors de l'instanciation.* En cas d'échec fatal, seuls les paramètres du taux de succès $\mu_{ins}^{\sharp}$ et du taux de non-fatalité des échecs $\mu_{ins}^{\triangledown}$ sont mis à jour.

$$\mu_{ins}^{\sharp} \leftarrow (1-\beta)\mu_{ins}^{\sharp} \tag{4.23}$$
$$\mu_{ins}^{\triangledown} \leftarrow (1-\beta)\mu_{ins}^{\triangledown}. \tag{4.24}$$

2. *En l'absence d'échec fatal lors de l'instanciation.* Soit $v_{ins}^{\sharp}$ le statut de l'exécution de cette phase de préparation (1 si succès, 0 si échec non-fatal). Soit $v_{prep,b_t}^{(1)}, ..., v_{prep,b_t}^{(K')}$ les valeurs des métriques autres que la durée sur les instances de services restées actives durant *toute la phase de préparation.* Soit $v_{ins}^{(\delta)}$ et $v_{prep}^{(\delta)}$ respectivement les durées de la phase d'instanciation et de la phase de préparation.

Le taux de succès est immédiatement mis à jour :

$$\mu_{ins}^{\sharp} \leftarrow (1-\beta)\mu_{ins}^{\sharp} + \beta v_{ins}^{\sharp}. \tag{4.25}$$

En cas d'échec non-fatal, le taux d'échec non-fatal est augmenté

$$\text{Si } v_{ins}^{\sharp} = 0, \mu_{ins}^{\triangledown} \leftarrow (1-\beta)\mu_{ins}^{\triangledown} + \beta. \tag{4.26}$$

Le statut $v_{ins}^{\sharp}$ de la phase de préparation n'est pas pris en compte lors de la mise à jour des paramètres restants :

$$\forall k \neq \delta, \mu_{ins,b_t}^{(k)} \leftarrow (1-\beta)\mu_{ins,b_t}^{(k)} + \beta \frac{v_{prep}^{(k)}}{v_{prep}^{(\delta)}} \tag{4.27}$$

4.2.1.2 États et probabilités à l'issue de la phase de préparation

Pour simplifier les notations, les exposants i, ψ et l'indice t sont omis dans la suite de cette sous-section sauf lorsqu'ils apportent une précision particulière.

1 - Si $\mathcal{Z}_{nouv} = \emptyset$. Lors d'une évaluation opportuniste ne faisant pas suite à un échec, le non-remplacement de services est systématiquement étudié. Dans une telle situation, la phase de préparation est ignorée. Aussi,

$$C_0^- = \emptyset \tag{4.28}$$
$$s_0^+ = s_t \tag{4.29}$$
$$p_0^+ = 1 \tag{4.30}$$

$\bar{P}_0^-$ n'est ici définie en aucun point.

2 - Performance durant l'instanciation des services Lorsque $\mathcal{Z}_{nouv} \neq \emptyset$, la fonction de performance de l'instanciation est appelée :

$$\begin{bmatrix} p_{ins}^+ \\ p_{ins}^- \\ \Delta m_{ins}^{(\delta)} \\ \Delta m_{ins}^{(\rho)} \\ \dot{m}_b^{(1)} \\ \vdots \\ \dot{m}_b^{(K')} \end{bmatrix} = h_{ins}(b) \tag{4.31}$$

On en déduit la probabilité d'échec fatal lors de l'instanciation :

$$p_{ins}^{\blacktriangledown} = 1 - p_{ins}^+ - p_{ins}^-. \tag{4.32}$$

Lorsqu'un échec non-fatal lors de l'instanciation est possible, l'état s_{ins}^- est considéré. Faute de modélisation particulière, on ne distingue pas les coûts et les durées d'instanciation suivant le succès ou l'échec non-fatal de l'instanciation. Par conséquent,

les métriques suivantes peuvent être calculées à l'issue de l'instanciation [6]. La métrique de temps ne considère que le temps d'instanciation :

$$m_{ins}^{(\delta)} = m_t^{(\delta)} + \Delta m_{ins}^{(\delta)}. \tag{4.33}$$

Les autres métriques ne considèrent que les valeurs des instances de services restées actives :

$$\forall k \neq \delta, m_{ins}^{(k)} = m_t^{(k)} + \dot{m}_{b_t}^{(k)} \times \Delta m_{ins}^{(\delta)}. \tag{4.34}$$

L'état s_{ins}^{-} a pour représentation :

$$s_{ins}^{-} = \left[x_t, y_t, x_{as,t}^{(1)}, y_{as,t}^{(1)}, \ldots, x_{as,t}^{(L)}, y_{as,t}^{(L)}, m_{ins}^{(1)}, \ldots, m_{ins}^{(K)}, b^{-}\right]^{\intercal}. \tag{4.35}$$

3 - Si aucun service de positionnement fourni par un robot assistant n'est dans $\mathcal{Z}_{nouv}$. Dans ce cas de figure, cette phase de préparation se limite donc à l'instanciation des services.

L'ensemble des états d'échec $\mathcal{C}_0^{-}$ et sa fonction de masse $\bar{P}_0^{-} : \mathcal{C}_0^{-} \to [0,1]$ sont :

$$\mathcal{C}_0^{-} = \begin{cases} \{s_{ins}^{-}\} & \text{si } (p_{ins}^{-} > 0) \wedge (p_{ins}^{\blacktriangledown} = 0) \\ \{s_{\varnothing}\} & \text{si } (p_{ins}^{-} = 0) \wedge (p_{ins}^{\blacktriangledown} > 0) \\ \{s_{ins}^{-}\} \cup \{s_{\varnothing}\} & \text{si } (p_{ins}^{-} > 0) \wedge (p_{ins}^{\blacktriangledown} > 0) \\ \emptyset & \text{sinon} \end{cases} \tag{4.36}$$

$$\bar{P}_0^{-}(s) = \begin{cases} p_{ins}^{\blacktriangledown} & \text{si } s = s_{\varnothing} \\ p_{ins}^{-} & \text{si } s = s_{ins}^{-} \end{cases} \tag{4.37}$$

L'état de succès $s_0^{+} = s_{ins}^{+}$ à l'issue de la phase de préparation a pour probabilité $p_0^{+} = p_{ins}^{+}$ et pour représentation :

$$s_0^{+} = s_{ins}^{+} = \left[x_t, y_t, x_{as,t}^{(1)}, y_{as,t}^{(1)}, \ldots, x_{as,t}^{(L)}, y_{as,t}^{(L)}, m_{ins}^{(1)}, \ldots, m_{ins}^{(K)}, b^{+}\right]^{\intercal}. \tag{4.38}$$

4 - Si le robot assistant ℓ fournit un service de positionnement compris dans $\mathcal{Z}_{nouv}$. Le déplacement du robot assistant à proximité du robot de la tâche de haut-niveau est ici modélisé comme une étape préparatoire.

1. *Détermination de sa destination.* La destination de ce déplacement correspond à la position d'observation du robot assistant ℓ (cf. figure 4.2). On fait donc appel à la fonction $\hbar_{as}^{(\ell)}$ (4.17).

$$q_{obs}^{(\ell)} = \begin{bmatrix} x_{obs}^{(\ell)} \\ y_{obs}^{(\ell)} \end{bmatrix} = \hbar_{as}^{(\ell)}(x_t, y_t, x_t, y_t, x_{as,t}^{(\ell)}, y_{as,t}^{(\ell)}) \tag{4.39}$$

2. *Déplacement du robot assistant.* Le déplacement préparatoire du robot assistant peut également donner lieu à une segmentation et une prédiction de dynamique sur chacun de ces segments. En reprenant la méthode de la sous-sous-section 4.2.2.2, il est possible d'obtenir les états successeurs $\breve{\mathcal{C}}^{(\ell)} \in \breve{\mathcal{S}}$ de l'état initial du robot assistant auxquels est associée la fonction de masse $\breve{P}^{(\ell)} : \breve{\mathcal{C}}^{(\ell)} \to [0,1]$. Similairement au cas du MDP non-opportuniste, le succès d'une transition depuis

6. On ignore pour l'instant l'éventualité d'un déplacement de robot assistant.

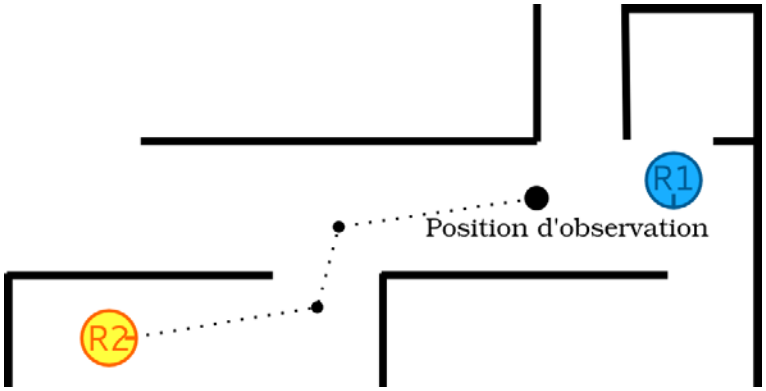


FIGURE 4.2 – Déplacement du robot assistant $R2$ vers sa position d'observation du robot principal $R1$.

l'état initial implique la réalisation complète de la tâche. Les transitions vers des états intermédiaires dénote aussi des échecs non-fatals. Néanmoins le remplacement des sous-services de cette instance de service de positionnement n'est pas considéré ici : nous nous intéressons qu'à la première transition. Par conséquent, l'horizon est $\breve{T} = 1$. De part les restrictions de notre scénario, ce service de positionnement fourni par un robot assistant fait systématiquement appel aux mêmes sous-services ; parmi ces sous-services se trouve un service de positionnement basé sur un télémètre laser embarqué sur ce robot assistant. L'échec de ce dernier service de positionnement entraîne donc celui du service de positionnement de niveau supérieur fourni par le robot assistant lui-même.
Soit $\breve{s}_\tau^{(\ell)} = \left[\breve{x}_\tau^{(\ell)}, \breve{y}_\tau^{(\ell)}, \breve{m}_\tau^{(\ell,1)}, \ldots, \breve{m}_\tau^{(\ell,K')}, \breve{m}_\tau^{(\ell,\delta)}\right]^\intercal$ l'état du MDP du service de déplacement du robot assistant. Le choix étant unique et l'horizon limité à une transition, les positions des autres robots assistants ne sont pas considérées tout comme la combinaison de services actuellement sélectionnée. La métrique de durée $\breve{m}_\tau^{(\ell,\delta)}$ est intégrée même si elle n'est pas considérée par le modèle d'évaluation associé à la tâche ψ. L'intérêt porté à ce MDP se concentre sur sa fonction de transition. L'état initial est $\breve{s}_0^{(\ell)} = \left[x_{as,t}^{(\ell)}, y_{as,t}^{(\ell)}, 0, \ldots, 0\right]^\intercal$. Ses états successeurs correspondant aux échecs (fatals ou non) $\breve{\mathcal{C}}^{(\ell)-} \in \breve{\mathcal{S}}$, leur fonction de masse $\breve{P}^{(\ell)-}$ ainsi que l'état de succès $\breve{s}_1^{(\ell)+} \in \breve{\mathcal{S}} \cup \{\varnothing\}$ et sa probabilité $\breve{p}_1^{(\ell)+}$ sont calculés par itération sur les segments, suivant la méthode décrite dans la sous-sous-section 4.2.2.2.

3. *Déduction de l'état de succès s_0^+, de sa probabilité p_0^+, des premiers états d'échec $\mathcal{C}_0^- \in \mathcal{S}$ et de leur fonction de masse $P_0^- : \mathcal{C}_0^- \to [0,1]$ pour le MDP de la tâche principale.*

   (a) Pour chaque état successeur non-fatal $\breve{s}_1^{(\ell,\gamma)} \in \breve{\mathcal{C}}^{(\ell)-} \cup \{\breve{s}^{(\ell)+}\} \setminus \{\varnothing\}$ du MDP du robot assistant, on en déduit les métriques des états du MDP principal correspondantes $m_{pre}^{(k,\gamma)}$. La métrique de temps néglige le temps d'instanciation au profit du temps de déplacement du robot assistant $\breve{m}_1^{(\ell,\delta,\gamma)}$:

$$m_{pre}^{(\delta,\gamma)} = m_t^{(\delta)} + \breve{m}_1^{(\ell,\delta,\gamma)}. \tag{4.40}$$

   Les autres métriques agrègent les valeurs liées au déplacement du robot assistant à celles associées aux instances de services restées actives. Pour cela,

elles utilisent des fonctions similaires à celles qu'emploient leurs sondes pour agréger les valeurs. Dans le cas de bruit, sa métrique $m_{pre}^{(\eta,\gamma)}$ est calculée de façon suivante en reprenant la formule (3.4) :

$$m_{pre}^{(\eta,\gamma)} = m_t^{(\eta)} + 10\log_{10}\left(\exp\left(\frac{\ln 10}{10} \times \frac{\breve{m}_1^{(\ell,\eta,\gamma)}}{\breve{m}_1^{(\ell,\delta,\gamma)}}\right) + \exp\left(\frac{\ln 10}{10} \times \dot{m}_{b_t}^{(\eta)}\right)\right) \times \breve{m}_1^{(\ell,\delta,\gamma)} \tag{4.41}$$

Pour la plupart des métriques, cette agrégation est une simple addition :

$$\forall k \neq \delta \neq \eta, m_{pre}^{(k,\gamma)} = m_t^{(k)} + \breve{m}_1^{(\ell,k,\gamma)} + \dot{m}_{b_t}^{(k)} \times \breve{m}_1^{(\ell,\delta,\gamma)}. \tag{4.42}$$

(b) Pour chaque état d'échec non-fatal $\breve{s}_1^{(\ell,\gamma)-} \in \breve{\mathcal{C}}^{(\ell)-} \setminus \{\breve{s}_\varnothing\}$, on en déduit un état d'échec non-fatal $s_{pre}^{(\gamma)-} \in \mathcal{S}$:

$$s_{pre}^{(\gamma)-} = \left[\begin{array}{ll} x_t, y_t, x_{as,t}^{(1)}, y_{as,t}^{(1)}, \ldots, & \breve{x}_1^{(\ell,\gamma)-}, \breve{y}_1^{(\ell,\gamma)-}, \ldots, \\ & x_{as,t}^{(L)}, y_{as,t}^{(L)}, m_{pre}^{(1,\gamma)-}, \ldots, m_{pre}^{(K,\gamma)-}, b^- \end{array}\right]^\intercal \tag{4.43}$$

L'ensemble des états du MDP principal résultant d'un échec non-fatal dans le MDP du robot assistant est

$$\mathcal{D}^- = \left\{\forall\gamma, s_{pre}^{(\gamma)-}\right\} \tag{4.44}$$

Les correspondances entre les états de $\mathcal{D}^-$ et ceux de $\breve{\mathcal{C}}^{(\ell)-}$ sont représentées par la fonction

$$\begin{array}{rccc} c_{pre}^- : & \mathcal{D}^- & \to & \breve{\mathcal{C}}^{(\ell)-} \setminus \{\breve{s}_\varnothing\} \\ & s_{pre}^{(\gamma)-} & \mapsto & \breve{s}_1^{(\ell,\gamma)-}. \end{array} \tag{4.45}$$

(c) De façon similaire, l'état de succès $s_0^+ = s_{pre}^+$ est déterminé à partir de $\breve{s}_1^{(\ell)+}$:

$$s_0^+ = s_{pre}^+ = \begin{cases} \left[\begin{array}{c} x_t \\ y_t \\ x_{as,t}^{(1)} \\ y_{as,t}^{(1)} \\ \vdots \\ \breve{x}_1^{(\ell)+} \\ \breve{y}_1^{(\ell)+} \\ \vdots \\ x_{as,t}^{(L)} \\ y_{as,t}^{(L)} \\ m_{pre}^{(1)+} \\ \vdots \\ m_{pre}^{(K)+} \\ b^+ \end{array}\right] & \text{si } \breve{s}_1^{(\ell)+} \neq \varnothing \\ \varnothing & \text{sinon.} \end{cases} \tag{4.46}$$

Sa probabilité de succès p_0^+ est

$$p_0^+ = p_{ins}^+ \times \breve{p}_1^{(\ell)+} \tag{4.47}$$

(d) Dans le cas d'un échec fatal du robot assistant, son MDP ne fournit pas de métrique. Aussi, l'état correspondant $s_{pre}^{\blacktriangledown -}$ est assimilé à l'état s_{ins}^{-}.

$$s_{pre}^{\blacktriangledown -} = s_{ins}^{-} \tag{4.48}$$

Sa probabilité $p_{pre}^{\blacktriangledown -}$ vaut

$$p_{pre}^{\blacktriangledown -} = \begin{cases} p_{ins}^{+} \times \breve{P}(\breve{s}_{\varnothing}) & \text{si } \breve{s}_{\varnothing} \in \breve{\mathcal{C}}^{(\ell)-} \\ 0 & \text{sinon} \end{cases} \tag{4.49}$$

(e) Enfin, on en déduit l'ensemble des états d'échecs et sa fonction de masse.

$$\mathcal{C}_0^{-} = \begin{cases} \mathcal{D}^{-} & \text{si } (p_{ins}^{-} + p_{pre}^{\blacktriangledown -} = 0) \wedge (p_{ins}^{\blacktriangledown} = 0) \\ \mathcal{D}^{-} \cup \{s_{ins}^{-}\} & \text{si } (p_{ins}^{-} + p_{pre}^{\blacktriangledown -} > 0) \wedge (p_{ins}^{\blacktriangledown} = 0) \\ \mathcal{D}^{-} \cup \{s_{\varnothing}\} & \text{si } (p_{ins}^{-} + p_{pre}^{\blacktriangledown -} = 0) \wedge (p_{ins}^{\blacktriangledown} > 0) \\ \mathcal{D}^{-} \cup \{s_{ins}^{-}\} \cup \{s_{\varnothing}\} & \text{si } (p_{ins}^{-} + p_{pre}^{\blacktriangledown -} > 0) \wedge (p_{ins}^{\blacktriangledown} > 0) \end{cases} \tag{4.50}$$

$$\bar{P}_0^{-}(s) = \begin{cases} p_{ins}^{\blacktriangledown} & \text{si } s = s_{\varnothing} \\ p_{ins}^{-} + p_{pre}^{\blacktriangledown -} & \text{si } s = s_{ins}^{-} \\ p_{ins}^{+} \times \breve{P}\left(c_{pre}(s)\right) & \text{si } s \in \mathcal{D}^{-} \end{cases} \tag{4.51}$$

En cas d'échec non-fatal survenu durant l'instanciation, le déplacement du robot est immédiatement arrêté. Son éventuel déplacement partiel est ignoré, ce qui permet d'introduire directement l'état s_{ins}^{-} dans l'ensemble $\mathcal{C}_0^{-}$.

Cette phase de préparation fournit l'itération 0 de la construction de l'ensemble des états successeurs et de sa fonction de masse. La sous-section suivante peut désormais s'intéresser au déplacement sur les segments.

4.2.2 Performance du déplacement sur plusieurs segments

Après la phase de préparation, cette sous-section s'intéresse à la seconde phase d'une transition : le déplacement du robot principal. La prédiction de performance de cette seconde phase repose sur une segmentation du déplacement.

Cette sous-section est organisée suivant deux axes. Dans un premier temps, elle propose un modèle de dynamique permettant une prédiction de performance sur un segment donné. Celui-ci organise ses paramètres suivant une grille (4.2.2.1). Enfin, elle en déduit les états successeurs de s_t et leurs probabilités associées à partir des états de la phase de préparation et des déplacements sur les segments (4.2.2.2).

4.2.2.1 Modèle de dynamique segmentée basé sur une grille

Ce modèle de dynamique réalise une approximation de la fonction $f_{dyn}^{(i)}$ (4.18) par le biais d'une estimation de paramètres. Ces paramètres sont organisés suivant une grille.

Cette sous-sous-section décrit ces paramètres puis montre comment ceux-ci sont sélectionnés et utilisés pour calculer les valeurs des métriques. L'estimation de ces paramètres est ensuite détaillée.

L'exposant ou l'indice i associé à l'action a_i sera généralement omis dans cette sous-sous-section.

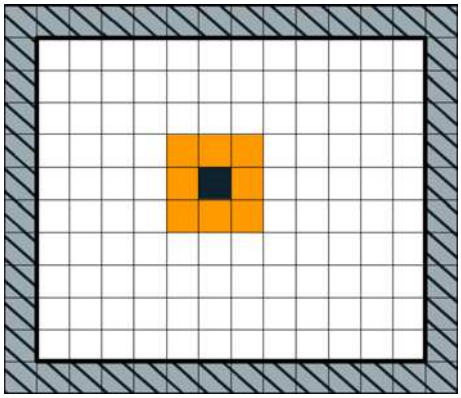


FIGURE 4.3 – *Grille, voisinage immédiat et marge extérieure.* Les huit cases du voisinage immédiat de la case noire sont représentés en orange. Une marge extérieure représentée en gris hachuré ajoute des cases supplémentaires que le robot ne peut pas atteindre mais qui fournissent des paramètres utiles.

Grille et paramètres La grille sur laquelle repose ce modèle vient partitionner le plan horizontal (x, y) de déplacement du robot. Nous supposons que cette grille est disposée de sorte qu'aucune case ne contienne les deux côtés d'un mur. À partir d'une position courante q_c et de la case c_c dans laquelle elle se situe, l'objectif est de prédire la dynamique du déplacement vers l'extrémité du segment q_d. q_d doit être située dans le voisinage immédiat de la case courante ou exceptionnellement dans la même case. Le voisinage immédiat d'une case est ici les huit cases qui l'entourent (voir la figure 4.3).

Ce modèle repose sur $8(2K'+4)\nu$ paramètres où K' représente le nombre de métriques autres que la durée d'exécution et ν le nombre de cases dans la grille appartenant à l'espace accessible [7]. La durée d'exécution a des paramètres comme les autres métriques même si elle ne fait pas partie du modèle d'évaluation de haut-niveau : sa valeur est nécessaire à la prédiction des autres. À chaque déplacement entre le centre de la case c_l et celui d'une case c_n voisine correspondent :

- *Un paramètre pour le taux de succès* $\mu^{\sharp}_{l\to n}$. Il correspond à la valeur moyenne d'une variable aléatoire ayant une distribution de Bernoulli.
- *Un paramètre pour le taux de non-fatalité des échecs* $\mu^{\triangledown}_{l\to n}$. De même, il estime l'espérance d'une distribution de Bernoulli.
- *Un paramètre pour la vitesse en cas de succès.* $\mu^{(\delta)+}_{l\to n}$ correspond à la vitesse moyenne observée lors les déplacements réussis depuis la case l vers la case n.
- *Un paramètre pour la durée de détection d'échec.* Ce paramètre $\mu^{(\delta)-}_{l\to n}$ correspond à la durée entre l'instant où la dernière position a été reçue et la déclaration de l'échec du sous-service de positionnement.
- *Deux paramètres par métrique k autre que la durée d'exécution* correspondant au cas d'un succès $\mu^{(k)+}_{l\to n}$ et d'un échec $\mu^{(k)-}_{l\to n}$. Ces paramètres correspondent aux moyennes de distributions supposées gaussiennes.

Calcul des métriques sur un segment

1. *Sélection des cases.* Les extrémités du déplacement de q_c à q_d sont associées aux cases c_c et c_d. La case c_c est toujours employée pour obtenir les paramètres tandis

7. Les cellules représentées en gris dans la figure 4.3 ne font pas partie de ce décompte car elles ne sont pas accessibles. Leur existence sera motivée dans le paragraphe suivant.

que c_d ne l'est pas systématiquement. Deux cas se présente pour c_d :

(a) $c_d = c_c$. Aucun paramètre ne correspond alors au couple (c_c, c_d). On utilise la direction $q_d - q_c$ pour choisir une case c_e dans le voisinage immédiat de c_c. Plus précisément, c_e est la case contenant la position $q_e = q_d + \lambda \times (q_d - q_c)$ où $\lambda \in R^{+*}$ est la plus petite valeur positive satisfaisant la condition $q_e \notin c_c$. Dans une perspective de généralisation, des cases supplémentaires ont été ajoutées à l'extérieur du bord (elles sont représentées en gris hachuré sur la figure 4.3) pour fournir les paramètres dont les cases à l'intérieur du bord peuvent avoir besoin.

(b) c_d *est dans le voisinage immédiat de* c_c. Par soucis d'homogénéité, c_e est également définie : $c_e = c_d$.

Les paramètres des différents modèles sont obtenus à partir du couple (c_c, c_e).

2. *Déduction des métriques à partir des paramètres.* Soit $p^+_{c\to d}$, $p^-_{c\to d}$ et respectivement les probabilités de succès et d'échec non-fatal sur le segment $[c, d]$.

$$p^+_{c\to d} = \mu^\sharp_{c\to e} \tag{4.52}$$
$$p^-_{c\to d} = (1 - p^+_{c\to d}) \times \mu^\triangledown_{c\to e} \tag{4.53}$$
$$\tag{4.54}$$

Dans un premier temps, on s'intéresse à la probabilité de succès $p^+_{c\to d}$. Si cette valeur est supérieure à τ^+, le cas d'un déplacement effectué avec succès est considéré. De même, le cas d'un échec non-fatal n'est étudié que si sa probabilité $p^-_{c\to d}$ est inférieure à τ^-.

Ensuite, la durée est évaluée (métrique $\delta = K' + 1$) en cas de succès ou d'échec non-fatal. En cas de succès, la durée $\Delta m^{(\delta)+}_{c\to d}$ est calculée comme étant le produit du paramètre de vitesse $\mu^{(\delta)+}$ et de la distance entre q_d et q_c :

$$\Delta m^{(\delta)+}_{c\to d} = \frac{\|q_d - q_c\|^2}{\mu^{(\delta)+}_{c\to e}} \tag{4.55}$$

En cas d'échec, seule la durée de détection d'échec est considérée :

$$\Delta m^{(\delta)-}_{c\to d} = \mu^{(\delta)-}_{c\to e} \tag{4.56}$$

Les autres métriques sont obtenues à partir de celle du temps :

$$\forall i \neq \delta, \Delta m^{(i)\alpha}_{c\to d} = \Delta m^{(\delta)\alpha}_{c\to d} \times \mu^{(i)\alpha}_{c\to e} \tag{4.57}$$

3. *Positions atteintes.* Ce modèle ne contient pas de paramètres pour estimer la position finale. Par conséquent, on suppose que la position q_d a été atteinte dans le cas d'un succès :

$$q^+_f = \begin{cases} q_d & \text{si } p^+_{c\to d} > \tau^+ \\ \varnothing & \text{sinon} \end{cases} \tag{4.58}$$

Si le succès est improbable, la position finale q^+_f n'est pas définie.

Concernant l'échec non-fatal, ne sachant à quelle position il a eu lieu, la position considérée est la position initiale q_c :

$$q^-_f = \begin{cases} q_c & \text{si } p^+_{c\to d} < \tau^- \\ \varnothing & \text{sinon} \end{cases} \tag{4.59}$$

De même, q^-_f n'est pas définie lorsque l'échec non-fatal est trop peu probable.

Mise à jour des paramètres à partir de métriques sur des segments de l'exécution Durant l'exécution du service de haut-niveau, le modèle de dynamique de déplacement associé à l'action a_i actuellement sélectionnée reçoit les valeurs de métriques sur des segments de l'exécution. Cette segmentation est supposée correspondre aux contraintes de ce modèle, à savoir que la seconde extrémité du segment soit dans le périmètre des huit cases entourant la case de la première extrémité. Un tel segment correspond au parcours observé du robot qui peut différer du parcours prédit.

On s'intéresse ici au déplacement de de q_l vers q_n. Soit c_l et c_o les cases dont les paramètres correspondent au déplacement de q_l vers q_n. En tant qu'estimateurs de l'espérance, les paramètres sont mis à jour à l'aide d'une moyenne incrémentale. On utilise ici un pas de mise à jour $\beta \in [0, 1]$ fixe pour l'ensemble de ces paramètres.

1. *En cas d'échec fatal.* Seuls les paramètres du taux de succès $\mu^{\sharp}_{l\to o}$ et du taux de non-fatalité des échecs $\mu^{\triangledown}_{l\to o}$ sont mis à jour.

$$\mu^{\sharp}_{c\to o} \leftarrow (1-\beta)\mu^{\sharp}_{l\to o} \tag{4.60}$$
$$\mu^{\triangledown}_{l\to o} \leftarrow (1-\beta)\mu^{\triangledown}_{l\to o} \tag{4.61}$$

2. *En l'absence d'échec fatal.* L'essentiel des paramètres correspondant aux cases c_l et c_o sont mis à jour. Soit $v^{\sharp}_{l\to n}$ le statut de l'exécution sur le segment (1 si succès, 0 si échec non-fatal). Soit $v^{(1)}_{l\to n}, ..., v^{(K')}_{l\to n}, v^{(\delta)}_{l\to n}$ les valeurs mesurées correspondant respectivement aux métriques requises par le modèle d'évaluation et à la durée sur ce segment. En cas d'échec non-fatal, il convient de considérer également la durée de détection d'échec $v^{(\delta)\triangledown}_{l\to n}$. α est déterminé à partir de $v^{\sharp}_{l\to n}$. Le taux de succès peut être immédiatement mis-à-jour :

$$\mu^{\sharp}_{l\to o} \leftarrow (1-\beta)\mu^{\sharp}_{l\to o} + \beta v^{\sharp}_{l\to n}. \tag{4.62}$$

En cas d'échec non-fatal, le taux d'échec non-fatal est augmenté

$$\mu^{\triangledown}_{l\to o} \leftarrow (1-\beta)\mu^{\triangledown}_{l\to o} + \beta. \tag{4.63}$$

La vitesse est mise à jour en cas de succès :

$$\mu^{(\delta)+}_{l\to o} \leftarrow (1-\beta)\mu^{(\delta)+}_{l\to o} + \beta\frac{\|q_o - q_l\|^2}{v^{(\delta)}_{l\to n}} \tag{4.64}$$

Si un échec non-fatal a eu lieu, la durée de détection d'échec est actualisée :

$$\mu^{(\delta)-}_{l\to o} \leftarrow (1-\beta)\mu^{(\delta)-}_{l\to o} + \beta v^{(\delta)\triangledown}_{l\to n} \tag{4.65}$$

Enfin, les paramètres des autres métriques sont mis à jour en tenant compte de $v^{(\delta)}_{l\to n}$:

$$\forall k \neq \delta, \mu^{(k)\alpha}_{l\to o} \leftarrow (1-\beta)\mu^{(k)\alpha}_{l\to o} + \beta\frac{v^{(k)}_{l\to n}}{v^{(\delta)}_{l\to n}} \tag{4.66}$$

4.2.2.2 États et probabilités à l'issue du déplacement sur plusieurs segments

À partir des modèles de dynamique segmentés et des résultats de la phase préparatoire, cette sous-sous-section peut désormais déduire les états successeurs de cette transition à partir de l'état s_t et du choix de l'action a_i. Plus précisément, son objectif est d'obtenir :

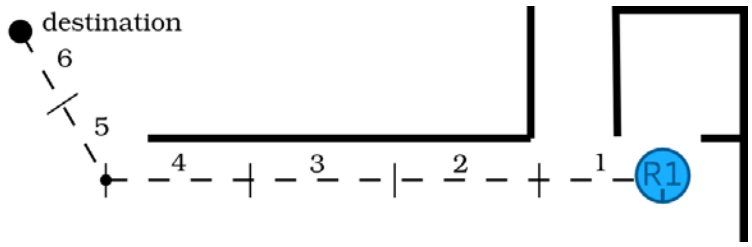


FIGURE 4.4 – Segmentation du déplacement du robot $R1$.

1. *L'ensemble d'états successeurs* $\mathcal{C}_t^{(i,\psi)} \subset \mathcal{S}$.
2. *Sa fonction de masse* $\bar{P}_t^{(i,\psi)} : \mathcal{C}_t^{(i,\psi)} \to [0,1]$ associant une probabilité à chacun de ces états.

Pour cela, le chemin de déplacement doit tout d'abord être segmenté. Ensuite ces segments sont itérés : leurs dynamiques sont obtenues à partir des états précédents et de la fonction $f_{dyn}^{(i)}$ spécifique à l'action a_i. La phase préparatoire fournit les éléments initiaux, à savoir l'ensemble d'états $\mathcal{C}_{t,0}^{(i,\psi)-}$, sa fonction de masse $\bar{P}_{t,0}^{(i,\psi)-} : \mathcal{C}_{t,0}^{(i,\psi)-} \to [0,1]$, l'état s_0^+ et sa probabilité p_0^+. Les exposants i et ψ ainsi que l'indice t sont omis dans la suite de cette sous-sous-section sauf lorsqu'ils apportent une précision particulière.

Segmentation Le chemin planifié est composé de Λ_c segments de niveau 1 reliant la position initiale à la destination. Ces segments sont de longueur variable et ne répondent pas directement aux contraintes des modèles de dynamique de déplacement : ils doivent donc être eux-mêmes segmentés en un second niveau. Il en résulte Λ_m segments de niveau 2 compatibles avec les modèles de dynamique de déplacement (voir figure 4.4).

Cette segmentation de niveau 2 est dépendante du modèle de dynamique utilisé. Pour ce modèle basé sur une grille, la segmentation d'un segment de niveau 1 $[q_l, q_o]$ part du point q_l. Elle cherche ensuite le point q_p situé sur ce segment le plus proche du centre d'une case de son voisinage sauf si ce point q_o est lui-même dans cette case. Dans ce dernier cas, q_o est directement utilisé comme extrémité du segment de niveau 2. Elle part ensuite du point q_p et recherche de façon similaire le point suivant tant que le point q_o n'est pas atteint.

Itérations des déplacements sur les segments Soit Λ_n le nombre de segments parcourus durant une transition. Sa valeur diffère selon que le remplacement puisse survenir de façon opportuniste ou non :

$$\Lambda_n = \begin{cases} M_m & \text{dans le cas opportuniste} \\ \Lambda_m & \text{sinon} \end{cases} \tag{4.67}$$

La construction de l'ensemble $\mathcal{C}$ et de sa fonction de masse s'appuie sur $n \leq \Lambda_n$ itérations où pour chaque itération $j \geq 1$, l'ensemble des états d'échec $\mathcal{C}_j^- \subset \mathcal{S}$, la fonction de masse $P_j^- : \mathcal{C}_j^- \to [0,1]$, l'état $s_j^+ \in \mathcal{S}$ atteint en cas de succès et sa probabilité $p_j^+ \in [0,1]$ sont calculés. Soit ξ_j le $j^{ème}$ segment de niveau 2.

Pour chaque itération $1 \leq j \leq n$,

1. *Calcul de la dynamique sur le segment* ξ_j. Soit $q_{c,j} = [x_{c,j}, y_{c,j}]^\intercal, q_{d,j} = [x_{d,j}, y_{d,j}]^\intercal$ les extrémités du segment ξ_j.

$$\begin{bmatrix} x^+_{f,j} \\ y^+_{f,j} \\ x^-_{f,j} \\ y^-_{f,j} \\ p^+_{c,j\to d,j} \\ p^-_{c,j\to d,j} \\ \Delta m^{(1)+}_{c,j\to d,j} \\ \vdots \\ \Delta m^{(K)+}_{c,j\to d,j} \\ \Delta m^{(1)-}_{c,j\to d,j} \\ \vdots \\ \Delta m^{(K)-}_{c,j\to d,j} \end{bmatrix} = f_{dyn}(\begin{bmatrix} x_{c,j} \\ y_{c,j} \\ x_{d,j} \\ y_{d,j} \end{bmatrix}) \tag{4.68}$$

On peut en déduire la probabilité d'échec fatal,

$$p^{\blacktriangledown}_{c,j\to d,j} = 1 - p^+_{c,j\to d,j} - p^-_{c,j\to d,j}. \tag{4.69}$$

2. *Positions des robots assistants.* Les robots assistants non-sollicités par cette composition de services sont supposés garder leurs positions (cf. sous-sous-section 4.2.0.1).

   Pour chaque robot assistant $\ell \leq L$,

   (a) *Le robot assistant* ℓ *traque le robot principal.* Sa nouvelle position est prédite par la fonction $\hbar^{(\ell)}_{as} : \mathbb{R}^6 \to \mathbb{R}^2$ (4.17) fournie. Pour chaque $\alpha \in \{+, -\}$,

$$\begin{bmatrix} x^{(\ell)\alpha}_{as,j} \\ y^{(\ell)\alpha}_{as,j} \end{bmatrix} = \hbar^{(\ell)}_{as}\left(x^+_{f,j-1}, y^+_{f,j-1}, x^{(\alpha)}_{f,j}, y^{(\alpha)}_{f,j}, x^{(\ell)+}_{as,j-1}, y^{(\ell)+}_{as,j-1}\right) \tag{4.70}$$

   avec $x^+_{f,0}, y^+_{f,0}, x^{(\ell)+}_{as,0}, y^{(\ell)+}_{as,0}$ directement extraits de s^+_0 suivant l'équation (4.72).

   (b) *Sinon,*

$$\begin{cases} x^{(\ell)\alpha}_{as,j} = x^{(\ell)}_{as,0} \\ y^{(\ell)\alpha}_{as,j} = y^{(\ell)}_{as,0} \end{cases} \tag{4.71}$$

3. *Calcul des nouveaux états.* L'état atteint en cas de succès est :

$$s^+_j = \begin{cases} \begin{bmatrix} x^+_{f,j} \\ y^+_{f,j} \\ x^{(1)+}_{as,j} \\ y^{(1)+}_{as,j} \\ \vdots \\ x^{(L)+}_{as,j} \\ y^{(L)+}_{as,j} \\ m^{(1)+}_{j-1} + \Delta m^{(1)+}_{c,j\to d,j} \\ \vdots \\ m^{(K)+}_{j-1} + \Delta m^{(K)+}_{c,j\to d,j} \\ b^+ \end{bmatrix} & \text{si } p^+_{c,j\to d,j} > \tau^+ \\ \varnothing & \text{sinon} \end{cases} \tag{4.72}$$

De même, l'état d'échec s'appuie sur les métriques de l'état de succès précédent :

$$s_j^- = \begin{cases} \begin{bmatrix} x_{f,j}^- \\ y_{f,j}^- \\ x_{as,j}^{(1)-} \\ y_{as,j}^{(1)-} \\ \vdots \\ x_{as,j}^{(L)-} \\ y_{as,j}^{(L)-} \\ m_{j-1}^{(1)+} + \Delta m_{c,j\to d,j}^{(1)-} \\ \vdots \\ m_{j-1}^{(K)+} + \Delta m_{c,j\to d,j}^{(K)-} \\ b^- \end{bmatrix} & \text{si } p_{c,j\to d,j}^- > \tau^- \\ \varnothing & \text{sinon} \end{cases} \tag{4.73}$$

4. *Calcul des probabilités associées à ces états.*

$$p_j^+ = \begin{cases} p_{j-1}^+ \times p_{c,j\to d,j}^+ & \text{si } (p_{c,j\to d,j}^+ > \tau^+) \wedge (p_{c,j\to d,j}^- > \tau^-) \\ p_{j-1}^+ & \text{si } p_{c,j\to d,j}^- \leq \tau^- \\ 0 & \text{sinon} \end{cases} \tag{4.74}$$

$$p_j^- = \begin{cases} p_{j-1}^+ \times p_{c,j\to d,j}^- & \text{si } (p_{c,j\to d,j}^+ > \tau^+) \wedge (p_{c,j\to d,j}^- > \tau^-) \\ p_{j-1}^+ & \text{si } p_{c,j\to d,j}^+ \leq \tau^+ \\ 0 & \text{sinon} \end{cases} \tag{4.75}$$

5. *Mise à jour de l'ensemble et de la fonction de masse.*

$$\mathcal{C}_j^- = \begin{cases} \mathcal{C}_{j-1}^- \cup \{s_j^-\} & \text{si } (s_j^- \neq \varnothing) \wedge (p_{c,j\to d,j}^{\blacktriangledown} = 0) \\ \mathcal{C}_{j-1}^- \cup \{s_j^-, s_\varnothing\} & \text{si } (s_j^- \neq \varnothing) \wedge (p_{c,j\to d,j}^{\blacktriangledown} > 0) \\ \mathcal{C}_{j-1}^- \cup \{s_\varnothing\} & \text{si } (s_j^- = \varnothing) \wedge (p_{c,j\to d,j}^{\blacktriangledown} > 0) \\ \mathcal{C}_{j-1}^- & \text{sinon} \end{cases} \tag{4.76}$$

$$\bar{P}_j^-(s) = \begin{cases} \bar{P}_{j-1}^-(s) & \text{si } s \in \mathcal{C}_{j-1}^- \setminus \{s_\varnothing\} \\ \bar{P}_{j-1}^-(s) + p_{j-1}^+ \times p_{c,j\to d,j}^{\blacktriangledown} & \text{si } (s = s_\varnothing) \wedge (s \in \mathcal{C}_{j-1}^-) \\ p_{j-1}^+ \times p_{c,j\to d,j}^{\blacktriangledown} & \text{si } (s = s_\varnothing) \wedge (s \notin \mathcal{C}_{j-1}^-) \\ p_j^- & \text{si } (s = s_j^-) \wedge (s_j^- \neq \varnothing) \end{cases} \tag{4.77}$$

6. *Évaluation des conditions d'arrêt.* La boucle s'arrête si $p_{c,j\to d,j}^+ \leq \tau^+$ ou si $j = \Lambda_n$.

Ensemble final À l'issue de N itérations, l'ensemble $\mathcal{C}$ et la fonction de masse $\bar{P}$ peuvent être déterminés.

$$\mathcal{C} = \begin{cases} \mathcal{C}_N^- \cup \{s_N^+\} & \text{si } s_N^+ \neq \varnothing \\ \mathcal{C}_N^- & \text{sinon} \end{cases} \tag{4.78}$$

$$\bar{P}(s) = \begin{cases} \bar{P}_N^-(s) & \text{si } s \in \mathcal{C}_N^- \\ p_N^+ & \text{si } s = s_N^+ \end{cases} \tag{4.79}$$

4.2.2.3 Discussion

De part les restrictions de notre scénario, certaines modélisations ont pu être économisées. Ces modélisations les plus notables sont celles de :

1. *L'apparition et la disparition de services en cours d'exécution.* Les robots assistants pourraient être momentanément indisponibles car étant sollicités ailleurs. De plus, leurs nouvelles positions dépendraient de ces sollicitations externes.
2. *L'échec non-fatal de services autres que ceux de positionnement.* Il conviendrait alors de distinguer leurs échecs et de leur associer des probabilités.

Cette modélisation d'une transition fournit un nombre *fini* d'états successeurs. Cette nature finie est essentielle aux méthodes d'estimation de valeurs employées dans la section suivante.

4.3 Estimation des valeurs

Suivant la définition des MDPs des deux problèmes de remplacement (section 4.1) et la modélisation de la fonction de transition (section 4.2), ces MDPs sont désormais entièrement modélisés. Cette section peut dès lors se consacrer à leur résolution par des méthodes de planification se basant sur les modèles de transition et de renforcement immédiat.

La finalité d'une planification peut aller de l'obtention d'une politique définie sur l'ensemble des états au simple choix de la meilleure action dans l'état courant. Dans le dernier cas, cette planification est dite *en ligne*. Elle permet de n'explorer que les états accessibles à partir de l'état courant à partir d'une ou de plusieurs itérations voire même un sous-ensemble d'entre eux. Son principal désavantage est de devoir être ré-effectuée dans chaque nouvel état rencontré contrairement à une planification dite hors-ligne aboutissant à une politique utilisable dans n'importe quel état. Néanmoins, ce type de planification convient à nos problèmes de part l'impossibilité de réutiliser une même politique entre les épisodes et l'absence de contrainte temps réel.

Notre objectif est donc d'obtenir une estimation précise des valeurs $Q(s_t, a)$ pour toutes les actions $a \in \mathcal{A}_{s_t}$ disponibles dans l'état courant s_t. L'action ayant la valeur la plus élevée est sélectionnée :

$$\pi^{(t)}(s_t) = \arg\max_{a \in \mathcal{A}_{s_t}} Q(s_t, a). \tag{4.80}$$

Partant du constat que les états successeurs sont obtenus en nombre fini et du caractère exceptionnel des transitions de plusieurs actions depuis le même état menant vers des états communs autres que l'état d'échec, nous proposons de modéliser la structure de recherche comme un arbre. Cet arbre est décrit dans la sous-section 4.3.1 et il est montré que sa dimension est généralement trop grande dans chacun des deux problèmes de remplacement pour envisager une recherche exhaustive. Nous proposons en conséquent dans la sous-section 4.3.2 d'effectuer une exploration dirigée de cet arbre de recherche par le biais d'une méthode de fouille stochastique d'arbre.

4.3.1 Arbre de recherche

L'arbre de recherche, représenté par la figure 4.5, est caractérisé par les éléments suivants :

1. *Il a pour racine l'état courant s_t.*
2. *À chaque état sont associées $|\mathcal{A}_s|$ actions.* Si l'état courant s_t est initial, c'est-à-dire si sa composante $b_t = \varnothing$, alors l'ensemble des actions disponibles est maximal : $\mathcal{A}_{s_t} = \mathcal{A}$. Pour les autres états, le nombre d'actions disponibles est généralement plus restreint. Soit Γ le nombre maximum d'instances de services dans un organigramme en cours d'exécution pouvant être remplacés [8]. Dans le cas d'un remplacement non-opportuniste, le nombre $\Upsilon_{max}^{(\mathcal{A})}$ pour un état non-initial correspond au nombre maximal de services pouvant remplacer Γ instances de services échouées. Dans le cas opportuniste, on considère tous les services susceptibles d'être sélectionnés puis remplacés. Pour ces deux problèmes, dans le pire cas,
$$\Upsilon_{max}^{(\mathcal{A})} \equiv \mathcal{O}(|\mathcal{Z}_{remp}|^{\Gamma}). \tag{4.81}$$
Néanmoins, l'échec simultané de toutes les instances de services pouvant être remplacées est généralement peu probable lorsque $\Gamma > 1$ tandis que toutes les combinaisons intégrant les mêmes services non remplaçables sont systématiquement considérées dans le cas opportuniste.
3. *Le choix d'une action a_i dans un état s_τ peut mener dans $|C_\tau^{(i)}|$ états différents.* Ces états possibles sont déterminés par le modèle de transition de la section 4.2. Le nombre maximum d'états successeurs possibles est
$$\Upsilon_{max}^{(C)} = \Lambda_n + 3. \tag{4.82}$$
Λ_n, défini par l'équation (4.67), correspond au nombre maximum de segments parcourus durant une transition et par conséquent aux nombres d'états d'échecs non-fatals possibles durant le déplacement. Viennent ensuite s'additionner le possible état d'échec non-fatal de la phase préparatoire, l'état de succès et l'état d'échec fatal.
4. *La profondeur maximale de cet arbre est $T - t$.* La profondeur maximale est atteinte lorsque par exemple, bien que l'échec soit inévitable, des actions autres que le rejet sont choisies. La profondeur minimale est 1 car l'échec fatal peut survenir dès la première transition et met un terme à l'exécution. En dehors de cet échec fatal, la profondeur $\Upsilon_{min}^{(pr)}$ est de 1 dans le cas non-opportuniste si la première transition est réussie. Dans le cas opportuniste, la profondeur minimale dépend du nombre de segments du chemin Λ_m et du nombre maximum de segments parcourus par transition M_m : $\Upsilon_{min}^{(pr)} = \max\left(1, \frac{\Lambda_m}{M_m}\right)$.
5. *Les renforcements sont obtenus uniquement dans les feuilles.*
6. *Sa taille dans le pire cas est donc approximativement de l'ordre de $\mathcal{O}\left((\Upsilon_{max}^{(\mathcal{A})} \times \Upsilon_{max}^{(C)})^{T-t}\right)$.*

La taille de l'arbre de recherche dans le pire cas a un effet dissuasif sur l'usage d'une méthode exacte naïve l'explorant de façon exhaustive. L'application par exemple de l'algorithme d'itération de valeur en horizon fini (cf. algorithme 1 de la sous-sous-section 4.1.1.2) énumère tous les états de l'arbre par niveau de profondeur en partant de $T - t$ afin de déterminer leurs valeurs optimales. Des méthodes d'exploration efficaces de cet arbre existent cependant, comme celle détaillée dans la section suivante.

8. Dans notre scénario, $\Gamma = 1$.

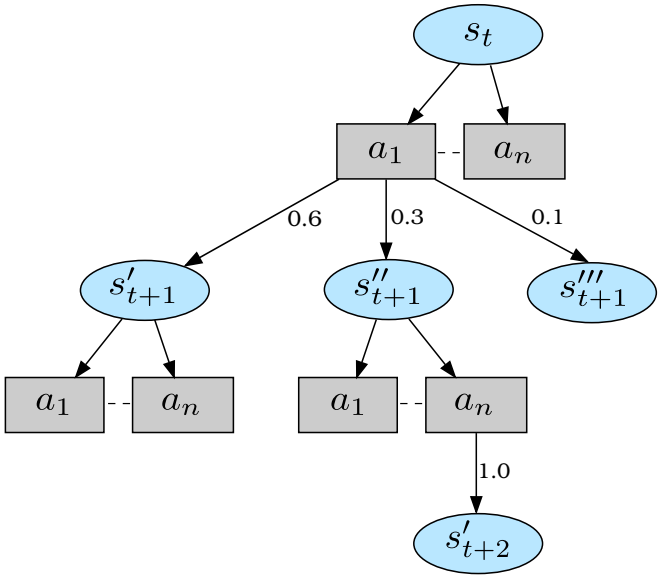


FIGURE 4.5 – *Représentation partielle de l'arbre de recherche.* Les états sont représentés par des ellipses et les actions par des rectangles. Les probabilités de transition suite au choix d'une action dans un état sont aussi représentées.

4.3.2 Fouille stochastique d'arbre

Cette sous-section s'intéresse à l'estimation des valeurs $Q(s_t, a)$ par une fouille stochastique de l'arbre de recherche. Cette fouille stochastique est une approche de type Monte-Carlo. De façon générale, les méthodes de Monte-Carlo sont utilisées pour estimer des intégrales en échantillonnant les variables aléatoires à intégrer puis en remplaçant ces intégrales par des sommes. Elles sont couramment utilisées pour estimer des espérances sous forme de moyennes. Au sein des MDPs, elles servent généralement à estimer l'espérance de gain dans un état en accumulant le gain reçu sur un épisode. Une telle espérance définit la valeur $V(s)$ d'un état (équation (4.3)). Il en va de même pour la valeur $Q(s, a)$ du couple état-action considérée dans cette section ; son estimation incrémentale est la suivante :

$$Q(s_\tau, a_\tau) \leftarrow Q(s_\tau, a_\tau) + \frac{1}{n(s_\tau, a_\tau)}[R_\tau - Q(s_\tau, a_\tau)]. \tag{4.83}$$

R_τ correspond au gain accumulé durant cet épisode depuis l'état s_τ. Dans nos MDPs, il s'agit simplement du renforcement reçu lors de la dernière transition de l'épisode ; aussi son indice est généralement omis. $n(s_\tau, a_\tau)$ correspond au nombre de fois où l'action a_τ a été choisie dans l'état s_τ.

Pour simuler un épisode en vue d'obtenir le gain R_t, il convient de distinguer deux niveaux d'échantillonnage :

1. *Échantillonnage des états successeurs.* Le simulateur échantillonne un état successeur tiré de l'ensemble $C_\tau^{(i)}$ du couple état-action (s_τ, a_i) en fonction de sa probabilité fournie par la fonction de masse $\bar{P}_\tau^{(i)}$. Cet échantillonnage permet de prioriser l'exploration des branches les plus probables au lieu de les traiter de façon indifférenciée comme le fait la méthode exacte précédemment évoquée.
2. *Échantillonnage des actions.* Les actions sélectionnées sont échantillonnées suivant une politique, qu'il s'agisse d'une politique par défaut ou de la politique de

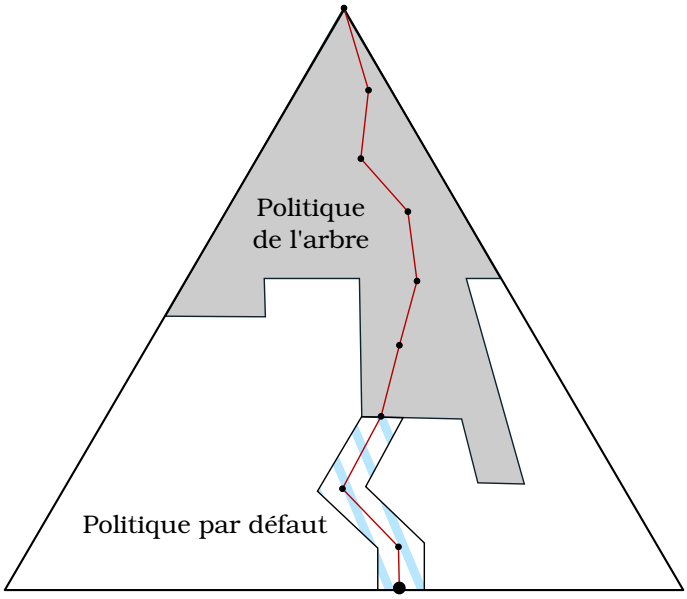


FIGURE 4.6 – *Politiques de l'arbre et par défaut.* La politique par défaut est appliquée sur les zones ne faisant pas partie de l'arbre, c'est-à-dire en dehors de la zone grise. Figure adaptée du tutoriel [Sebag 2012].

l'arbre (cf. figure 4.6). La politique par défaut est appliquée lorsque l'état courant ne fait pas partie de l'arbre ou que certaines actions associées à cet état n'en font pas partie. La politique par défaut la plus courante est la marche aléatoire uniforme : elle sélectionne les actions suivant une loi uniforme. Une telle politique est cependant insuffisante pour assurer une exploration efficace. Les méthodes de fouille stochastique d'arbre (*Monte-Carlo Tree Search*, MCTS) [Coulom 2006] exploitent les retours des épisodes précédents afin d'améliorer la politique d'exploration de l'arbre de recherche en se focalisant sur les actions les plus prometteuses. Parmi ces méthodes, nous proposons d'employer l'algorithme *Upper Confidence bounds applied to Trees* (UCT).

Upper Confidence bounds applied to Trees (UCT) UCT [Kocsis & Szepesvári 2006] est l'algorithme de type MCTS le plus fréquemment employé. Son introduction a eu un impact majeur dans le domaine des jeux à forte dimensionnalité comme le Go [Gelly & Silver 2007]. UCT effectue un échantillonnage sélectif des actions en formalisant chacune de ces sélections comme un problème de bandits multi-bras.

Bandit multi-bras À l'image d'un bandit-manchot de casino disposant de plusieurs bras, un problème de bandit multi-bras vise à maximiser le gain cumulé sur un nombre fixe d'interactions sans disposer d'information initiale sur l'espérance de gain associée à chaque bras. La performance d'un algorithme de bandit est évaluée selon le regret, c'est-à-dire selon la différence entre le gain obtenu à partir des actions choisies et le gain optimal obtenu en sélectionnant à chaque fois le bras optimal. Face à l'ignorance initiale et à l'objectif de minimisation du regret, ce problème doit effectuer un compromis entre l'exploration et l'exploitation. L'exploration vise à tester des bras méconnus en prenant le risque que les retours reçus soient mauvais. A contrario, l'exploitation est une tendance conservatrice privilégiant les meilleures actions connues quitte à ce qu'elles

soient sous-optimales.

Le principe d'optimisme face à l'incertitude (*Optimism in the Face of Uncertainty*, OFU) permet d'effectuer un compromis efficace entre ces deux tendances. Pour résoudre ce problème de bandit, ce principe incite à choisir l'action ayant la meilleure borne supérieure de confiance. L'algorithme *Upper Confidence Bound* (UCB1) [Auer et al. 2002] calcule cette borne de la façon suivante :

$$U_t(a) = r_t(a) + \mathcal{R}\sqrt{\frac{2\ln t}{n_t(a)}} \tag{4.84}$$

où $\mathcal{R}$ est la valeur maximale de renforcement immédiat possible et $n_t(a)$ est le compteur du nombre de fois où le bras a a été sélectionné.

Exploration de l'arbre et estimation des valeurs UCT s'appuie sur UCB pour calculer ses bornes supérieures de confiance. Il associe à chaque couple état-action (s, a) de l'arbre $\mathcal{T}$ un compteur $n(s, a)$ du nombre de visites et une valeur $Q(s, a)$.

Initialement l'arbre $\mathcal{T}$ ne contient que l'état courant s_t. Chaque épisode se déroule de la façon suivante.

1. *Un épisode commence dans l'état s_t correspondant à la $j^{\text{ème}}$ sélection.* Initialement,

$$\tau \leftarrow j; \quad \tilde{s}_\tau \leftarrow s_t \tag{4.85}$$

2. Tant que l'état $\tilde{s}_\tau$ n'est pas terminal et que $\tau < T$:
   (a) *S'il existe dans l'état courant des actions qui ne sont pas représentées dans l'arbre : $\exists a \in \mathcal{A}_{\tilde{s}_\tau}$ tel que $(\tilde{s}_\tau, a) \notin \mathcal{T}$.* Dans ce cas, une politique de type marche aléatoire est utilisée pour ces actions :

$$\pi_{uni}(\tilde{s}_\tau, a) = \begin{cases} |\{a' \in \mathcal{A}_{s_\tau} | (\tilde{s}_\tau, a') \notin \mathcal{T}\}|^{-1} & \text{si } (\tilde{s}_\tau, a) \notin \mathcal{T} \\ 0 & \text{sinon} \end{cases} \tag{4.86}$$

   Une action $a_\tau^{(i)}$ est ensuite tirée de cette politique stochastique $\pi_{uni}(\tilde{s}_\tau, .)$:

$$a_\tau^{(i)} \sim \pi_{uni}(\tilde{s}_\tau, .) \tag{4.87}$$

   et le couple $(\tilde{s}_\tau, a_\tau^{(i)})$ est ajouté à l'arbre $\mathcal{T}$. Ce nouveau couple état-action se voit affecté un compteur $n(\tilde{s}_\tau, a_\tau^{(i)})$ du nombre de visites et une valeur $Q(\tilde{s}_\tau, a_\tau^{(i)})$. Les valeurs initiales sont :

$$n(\tilde{s}_\tau, a_\tau^{(i)}) \leftarrow 1 \tag{4.88}$$
$$Q(\tilde{s}_\tau, a_\tau^{(i)}) \leftarrow 0 \tag{4.89}$$

   (b) Sinon, l'action est choisie via une variante d'UCB1 :

$$Q^{\oplus}(\tilde{s}_\tau, a) = Q(\tilde{s}_\tau, a) + c\sqrt{\frac{\ln n(\tilde{s}_\tau)}{n(\tilde{s}_\tau, a)}} \tag{4.90}$$
$$a_\tau^{(i)} = \pi^{\oplus}(\tilde{s}_\tau) = \arg\max_{a \in \mathcal{A}_s} Q^{\oplus}(\tilde{s}_\tau, a) \tag{4.91}$$
$$n(\tilde{s}_\tau, a_\tau^{(i)}) \leftarrow n(\tilde{s}_\tau, a_\tau^{(i)}) + 1 \tag{4.92}$$

   avec $n(\tilde{s}_\tau) = \sum_{a' \in \mathcal{A}_{s_\tau}} n(\tilde{s}_\tau, a')$, $\pi^{\oplus}$ la politique déterministe de l'arbre et c le facteur d'exploration.

(c) *L'état suivant est obtenu par échantillonnage.*

$$s_{\tau+1} \sim \bar{P}_{\tau}^{(i)}(.) \tag{4.93}$$

$$\tau \leftarrow \tau + 1 \tag{4.94}$$

3. *À la fin de l'épisode.* Seul le dernier renforcement reçu est pris en compte car tous les autres sont nuls (cf. équation (4.11)). Par conséquent, pour tous les couples états-actions rencontrés durant cet épisode reçoivent le gain R et leurs valeurs peuvent être mises à jour suivant la formule (4.83). Ce gain est

$$R = r_{s_{\tau-1}}(a_{\tau-1}^{(i)}, \tilde{s}_{\tau}) \tag{4.95}$$

avec $\tilde{s}_{\tau-1}$ et $\tilde{s}_{\tau}$ les deux derniers états de l'épisode et $a_{\tau-1}^{(i)}$ la dernière action sélectionnée.

Cet algorithme peut être arrêté à n'importe quel moment. Dans notre approche, l'arrêt a lieu lorsque la durée η a été atteinte. Les résultats sont présentés dans la section suivante.

4.4 Simulations

L'évaluation de notre approche s'appuie sur le scénario décrit dans la sous-sous-section 3.1.3, consacré au déplacement du robot R1 au sein d'un étage.

Cette section introduit la modélisation de l'environnement et des organigrammes dans notre simulateur (4.4.1) avant d'effectuer diverses expériences (4.4.2, 4.4.3, 4.4.4, 4.4.5, 4.4.6 et 4.4.7).

4.4.1 Modélisation de l'environnement et des organigrammes

4.4.1.1 Bâtiment

L'étage de ce scénario, représenté par la figure 4.7, est composé de sept pièces. Il a une largeur et une longueur de 8 et 10m. La grille utilisée par les modèles de dynamique est de dimension 7 par 7. Ses lignes sont alignées avec les murs (cf. figure 4.8).

4.4.1.2 Organigrammes

Six services de positionnement fixes sont déployés dans l'environnement. La modélisation du comportement des organigrammes résultants est minimale et arbitraire. Les performances des six organigrammes résultants sont modélisées comme étant constantes dans chacune des pièces. Toutes les pièces, à l'exception des WC et la chambre 2, sont couvertes par au moins un de ces services de positionnement. Certains d'entre eux sont disponibles dans des pièces connexes et permettent ainsi la transition d'une pièce à l'autre sans échec.

Le robot assistant est introduit dans l'expérience 4.4.7 et fournit un service de positionnement dans toutes les pièces sauf les WC. Cette dernière pièce est par conséquent la seule à n'être couverte par aucun service de positionnement.

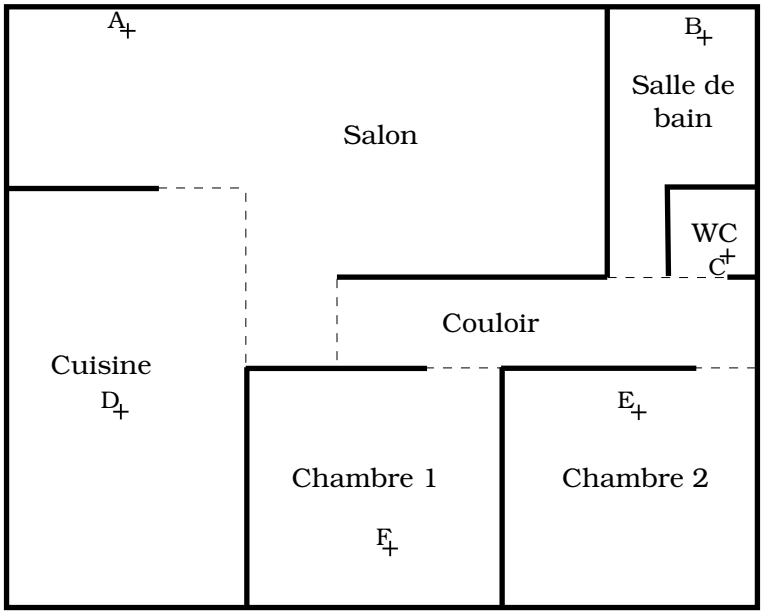


FIGURE 4.7 – Étage du scénario

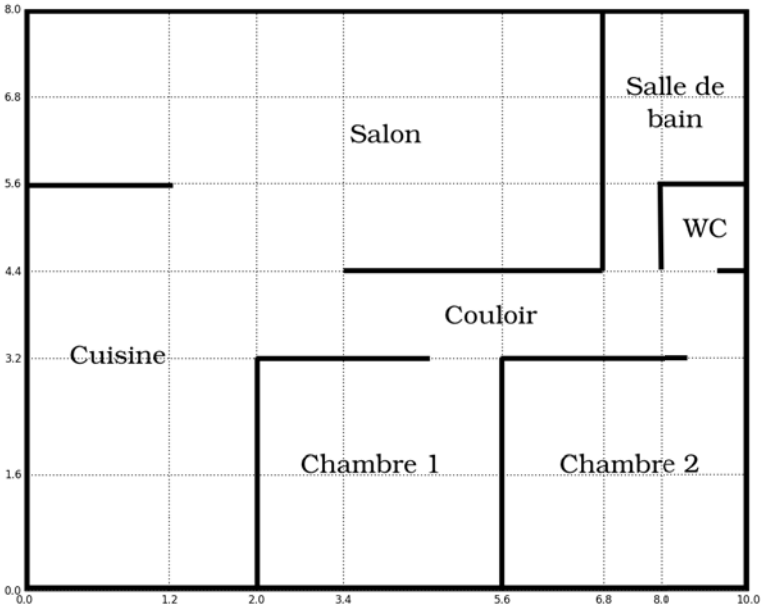


FIGURE 4.8 – Grille utilisée par les modèles de dynamique

Coût et bruit L'activation des services de déplacement du robot R1 et de positionnement entraînent deux coûts fixes par unité de temps. Ces coûts s'additionnent et forment le coût de l'organigramme. Le bruit de chaque organigramme actif est modélisé comme étant constant pour chaque unité de temps. Les valeurs de coût de bruit des différents organigrammes sont présentées dans la table 4.1.

Service de positionnement	Coût (unité par s)	Bruit (dB)
S1	50	56
S2	50	56
S3	80	53
S4	80	53
S5	50	65
S6	35	53
Robot assistant (Ra)	100	56

TABLE 4.1 – Coûts et bruits par seconde des différents organigrammes

Phase préparatoire La sélection d'un nouveau service de positionnement est modélisée comme entraînant un arrêt de 5s. Le bruit du robot R1 en position d'arrêt est supposé rester nul durant cette période tandis que le coût lié à son activation reste constant. Dans l'expérience 4.4.7, l'introduction du service de positionnement d'un robot assistant augmente cette durée d'arrêt et introduit un coût et un bruit supplémentaires dus à son déplacement vers le robot R1.

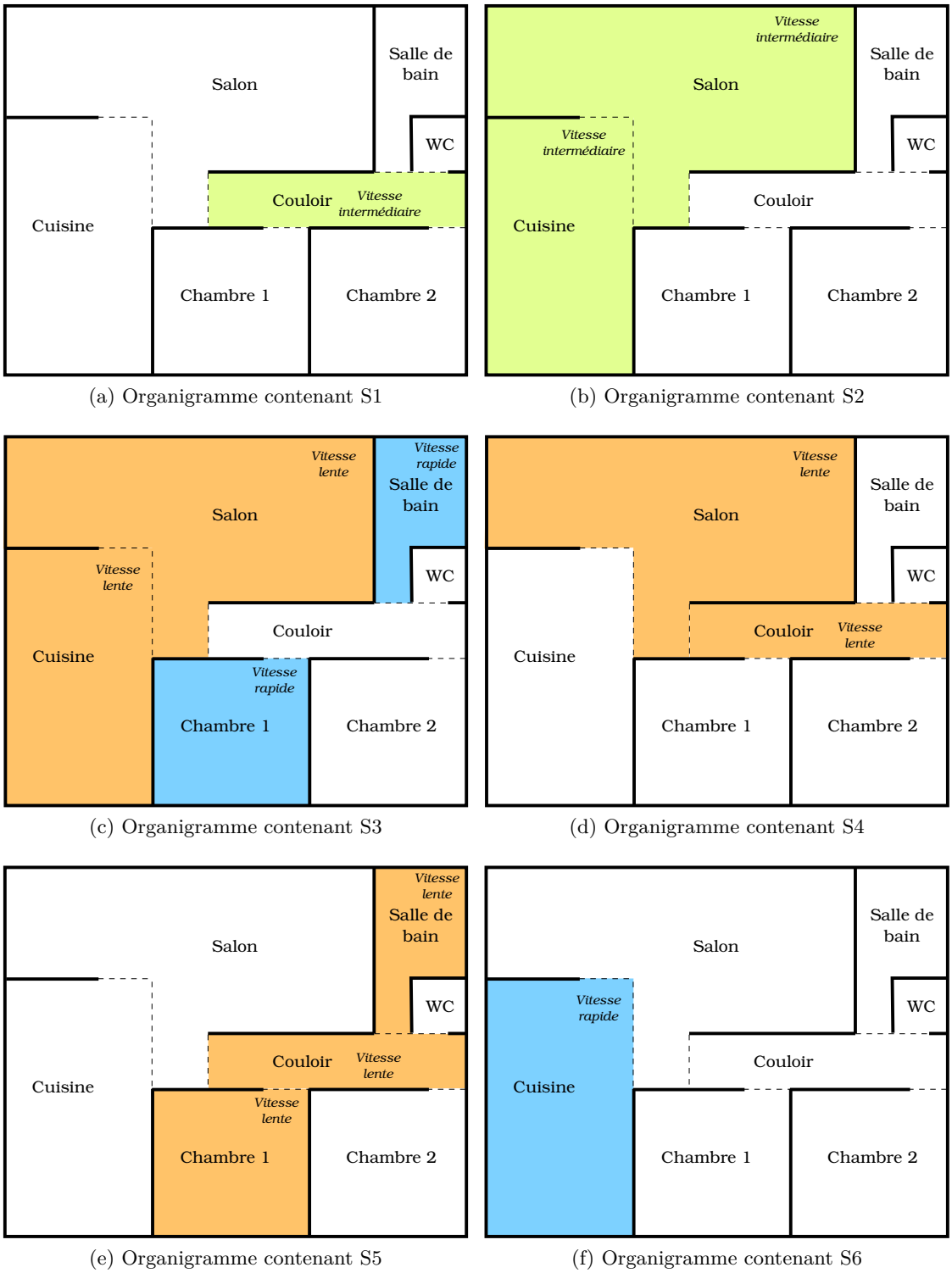
Vitesse Ces performances sont spécifiées dans le tableau 4.2 et sont représentées par les figures 4.9 et 4.10.

Service de positionnement	Vitesse (m/s)						
	Salon	Cuisine	Couloir	Salle de bains	Chambre 1	Chambre 2	WC
S1	-	-	0.23	-	-	-	-
S2	0.23	0.23	-	-	-	-	-
S3	0.20	0.20	-	0.3	0.3	-	-
S4	0.20	-	0.20	-	-	-	-
S5	-	-	0.20	0.20	0.20	-	-
S6	-	0.3	-	-	-	-	-
Ra	0.10	0.10	0.10	0.10	0.10	0.10	-

TABLE 4.2 – Vitesses de déplacement des différents organigrammes

Échec Le temps de détection d'un échec est fixé à $15s$. Durant cette période, le coût et le bruit moyens restent les mêmes que durant le déplacement. Aucun échec fatal ne survient.

Normalisation des métriques en paramètres de QdS Les métriques de bruit moyen, de coût moyen et de durée sont normalisées à partir des fonctions (3.10), (3.11) et (3.12). Leurs valeurs de références ou paramètres constants sont :



(a) Organigramme contenant S1

(b) Organigramme contenant S2

(c) Organigramme contenant S3

(d) Organigramme contenant S4

(e) Organigramme contenant S5

(f) Organigramme contenant S6

FIGURE 4.9 – Vitesses de déplacement des différents organigrammes faisant appel à des services de positionnement fixes

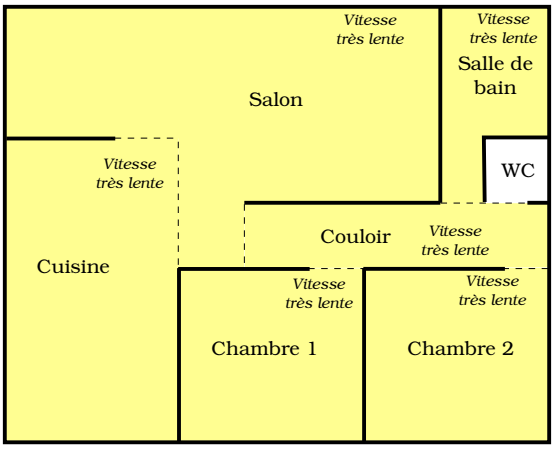


FIGURE 4.10 – Vitesse de déplacement de l'organigramme faisant appel au service de positionnement du robot assistant

1. *Coût moyen :* $r_{coût} = 30$ unités par seconde.
2. *Bruit moyen :* $r_{bruit} = 50$ dB.
3. *Vitesse maximale :* $v_{max} = 0.3$ m/s.

Fonctions d'agrégation en un score Les expériences 4.4.2 et 4.4.3 se restreignent à une seule métrique, la durée d'exécution. Le score associe un poids de 1.0 à la valeur de son paramètre de QdS et des poids nuls aux autres paramètres de QdS.

4.4.1.3 Initialisation des paramètres des sous-modèles

Le principe général adopté pour l'initialisation des paramètres des sous-modèles de la fonction de transition est celui de l'optimisme face à l'incertitude. Aussi, les valeurs d'initialisation sont :

1. *Probabilités de succès des phases de préparation et de déplacement :* $\mu^{\sharp}_{ins} = \mu^{\sharp}_{l\to n} = 1.0$.
2. *Probabilités de non-fatalité d'un échec lors de ces deux phases :* $\mu^{\triangledown}_{ins} = \mu^{\triangledown}_{l\to n} = 1.0$.
3. *Vitesse :* $\mu^{(\delta)+}_{l\to n} = 2.0 \times v_{max}$.
4. *Durée de détection de l'échec du service de positionnement :* $\mu^{(\delta)-}_{l\to n} = 0.0$
5. *Durée d'instanciation :* $\mu^{(\delta)}_{ins,b_t} = 0.0$
6. *Métriques de bruit et de coût des deux phases* ($\alpha \in \{-,+\}$) *:* $\mu^{(k)}_{ins,b_t} = \mu^{(k)\alpha}_{l\to n} = 0.0$

Le pas de mise à jour est le même pour tous ces paramètres $\beta = 0.3$.

Cas de la phase d'approche du robot assistant La prédiction de performance lors de cette phase d'approche sollicite le modèle de dynamique d'un organigramme spécifique correspondant au déplacement du robot assistant. Comme indiqué dans la sous-sous-section 3.1.3, cet organigramme fait toujours appel à un même service de positionnement embarqué sur le robot mobile. Nous supposons que le modèle de dynamique correspondant à cet organigramme a été appris au préalable. La vitesse d'approche du

robot assistant est constante et vaut 0.23 m/s. Son bruit moyen et son coût moyen ont également des valeurs constantes, respectivement 55 dB et 40 unités par seconde.

4.4.2 Requête fixe sans remplacement opportuniste

La première expérience consiste à répéter le déplacement du robot de la position A à la position B (cf. figure 4.7). Ce déplacement implique la traversée du salon, du couloir et de la salle de bains. Étant donné qu'aucun service de positionnement ne couvre ces trois pièces, au moins un remplacement doit avoir lieu. Le remplacement opportuniste n'est pour l'instant pas considéré. De ce fait, tous les remplacements font suite à un échec.

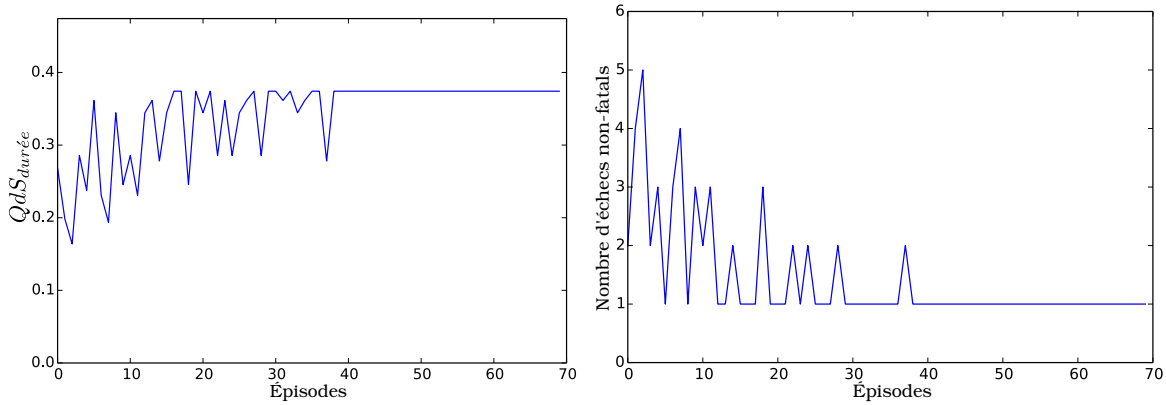
Le tableau 4.3 détaille les séquences et les scores, prédits ou obtenus, des premiers épisodes d'une même simulation avec des paramètres $c = 0.3$ et $\eta = 1$s. De part le faible pas de mise à jour ($\beta = 0.3$), les probabilités de succès restent importantes suite à un premier échec et les paramètres de succès des métriques, qui n'ont pu être mis à jour faute de succès, restent très optimistes. De ce fait, tous les organigrammes sont sélectionnés à plusieurs reprises en dehors de leur zone de couverture, jusqu'à ce que leurs probabilités de succès deviennent faibles. On constate néanmoins une convergence vers les deux meilleures séquences `(S4 S3)` et `(S2 S5)` avec des scores respectifs de 0.437 et de 0.425. L'échec étant coûteux en temps, il est évité autant que possible, comme le montrent les sous-figures 4.22a et 4.22b.

Si les comportements des organigrammes sont déterministes, la prédiction de valeur est, quant à elle, stochastique. Néanmoins, l'écart-type observé après convergence est très faible (de l'ordre de 10^{-3} sur les dix dernières exécutions). La figure 4.12 présente l'évolution des valeurs Q prédites et les bonus lors de la prédiction effectuée dans l'état initial au cours du dernier épisode. La valeur de c engendre une exploration qui permet une bonne estimation des valeurs des trois premiers organigrammes S2, S3 et S4. Le meilleur organigramme se démarque toutefois des autres en nombre de sélections, comme le montre la figure 4.13. Lors des derniers épisodes de cette expérience, ces estimations restent légèrement inférieures aux valeurs observées. Ceci est dû à l'exploration d'actions sous-optimales qui impactent directement le calcul des moyennes que sont les valeurs Q.

Les figures 4.14 et 4.15 représentent les valeurs des paramètres de vitesse et durée de détection d'échec à l'issue du 70^e épisode. Les estimations des paramètres de vitesse des organigrammes contenant le service S2, S3, S4 ou S5 ont convergé alors qu'un seul paramètre de l'organigramme de S1 est proche de sa valeur cible 0.23 m/s. En effet, deux segments du chemin planifié correspondent à la première case du côté gauche du couloir. Ceci entraîne un doublement des mises à jour des paramètres associés à cette case en cas de succès que les autres cases du couloir. Cette différence de fréquence est notable pour l'organigramme de S1 car celui-ci a été peu sélectionné dans le couloir. Par ailleurs, on peut distinguer, au travers de ces deux figures, les trois cases correspondant aux positions où les sélections ont lieu : tous les organigrammes y ont mis à jour leurs paramètres d'échec ou de succès. Ces positions sont la position initiale et les points de jonction entre deux pièces parcourus.

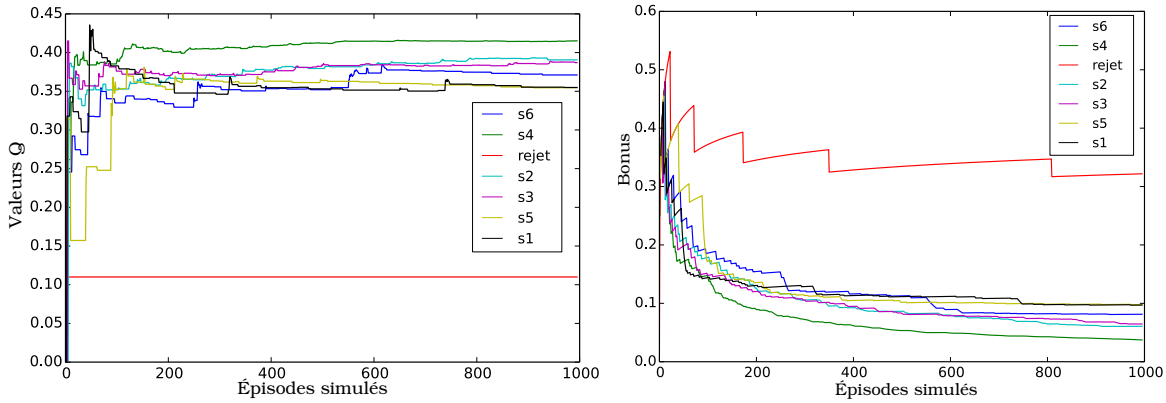
Itération	Séquence observée	Séquence prédite	Score observé	Score prédit
1	s4 s6 s5	s4	0.341	1.000
2	s1 s3 s2 s1 s3	s1	0.278	1.000
3	s6 s5 s6 s5 s2 s5	s6	0.248	1.000
4	s4 s2 s3	s4	0.357	0.995
5	s3 s6 s1 s3	s3	0.314	0.983
6	s2 s5	s2 s5	0.425	0.966
7	s1 s4 s2 s3	s1 s6	0.308	0.902
8	s5 s1 s6 s4 s3	s5 s5 s5 s6 s3	0.274	0.839
9	s3 s5	s3 s5	0.410	0.777
10	s2 s1 s6 s3	s2 s1 s6	0.321	0.685
11	s5 s4 s3	s5 s5 s5 s5	0.357	0.662
12	s6 s1 s4 s3	s6 s1 s4 s6	0.308	0.591
13	s3 s5	s3 s5	0.410	0.607
14	s2 s5	s2 s5	0.425	0.558
15	s5 s2 s5	s5 s4 s3	0.350	0.527
16	s3 s5	s3 s6 s1 s6 s1	0.410	0.514
17	s4 s3	s4 s3	0.437	0.522
18	s4 s3	s4 s3	0.437	0.487
19	s1 s2 s1 s3	s1 s6 s5 s6	0.321	0.484
20	s4 s3	s4 s3	0.437	0.456
21	s3 s5	s3 s5	0.410	0.466
22	s4 s3	s4 s3	0.437	0.441
23	s6 s4 s3	s6 s6 s6 s6	0.357	0.461
24	s2 s5	s2 s6 s5	0.425	0.460
25	s5 s4 s3	s5 s4 s6 s3	0.357	0.432
26	s3 s5	s3 s6 s6 s6	0.410	0.417
27	s2 s5	s2 s5	0.425	0.428
28	s4 s3	s4 s3	0.437	0.429
29	s1 s4 s3	s1 s4 s3	0.357	0.428
30	s4 s3	s4 s3	0.437	0.416
31	s4 s3	s4 s3	0.437	0.408
32	s2 s5	s2 s5	0.425	0.420
33	s4 s3	s4 s3	0.437	0.415
34	s3 s5	s3 s5	0.410	0.412
35	s2 s5	s2 s6 s5	0.425	0.423
36	s4 s3	s4 s3	0.437	0.409
37	s4 s3	s4 s3	0.437	0.414
38	s2 s6 s5	s2 s6 s5	0.350	0.417
39	s4 s3	s4 s3	0.437	0.413
40	s4 s3	s4 s3	0.437	0.408
65	s4 s3	s4 s3	0.437	0.414
66	s4 s3	s4 s3	0.437	0.411
67	s4 s3	s4 s3	0.437	0.410
68	s4 s3	s4 s3	0.437	0.415
69	s4 s3	s4 s3	0.437	0.409
70	s4 s3	s4 s3	0.437	0.415

TABLE 4.3 – *Déroulement des 70 premiers épisodes de l'expérience 4.4.2.* Seules les prédictions de séquence et de score ayant eu lieu dans l'état initial sont représentées.



(a) $QdS_{durée}$ observée (b) Nombre d'échecs détectés par épisode

FIGURE 4.11 – *Durée observée et échecs non-fatals survenus lors de l'expérience 4.4.2.*



(a) Valeurs Q prédites (b) Bonus de la borne UCB

FIGURE 4.12 – *Valeurs Q et bonus lors de la première sélection de l'épisode 70 de l'expérience 4.4.2.*

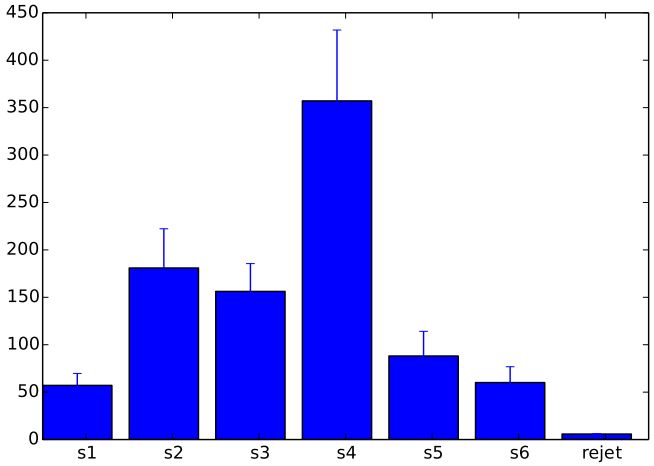
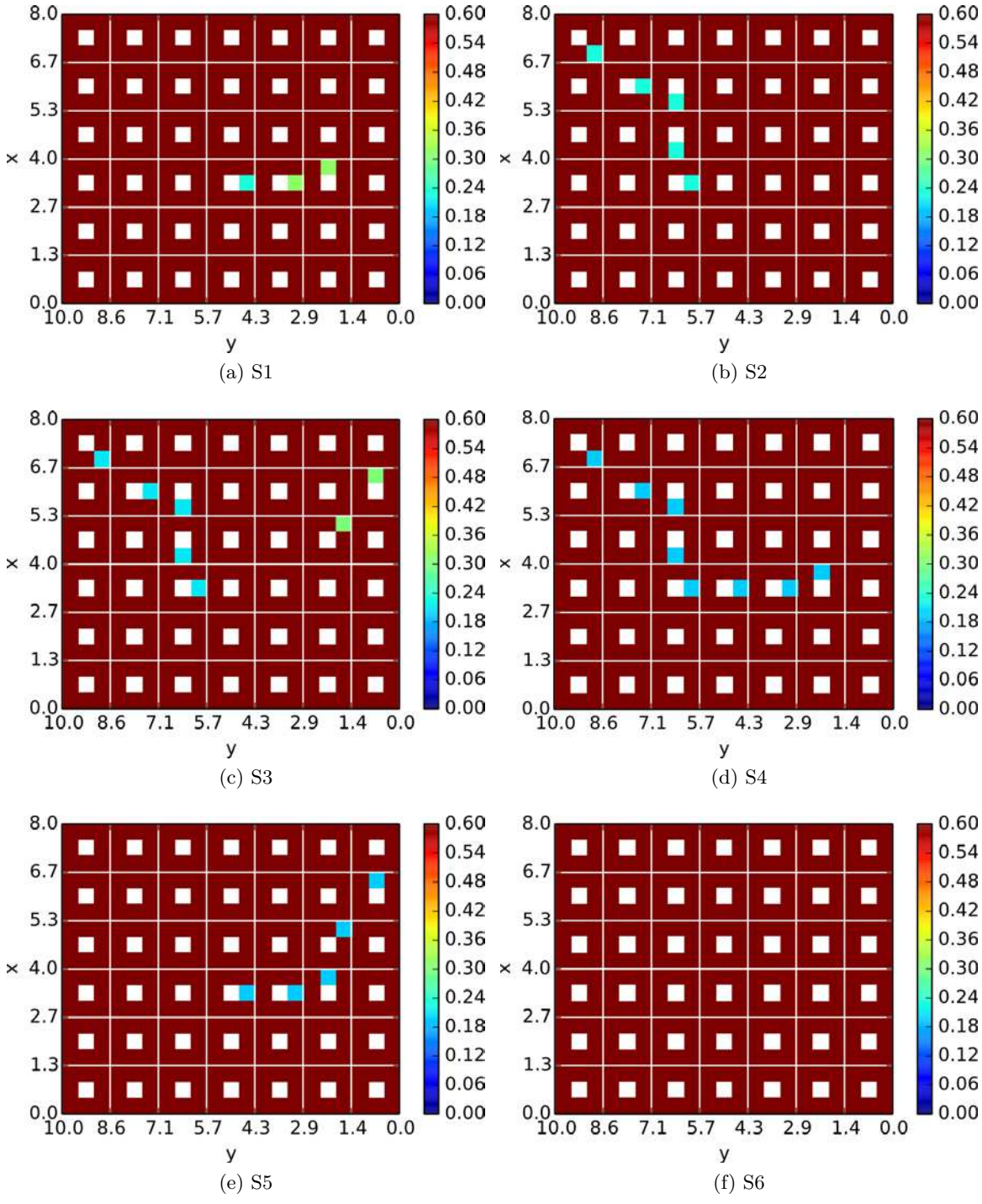


FIGURE 4.13 – *Organigrammes sélectionnés dans l'état initial lors des phases de simulation des dix derniers épisodes de l'expérience 4.4.2.*

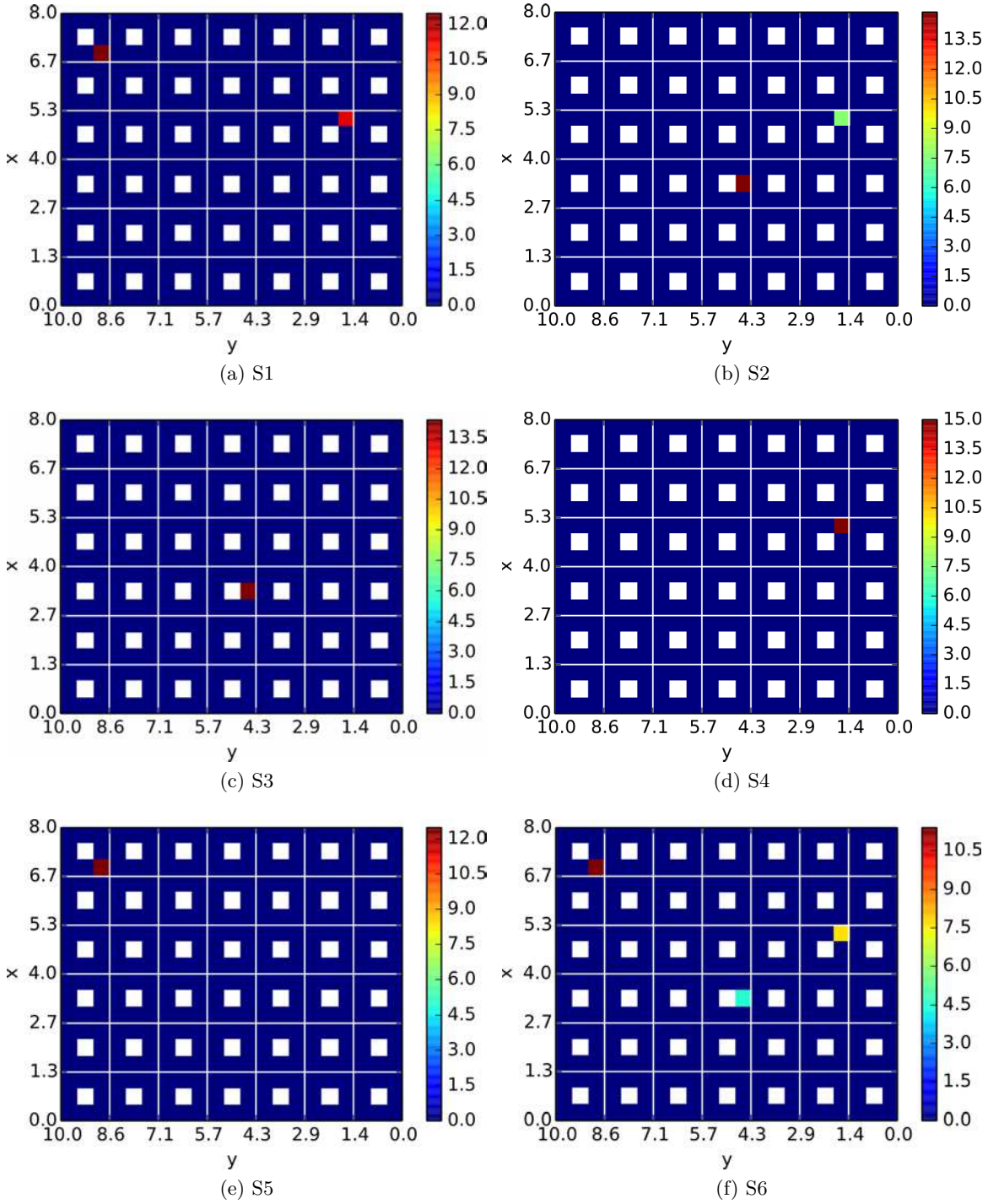


(a) S1 (b) S2

(c) S3 (d) S4

(e) S5 (f) S6

FIGURE 4.14 – *Paramètres de vitesse à l'issue du* 70^e *épisode de l'expérience 4.4.2.* Chaque case de la grille est divisée en 9 sous-cases. Seule la sous-case centrale (en blanc) ne représente aucun paramètre. Chacune des autres sous-cases correspond au paramètre de vitesse associé au déplacement de la case vers l'autre case connexe à cette sous-case.



(a) S1 (b) S2

(c) S3 (d) S4

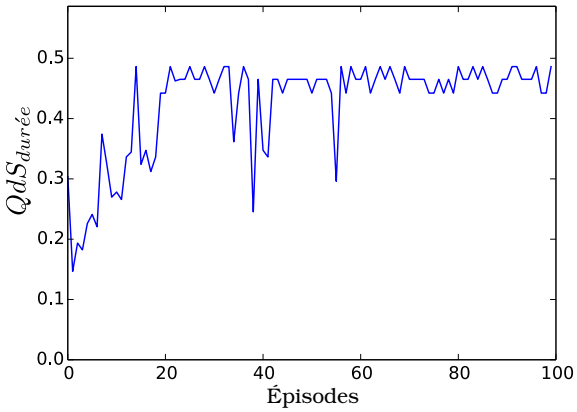
(e) S5 (f) S6

FIGURE 4.15 – *Paramètres de durée d'échec à l'issue du 70^e épisode de l'expérience 4.4.2.*

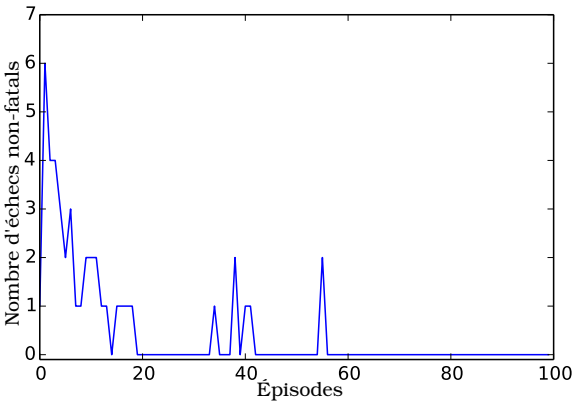
4.4.3 Requête fixe avec remplacement opportuniste

Cette seconde expérience reprend la même requête que l'expérience précédente et effectue des sélections opportunistes tous les deux segments de niveau 2. Les paramètres par défaut de cette expérience sont $c = 0.3$ et $\eta = 2$s. De même, seule la durée est considérée.

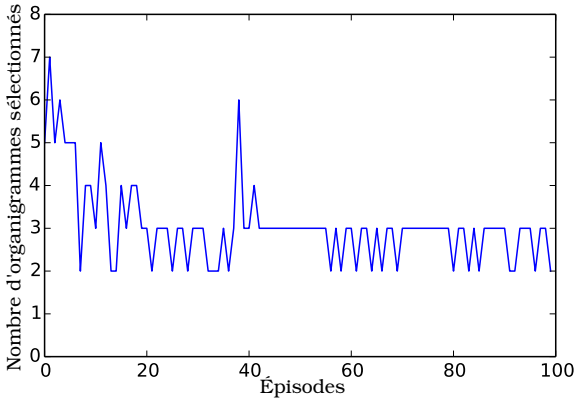
L'introduction de remplacements opportunistes permet d'améliorer la performance (cf. figure 4.16a) : les meilleures séquences observées sont `(S4 S4 S4 S4 S4 S3)`, `(S2 S2 S4 S4 S4 S3)` et `(S3 S3 S4 S4 S4 S3)`. Elles ont respectivement des scores de 0.538, 0.519 et 0.498. Les échecs non-fatals sont désormais évités (cf. sous-figure 4.16b). Ces trois meilleures séquences impliquent la sélection de deux ou de trois organigrammes au cours d'un épisode (cf. la sous-figure 4.16c).



(a) $QdS_{durée}$ observée



(b) Nombre d'échecs détectés par épisode



(c) Nombre d'organigrammes sélectionnés par épisode

FIGURE 4.16 – *Durée observée, échecs non-fatals et sélections survenus lors de l'expérience 4.4.3 avec* $c = 0.3$. Seules les sélections impliquant l'instanciation d'un nouvel organigramme sont comptabilisées dans la sous-figure 4.16c.

Les trois meilleures séquences effectuent toutes un choix différent dans l'état initial. De part les faibles écarts entre les scores de ces séquences, la profondeur de l'arbre (6 sélections) et le temps de calcul limité (2s), le service de positionnement sélectionné dans l'état initial oscille entre S2, S3 et S4 (cf. figure 4.16a). Les sélections dans les états postérieurs donnent les mêmes résultats, sachant le premier organigramme sélectionné, lors des 40 derniers épisodes. La fouille de l'arbre de recherche est en effet plus aisée car

celui-ci est plus restreint et le coût engendré par le remplacement d'un organigramme déjà sélectionné favorise son maintien.

4.4.4 Requêtes aléatoires uniformes

Dans cette troisième expérience, les 400 premières requêtes ont des positions initiales et des destinations générées aléatoirement suivant une loi uniforme. Les sélections sont de type non-opportuniste et les paramètres sont $c = 0.3$ et $\eta = 1$s. Tous les paramètres de QdS sont considérés et aucun humain n'est supposé être à proximité d'un chemin de déplacement du robot. La performance des sélectionneurs est évaluée à différents stades de leur apprentissage à partir de l'ensemble de 7 requêtes prédéfinies correspondant aux déplacements entre des points représentés dans la figure 4.7.

Les figures 4.18 et 4.19 représentent respectivement les valeurs des paramètres de vitesse et de probabilité de succès à l'issue du 407^e épisode d'une des simulations de cette expérience. Les pièces couvertes par chacun des services de positionnement se distinguent la figure 4.18 par des valeurs de vitesse inférieures à la valeur par défaut. Dans ces zones de couverture, beaucoup de paramètres ont encore des valeurs fortement optimistes, dues à la valeur modérée de β et à la faible fréquence de parcours de certaines cases. Suivant la modélisation de notre simulateur, les probabilités de succès réelles sont binaires (elles valent 0 ou 1.0) car aucune case de la grille ne se situe entre deux pièces. Néanmoins, peu d'entre elles ont des valeurs inférieures à 0.1. Ces valeurs faibles se retrouvent principalement aux points de jonction aux frontières des zones de couverture des services de positionnement fréquemment sélectionnés mais aussi dans une pièce hors de toute couverture, la chambre 2. La présence de nombreuses valeurs comprises entre 0.3 et 0.7 est le résultat du compromis exploration-exploitation en vigueur au niveau des MDPs. L'exploration est favorisée par les valeurs initiales optimistes des modèles de dynamique locale, tandis que le sélectionneur se comporte de façon gloutonne *(greedy)* en choisissant le meilleur organigramme au vu de sa connaissance. Dès que la probabilité de succès est trop faible pour offrir à l'organigramme d'un service de positionnement hors de sa zone de couverture une valeur Q supérieure aux autres, celui-ci n'est plus sélectionné et ses paramètres ne sont plus mis à jour dans cette case. Ceci explique la présence de nombreuses valeurs comprises entre 0.3 et 0.7.

Il apparaît aussi nettement que certaines zones sont plus régulièrement fréquentées que d'autres et ont par conséquent des paramètres plus à jour. Ces zones sont principalement le couloir et les entrées des pièces, comme le montre la figure 4.17 [9].

Requêtes de test La performance des sélectionneurs est évaluée à différents stades de leur apprentissage. Cette évaluation repose sur 7 requêtes prédéfinies correspondant aux déplacements entre des points représentés dans la figure 4.7. Les scores obtenus pour chacune de ces requêtes sont présentés dans la figure 4.20.

Les requêtes `b,f,g` ont leur position initiale ou leur destination dans une pièce non couverte par un service de positionnement (C est dans les toilettes et E dans la chambre 2). Il en résulte un score nul sauf pour la requête `f` à partir de l'épisode 120 qui est alors rejetée. Ce rejet a lieu dès lors que la probabilité de succès sollicitée dans la première case de la chambre 2 accessible depuis le couloir devienne suffisamment faible pour que

9. Elle ne représente pas la chambre 2 et les WC car elle ne comptabilise que les segments parcourus avec succès.

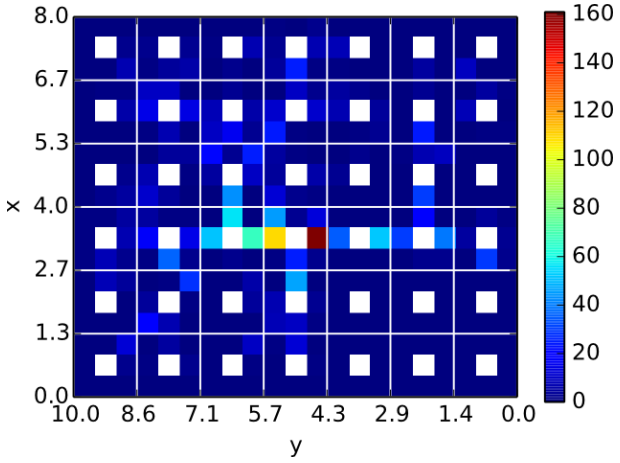


FIGURE 4.17 – *Nombre de segments parcourus avec succès lors de l'expérience 4.4.4.*

l'espérance de gain de tout déplacement soit inférieure à α_{rej}. En revanche, les paramètres de probabilité de succès du segment de sortie de la chambre 2 emprunté par la requête `g` et du segment entrant dans les WC n'ont pas été suffisamment mis à jour pour que le rejet soit sélectionné.

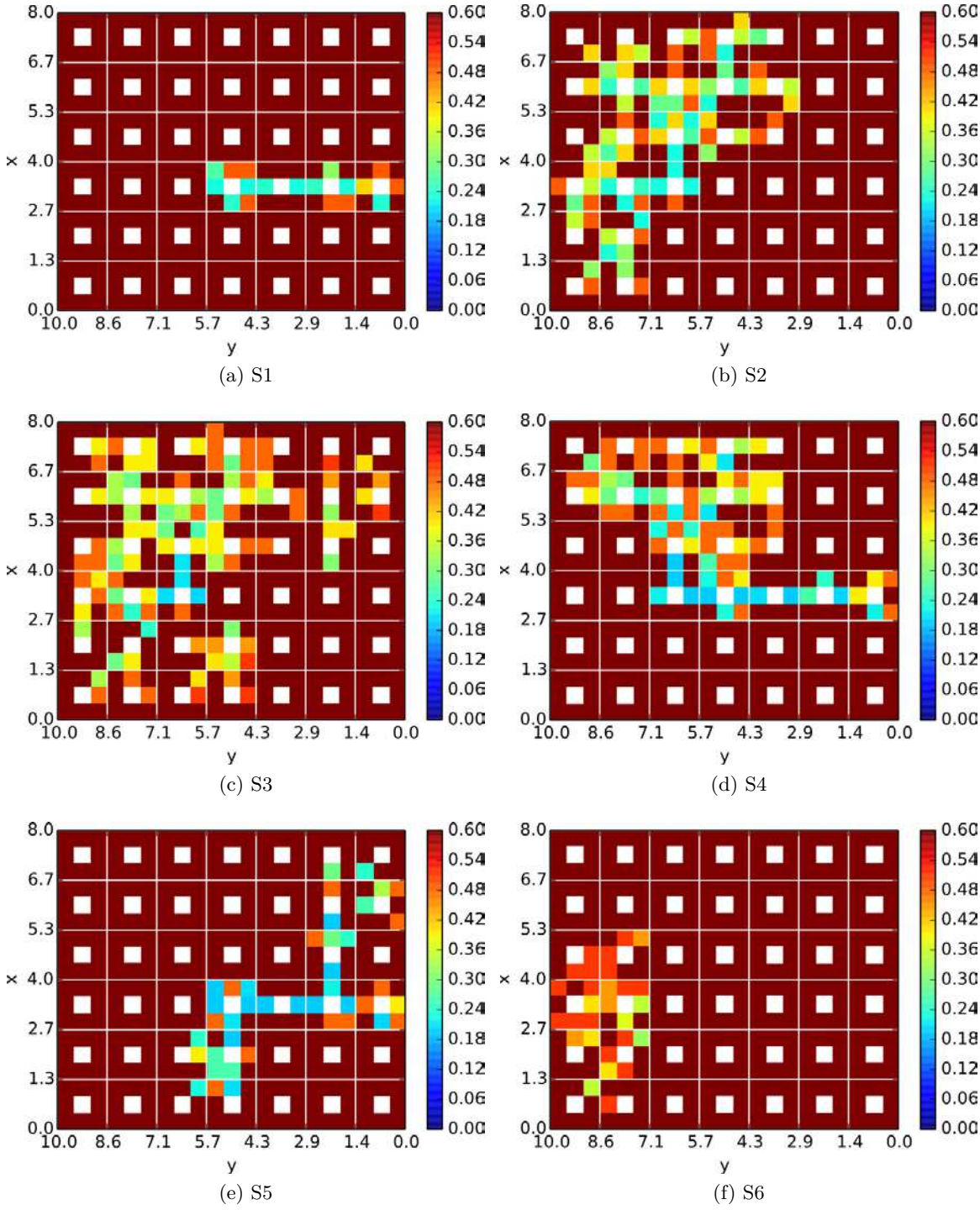
La requête `e` admet une solution simple que le sélectionneur choisit à partir de l'épisode 40. Il s'agit du choix de l'organigramme du service S5 qui ne provoque aucun échec nécessitant un remplacement sur le trajet allant de la chambre 1 à la salle de bains. Enfin, la performance de la requête `a` est plus stable que celles des requêtes `c` et `d` car son déplacement emprunte plusieurs sous-chemins fréquentés : le corridor de sortie de la salle de bains, le couloir ainsi que l'entrée du salon.

4.4.5 Modèles de dynamique locale connus

Cette expérience s'intéresse aux effets des paramètres de l'algorithme de fouille stochastique, le facteur d'exploration c et le temps de simulation η. Les modèles de dynamique locale sont supposés être connus. 20 requêtes de test sont générées aléatoirement, avec des positions initiales et des destinations tirées d'une loi uniforme. Aucun humain n'est à proximité des chemins parcourus par les robots.

Dans un premier temps, n est fixé à 2s et des valeurs de c allant de 0 à 1.6 sont comparées. La sous-figure 4.21a présente les scores moyens obtenus pour chaque valeur de c durant l'exécution des 20 requêtes. Dans ces conditions, la valeur de c semble avoir peu d'importance lorsque $c \geq 0.2$. Ceci semble indiquer que la découverte de bonnes solutions ne nécessite pas, dans de nombreux cas, une exploration importante. Ensuite, l'effet de la durée de simulation η sur le score moyen est étudié pour trois valeurs de c (cf. sous-figure 4.21b). Cette durée est, bien entendu, dépendante de l'implémentation et du processeur utilisé. Le sélectionneur a été implémenté en Python [10] et exécuté dans un processus à un seul thread sur un processeur i5 M430 fréquencé à 2.27 GHz. La construction de l'arbre de recherche est l'étape la plus coûteuse dans cette implémentation. Des mécanismes de cache ont été mis en place pour limiter son impact. Toutefois, d'importantes optimisations de cette implémentation restent possibles. Au vu de ces résultats, la durée $\eta = 2$s est retenue comme compromis entre le temps de sélection et la qualité

10. Des détails sur cette implémentation sont disponibles en annexe B.2.

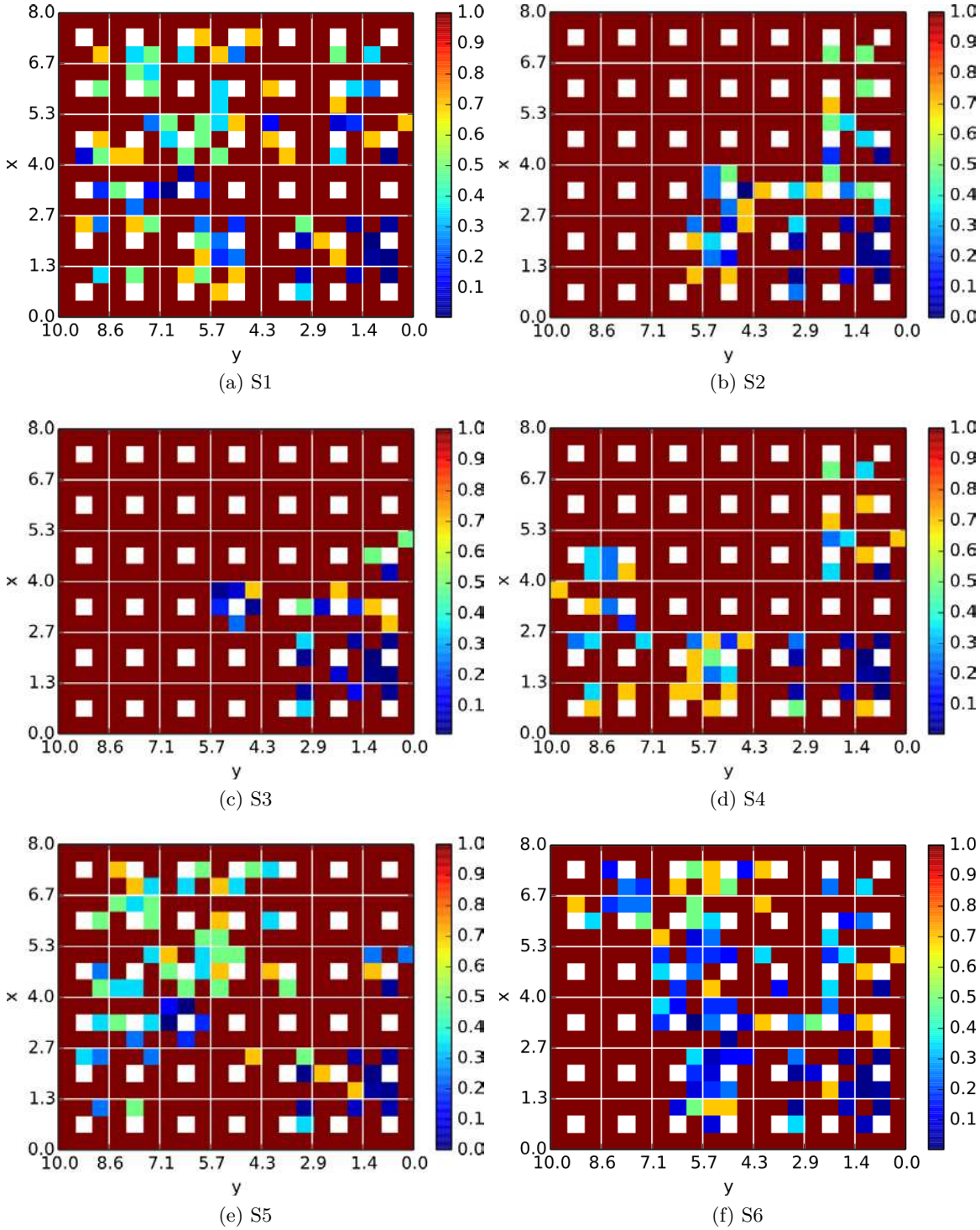


(a) S1 (b) S2

(c) S3 (d) S4

(e) S5 (f) S6

FIGURE 4.18 – *Paramètres de vitesse à l'issue du* 407^{e} *épisode de l'expérience 4.4.4.*

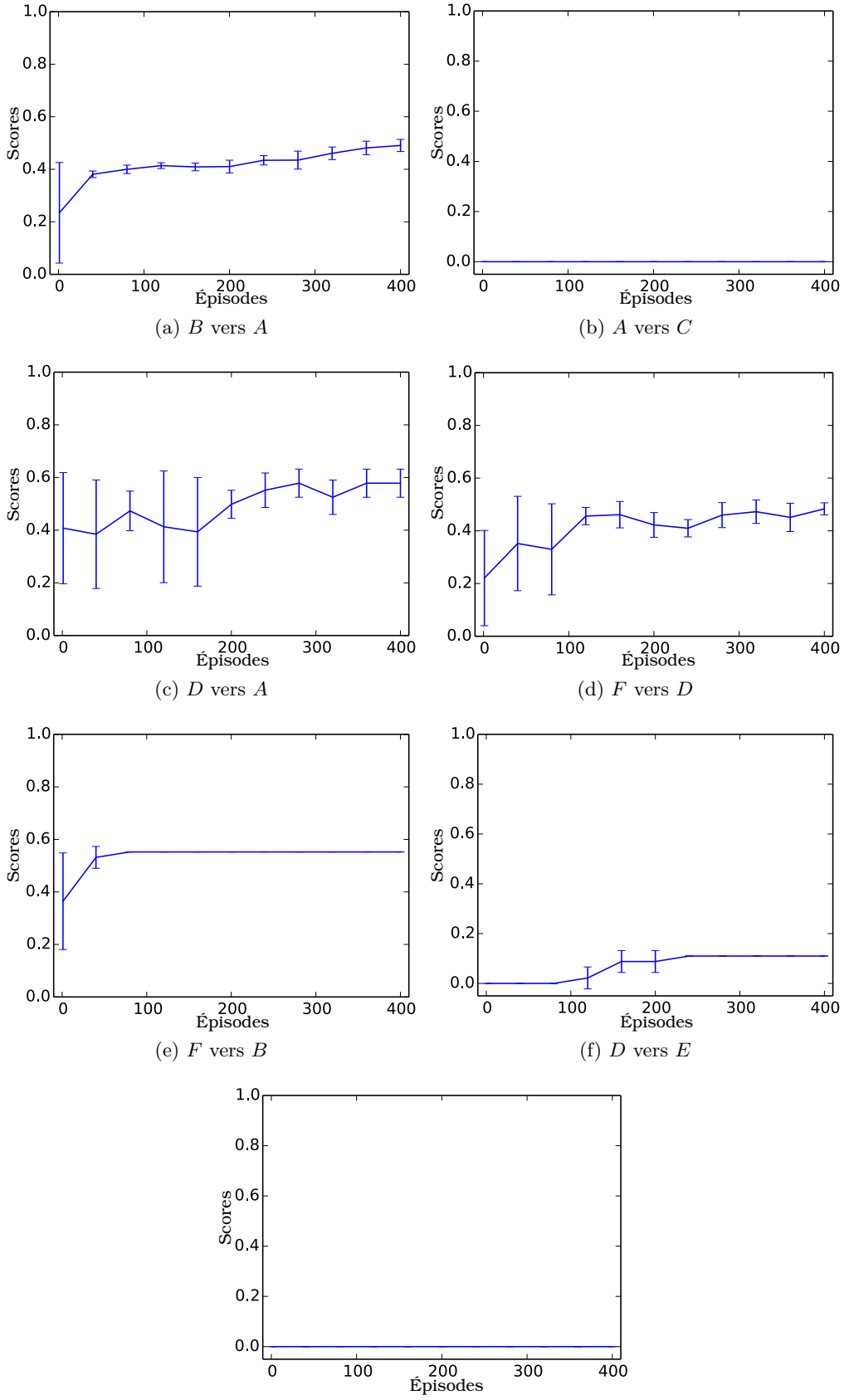


(a) S1 (b) S2

(c) S3 (d) S4

(e) S5 (f) S6

FIGURE 4.19 – *Probabilités de succès à l'issue du* 407^{e} *épisode de l'expérience 4.4.4.*



(a) B vers A

(b) A vers C

(c) D vers A

(d) F vers D

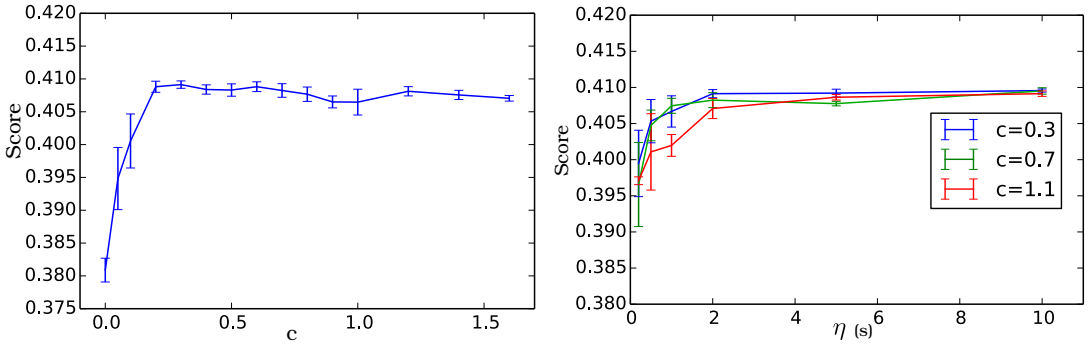
(e) F vers B

(f) D vers E

(g) E vers B

FIGURE 4.20 – *Scores des requêtes de test de l'expérience 4.4.4.* Les moyennes et les écart-types ont été obtenus à partir de 5 simulations indépendantes.

des sélections effectuées.



(a) Score en fonction de c. $\eta = 2$s

(b) Score en fonction de η

FIGURE 4.21 – *Influence du facteur d'exploration c et du temps de simulation η lors de l'expérience 4.4.5.* Les scores représentés sont les scores moyens obtenus durant l'exécution des 20 requêtes. Les écart-types ont été calculés à partir de 4 validations effectuées sur le même jeu de requêtes.

4.4.6 Changement de fonction de score

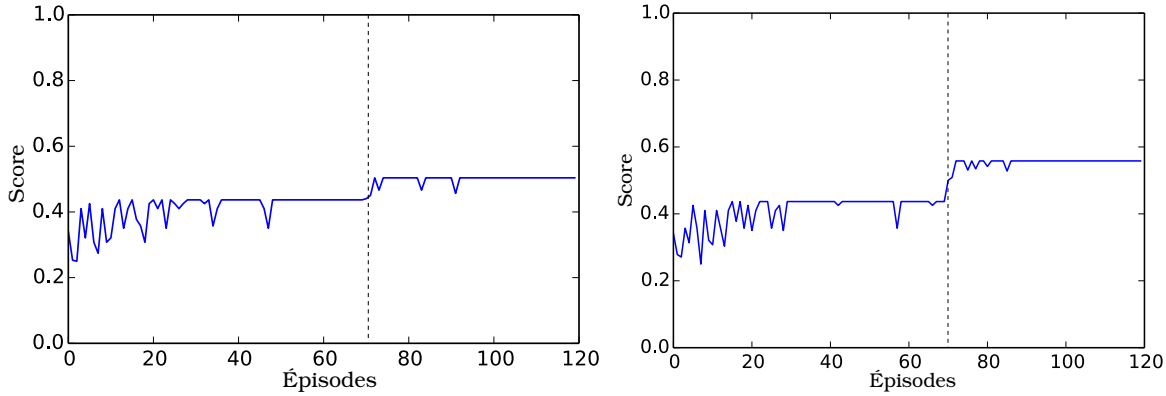
Cette expérience se base sur les modèles de dynamique locale appris dans l'expérience 4.4.2, et étudie les séquences sélectionnées suivant les deux fonctions de scores considérées par le modèle d'évaluation. Durant cette précédente expérience, seul le paramètre de QdS de la durée était considéré. Ces deux fonctions de scores correspondent respectivement à l'absence et à la présence d'humains à proximité du chemin du robot. Elles se distinguent par les poids donnés aux paramètres de QdS (cf. table 3.2).

La figure 4.22 présente l'évolution des scores des 50 épisodes suivant le changement de la fonction de score à l'épisode 71. Le remplacement d'organigrammes est non-opportuniste. Dans chacune des fonctions, ce changement est suivi d'une courte phase d'exploration où un nombre restreint d'alternatives sont évaluées. Dans le cas où un humain est à proximité du chemin du robot, la meilleure séquence choisie est `(S4 S3)` avec un score observé de 0.56. Dans le cas contraire, l'autre fonction de score privilégie la séquence `(S2 S5)` avec un score de 0.50. Les modèles de dynamique locale peuvent donc être directement réutilisés entre les différentes fonctions de score.

4.4.7 Introduction du robot assistant

Cette dernière expérience introduit le robot assistant afin de permettre au robot principal de mener à bien son déplacement du salon (position `A`) vers la chambre 2 (position `E`). En effet, cette introduction est nécessaire car le robot assistant est le seul à fournir un service de positionnement dans la chambre 2. La tentative précédente d'accéder à cette destination dans l'expérience 4.4.4 a mené au rejet du déplacement. Cette expérience vise à montrer l'importance de la position courante du robot assistant lors de la sélection.

Deux positions initiales du robot assistant sont considérées : dans la salle de bains (position B) et à proximité du robot principal, dans le salon (proximité de la position A). Le remplacement d'organigrammes est ici non-opportuniste. Les modèles de dynamique



(a) Pas d'humain à proximité (b) Humain à proximité

FIGURE 4.22 – *Changement de fonction de score lors de l'expérience 4.4.6.* Après un apprentissage de 70 épisodes effectué dans les conditions de l'expérience 4.4.2, chaque figure représente l'activation d'une fonction de score.

locale sont par ailleurs supposés être connus. Aucun humain n'est à proximité des chemins parcourus par les robots.

Lorsque le robot assistant est dans la salle de bains, le service S4 est tout d'abord sélectionné puis échoue à l'entrée de la chambre 2 et est alors remplacé par le robot assistant. Le score observé pour cette séquence est de 0.36.

Lorsque le robot assistant est initialement à proximité du robot principal, il est directement sélectionné et est utilisé durant toute l'exécution malgré la faible vitesse de déplacement et le coût important de l'organigramme résultant. Ce choix est préféré à la séquence `(S4 Ra)` car il n'entraîne pas d'échec non-fatal ni de longue phase d'approche. Cette exécution obtient un score de 0.29.

4.5 Conclusion

Ce chapitre vient compléter notre étude sur la sélection de services dans le cadre de la coordination étroite en s'intéressant à un exemple nécessitant le remplacement d'un de ses sous-services continus. Nos contributions sont les suivantes :

1. *Ce problème de sélection de services a donné lieu à deux formalisations basées sur des processus de décision markoviens.* Ces MDPs correspondent à deux types de remplacement : le remplacement ne survenant que suite à un échec et le remplacement opportuniste.
2. *Un problème de dimensionnalité de l'espace d'état a été mis en évidence.* Cette dimensionnalité rend inadaptées les méthodes consistant à apprendre directement une fonction de valeur ou une fonction de transition. Les MDPs proposés tiennent compte de ce problème et sont spécifiques à chaque requête de service de haut-niveau.
3. *Une modélisation des fonctions de transition de ces MDPs a été proposée pour affronter ce problème de dimensionnalité.* Elle segmente le traitement de la requête de haut-niveau et fait appel à des sous-modèles de dynamique locale. Des

sous-modèles discrétisés par une grille ont été proposés pour modéliser la performance lors du déplacement sur un segment. Ces sous-modèles peuvent être réutilisés dans plusieurs modèles d'évaluation et requêtes de haut-niveau.

4. *La méthode de fouille stochastique d'arbre UCT a été choisie pour estimer les valeurs des organigrammes disponibles dans l'état courant.* Elle permet de parcourir efficacement l'arbre de recherche dont la taille peut être importante.

Conclusion générale et perspectives

Cette thèse s'est intéressée à la question de la sélection de services dans le cadre de la robotique ambiante. Ce domaine, à la croisée de l'intelligence ambiante et de la robotique mobile, introduit deux exigences : la sélection doit (i) être centrée sur l'utilisateur en impliquant ce dernier dans le choix des critères d'évaluation, (ii) prendre en charge des compositions de services nécessitant une coordination étroite. Cette coordination étroite, non investie par la littérature consacrée aux services logiciels, est à l'origine d'un changement architectural ayant un impact important sur la sélection de services. En effet, elle exige que les compositions soient hiérarchiques et introduit un couplage non-fonctionnel entre les performances des services. Les méthodes habituelles de sélection globale de services, prévues pour des plans d'actions faiblement coordonnées, ne sont pas adéquates pour répondre à ces nouvelles exigences.

Ce cadre robotique peut exiger, par ailleurs, que la sélection soit contextuelle. La navigation d'un robot faisant appel à un service de positionnement est un exemple de tâche où cette contextualité est primordiale lors de la prédiction de performance. Les performances des services de positionnement ne sont en effet pas constantes dans l'espace dans lequel le robot évolue. Ces services admettent des zones de non-couverture et ne sont donc pas systématiquement disponibles durant l'intégralité de l'exécution de haut-niveau. Leur possible remplacement implique de considérer ce type de sélection comme un problème de prise de décision séquentielle et de représenter les états intermédiaires d'échecs non-fatals. Ce problème a été formalisé comme un processus de décision markoviens à horizon fini.

Cette contextualité importante engendre un problème de dimensionnalité de l'espace d'états. Celui-ci empêche l'apprentissage direct de fonctions de valeur ou de fonctions de transition. L'approche proposée exploite la logique métier de la tâche de haut-niveau en extrayant le chemin suivi par le robot. Celui-ci est calculé par le service de planification de chemin de l'organigramme. À partir de ce chemin, cette approche effectue une segmentation du déplacement et exploite des modèles de dynamique locale pour prédire les transitions entre les sélections. Ces modèles de dynamique locale, spécifiques à un organigramme et une métrique donnés, sont indépendants du reste du contexte, ce qui permet leur réutilisation avec des requêtes et des modèles d'évaluation différents. Le partage d'expérience entre les exécutions est suffisant pour répondre à ce problème de dimensionnalité. Les valeurs Q sont ensuite estimées à partir d'une fouille stochastique de l'arbre. Cette fouille pouvant être interrompue à n'importe quel moment, elle est en mesure de garantir un temps de simulation borné tout en permettant d'effectuer des sélections quasi-optimales lorsque le nombre de candidats est modéré.

Perspectives

Enjeux combinatoires liés aux organigrammes Plusieurs questions concernant les enjeux combinatoires de ce type de sélection globale ont été soulevées dans la sous-sous-section 3.2.2.3. Bien qu'elles n'aient été traitées qu'*a minima* dans ce manuscrit, elles mériteraient néanmoins une étude approfondie.

Expérimentation L'expérimentation en laboratoire de notre approche permettrait, dans un premier temps, de connaître plus en détail les difficultés auxquelles les robots sont confrontés. Notre modèle de dynamique locale pourrait alors être amélioré afin de mieux en rendre compte. De nouveaux modèles d'évaluation et de nouvelles métriques pourraient également être proposées en se basant sur le vécu des utilisateurs. Une telle expérimentation nécessite cependant un effort de développement conséquent afin que l'ensemble des fonctionnalités robotiques et de mesure soient disponibles sous forme de services logiciels.

États partiellement observables La modélisation des services de positionnement que nous proposons dans ce manuscrit suppose que ceux-ci ne fournissent que la position la plus probable. Cependant, il se peut que la distribution de probabilités de cette position soit multimodale et que plusieurs modes aient des probabilités non négligeables. L'état est alors dit partiellement observable car plusieurs positions non-voisines sont probables. Toutefois, un problème de MDP partiellement observable *(Partially Observable MDP)* peut être ramené à un simple MDP en intégrant l'état de croyance dans la représentation de l'état. Pour cela, l'estimation de la position courante dans l'état du MDP peut être remplacée par une représentation de sa probabilité de distribution. La fonction de transition résultante est alors plus complexe.

Nouveaux modèles de dynamique locale Des modèles de dynamique locale effectuant une tâche de régression pourraient être utilisés. De part le caractère local des dynamiques prédites, les approches non-paramétriques comme les processus gaussiens semblent être adaptées à cette tâche. Ces régressions seraient susceptibles d'offrir un meilleur rapport entre la précision des prédictions et la rapidité d'apprentissage que son équivalent basé sur le réglage de la taille des cases. Néanmoins, la mise en œuvre de ce type d'approche semble être plus complexe que celle de notre modèle de dynamique locale basé sur une grille.

Apprentissage à vie Les conditions environnementales sont susceptibles d'évoluer de part l'apparition ou le déplacement d'obstacles. Ce changement de la dynamique du système introduit une problématique d'apprentissage à vie. L'approche que nous avons proposée se limite à l'exploration initiale de l'environnement par le biais des valeurs initiales optimistes données aux paramètres des modèles de dynamique locale. Il conviendrait donc d'introduire un nouveau mécanisme permettant une exploration continue suffisante. Par ailleurs, si un modèle non-paramétrique venait à être utilisé, il serait alors nécessaire d'effectuer une sélection d'exemples afin de limiter sa taille en mémoire et de ne pas pénaliser sa performance.

Bibliographie

Abbeel, P., Coates, A., Quigley, M., & Ng, A. (2007). An application of reinforcement learning to aerobatic helicopter flight. *Advances in neural information processing systems*.

Abowd, G. (2012). What next, ubicomp ? celebrating an intellectual disappearing act. In *International conference on Ubiquitous computing*, Pittsburgh, USA.

Abowd, G., Bobick, A., Essa, I., Mynatt, E., & Rogers, W. (2002). The aware home : A living laboratory for technologies for successful aging. In *International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*.

Ahmadi, M. & Stone, P. (2006). A multi-robot system for continuous area sweeping tasks. In *IEEE International Conference on Robotics and Automation*, (pp. 1724 –1729).

Alami, R., Robert, F., Ingrand, F., & Suzuki, S. (1995). Multi-robot cooperation through incremental plan-merging. In *IEEE International Conference on Robotics and Automation*, (pp. 2573 –2579).

Alarcón, R. & Wilde, E. (2010a). From RESTful services to RDF : connecting the web and the semantic web. Technical report, UC Berkeley, CA, USA.

Alarcón, R. & Wilde, E. (2010b). RESTler : crawling RESTful services. In *International conference on World Wide Web*, (pp. 1051–1052).

Alarcón, R., Wilde, E., & Bellido, J. (2011). Hypermedia-driven RESTful service composition. In *International Conference on Service-Oriented Computing*, (pp. 111–120).

Alrifai, M., Risse, T., Dolog, P., & Nejdl, W. (2008). A scalable approach for QoS-Based web service selection. In *ICSOC Workshop*, (pp. 190–199).

Ambroszkiewicz, S., Bartyna, W., Faderewski, M., & Terlikowski, G. (2010). Multirobot system architecture : environment representation and protocols. *Bulletin of the Polish Academy of Sciences-Technical Sciences*.

Amin, A., Colman, A., & Grunske, L. (2012). Statistical detection of QoS violations based on CUSUM control charts. In *International Conference on Performance Engineering*, (pp. 97–108).

Amin, A., Grunske, L., & Colman, A. (2012). An automated approach to forecasting QoS attributes based on linear and non-linear time series modeling. In *International Conference on Automated Software Engineering*, (pp. 130–139).

Amundsen, M. (2010). H factor : Hypermedia types.

Amundsen, M. (2011). *Building Hypermedia APIs With HTML5 and Node*. O'Reilly Media.

Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., & Xu, M. (2007). Web services agreement specification (WS-Agreement). In *Global Grid Forum*.

Au, T.-C., Kuter, U., & Nau, D. (2005). Web service composition with volatile information. In *International Semantic Web Conference*, (pp. 52–66).

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, *47*(2), 235–256.

Augusto, J. (2007). Ambient intelligence : the confluence of ubiquitous/pervasive computing and artificial intelligence. *Intelligent Computing Everywhere*, 213–234.

Bahl, P. & Padmanabhan, V. (2000). RADAR : an in-building RF-based user location and tracking system. In *INFOCOM*, (pp. 775 –784).

Baresi, L. & Guinea, S. (2005). Towards dynamic monitoring of WS-BPEL processes. In *International Conference on Service-Oriented Computing*.

Baresi, L., Guinea, S., & Plebani, P. (2006). WS-Policy for service monitoring. In *Technologies for E-Services*, volume 3811, (pp. 72–83). Springer Berlin Heidelberg.

Beckwith, R. & Lederer, S. (2003). Designing for one's dotage : UbiComp and residential care facilities. Irvine, CA, USA.

Bell, G. & Dourish, P. (2006). Yesterday's tomorrows : notes on ubiquitous computing's dominant vision. *Personal and Ubiquitous Computing*, *11*(2), 133–143.

Bellido, J., Alarcon, R., & Sepulveda, C. (2011). Web linking-based protocols for guiding RESTful M2M interaction. In *International Workshop on Lightweight Integration on the Web*, (pp. 74–85)., Paphos, Cyprus.

Bellman, R. (1957). *Dynamic Programming*. Princeton University.

Ben Mabrouk, N., Beauche, S., Kuznetsova, E., Georgantas, N., & Issarny, V. (2009). QoS-Aware service composition in dynamic service oriented environments. *Middleware 2009*, 123–142.

Ben Mabrouk, N., Georgantas, N., & Issarny, V. (2009). A semantic end-to-end QoS model for dynamic service oriented environments. In *ICSE Workshop on Principles of Engineering Service Oriented Systems*, (pp. 34–41)., Vancouver, BC, Canada.

Ben Mokhtar, S., Preuveneers, D., Georgantas, N., Issarny, V., & Berbers, Y. (2008). EASY : Efficient semAntic service discoverY in pervasive computing environments with QoS and context support. *Journal of Systems and Software*.

Berners-Lee, T., Connolly, D., Kagal, L., Scharf, Y., & Hendler, J. (2008). N3logic : A logical framework for the world wide web. *Theory Pract. Log. Program.*, *8*(3), 249–269.

Blake, M., Remy, S., Wei, Y., & Howard, A. (2011). Robots on the web. *IEEE Robotics Automation Magazine*, *18*(2), 33 –43.

Bonitasoft (2012). Bonitasoft - open source workflow & BPM software.

Bouillard, A., Rosario, S., Benveniste, A., & Haar, S. (2009). Monotonicity in service orchestrations. *Applications and Theory of Petri Nets*, 263–282.

Box, D., Curbera, F., et al. (2004). Web services addressing (WS-Addressing). Technical report, W3C.

Brdiczka, O. (2007). *Learning Situation Models for Providing Context-Aware Services*. PhD thesis, Institut National Polytechnique de Grenoble (INPG).

Broxvall, M., Seo, B. S., & Kwon, W. Y. (2007). The peis kernel : A middleware for ubiquitous robotics. In *IROS Workshop on Ubiquitous Robotic Space Design and Applications*.

Burgard, W., Moors, M., Stachniss, C., & Schneider, F. (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, *21*(3), 376 – 386.

Canfora, G., Di Penta, M., Esposito, R., & Villani, M. L. (2005). An approach for QoS-aware service composition based on genetic algorithms. In *Conference on Genetic and Evolutionary Computation*, (pp. 1069–1075).

Cao, H., Lacroix, S., & Ingrand, F. (2012). Planification d'une mission d'observation par allocation de tâches hiérarchiques pour une équipe de robots hétérogènes. In *RFIA*, Lyon, France.

Cardoso, J., Sheth, A., Miller, J., Arnold, J., & Kochut, K. (2004). Quality of service for workflows and web service processes. *Web Semantics : Science, Services and Agents on the World Wide Web*, *1*(3), 281–308.

Cardozo, E., Guimaraes, E., Rocha, L., Souza, R., Paolieri, F., & Pinho, F. (2010). A platform for networked robotics. In *International Conference on Intelligent Robots and Systems*, (pp. 1000–1005).

Carzaniga, A., Rosenblum, D., & Wolf, A. (2000). Content-based addressing and routing : A general model and its application. Technical Report CU-CS-902-00, Department of Computer Science, University of Colorado.

Castellanos, M., Casati, F., Dayal, U., & Shan, M.-C. (2004). A comprehensive and automated approach to intelligent business processes execution analysis. *Distributed and Parallel Databases*, *16*(3), 239–273.

Cavallo, B., Di Penta, M., & Canfora, G. (2010). An empirical comparison of methods to support QoS-aware service selection. In *International Workshop on Principles of Engineering Service-Oriented Systems*, (pp. 64–70).

Chaari, S., Badr, Y., & Biennier, F. (2008). Enhancing web service selection by QoS-based ontology and WS-policy. In *ACM symposium on Applied computing*, (pp. 2426–2431).

Chaimowicz, L., Kumar, V., & Campos, M. F. M. (2004). A paradigm for dynamic coordination of multiple robots. *Autonomous Robots*, *17*(1), 7–21.

Chalmers, M., MacColl, I., & Bell, M. (2003). Seamful design : showing the seams in wearable computing. In *Eurowearable, 2003*, (pp. 11 –16).

Chandra, A., Gong, W., & Shenoy, P. (2003). Dynamic resource allocation for shared data centers using online measurements. In *International Workshop on Quality of Service*, (pp. 381–398).

Chappell, D. (2004). *Enterprise Service Bus : Theory in Practice*. O'Reilly Media.

Chen, Y., Sabnis, A., & Garcia-Acosta, M. (2009). Design and performance evaluation of a service-oriented robotics application. In *ICDCS Workshop*, (pp. 292 –299).

Chollet, S., Lestideau, V., Lalanda, P., Moreno-Garcia, D., & Colomb, P. (2010). Heterogeneous service selection based on formal concept analysis. In *IEEE Congress on Services*, (pp. 367–374)., Los Alamitos, CA, USA.

Cogrel, B., Daachi, B., & Amirat, Y. (2011). Impact-based contextual service selection in a ubiquitous robotic environment. In *IEEE International Conference on Ubiquitous Robots and Ambient Intelligence*, (pp. 313–318)., Seoul, South Korea.

Cogrel, B., Daachi, B., Amirat, Y., & Chibani, A. (2011). Sélection de services basée sur l'impact en environnement ubiquitaire. In *Journées nationales Ubimob*, Toulouse, France.

Coulom, R. (2006). Efficient selectivity and backup operators in monte-carlo tree search. In *International Conference on Computer and Games*, Turin, Italy.

Coutaz, J. & Crowley, J. (2008). Plan "intelligence ambiante" : Défis et opportunités. *Document de réflexion conjoint du comité d'experts «Informatique Ambiante» du département ST2I du CNRS et du Groupe de Travail «Intelligence Ambiante» du Groupe de Concertation Sectoriel (GCS3) du Ministère de l'Enseignement Supérieur et de la Recherche*.

Crnkovic, I., Sentilles, S., Vulgarakis, A., & Chaudron, M. (2011). A classification framework for software component models. *IEEE Transactions on Software Engineering*, *37*(5), 593 –615.

Cuny, F. (2011). SPORE : SPecification to a POrtable rest environment.

Decker, G., Kopp, O., Leymann, F., & Weske, M. (2007). BPEL4Chor : Extending BPEL for modeling choreographies. In *IEEE International Conference on Web Services*, (pp. 296 –303).

Dehousse, S., Faulkner, S., Herssens, C., Jureta, I. J., & Saerens, M. (2009). Learning optimal web service selections in dynamic environments when many quality-of-service criteria matter. In *Machine Learning* (pp. 207–229). InTech.

Derdour, M., Roose, P., Dalmau, M., Ghoualmi Zine, N., & Alti, A. (2010). MMSA : Metamodel multimedia software architecture. *Advances in Multimedia*, 1–17.

Dertouzos, M. (1999). The future of computing. *Scientific American*, *281*(2), 36–47.

Dey, A. K. (2001). Understanding and using context. *Personal And Ubiquitous Computing*.

Dey, A. K., Hamid, R., Beckmann, C., Li, I., & Hsu, D. (2004). a CAPpella : programming by demonstration of context-aware applications. In *SIGCHI Conference on Human Factors in Computing Systems*, (pp. 33–40).

Di Nitto, E., Ghezzi, C., Metzger, A., Papazoglou, M., & Pohl, K. (2008). A journey to highly dynamic, self-adaptive service-based applications. *Automated Software Engineering*, *15*(3-4), 313–341.

Dorigo, M., Floreano, D., Gambardella, L., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., et al. (2012). Swarmanoid : a novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*.

Doshi, P., Goodwin, R., Akkiraju, R., & Verma, K. (2004). Dynamic workflow composition using markov decision processes. In *IEEE International Conference on Web Services*, (pp. 576– 582).

Dourlens, S., Ramdane-Cherif, A., & Monacelli, E. (2013). Multi levels semantic architecture for multimodal interaction. *Applied Intelligence*, 1–14.

Fielding, R. (2007). A little REST and relaxation. Zurich, Switzerland.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. PhD thesis.

Fielding, R. T. (2008a). Economies of scale.

Fielding, R. T. (2008b). REST APIs must be hypertext-driven.

Filliat, D. (2011). Robotique mobile. Technical report, ENSTA ParisTech.

Fitzpatrick, B., Slatkin, B., & Atkins, M. (2010). PubSubHubbub core 0.3. Working draft.

Frystyk Nielsen, H. & Chrysanthakopoulos, G. (2007). Decentralized software services Protocol–DSSP. *Microsoft Corporation*.

Garlan, D., Cheng, S.-W., Huang, A.-C., Schmerl, B., & Steenkiste, P. (2004). Rainbow : architecture-based self-adaptation with reusable infrastructure. *Computer*, *37*(10), 46–54.

Gelly, S. & Silver, D. (2007). Combining online and offline knowledge in UCT. In *International Conference on Machine Learning*, (pp. 273–280).

Georgantas, N., Issarny, V., Mokhtar, S. B., Bromberg, Y.-D., Bianco, S., Thomson, G., Raverdy, P.-G., Urbieta, A., & Cardoso, R. S. (2010). Middleware architecture for ambient intelligence in the networked home. In *Handbook of Ambient Intelligence and Smart Environments* (pp. 1139–1169). Springer.

Gerkey, B. P. & Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, *23*(9), 939–954.

Ghallab, M. (2004). An overview of planning technology in robotics. In *Annual German Conference on AI (KI)*, (pp. 29–49).

Glad, A., Simonin, O., Buffet, O., & Charpillet, F. (2008). Theoretical study of ant-based algorithms for multi-agent patrolling. In *European Conference on Artificial Intelligence*, (pp. 626–630).

Godse, M., Bellur, U., & Sonar, R. (2010). Automating QoS based service selection. In *IEEE International Conference on Web Services*, (pp. 534–541).

Greenfield, A. (2006). *Everyware : The Dawning Age of Ubiquitous Computing* (1st ed.). New Riders Publishing.

Gritti, M. (2008). *A Mechanism for Automatic Self-Configuration of Software Components in Robot Ecologies*. Licentiate thesis.

Gupta, S., Reynolds, M. S., & Patel, S. N. (2010). ElectriSense : single-point sensing using EMI for electrical event detection and classification in the home. In *International conference on Ubiquitous computing*, (pp. 139–148).

Ha, Y.-G., Sohn, J.-C., & Cho, Y.-J. (2005). Service-oriented integration of networked robots with ubiquitous sensors and devices using the semantic web services technology. In *Intelligent Robots and Systems*, (pp. 3947–3952).

Hadley, M. J. (2006). Web application description language (WADL). Technical report, Sun Microsystems, Inc., Mountain View, CA, USA.

Hayet, J., Lerasle, F., & Devy, M. (2007). A visual landmark framework for mobile robot navigation. *Image and Vision Computing*, *25*(8), 1341–1351.

Hock-koon, A. & Oussalah, M. (2011). Vers une meilleure compréhension de la différence théorique entre un composant et un service. *INFORSID*.

Hohpe, G. & Woolf, B. (2003). *Enterprise Integration Patterns : Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman Publishing Co.

Hong, X., Nugent, C., Mulvenna, M., Mcclean, S., Scotney, B., & Devlin, S. (2009). Evidential fusion of sensor data for activity recognition in smart homes. *Pervasive and Mobile Computing*, *5*(3), 236–252.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., & Dean, M. (2004). SWRL : A semantic web rule language combining OWL and RuleML. W3C member submission.

Hossain, M. A., Atrey, P. K., & El Saddik, A. (2008). Gain-based selection of ambient media services in pervasive environments. *Mob. Netw. Appl.*, *13*(6), 599–613.

Hourdin, V., Tigli, J.-Y., Lavirotte, S., Rey, G., & Riveill, M. (2008). SLCA, composite services for ubiquitous computing. In *International Conference on Mobile Technology*, (pp.1̃)., Yilan, Taiwan.

Huang, Y. & Gannon, D. (2006). A comparative study of web services-based event notification specifications. In *International Conference on Parallel Processing Workshops*, (pp. 8 pp. –14).

Hwang, S.-Y., Wang, H., Tang, J., & Srivastava, J. (2007). A probabilistic approach to modeling and estimating the QoS of web-services-based workflows. *Information Sciences*, *177*(23), 5484–5503.

IBM (2000). Web services architecture overview.

IETF (1999a). Hypertext transfer protocol - HTTP/1.1. Request For Comment 2616.

IETF (1999b). Service location protocol, version 2. Request For Comment 2608.

IETF (2005). The atom syndication format. Request For Comment 4287.

IETF (2010). Web linking. Request For Comment 5988.

IETF (2012). URI template. Request For Comment 6570.

IETF (2013a). DNS-Based service discovery. Request For Comment 6763.

IETF (2013b). Multicast DNS. Request For Comment 6762.

ISO (2008). UPnP device architecture – part 1 : UPnP device architecture version 1.0. ISO Standard ISO/IEC 29341-1 :2008.

ISTAG (2001). Scenarios for ambient intelligence in 2010. Technical report.

ISTAG (2003). Ambient intelligence : from vision to reality. Technical report.

Jackson, J. (2007). Microsoft robotics studio : A technical introduction. *IEEE Robotics & Automation Magazine*, *14*(4), 82–87.

Jammes, F., Mensch, A., & Smit, H. (2005). Service-oriented device communications using the devices profile for web services. In *International workshop on Middleware for pervasive and ad-hoc computing*, (pp. 1–8).

Jean, S., Losavio, F., Matteo, A., & Levy, N. (2012). Prise en compte des standards de qualité et des préférences utilisateurs pour la modélisation des propriétés non-fonctionnelles dans OWL-S. *Technique et Science Informatiques*, *31*(1), 39–69.

Junghans, M. & Agarwal, S. (2010). Web service discovery based on unified view on functional and non-functional properties. In *IEEE International Conference on Semantic Computing*, (pp. 224 –227).

Junghans, M., Agarwal, S., & Studer, R. (2010). Towards practical semantic web service discovery. In *ECSW*, (pp. 15–29).

Kaldeli, E., Lazovik, A., & Aiello, M. (2011). Continual planning with sensing for web service composition. In *AAAI Conference on Artificial Intelligence*.

Kanda, T., Glas, D. F., Shiomi, M., Ishiguro, H., & Hagita, N. (2008). Who will be the customer ? : a social robot that anticipates people's behavior from their trajectories. In *International conference on Ubiquitous computing*, (pp. 380–389).

Kaye, D. (2003). *Loosely Coupled : The Missing Pieces of Web Services*. RDS Strategies LLC.

Keller, A. & Ludwig, H. (2003). The WSLA framework : Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, *11*(1), 57–81.

Kim, B. K., Miyazaki, M., Ohba, K., Hirai, S., & Tanie, K. (2005). Web services based robot control platform for ubiquitous functions. In *IEEE International Conference on Robotics and Automation*, (pp. 691 – 696).

Kim, D., Park, S., Jin, Y., Chang, H., Park, Y.-S., Ko, I.-Y., Lee, K., Lee, J., Park, Y.-C., & Lee, S. (2006). SHAGE : a framework for self-managed robot software. In *International workshop on Self-adaptation and self-managing systems*, (pp. 79–85).

Klusch, M., Fries, B., Khalid, M., & Sycara, K. (2005). OWLS-MX : Hybrid OWL-S service matchmaking. In *International AAAI Fall Symposium on Agents and the Semantic Web*.

Kocsis, L. & Szepesvári, C. (2006). Bandit based monte-carlo planning. In *European Conference on Machine Learning*, (pp. 282–293).

Kopecky, J., Gomadam, K., & Vitvar, T. (2008). hRESTS : An HTML microformat for describing RESTful web services. In *International Conference on Web Intelligence and Intelligent Agent Technology*, (pp. 619 –625).

Kopp, O., van Lessen, T., & Nitzsche, J. (2008). The need for a choreography-aware service bus. In *European Young Researchers Workshop on Service Oriented Computing*, (pp. 30–36).

Kourtesis, D. & Paraskakis, I. (2008). Combining SAWSDL, OWL-DL and UDDI for semantically enhanced web service discovery. (pp. 614–628).

Krafzig, D., Banke, K., & Slama, D. (2005). *Enterprise SOA : Service-Oriented Architecture Best Practices*. Prentice Hall Professional.

Krakowiak, S. (2006). Introduction à l'intergiciel. *Intergiciel et construction d'application réparties*, 1–21.

Krill, P. (2005). Microsoft, IBM, SAP discontinue UDDI registry effort.

Lathem, J., Gomadam, K., & Sheth, A. (2007). SA-REST and (s)mashups : Adding semantics to RESTful services. In *International Conference on Semantic Computing*, (pp. 469 –476).

Leake, D. B. & Gary, J. (2008). AI magazine poster : The AI landscape. *AI Magazine*, *29*(2), 3.

LeBlanc, K. & Saffiotti, A. (2008). Cooperative anchoring in heterogeneous multi-robot systems. In *IEEE International Conference on Robotics and Automation*, (pp. 3308 –3314).

Leitner, P., Wetzstein, B., Rosenberg, F., Michlmayr, A., Dustdar, S., & Leymann, F. (2009). Runtime prediction of service level agreement violations for composite services. In *ICSOC/ServiceWave Workshops*, (pp. 176–186).

Leymann, F. (2005). The (service) bus : Services penetrate everyday life. In *International Conference on Service-Oriented Computing*, (pp. 12–20).

Lieberman, H. & Selker, T. (2000). Out of context : Computer systems that adapt to, and learn from, context. *IBM Systems Journal*, *39*(3), 617–632.

Lundh, R., Karlsson, L., & Saffiotti, A. (2008). Autonomous functional configuration of a network robot system. *Robotics and Autonomous Systems*, *56*, 819–830.

Majedi, M., Osman, K., & Boyd, M. (2008). A generic service oriented architectural model for pervasive applications : A case study in internet-based multiple robot control. In *International Conference on Pervasive Computing and Applications*, volume 1, (pp. 54 –59).

Mao, G., Fidan, B., & Anderson, B. D. (2007). Wireless sensor network localization techniques. *Computer Networks*, *51*(10), 2529–2553.

Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., & Sycara, K. (2005). Bringing semantics to web services : The OWL-S approach. In *International Workshop on Semantic Web Services and Web Process Composition*, (pp. 26–42).

Maximilien, E. & Singh, M. (2005). Multiagent system for dynamic web services selection. In *Workshop on Service-Oriented Computing and Agent-Based Engineering*, (pp. 25–29).

Maximilien, E. M. & Singh, M. P. (2004). A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, *8*(5), 84– 93.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., & Wilkins, D. (1998). PDDL- the planning domain definition language. Technical report, AIPS-98 Planning Committee.

McIlraith, S., Son, T., & Zeng, H. (2001). Semantic web services. *IEEE Intelligent Systems*, *16*(2), 46 – 53.

McNamara, L., Mascolo, C., & Capra, L. (2006). Trust and mobility aware service provision for pervasive computing. In *International Workshop on Requirements and Solutions for Pervasive Software Infrastructures*, (pp. 603–610).

Menascé, D. A., Barbara, D., & Dodge, R. (2001). Preserving QoS of e-commerce sites through self-tuning : a performance model approach. In *ACM conference on Electronic Commerce*, (pp. 224–234)., New York, NY, USA. ACM.

Merle, P. (2003). OpenCCM : The open CORBA components platform. In *ObjectWeb Conference*, Rocquencourt, France.

Michlmayr, A., Rosenberg, F., Leitner, P., & Dustdar, S. (2009). Comprehensive QoS monitoring of web services and event-based SLA violation detection. In *International Workshop on Middleware for Service Oriented Computing*, (pp. 1–6)., Urbana Champaign, Illinois, USA.

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). FastSLAM : A factored solution to the simultaneous localization and mapping problem. *National conference on Artificial Intelligence*, 593–598.

Morisset, B. & Ghallab, M. (2008). Learning how to combine sensory-motor functions into a robust behavior. *Artificial Intelligence*, *172*(4–5), 392–412.

Musser, J. (2011). Open APIs - state of the market 2011.

Muñoz-Salinas, R., Aguirre, E., & García-Silvente, M. (2007). People detection and tracking using stereo vision and color. *Image and Vision Computing*, *25*(6), 995–1007.

Nahrstedt, K. & Balke, W.-T. (2004). A taxonomy for multimedia service composition. In *ACM International conference on Multimedia*, (pp. 88–95).

Nau, D., Au, T., Ilghami, O., Kuter, U., Murdock, J., Wu, D., & Yaman, F. (2003). SHOP2 : An HTN planning system. *Journal of Artificial Intelligence Research*, *20*, 379–404.

Nentwich, C., Capra, L., Emmerich, W., & Finkelstein, A. (2002). xlinkit : a consistency checking and smart link generation service. *ACM Transactions on Internet Technology*, *2*(2), 151–185.

OASIS (2002). UDDI version 3.0. OASIS standard.

OASIS (2006). OASIS web services notification.

OASIS (2007a). Web services business process execution language version 2.0. OASIS standard.

OASIS (2007b). WS-SecurityPolicy 1.2. OASIS standard.

OASIS (2007c). WS-Trust 1.3. OASIS standard.

OASIS (2009a). OASIS devices profile for web services (DPWS). OASIS standard.

OASIS (2009b). Web services dynamic discovery (WS-Discovery) 1.1. OASIS standard.

OASIS (2009c). Web services reliable messaging (WS-ReliableMessaging) version 1.2. OASIS standard.

OMG (2010). BPMN 2.0 by example. Technical report.

OMG (2011). BPMN 2.0. OMG standard.

Paolucci, M., Kawamura, T., Payne, T. R., & Sycara, K. (2002a). Importing the semantic web in UDDI. In *International Workshop on Web Services, E-Business, and the Semantic Web*, (pp. 225–236).

Paolucci, M., Kawamura, T., Payne, T. R., & Sycara, K. (2002b). Semantic matching of web services capabilities. In *International Semantic Web Conference*, (pp. 333–347).

Papazoglou, D. M. (2012). *Web Services and SOA : Principles and Technology* (2 ed.). Prentice Hall.

Parker, L. E., Chandra, M., & Tang, F. (2005). Enabling autonomous sensor-sharing for tightly-coupled cooperative tasks. In *International Workshop on Multi-Robot Systems*, (pp. 119–130).

Patel, S. N., Reynolds, M. S., & Abowd, G. D. (2008). Detecting human movement by differential air pressure sensing in HVAC system ductwork : An exploration in infrastructure mediated sensing. In *Pervasive Computing*, (pp. 1–18).

Pautasso, C. (2009). RESTful web service composition with BPEL for REST. *Data & Knowledge Engineering*, *68*(9), 851–866.

Pautasso, C. & Wilde, E. (2009). Why is the web loosely coupled ? a multi-faceted metric for service design. In *International conference on World Wide Web*, (pp. 911–920).

Pedrinaci, C. & Domingue, J. (2010). Toward the next wave of services : linked services for the web of data. *Journal of Universal Computer Science*, *16*(13), 1694–1719.

Peer, J. (2004). A PDDL based tool for automatic web service composition. *Principles and Practice of Semantic Web Reasoning*, 149–163.

Petrovskaya, A. & Ng, A. (2007). Probabilistic mobile manipulation in dynamic environments, with application to opening doors. In *International Conference on Artificial Intelligence*.

Pistore, M., Marconi, A., Bertoli, P., & Traverso, P. (2005). Automated composition of web services by planning at the knowledge level. In *International Joint Conference on Artificial Intelligence*, volume 19, (pp. 1252).

Priyantha, N. (2005). *The cricket indoor location system*. PhD thesis, Massachusetts Institute of Technology.

Prüter, S., Golatowski, F., & Timmermann, D. (2009). Adaptation of resource-oriented service technologies for industrial informatics. In *Conference of IEEE Industrial Electronics*, (pp. 2399 –2404).

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Ng, A. (2009). ROS : an open-source robot operating system. In *ICRA workshop on open source software*, volume 3.

Ramos, C., Augusto, J. C., & Shapiro, D. (2008). Ambient intelligence—the next step for artificial intelligence. *IEEE Intelligent Systems*, *23*(2), 15–18.

Rashid, J., Broxvall, M., & Saffiotti, A. (2012). A middleware to integrate robots, simple devices and everyday objects into an ambient ecology. *Pervasive and Mobile Computing*, *8*(4), 522–541.

Richardson, L. & Amundsen, M. (2013). *RESTful Web APIs*. O'Reilly Media.

Richardson, L. & Ruby, S. (2008). *RESTful Web Services*. O'Reilly Media.

Rogers, Y. (2006). Moving on from weiser's vision of calm computing : Engaging UbiComp experiences. In *International conference on Ubiquitous computing*, (pp. 404–421).

Rosario, S. (2009). *Quality of Service issues in compositions of Web Services*. PhD thesis, Université de Rennes 1, Rennes, France.

Rosario, S., Benveniste, A., & Jard, C. (2009). Flexible probabilistic QoS management of transaction based web services orchestrations. In *International Conference on Web Services*, (pp. 107 –114).

Russell, S. & Norvig, P. (2009). *Artificial Intelligence : A Modern Approach* (3 ed.). Prentice Hall.

Saaty, T. L. (1980). Analytic hierarchy process. In *Encyclopedia of Biostatistics*. John Wiley & Sons, Ltd.

Saffiotti, A. & Broxvall, M. (2005). PEIS ecologies : ambient intelligence meets autonomous robotics. In *Joint conference on Smart objects and ambient intelligence*, (pp. 277–281)., Grenoble, France.

Saffiotti, A., Broxvall, M., Gritti, M., LeBlanc, K., Lundh, R., Rashid, J., Seo, B. S., & Cho, Y. J. (2008). The PEIS-ecology project : vision and results. In *International Conference on Intelligent Robots and Systems*, (pp. 2329–2335).

Sanfeliu, A., Hagita, N., & Saffiotti, A. (2008). Network robot systems. *Robotics and Autonomous Systems*, *56*(10), 793–797.

Sebag, M. (2012). Monte-carlo tree search.

Sirin, E., Parsia, B., Wu, D., Hendler, J., & Nau, D. (2004). HTN planning for web service composition using SHOP2. *Web Semantics : Science, Services and Agents on the World Wide Web*, *1*(4), 377–396.

Srivastava, B. & Koehler, J. (2003). Web service composition-current solutions and open problems. In *ICAPS Workshop on Planning for Web Services*, volume 35.

Stiegler, B. (2012). *États de choc : Bêtise et savoir au XXIe siècle*. Fayard.

Stollberg, M., Hepp, M., & Hoffmann, J. (2007). A caching mechanism for semantic web service discovery. In *International Semantic Web Conference*, (pp. 480–493).

Sun, G., Chen, J., Guo, W., & Liu, K. J. (2005). Signal processing techniques in network-aided positioning : a survey of state-of-the-art positioning designs. *IEEE Signal Processing Magazine*, *22*(4), 12– 23.

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning : An Introduction*. MIT Press.

Szepesvari, C. (2010). *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers.

Tan, Z., Lin, C., Yin, H., Hong, Y., & Zhu, G. (2003). Approximate performance analysis of web services flow using stochastic petri net. In *International workshop on Grid and Cooperative Computing*, (pp. 193–200)., Shanghai, China.

Tang, F. & Parker, L. (2005). ASyMTRe : Automated synthesis of multi-robot task solutions through software reconfiguration. In *IEEE International Conference on Robotics and Automation*, (pp. 1501 – 1508)., Barcelona, Spain.

Tari, K., Amirat, Y., Chibani, A., Yachir, A., & Mellouk, A. (2010). Context-aware dynamic service composition in ubiquitous environment. In *IEEE International Conference On Communications*, South Africa.

Thrun, S., Fox, D., Burgard, W., & Dellaert, F. (2001). Robust monte carlo localization for mobile robots. *Artificial Intelligence*, *128*(1–2), 99–141.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., et al. (2007). Stanley : The robot that won the DARPA grand challenge. *The 2005 DARPA Grand Challenge*, 1–43.

Tigli, J.-y., Hourdin, V., Cheung-foo wo, D., Callegari, E., & Riveill, M. (2009). WComp middleware for ubiquitous computing : Aspects and composite event-based web services. *Annals of telecommunications*, 197–214.

Tigli, J.-Y., Lavirotte, S., Rey, G., Hourdin, V., Ferry, N., Vergoni, C., & Riveill, M. (2012). Low response time context awareness through extensible parameter adaptation with ORCA. *Annals of telecommunications*, *67*(7-8), 313–327.

Vanel, J.-M. (2010). N3 rules good practices - bypass semantic web limitations.

Vanrompay, Y., Rigole, P., & Berbers, Y. (2008). Genetic algorithm-based optimization of service composition and deployment. In *International workshop on Services integration in pervasive environments*, (pp. 13–18)., Sorrento, Italy.

Verborgh, R., Steiner, T., Deursen, D., Van de Walle, R., & Valles, J. (2011). Efficient runtime service discovery and consumption with hyperlinked RESTdesc. In *International Conference on Next Generation Web Services Practices*, (pp. 373 –379).

Verborgh, R., Steiner, T., Van Deursen, D., De Roo, J., Van de Walle, R., & Gabarró Vallés, J. (2011). Description and interaction of RESTful services for automatic discovery and execution. In *FTRA International Workshop on Advanced Future Multimedia Services*.

Vinoski, S. (2008). Serendipitous reuse. *IEEE Internet Computing*, *12*(1), 84 –87.

Vitvar, T., Kopecký, J., Viskova, J., & Fensel, D. (2008). WSMO-Lite annotations for web services. In *European Semantic Web Conference*, (pp. 674–689).

W3C (2001). Web service definition language (WSDL) 1.1. W3C note.

W3C (2003). QoS for web services : Requirements and possible approaches.

W3C (2004a). RDF primer. W3C recommendation.

W3C (2004b). RDF vocabulary description language 1.0 : RDF schema. W3C recommendation.

W3C (2006). Web services choreography description language : Primer. W3C working draft.

W3C (2007a). Semantic annotations for WSDL and XML schema. W3C recommendation, W3C.

W3C (2007b). SOAP version 1.2 part 1 : Messaging framework (second edition). W3C recommendation, W3C.

W3C (2007c). Web services policy 1.5 - framework (WS-Policy). W3C recommendation, W3C.

W3C (2008). SPARQL query language for RDF 1.0. W3C recommendation.

W3C (2009). OWL 2 web ontology language. W3C recommendation.

W3C (2010). RIF core dialect. W3C recommendation.

W3C (2011). Web services metadata exchange (WS-MetadataExchange). W3C recommendation.

Waibel, M., Beetz, M., Civera, J., D'Andrea, R., Elfring, J., Galvez-Lopez, D., Haussermann, K., Janssen, R., Montiel, J., Perzylo, A., Schiessle, B., Tenorth, M., Zweigle, O., & van de Molengraft, R. (2011). RoboEarth - a world wide web for robots. *IEEE Robotics Automation Magazine*, *18*(2), 69 –82.

Wang, Y. & Vassileva, J. (2007). A review on trust and reputation for web service selection. In *International Conference on Distributed Computing Systems Workshops*.

Want, R., Pering, T., Danneels, G., Kumar, M., Sundar, M., & Light, J. (2002). The personal server : Changing the way we think about ubiquitous computing. (pp. 194–209).

Weiser, M. (1991). The computer for the 21st century. *Scientific American*.

Weiser, M. (1994). The world is not a desktop. *Interactions*.

Weiser, M. & Brown, J. S. (1996). The coming age of calm technology. *Beyond Calculation*.

WfMC (1996). Workflow management coalition glossary and terminology.

Wordnik (2012). Swagger : A simple, open standard for describing REST APIs with JSON.

Yu, T., Zhang, Y., & Lin, K.-J. (2007). Efficient algorithms for web services selection with end-to-end QoS constraints. *ACM Transactions on the Web*, *1*(1).

Zaidenberg, S. (2009). *Apprentissage par renforcement de modèles de contexte pour l'informatique ambiante*. PhD thesis, Laboratoire d'Informatique de Grenoble.

Zaidenberg, S. & Reignier, P. (2011). Reinforcement learning of user preferences for a ubiquitous personal assistant. Advances in Reinforcement Learning. InTech.

Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J., & Chang, H. (2004). QoS-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, *30*(5), 311–327.

Zeng, L., Lei, H., & Chang, H. (2007). Monitoring the QoS for web services. In *International Conference on Service-Oriented Computing*, volume 4749, (pp. 132–144).

Zeng, L., Lingenfelder, C., Lei, H., & Chang, H. (2008). Event-driven quality of service prediction. In *International Conference on Service-Oriented Computing*, (pp. 147–161).

Zhu, F., Mutka, M., & Ni, L. (2005). Service discovery in pervasive computing environments. *IEEE Pervasive Computing*, *4*(4), 81 – 90.

Zouba, N., Bremond, F., & Thonnat, M. (2009). Multisensor fusion for monitoring elderly activities at home. In *IEEE International Conference on Advanced Video and Signal Based Surveillance*, (pp. 98 –103).

Annexe A

Compléments sur les services Web

A.1 Facettes du couplage fonctionnel

La classification proposée par Pautasso et Wilde [Pautasso & Wilde 2009] s'inscrit dans le contexte du débat entre les approches REST et WS-* et vise à orienter les choix techniques effectués en interne de ces deux approches. Elle se présente sous douze facettes :

1. *Découverte de services.* Exige-t'on des services qu'ils s'inscrivent eux-mêmes auprès d'un annuaire central ? Peuvent-ils être découverts par d'autres dispositifs comme des robots d'indexation suivant des liens ? Dans le premier cas, les services observent le respect d'une consigne stricte, ce qui n'est possible que dans un environnement contrôlé.
2. *Identification des entités manipulées.* L'identifiant d'une entité est-il dépendant d'un contexte d'utilisation ou bien global ?
3. *Mise en relation avec un service (binding).* Le choix d'un service est-il effectué en amont, au moment de déploiement voire de la conception, ou au dernier moment à l'exécution ? Bien qu'une mise en relation tardive semble aller dans le sens d'un plus faible couplage, un couplage étroit peut apparaître entre le client et le solveur si celui-ci est sollicité à chaque invocation.
4. *Interopérabilité des plateformes.* Un service est-il capable d'interagir avec d'autres services utilisant d'autres technologies de plateforme ? Un faible couplage implique ici la mise en place et le strict respect de standards.
5. *Interaction.* Deux services doivent-ils être disponibles en même temps pour pouvoir interagir ? Autrement dit, l'interaction est-elle synchrone ou asynchrone ? Une communication synchrone est bloquante pour le client et nécessite une réponse quasi-immédiate, ce qui présente également une forte contrainte pour le serveur.
6. *Orientation de l'interface entre le client et le serveur.* Les hypothèses entre le client et le serveur se limitent-elles à la définition du protocole ou impliquent-elles également l'utilisation d'une couche intergicielle ? Ces deux cas correspondent respectivement aux interfaces verticales et horizontales (voir figure A.1). L'interface verticale autorise une plus grande hétérogénéité entre les implémentations, le choix d'un composant intermédiaire de bas-niveau côté client restant strictement interne.

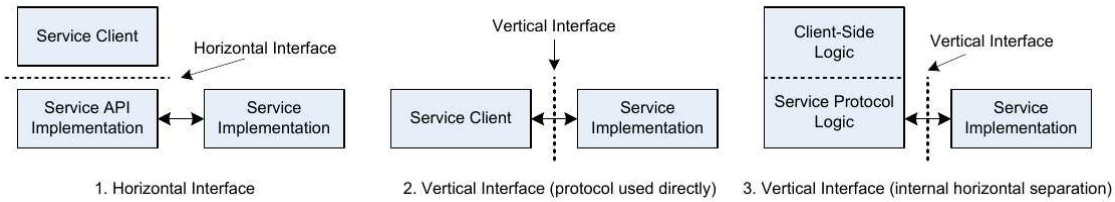


FIGURE A.1 – Orientations de l'interface client-serveur, dérivé de [Pautasso & Wilde 2009]

7. *Modèle.* Les documents échangés supposent-ils un modèle partagé ou sont-ils auto-descriptifs ?
8. *Granularité.* Les services ont-ils une interface fine ou grossière ? Une interface grossière vise à minimiser le nombre d'interactions afin de réduire le couplage au prix d'un plus grand volume de données échangées.
9. *État.* Les interactions mémorisent-elles un état ? La gestion d'un état partagé implique de nombreux mécanismes, accentuant ainsi le couplage.
10. *Évolution.* Les services offrent-ils une compatibilité descendante et ascendante ?
11. *Génération de code.* L'automatisation de l'interaction induit-elle de la génération de code statique ?
12. *Conversation.* L'invocation d'un service peut nécessiter plusieurs interactions, formant une conversation. Les mécanismes permettant de diriger une conversation sont-ils spécifiés au préalable ou bien découverts à l'exécution ?

Comparaison REST-HTTP et WS-* Ces deux approches se distinguent par les hypothèses qu'elles se donnent et, par conséquent, par leurs niveaux de couplages. L'architecture REST-HTTP a été comparée à l'approche WS-* couplée à un bus de service d'entreprise (ESB) [Pautasso & Wilde 2009] [1]. Leurs degrés de couplage sont résumés par la figure A.2.

On constate que l'exigence de couplage faible est dans l'ensemble plus prononcée pour les services REST-HTTP que pour son approche concurrente. WS-* a une orientation prescriptive [Pautasso & Wilde 2009], où l'utilisation privilégiée est celle de services au sein d'entreprises se pliant à des politiques et ayant des relations contractuelles [Vinoski 2008]. Ceci contraste avec l'hypothèse d'un monde ouvert à l'échelle du Web soutenue par REST, où les interactions sont ad-hoc [Fielding 2007, Pautasso & Wilde 2009] ; son orientation est alors davantage descriptive [Pautasso & Wilde 2009].

A.2 Services SOAP

A.2.1 Web Service Description Language (WSDL)

Web Service Description Language (WSDL) est disponible en deux versions : 1.1 [W3C 2001] et 2.0 [W3C 2007c], sous forme respectivement de note et de recommandation du W3C. La version la plus courante est 1.1 car elle est la seule à être supportée par le

1. L'étude incluait également l'architecture RPC basée sur HTTP que nous reproduisons pas ici.

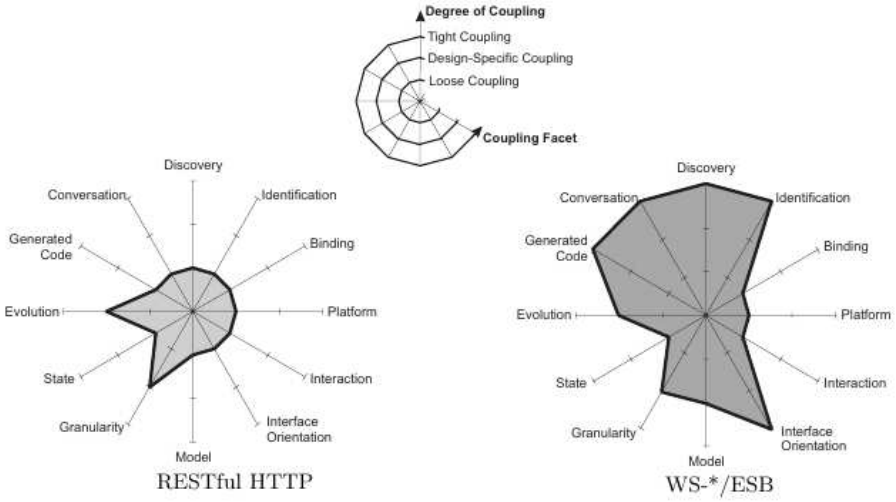


FIGURE A.2 – Degrés de couplage de différentes technologies de service Web. D'après [Pautasso & Wilde 2009]

standard d'orchestration BPEL *(Business Process Execution Language)* dans sa version 2.0.

WSDL décrit la localisation d'un service, les opérations qu'il fournit ainsi que les détails nécessaires à son invocation. Chaque opération du service fournit une fonctionnalité. Lorsque celui-ci est accessible dans le style RPC, une opération peut être vue comme une méthode distante à appeler.

Plus précisément, une description WSDL est composée d'une partie abstraite et d'une partie concrète. La partie abstraite spécifie les types de données échangées, la structure des messages ainsi que les opérations qu'elle regroupe en types de port. La partie concrète se charge de définir les extrémités communicantes *(endpoints)* à partir de leur adresse et des protocoles concrets utilisés pour fournir le type de port leur étant assigné. Enfin un service est défini comme l'ensemble des endpoints. La présence de plusieurs endpoints au sein d'un même service peut permettre le support de différents protocoles tout comme l'offre de différents niveaux de qualité pour un même service.

Une description WSDL est strictement syntaxique ; seule la structure de ses fonctionnalités est décrite par l'intermédiaire des schémas XSD impliqués. Ceci est suffisant pour générer automatiquement du code client, mais cette pratique est déconseillée car elle crée un couplage fort [2]. Des extensions sémantiques seront présentées dans la section 2.2. Par ailleurs, elle n'indique pas l'ordre d'exécution de ses opérations ; ceci est généralement géré au niveau de l'orchestration BPEL (voir la section 2.3).

Elle spécifie par contre les patrons d'échanges de message (MEP [3]) à partir de l'ordre des éléments d'entrée `wsdl:input` et de sortie `wsdl:output` au sein du bloc d'une opération. Ainsi l'échange de messages peut être un simple envoi (entrée uniquement), une requête suivie d'une réponse (entrée puis sortie), une notification (sortie uniquement) ou une sollicitation suivie d'une réponse (sortie puis entrée).

2. Cf. la facette *génération de code* de l'annexe A.1
3. *Message Exchange Patterns*

A.2.2 Extensions WS-*

Cette sous-section apporte des précisions sur les extensions WS-* décrites dans la sous-sous-section 2.1.2.3 et présente les extensions WS-Security et WS-ReliableMessaging.

Adressage des messages et des extrémités communicantes : WS-Addressing SOAP ne fournit pas en lui-même l'adressage des extrémités communicantes mais le relègue à la couche de transport (par exemple aux entêtes HTTP ou SMTP), ce qui limite le nombre de patrons de communication possibles. La spécification WS-Addressing [Box et al. 2004] vient ajouter cette fonctionnalité, appelée *EndPoint Reference* (EPR), en l'intégrant aux entêtes SOAP et aux descriptions WSDL. Cette fonctionnalité est rendue indépendante de la couche de transport [Papazoglou 2012]. Il devient possible de désigner l'émetteur, la destination mais aussi les futurs destinataires de la réponse, d'une éventuelle erreur ou d'un accusé de réception.

Au niveau de WSDL, la description des extrémités communicantes est augmentée en permettant d'adresser dynamiquement les instances de services via la propriété `resourceID` mais aussi en autorisant l'actualisation d'assertions de politique WS-Policy.

Enfin cette normalisation apporte plusieurs éléments utiles aux interactions à état *(stateful)* : l'identification d'un message est maintenant standard et les relations entre messages lors de conversations peuvent être explicitées. L'usage d'états dans les interactions est controversé car il augmente le couplage entre le client et le serveur [Fielding 2000, Richardson & Ruby 2008, Pautasso & Wilde 2009] : le serveur est contraint à mémoriser l'état de la session du client, ce qui peut limiter fortement son passage à l'échelle. Enfin la résilience de l'application est diminuée en cas de problème côté serveur : l'état ne peut être récupéré [Fielding 2000]. En termes d'effets recherchés, il facilite le contrôle du serveur sur une séquence imposée d'étapes et augmente la concision des messages échangés.

Communication événementielle : WS-Notification et WS-Eventing Un autre mode de communication fréquente dans les architectures distribuées est la communication événementielle entre services de type publication-souscription *(publish-subscribe)* [Carzaniga et al. 2000]. Un service intéressé par les événements sur un sujet ou émanant d'un service particulier peut être notifié lorsqu'un nouveau message (appelé événement) est publié moyennant l'inscription de son EPR auprès du service émetteur ou d'un intermédiaire que ce dernier a désigné. Pour cela, l'OASIS et le W3C ont publié deux spécifications concurrentes et non-compatibles : WS-Notification [OASIS 2006] et WS-Eventing [W3C 2011]. WS-Notification, par l'intermédiaire de ses trois spécifications offre des fonctionnalités plus riches que celles de WS-Eventing [Huang & Gannon 2006], notamment en matière de définition de sujets d'intérêt (WS-Topic). Les mécanismes de filtrage et de définition de sujets d'intérêts sont limités à un niveau syntaxique du fait des technologies XML usitées tels que XSD pour les sujets et XPath pour les expressions régulières de filtrage.

Politiques : WS-Policy La communication entre services exige régulièrement l'échange de propriétés supplémentaires à celles décrites dans les descriptions WSDL. Ces propriétés prennent la forme d'assertions qui expriment des préférences, des exigences ou des capacités et sont regroupées en ensembles cohérents, appelés politiques [Papazoglou 2012].

Ces politiques couvrent divers domaines comme la performance, la sécurité, la fiabilité des messages et la gestion de transactions et s'appuient sur un framework standardisé par le W3C : WS-Policy [W3C 2007c].

En général, le client, le serveur et le service ont chacun leur propre politique qu'ils s'échangent sous forme d'entêtes de message ou par l'intermédiaire d'un protocole dédié : WS-MetadataExchange [W3C 2011]. Ces politiques sont ensuite conciliées en une politique effective par des opérations d'intersection [Baresi et al. 2006].

Plusieurs extensions ont été standardisées par l'OASIS dans des domaines particuliers comme la sécurité (WS-SecurityPolicy [OASIS 2007b]) et la fiabilité des messages (WS-ReliableMessaging [OASIS 2009b]). Chaari et al. [Chaari et al. 2008] ont également proposé une extension permettant la confrontation entre les exigences d'un client et les capacités d'un fournisseur en termes de paramètres de Qualité de Service (QoS).

Sécurité : WS-Security Conformément au principe de SOAP, la sécurité s'exerce avant tout au niveau des messages et non au niveau du transport [4].

La famille WS-Security [W3C 2007c] offre un riche ensemble de spécifications répondant aux besoins des entreprises, réputés complexes [Papazoglou 2012, Richardson & Ruby 2008]. On y retrouve de nombreux standards de la sécurité comme XML encryption et XML signature pour le chiffrement et la signature de nœuds XML, les assertions SAML pour demander et affirmer des faits ainsi qu'XACML pour le contrôle d'accès. Par ailleurs, le mécanisme de jeton est couramment utilisé pour l'autorisation. Les exigences, capacités et préférences des services sont regroupées sous forme de politiques de sécurité spécifiées à l'aide de l'extension WS-SecurityPolicy.

Des mécanismes de fédération (WS-Federation [OASIS 2009c]) et de courtage de confiance (WS-Trust [OASIS 2007c]) permettent la communication sécurisée entre services de différentes entreprises. Toutefois, ces mécanismes de confiance repose essentiellement sur des infrastructures à clés publiques qui sont accessibles aux entreprises mais beaucoup plus compliqués à mettre en œuvre pour des particuliers ou des objets communicants de l'Internet des Objets. Pour des raisons de simplicité, nos contributions ne traiteront pas la question difficile mais majeure de la sécurité au sein des systèmes distribués.

Fiabilité des messages : WS-ReliableMessaging Les protocoles de transport comme HTTP n'étant pas fiables, l'envoi d'un message SOAP ne garantit pas de base sa réception. Par ailleurs, à cause des intermédiaires, il peut arriver que le message soit dupliqué. Ne bénéficiant pas d'opération idempotente, SOAP a nécessité l'élaboration d'une extension spécifique, WS-ReliableMessaging [OASIS 2009b], pour l'envoi de messages sensibles. Cette extension, indépendante du protocole de transport, gère la numérotation des messages et les accusés de réception au prix bien entendu d'un surcoût notable.

4. L'usage d'un chiffrement TLS (Transport Layer Security) est notoirement insuffisant car SOAP est prévu pour avoir des nœuds intermédiaires : TLS n'est donc pas en mesure de lui assurer un sécurité de bout en bout [Papazoglou 2012].

A.3 Services REST-HTTP

A.3.1 Interface uniforme

L'interface uniforme repose sur les verbes du protocole HTTP ansi que sur ses codes de retour [Alarcón et al. 2011]. Les verbes permettent les usages suivants [Richardson & Ruby 2008] :

1. Réception d'une représentation de la ressource : `GET`
2. Création d'une nouvelle ressource : `PUT` pour une nouvelle ressource, `POST` auprès d'une ressource existante
3. Modification d'une ressource existante : `PUT`
4. Suppression d'une ressource existante : `DELETE`
5. Récupération des métadonnées uniquement : `HEAD`
6. Information sur les méthodes supportées : `OPTIONS`

Conformément au style REST, les verbes `GET`,`PUT` et `POST` réalisent des transferts d'état au travers de représentations. En effet, `GET` reçoit une représentation de la ressource fournie par le serveur, tandis que `POST` et `PUT` consistent en l'envoi d'une représentation afin de créer ou de mettre à jour l'état d'une ressource. Concernant la méthode `POST`, l'architecture REST-HTTP encourage deux usages mutuellement très proches : la création d'une ressource subordonnée et l'ajout d'information à l'état d'une ressource [Richardson & Ruby 2008]. Par ailleurs, la spécification d'HTTP 1.1 [IETF 1999a] prévoit un troisième usage où le serveur effectue un traitement et renvoie le résultat sans créer ni altérer de ressource. HTTP RPC et SOAP/HTTP ont recours systématiquement à cet usage.

Propriétés des méthodes Cet usage des méthodes HTTP offre deux propriétés importantes : `GET` et `HEAD` sont sûres et `GET`, `HEAD`, `PUT` et `DELETE` sont idempotentes. En effet, `GET` et `HEAD` ne changent pas l'état du serveur tandis que l'envoi de plusieurs requêtes identiques `GET`, `HEAD`, `PUT` ou `DELETE` au serveur conduit au même résultat qu'une seule de ces requêtes. Ces propriétés sont importantes pour un réseau non fiable comme Internet : lorsqu'une requête n'a pas reçu de réponse, il suffit de la réitérer sans crainte d'effet de bord. Seule la méthode `POST` n'est pas idempotente et doit donc être utilisée à bon escient.

A.3.2 Web Application Description Language (WADL)

WSDL intègre le protocole HTTP dans sa version 2.0. Cependant, il n'est pas adéquat pour représenter les ressources car celles-ci peuvent être générées dynamiquement, et chacune d'entre elles disposent d'une ou de plusieurs méthodes. Il favorise l'exposition d'un nombre très restreint de ressources ayant une URI pré-déterminée.

Dans la lignée de WSDL, Hadley [Hadley 2006] a proposé un langage de description syntaxique destiné aux services REST : WADL *(Web Application Description Language)*. Il prend en considération des ensembles de ressources, qu'il désigne à l'aide de patrons d'URI (URI Templates [IETF 2012]) et leur associe une opération. Comme en WSDL, ces opérations sont décrites par leurs entrées et leurs sorties qui, lorsqu'elles sont au format XML, peuvent être décrites par un schéma XSD.

Annexe B

Prototypes réalisés

B.1 Prototype d'architecture orientée service

Les exigences et les recommandations architecturales proposées dans la sous-section 3.3.1 s'appuient sur l'expérience tirée du prototype suivant.

Ce prototype a été implémenté en C++ à partir du cadriciel *(framework)* ARéVi [1]. L'origine du choix de ce cadriciel remonte au besoin de modéliser le comportement de services de positionnement en effectuant des lancers de rayons des capteurs vers un objet suivi. Le lancer de rayon est une fonctionnalité offerte par ARéVi, qui fournit une interface de haut-niveau au-dessus d'OpenGL. Notre bonne connaissance préalable du cadriciel nous a permis de répondre à une première échéance courte. ARéVi contient également un certain d'autres fonctionnalités utiles à notre modélisation : la communication événementielle du type souscription-publication, la réification des classes qui permet d'effectuer des tests de subsomption ainsi qu'un ordonnanceur permettant à divers agents d'évoluer à des fréquences différentes. Il nous décharge également de la complexe gestion de la mémoire en fournissant un ramasse-miettes (*garbage collector*).

À l'occasion de ce prototypage, un nombre important de fonctionnalités ont été modélisées comme des services logiciels. Outre les services impliqués dans le déplacement de robot, il existe, en autres des services de mesure de métriques, d'observation du contexte, de sélection de services, de normalisation de métrique et d'évaluation. Par ailleurs, une place centrale a été accordée à la sélection de services : toutes les tâches requises (à l'exception des requêtes de méta-sélection de services) font l'objet d'une sélection de services. Il apparaît que dans une majorité de cas, cette sélection est triviale car il n'existe qu'une seule alternative pour réaliser la tâche requise. La découverte de services vérifie en effet que les services candidats sont disponibles et qu'ils supportent la tâche requise et non simplement le type de tâche requis. Par exemple, la présence de plusieurs robots mobiles fournissant un service de déplacement n'entraîne pas de concurrence lorsqu'il s'agit de déplacer un robot donné. Un client ordinaire n'a besoin que de connaître un méta-sélectionneur. Seuls les services de sélection de services, comme ce service de méta-sélection, ont besoin de faire appel aux fonctionnalités de découverte de services. Cette découverte repose sur un annuaire central où tous les fournisseurs de services vont inscrire leurs services.

Les instances de services sont réifiées comme des objets. Ce prototype étant mono-

1. `http://svn.cerv.fr/trac/AReVi/`

processus, leurs adresses mémoire peuvent être partagées et utilisées comme des identifiants déréférençables (à l'image des URI HTTP pour les services Web). Les instances de service de certains types en cours d'exécution sont référencées dans des annuaires spécialisés. Aussi, par exemple, un client, intéressé par un flot de données particulier, peut souscrire à une instance en cours d'exécution qui traite une requête émise par un autre client.

Le remplacement d'instances de sous-service ayant échoué est une pratique courante dans ce prototype. Néanmoins, le remplacement ne doit pas être effectué de façon systématique pour n'importe quel type de service. Si une instance de service de mesure échoue au milieu de l'exécution de l'instance de service observée, celle-ci ne doit pas être remplacée par une autre instance ayant des valeurs initiales nulles.

B.2 Sélectionneur

Le sélectionneur utilisé dans les simulations de la section 4.4 a été implémenté en Python. Ce prototype se présente comme un module Python qui, à l'avenir, pourrait être englobé dans un service Web. Il offre une interface de programmation simple supportant les identifiants de services, d'instances, de tâches sous forme de chaînes de caractères arbitraires. Dans la version actuelle, les alternatives doivent être fournies au module en même temps que les requêtes. Cette interface a été utilisée par implémenter un service de sélection de services [2] au sein de notre prototype d'architecture orientée service réalisée en C++ (cf. section précédente B.1). Cette intégration a été rendue possible par les *bindings* CPython qui permettent l'intégration de code Python au sein de programmes écrits en C/C++. Cette implémentation s'appuie partiellement sur les librairies Scipy, Numpy et PyGraph. Matplotlib et IPython ont été, quant à eux, d'une grande aide pour visualiser nos résultats.

Pour des besoins de performance, des mécanismes de cache ont été mis en place afin d'accélérer les simulations de type *Monte-Carlo Tree Search*. Cependant, la construction de l'arbre de recherche reste la tâche la plus coûteuse et il conviendrait d'en proposer une réécriture avec des structures de données de plus bas-niveau.

2. Celui-ci n'a cependant pas été utilisé dans nos simulations. Ces simulations ont été entièrement implémentées en Python.