Canevas sémantique et méthodologie formelle pour le développement des applications ambiantes multi-domaine

Mohamed Hilia

UNIVERSITÉ PARIS-EST

École Doctorale Mathématiques et STIC
Laboratoire Images Signaux et Systèmes Intelligents
LISSI (EA 3956)

T H È S E

Spécialité : Informatique

Présentée par

Mohamed Hilia

Canevas Sémantique & Méthodologie Formelle pour le Développement des Applications Ambiantes Multi-domaine

Soutenue le : 16 décembre 2013

Devant le jury composé de :

Présidente du jury	:	Claude Godart	Professeur à l'Université de Lorraine
Rapporteurs	:	Khalil Drira	Directeur de recherche à l'Université de Toulouse
		Salima Benbernou	Professeur à l'Université Paris Descartes
Examinateurs	:	Kamel Barkaoui	Professeur au CNAM
		Dimitris Plexousakis	Professeur à l'Université de Crète
		Abdelghani Chibani	Maître de conférences à l'Université Paris-Est Créteil
Directeur	:	Karim Djouani	Professeur à l'Université Paris-Est Créteil
Co-Directeur	:	Yacine Amirat	Professeur à l'Université Paris-Est Créteil
Invité	:	Thierry Winter	Directeur de recherche à Bull-Evidian

Résumé de la thèse L'intelligence ambiante [1] est considérée comme l'une des évolutions majeures de l'informatique ubiquitaire. Elle vise la mise en œuvre ou des applications pour des environnement dits ambiants ou des espaces intelligents permettant d'améliorer la vie quotidienne des utilisateurs, leur bien-être et leur sécurité. Un environnement ambiant dispose d'une multitude d'équipements (capteurs, actionneurs), fournissant des services ubiquitaires atomiques qui sont distribués et hétérogènes. Ces services atomiques, disponibles dans ce type d'environnement, ne répondent pas directement ou complètement à l'ensemble des exigences et des besoins utilisateurs. Par ailleurs, des services de haut niveau peuvent impliquer l'interaction de services ubiquitaires appartenant et/ou contrôlés par différentes structures ou domaines (i.e. environnement multi-domaine). Les travaux de cette thèse concernent les problématiques liés à l'agrégation des services ubiquitaires atomiques dans des coopérations et des compositions dans un environnement ambiant multi-domaine afin de répondre aux besoins utilisateurs. Un intérêt particulier est porté à l'interopérabilité sémantique et comportementale des compositions de services ainsi que leur spécification dans un modèle formel basé sur la logique constructive $\mathcal{BCDL}_0$. la contribution de la présente thèse concerne la réalisation d'un canevas sémantique permettant la specification et la vérification formelles des processus coopératifs avec l'assistant à la preuve Isabelle/HOL. Le canevas sémantique proposé consiste en une ontologie de coopération extensible et en un langage de spécification de services fiables "*Sound*" ainsi que des modules de mise en correspondance vers des infrastructures cibles.

[1] Ambient Intelligence

Abstract Ambient Intelligence is considered as the major application domain of ubiquitous computing. It aims at implementing intelligent environments to improve the daily activities, the well-being and the safety of users. An ambient environment has a multitude of devices (sensors, actuators), providing atomic ubiquitous services which are distributed and heterogeneous. These atomic services available in these environments, do not directly or fully meet all the requirements and users' needs. Moreover, high-level services may involve the interaction of ubiquitous services owned and/or controlled by different structures or domains (i.e. multi-domain environment). The present thesis focuses on the problems related to the aggregation of atomic ubiquitous services in an ambient multi-domain environment, by cooperation and composition, in order to meet users' needs. This study concerns the semantic and behavioral interoperability of high-level composite services and their formal specification. The later, is based on a the Basic Constructive Description Logic, namely, $\mathcal{BCDL}_0$ formal model. The contribution of the present work concerns the realisation of a semantic framework for the design and the formal verification of cooperative processes under the interactive theorem prover Isabelle/HOL. The proposed semantic framework consists of an extensible cooperation ontology, a sound formal specification language of ubiquitous services and a software components for mapping the formally proved cooperative processes to the targeted infrastructures.

Liste de publications

- Mohamed Hilia, Abdelghani Chibani, Karim Djouani and Yacine Amirat. "Formal Specification and Verification For Ubiquitous Robotic Services Composition", In 4th Workshop on Formal Methods for Robotics and Automation in conjunction with **(RSS 2013**), Berlin,Germany, 2013.

- Mohamed Hilia, Abdelghani Chibani, Karim Djouani. "Trends and Challenges in Formal Specification and Verification of Services Composition in Ambient Assisted Living Applications". Springer. The 4th International Conference on Ambient Systems, Networks and Technologies **(ANT 2013)**, June 25-28, 2013, Halifax, Nova Scotia, Canada, pages 540-547

- Mohamed Hilia, Abdelghani Chibani, Karim Djouani and Yacine Amirat. "Semantic Service Composition Framework for Multidomain Ubiquitous Computing Applications". 10th International Conference on Service Oriented Computing **(ICSOC 2012)** Nov. 12-15 Shanghai, China 2012, pages 450-467

- Mohamed Hilia, Abdelghani Chibani, Yacine Amirat and Karim Djouani. "Cross-Organizational Cooperation Framework for Security Management in Ubiquitous Computing Environment". 23rd IEEE International Conference on Tools with Artificial Intelligence **(ICTAI 2011)** Nov. 7-9, 2011, Boca Raton, Florida, USA 2011, pages 464-471

Remerciements

Je tiens à remercier mes directeurs de thèse, M Djouani Karim et M Amirat Yacine, pour la proposition de cette thèse et pour leur directives humaines et scientifiques afin de réussir ces travaux de thèse.

Je remercie Mme Salima Benbernou et M Khalil Drira, rapporteurs, d'avoir accepté la charge de rapporter ce travail de thèse ainsi que pour l'intérêt et les critiques portés à ces travaux.

Je remercie M Kamel Barkaoui, M Dimitris Plexousakis et M Claude Godart, examinateurs, pour leurs nombreuses questions et leurs suggestions sur des perspectives à ces travaux.

Je tiens à remercier le directeur de recherche d'Evidian, M Thierry Winter, pour sa confiance et son soutien.

Je tiens à remercier M Abdelghani Chibani d'être présent dans les périodes de crise, pour ses conseils ainsi que les discussions menées pendant toute la période de thèse.

Un remerciement pour mes parents, ma famille et mes amis en particulier ceux au laboratoire LISSI de m'avoir encouragé jusqu'aux derniers moments et d'avoir vécu avec moi cette aventure de thèse pendant ces longues années.

Tables des matières

Liste des figures

Liste des tableaux

1

Introduction générale

Les travaux de recherche présentés ici s'inscrivent dans le cadre du domaine de l'informatique ambiante[1]. Ils portent, plus particulièrement sur la description et la validation formelle des compositions de services ubiquitaires et la construction de services composites corrects (valide au sens des preuves formelles) dans un environnement ambiant multi-domaine. Un environnement ambiant dispose d'une multitude d'équipements (capteurs, actionneurs), fournissant des services ubiquitaires distribués et de nature hétérogène. Ces services atomiques disponibles dans ce type d'environnement, ne répondent pas directement ou complètement à toutes les exigences et les besoins des utilisateurs. Ces services atomiques sont adaptés ou agrégés d'une manière ou d'une autre afin de proposer de nouvelles fonctionnalités *intelligentes* dans l'environnement comme le service de notification de messages qui s'adapte à la situation de l'utilisateur. Ces services ubiquitaires enrichissent la multitude de services disponibles, afin de satisfaire les requêtes de plus haut niveau de l'utilisateur, en lui offrant des services adaptés. Cependant, ces derniers de moyen et de bas niveaux ne sont pas toujours en mesure de répondre aux nouvelles exigences des utilisateurs qui sont de plus en plus complexes et plus critiques. Prenons l'exemple d'un environnement ambiant dans le cadre de la santé. Les utilisateurs ont besoin de services dans leur domicile qui sont capables d'assurer le suivi des programmes de rééducation en plus du suivi médical classique. Ces services de haut niveau impliquent l'interaction de services appartenant et/ou contrôlés par différentes structures (i.e. domaines) à savoir l'hôpital, le centre d'urgence, le centre de kinésithérapie et le domicile. En effet, ces services de haut niveau peuvent être construits par composition des services offerts par plusieurs environnements ambiants. Ils peuvent être construits aussi par l'agrégation (ou la coopération) des services offerts par différent domaines ambiants, en utilisant les moyens et les compétences de chaque domaine. Dans le cas d'utilisation présenté pour démontrer l'intérêt des concepts et des approches proposées dans le cadre de ce travail de recherche, on considère que les services atomiques, propres à chaque domaine ambiant, sont déjà en place. Parmi ces services, on peut citer les services de suivi des patients au niveau de l'hôpital, les services de gestion des contenus

[1] Ambiante Intelligence

des programmes de rééducation, au niveau du centre de kinésithérapie ou les services ubiquitaires offerts au niveau de la maison intelligente, domicile du patient, comme le service de localisation, le service de notification, le service de diffusion de flux vidéo ou les services offerts par les robots d'assistance.

La coopération entre des services offerts par des domaines hétérogènes, distribués et indépendants engendre des problématiques nouvelles dans un contexte dit *environnement ambiant multi-domaine*. La composition de ces services dans des processus coopératifs de plus haut niveau est le résultat des coopérations multi-domaine à base de services. Généralement, dans les systèmes actuels d'intelligence ambiante les équipements sont connectés sur des réseaux locaux. Cependant, aucune contrainte sur les aspects de sécurité en particulier sur le contrôle d'accès et d'usage (i.e. l'authentification et l'autorisation) n'est discutée ici. Par ailleurs, dans un système multi-domaine où les domaines sont indépendants, des problématiques sur la confidentialité des données doivent être adressées à savoir la non révélation du vocabulaire utilisé dans la spécification des processus internes ainsi que leur structure qui rend accessible le savoir-faire d'un domaine. La criticité du système, dont l'utilisateur est le centre d'intérêt, impose une certaine rigueur dans sa conception. Par conséquent, l'étape de la vérification formelle de la propriété de *correction* des processus coopératifs issues des coopérations est indispensable. Dans ces travaux de thèse, on propose un canevas sémantique pour enrichir l'environnement ambiant multi-domaine avec des services composites corrects de plus haut niveau. Ce canevas aidera à leurs spécifications et à leurs conceptions à travers une composition statique. Ces compositions sont organisées dans des chorégraphies de services appartenant à plusieurs domaines. Du moment ou notre conception des services de haut niveau est en place, d'autres mécanismes et techniques permettent de relier les descriptions sémantiques et formelles aux services concrets dans les domaines cible.

1.1 Structure du document

L'objectif de la thèse est la proposition d'un canevas permettant la mise en œuvre des applications ubiquitaire multi-domaine. Le canevas doit assurer un certain nombre de propriétés.

- Représenter la sémantique partagée par les différents domaines participant à la coopération.
- Préserver le savoir-faire de chacun des domaines par l'exposition des vues sur les services internes.
- Proposer l'utilisation d'un langage formel pour la spécification des vues et démontrer la fiabilité [2] de ce langage.

[2] Soundness

- Assurer la vérification formelle de la composition multi-domaine des vues sur les services, cela implique l'assurance de la propriété de correction [3] de la coopération.

Chapitre 2 : État de l'art : Informatique ubiquitaire et composition de services

Dans ce chapitre, on présente l'informatique ubiquitaire orientée services et les avantages des architectures orientées services dans la mise en œuvre des applications ubiquitaires. Cette mise en œuvre nécessite des formalismes et des langages de description ainsi que des plateformes d'exécution. Les formalismes et les technologies permettant le développement des applications ambiantes à base de services seront passés en revue. Cependant, les approches de description de services proposées restent à un niveau syntaxique (e.g. WSDL) et manquent de sémantique, ce qui entrave leurs exploitations dans certaines tâches d'analyse, d'automatisation (e.g. la composition de service) et de raisonnement (e.g. la preuve formelle). Cette problématique est partiellement résolue à l'aide des technologies du web sémantique. Par conséquent, de nouveaux langages de modélisation (i.e. description) ont été proposés ainsi que des méthodes et des techniques permettant l'utilisation des descriptions de services dits *atomiques* pour construire de nouveaux services de haut niveau dits *composites*. Une partie de ce chapitre est consacrée à l'état de l'art sur les méthodes de composition de services et en particulier celles proposées dans le contexte de l'intelligence ambiante. Parmi ces méthodes, on s'intéresse à celles basées sur la preuve de théorèmes et la construction des processus coopératifs (i.e. multi-domaine) corrects. Par ailleurs, l'intérêt des langages de spécification de services avec une sémantique formelle permettant de réaliser des compositions de services corrects sera discuté pour montrer le besoin d'un canevas sémantique et d'une méthodologie formelle pour le développement des services composites abstraits dans un environnement ambiant multi-domaine.

Chapitre 3 : Vers un canevas sémantique pour la composition de services multi-domaine

Dans ce chapitre, les concepts clés utilisés tout au long de ce manuscrit sont définis, avant de donner une liste non exhaustive des exigences auxquelles un canevas de coopération dans un environnement multi-domaine doit répondre. On se focalisera particulièrement sur les exigences d'interopérabilité sémantique et comportementale avec la préservation de l'étanchéité et de la confidentialité relatives au savoir-faire d'un domaine comme la flexibilité, la réutilisation et la preuve de la propriété de correction par rapport à la spécification de la coopération multi-domaine. Une approche hybride est proposée par la suite, combinant les avantages des trois

[3] Correctness

catégories d'approches discutées dans ce chapitre. Premièrement, l'approche proposée consiste en un modèle sémantique utilisé dans la spécification des processus de coopération. Cette spécification se base sur un système formel et une ontologie de coopération de haut niveau. Deuxièmement, elle permet de vérifier formellement les processus coopératifs ainsi spécifiés. Finalement, ce canevas présente l'avantage de modéliser et de séparer, au niveau du modèle qu'il propose, les deux types d'actions qu'on considère indispensables dans un environnement multi-domaine, à savoir, les *actions métiers* ou fonctionnelles et les actions non-fonctionnelles, ou les actions de configuration dites aussi *actions de provisioning*.

Chapitre 4 : Spécification et validation formelle sur l'outil Isabelle/HOL

Dans ce chapitre on présente une variante des logiques de description constructive appelée *logique de description constructive basique*, notée $\mathcal{BCDL}_0$. Cette logique sera employée lors de la spécification et la validation formelle du modèle sémantique introduit dans le chapitre 3. Dans une première partie, une introduction aux logiques de description ainsi qu'un aperçu sur les méthodes formelles sont présentés. Par ailleurs, les limites des sémantiques classiques par rapport à la preuve formelle de théorèmes et le besoin de passer à des sémantiques constructives sont discutés. Cette sémantique constructive dite aussi *intuitionniste* a de nombreux avantages à savoir, le respect du paradigme *proofs-as-program*, la capacité de donner des sémantiques au calcul de preuves ainsi que le support des techniques de la preuve de théorèmes utilisées pour la vérification de la *correction* des spécifications d'un système donné. La mise en œuvre du système formel basé sur $\mathcal{BCDL}_0$ et la preuve de sa fiabilité [4] dans l'assistant à la preuve Isabelle/HOL sont achevés. Dans une deuxième partie, le système formel est utilisé pour la spécification du modèle sémantique proposé dans le chapitre 3 ainsi que la formalisation des vues abstraites sur les services et les processus multi-domaine. La spécification formelle de la composition de services multi-domaine, la formalisation du langage de composition ainsi que l'implémentation et la preuve dans Isabelle/HOL sont présentés. Finalement, un scénario relatif au domaine de la santé est présenté en se basant sur l'approche formelle proposée. La preuve interactive dans Isabelle/HOL permettra d'évaluer l'impact d'un changement de la composition des processus (services) coopératifs sur la propriété de *correction*[5].

Chapitre 5 : Implémentation et expérimentation

Dans ce chapitre, la faisabilité du canevas sémantique proposé, pour le développement des applications dans un environnement d'intelligence ambiante multi-domaine est discutée. Ces

[4] Soundness
[5] Correctness

applications se basent essentiellement sur la définition des services et de leur composition dans des contextes particuliers afin de répondre à des besoins utilisateurs dans ces environnements. Ce chapitre présente la mise en œuvre de notre étude de cas développée au sein du laboratoire LISSI[6] en collaboration avec l'hôpital UPEC[7] Henry Mondor. Cette étude permettra la validation de la méthodologie et du canevas sémantique proposés dans les chapitres trois et quatre. Ce chapitre, présente aussi les aspects techniques et technologiques ainsi que les outils d'implémentation pour la mise en œuvre de cette étude de cas. Cette dernière est représentée par un scénario dans le domaine médical. Ce scénario, concerne la coopération entre plusieurs domaines ambiants pour faire réussir la gestion des programmes d'auto-rééducation des personnes âgées à domicile.

Chapitre 6 : Conclusion et Perspectives

Dans ce chapitre, les conclusions relatives aux travaux menés dans le cadre de la présente thèse sont discutées avant de donner quelques perspectives et directions futures à ces travaux.

[6]Laboratoire Images, Signaux et Systèmes Intelligents
[7]Université Paris-Est Créteil

2

État de l'art : Informatique ubiquitaire et composition de services

Sommaire du chapitre

Introduction

L'objectif principal des services ubiquitaires est l'amélioration de la vie quotidienne et l'assurance du bien être des utilisateurs dans leurs environnements. La construction de nouveaux services est réalisée par l'agrégation (*la composition*) de services disponibles, en prenant en compte plusieurs paramètres, comme la situation des utilisateurs, leurs états, l'état de l'environnement, etc. On remarque que, la notion de service, qu'il soit atomique ou composé, est centrale dans la définition des applications dans un environnement d'intelligence ambiante (AmI)[1]. Cependant, dans certains cas, la proposition d'un service composite implique l'accès et l'usage des services atomiques appartenant à plusieurs organisations (i.e. domaines). Dans ce cas, les services disponibles dans plusieurs environnements ont besoin d'être composés dans le but de coopérer afin de répondre aux besoins exprimés. En effet, cette coopération se réalise par la composition de services exposés par ces différents domaines sous forme de vues abstraites on parle alors, d'une composition de services multi-domaine.

2.1 Informatique ubiquitaire orientée services

L'informatique ubiquitaire [2] représente une nouvelle vision des systèmes d'information où l'interaction et la communication entre les utilisateurs et l'environnement physique s'effectue de manière transparente par l'intermédiaire de dispositifs "intelligents" disponibles dans l'environnement physique [Ramos et al., 2008]. Le concept de "Informatique Ubiquitaire" initialement introduit en 1991 par Wieser, a donné un nouvel élan aux systèmes informatiques vers l'informatique omniprésente ou ubiquitaire [Weiser, 1991]. Cette nouvelle vision de l'informatique a permis d'élargir le spectre des applications utilisateurs ainsi que des modes d'interaction possibles afin d'améliorer les activités quotidiennes. Bien que ce domaine a plus d'une vingtaine d'années, il continue toujours à poser des défis de recherche, en l'occurrence la prise en compte de l'hétérogénéité de l'environnement, de la sensibilité au contexte, de l'adaptation à la dynamique de l'environnement, du raisonnement sémantique, des interactions multi-domaine, de la gestion des identités des utilisateurs, du contrôle d'accès et d'usages, de la protection de la vie privée des utilisateurs, etc.

On considère que les utilisateurs sont entourés des équipements "intelligents" qui encapsulent des services qui sont accessibles de n'importe quel endroit et à n'importe quel moment. Le bon fonctionnement de ces services nécessite de localiser les utilisateurs, pour comprendre leur environnement et d'anticiper de manière proactive leurs besoins.

La notion de service qu'il soit élémentaire ou complexe, est centrale dans la mise en œuvre des architectures des applications de l'informatique ubiquitaire [Miguel and Roberto, 2009]. Toute

[1] Ambient Intelligence
[2] Ubiquitous Computing

entité, identifiée comme un objet (i.e. objet virtuel ou matériel) ayant la capacité de communiquer à travers le réseau, est considérée comme étant un fournisseur de services ou un consommateur/demandeur de services. D'autres types de services sont fournis par ces dispositifs via des agents souvent appelés des actionneurs, comme des effecteurs simples tel qu'un mécanisme d'ouverture automatique d'une porte ou complexe tel qu'un robot mobile. Ces effecteurs sont capables d'agir sur l'environnement en exécutant des actions ou des tâches utilisateurs. Ils sont capables aussi de raisonner afin d'adapter l'exécution des services pour les utilisateurs dans de tels environnements. Un service par définition est une entité logicielle autonome qui fournit une fonction bien définie permettant, par exemple, de remonter les mesures des capteurs, de contrôler l'exécution des tâches effectuées par des actionneurs (e.g. robots), de fusionner et d'analyser de données multi-capteur, ou accomplir des tâches complexes pour assister les usagers. Le concept de service permet d'encapsuler les fonctionnalités des infrastructures matérielles et logicielles à travers des interfaces abstraites permettant de faciliter leur utilisation. Un service peut être publié et rendu disponible pour être invoqué par d'autres modules logiciels (services ou applications). Il est défini aussi par un contrat (appelé aussi interface) qui est une spécification abstraite de ses fonctionnalités décrivant ce que fait le service, comment l'invoquer et éventuellement quelles sont ses propriétés non fonctionnelles.

La spécification des architectures des applications de l'informatique ubiquitaire repose principalement sur les standards de l'informatique orientée services comme SOA[3] [Tigli et al., 2011] ou SOC [4]. En particulier ceux traitant de la description et de la découverte de services ainsi que leur composition de plus haut niveau répondant à des besoins des utilisateurs dans des environnements du quotidien comme les aéroports, les hôpitaux, les domiciles, etc. On parle alors de service ubiquitaire multi-domaine. La description de ces services, de leurs compositions ainsi que la vérification de la correction de ces compositions sont essentielles [Miguel and Roberto, 2009, Preuveneers and Novais, 2012], notamment dans deux domaines d'application majeurs qui sont l'intelligence ambiante [Cook et al., 2009] et la robotique ubiquitaire [Saffiotti and Broxvall, 2005] ayant comme fondement l'informatique ubiquitaire orientée services.

2.1.1 Intelligence Ambiante

L'intelligence ambiante [5] est considérée comme l'une des évolutions majeures de l'informatique ubiquitaire. Elle vise la mise en œuvre d'espaces intelligents permettant d'améliorer la vie quotidienne des utilisateurs, leur bien être et leur sécurité [Cook et al., 2009]. Les systèmes d'intelligence ambiante sont souvent mis en œuvre à l'aide d'agents capables d'exécuter des tâches de manière pro-active en tenant compte du contexte des utilisateurs [Ramos et al., 2008], (Figure 2.1). Par

[3]Service Oriented Architecture
[4]Service Oriented Computing
[5]Ambient Intelligence

ailleurs, l'intelligence ambiante repose sur plusieurs techniques de planification, de raisonnement et de représentation des connaissances proposées par l'IA [Russell and Norvig, 2009].

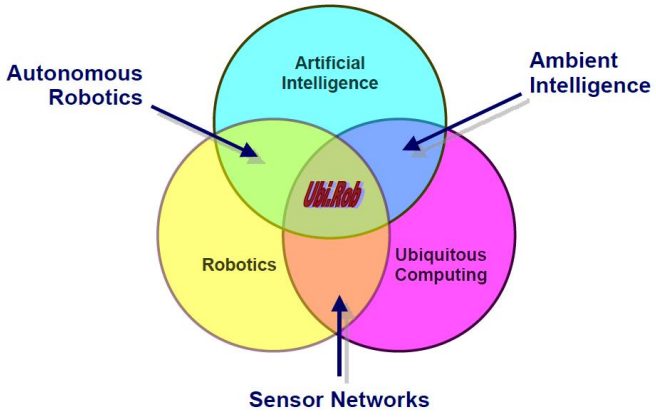


Figure 2.1: Robotique Ubiquitaire [Saffiotti and Broxvall, 2005]

2.1.2 Robotique ubiquitaire

L'émergence des robots dans les applications d'intelligence ambiante permet d'envisager des services plus élaborés pour effectuer des tâches simples ou complexes en coopérant, interagissant avec les dispositifs disponibles dans leur environnement [Saffiotti and Broxvall, 2005] afin d'assurer des activités quotidiennes, ou simplement de mettre à disposition des utilisateurs des services qui assurent leur bien être tout en garantissant leur sécurité (Safety). La robotique ubiquitaire est un paradigme émergent qui vise à offrir à l'utilisateur des services de manière transparente, en tout lieu et en tout instant. Le projet *PEIS-Ecology*[6] présente cette vision d'un système qui combine la discipline de la robotique et de l'intelligence ambiante [Saffiotti et al., 2008] permettant la construction de robots intelligents au service de l'utilisateur. Cette vision fournit donc une nouvelle approche pour mettre au services des utilisateurs des système robotiques, par l'intégration des robots dans un environnement d'intelligence ambiante [Saffiotti and Broxvall, 2005] d'une manière transparente. Les domaines de la robotique autonome et de l'intelligence ambiante convergent vers la vision d'environnements intelligents robotiques, ou la robotique omniprésente, par l'exécution de tâches via la coopération de nombreux dispositifs robotiques en réseau.

[6]Ecology of Physically Embedded Intelligent Systems

2.2 Fondement des architectures orientées services

Les architectures orientées services (SOA) ont été adoptées initialement pour la mise en œuvre des systèmes distribués selon le modèle serveur à serveur ou B2B dans le monde des SI des entreprises [Papazoglou, 2008]. Ces architectures ont été utilisées dans d'autres domaines tels que l'ubiquité grâce à leur avantage en terme d'interopérabilité, le faible couplage, etc. [Helal, 2002]. Les architectures SOA sont capables de gérer à un niveau de détail défini l'hétérogénéité dans les environnements AmI via une description standardisée des objets communicants hétérogènes. L'organisation des fonctionnalités des services augmente aussi la modularité (service autonomes) et offre de flexibilité aux concepteurs, en leur permettant de combiner les différents modules pour répondre aux besoins métiers. L'utilisation de la communication par échange de messages entres les fournisseurs et les consommateurs de services, permet de garantir un faible couplage entre les modules des architectures orientées services.

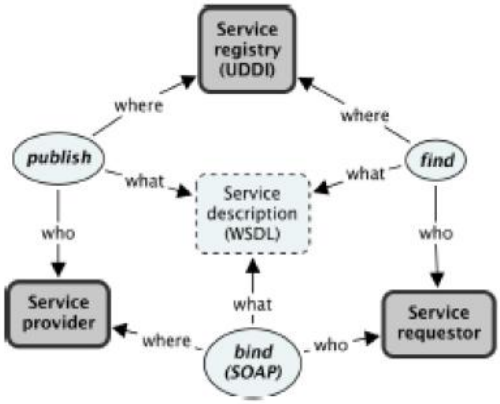


Figure 2.2: Architecture SOA [Törmä et al., 2008]

L'interaction entre un consommateur et un fournisseur de service est régie par un contrat négocié par avance. Ce contrat est exprimé dans un langage déclaratif indépendant de tout langage de programmation. Ce qui permet de s'abstraire de l'implémentation du service. Les demandeurs et les fournisseurs de services ne connaissent rien de leurs implémentations respectives. Les demandeurs ne référencent pas directement des instances particulières de services qui leur sont nécessaires mais leur font indirectement référence par l'intermédiaire de la description de leurs caractéristiques. A l'aide de cette dernière, les demandeurs sont en mesure de localiser/découvrir dynamiquement les instances de services disponibles dans leur environnement.

Plusieurs projets européen ont eu comme objectif de proposer des architectures orientées services

pour mettre en œuvre des applications ubiquitaires pour l'intelligence ambiante, comme par exemple Amigo [Gérodolle and Bottaro, 2006], MonAmI [MonAmI, 2011] ou SODA [Itea2, 2008]. Le projet européen SODA a développé un écosystème global basé sur une architecture orientée services pour gérer l'interopérabilité entre des équipements hétérogènes en utilisant les services web [Itea2, 2008]. L'encapsulation des fonctionnalités offertes dans un environnement ambiant dans des services, n'est surement pas la seule motivation derrière l'adoption de SOA. Les services peuvent être publiés dans des annuaires, pour faciliter leur découverte par des demandeurs.

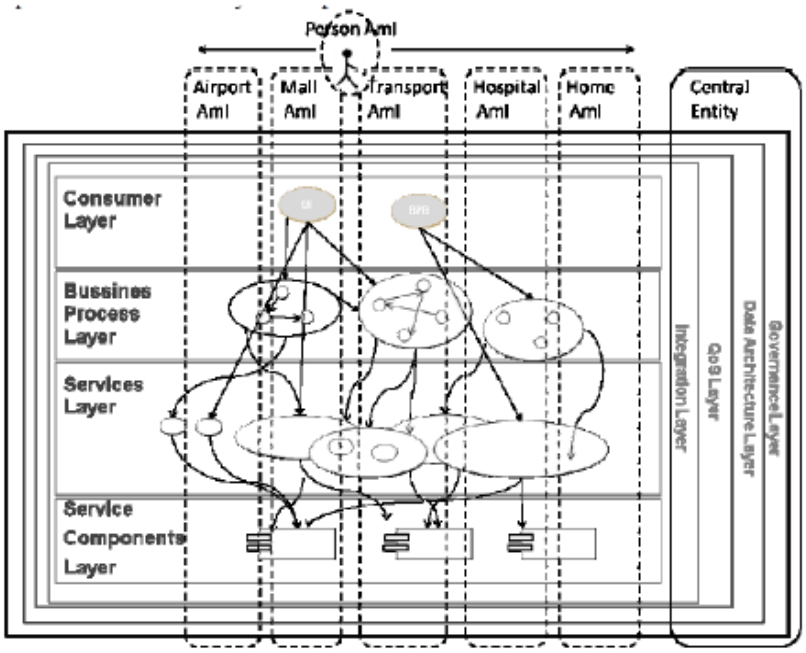


Figure 2.3: Environnement ambiant multi-domaine et SOA [Miguel and Roberto, 2009]

La Figure 2.3 montre une architecture de mise en œuvre des scénarii AmI via une approche à base de services dans un environnement multi-domaine. Chaque domaine constitue un environnement fréquenté au quotidien par des utilisateurs. Ces domaines ont pour but de réaliser des coopérations afin de servir au mieux les utilisateurs et de leur proposer des nouveaux services. Dans cette architecture, il est nécessaire que chaque domaine implémente les couches logicielles suivantes.

- La couche des composants exposés sous forme de services. Toutes les implémentations des fonctionnalités atomiques de chaque domaine résident dans cette couche. Elle correspond à l'ensemble des comportements de services offerts par un domaine.
- La couche service permet de mettre des services à la disposition des autres domaines. Chaque domaine fournit obligatoirement des interfaces d'invocation de ces services et les politiques connexes (i.e. politique d'accès et d'usage).

- La couche des processus métiers, ces processus métiers de chaque domaine ambiant seront présentés sur cette couche.
- La couche des consommateurs, l'interface utilisateur qui désire recevoir les services ainsi que les systèmes qui nécessitent d'avoir des informations sur des services afin de les invoquer durant l'exécution des processus métiers.
- La couche d'integration, sa fonction principale sera d'inter-connecter les différents domaines AmI.
- La couche responsable sur la gestion de la qualité de services, cette couche a la responsabilité de contrôler la qualité des services offerts. Cette couche permet la définition et le contrôle des exigences non fonctionnelles (e.g. SLA[7]).
- Enfin, la couche des données, cette couche est responsable sur la définition des structures de données spécifiques à chaque domaine, les protocoles de communication, etc. Le contenu de cette couche servira comme base pour la construction de l'intelligence métier dans les scénarii AmI inter-domaine.

2.2.1 Catégories de services

Dans les environnements ubiquitaires multi-domaine, les modalités d'accès et d'usage aux différents ressources et services ne sont pas connues a priori et sont contrôlées par des politiques. La mise en œuvre de ces dernières implique des processus de configuration (provisioning) permettant de donner les droits d'accès et d'usage nécessaires (i.e. l'authentification et l'autorisation) aux usagers. On distingue deux catégories de services; les services fonctionnels ou métiers et les services de configuration. *Les services fonctionnels ou métiers* sont responsables sur la partie applicative du système ambiant et ne se préoccupent pas des aspects non-fonctionnels dont les services de provisioning sont responsables. Dans ces travaux on s'intéresse à circonscrire les notions de provisionnement dans le cadre de l'intelligence ambiante, qui est une étape indispensable pour permettre aux différents composants de l'environnement de communiquer, de collaborer et de coopérer dans le but d'atteindre des objectifs communs. *Les services de configuration* sont responsables sur la gestion des configurations des paramètres de fonctionnement des services métiers ainsi que des paramètres de qualité de services (QoS). Des standards existent dans l'état de l'art et sont utilisés dans le milieu industriel pour développer ce type de services comme WS-Management [8], SPML [9], SAML [10].

[7]Service Level Agreement
[8]Web Service Management : est devenu le 28 juin 2013 un standard ISO/IEC sous la référence ISO/IEC 17963:2013
[9]Service Provisioning Markup Language
[10]Security Assertion Markup Language

2.2.2 Description de services

Un service possède une description sémantique abstraite et une ou plusieurs implémentations qu'on appelle services concrets.

Service abstrait dans la vision de l'environnement informatique ubiquitaire décrit par Weiser, [Weiser, 1991], l'environnement est peuplé par des dispositifs intelligents qui s'adaptent au contexte utilisateur, afin de répondre aux besoins des utilisateurs et s'occuper d'eux. Cependant, nous sommes encore loin de cette vision parce que les dispositifs actuels décrivent leurs fonctionnalités de façon très simple, et non pas d'une manière expressive, et il est donc difficile d'automatiser le comportement de l'environnement. Dans ce travail de thèse on utilise une approche de représentation de services fondée sur les pré-conditions (l'état nécessaire du monde avant d'exécuter le service) et les effets (le changement de l'état du monde après l'exécution du service). Cette approche a été largement adoptée pour la représentation de services dans les environnements ubiquitaires [Urbieta et al., 2008a] ainsi que par de nombreux travaux dans la planification et la composition des actions dans le domaine de la robotique [Russell and Norvig, 2009].

Service concret représente une implémentation de la description abstraite d'un service dans une plateforme logicielle cible (e.g. UPnP, DPWS). Cette implémentation permet d'encapsuler l'appel de fonctions de traitement de données, ou des fonctions d'accès aux capteurs et effecteurs de l'environnement physique (e.g. Lecteur UPnP) [Stavropoulos et al., 2011a]. Un service concret est par défaut localisable et accessible via un protocole de communication standard, notamment HTTP pour les services web.

2.2.3 Service web

Les services web constituent la technologie phare et la plus utilisée pour le développement d'architectures orientées services. Les services web sont conçus pour garantir l'interopérabilité entre des services implémentés avec des technologies hétérogènes. Ils possèdent des descriptions d'interface qui peuvent être publiées par des agents logiciels à travers Internet. L'invocation des services web ressemble aux appels de procédure distante (RPC) sur l'Internet mais en utilisant les protocoles et les langages de représentation des données web existants en particulier HTTP, URI et XML [Törmä et al., 2008]. Un service web, est une entité logicielle fournissant une ou plusieurs fonctionnalités accessibles via le web, dans le but de faciliter l'interopérabilité et la coordination de différents agents logiciels sur le réseau. Un service est constitué principalement des descriptions d'interfaces qui peuvent être publiées, localisées et invoquées à travers le web. La spécification du consortium W3C précise une différence entre la spécification abstraite d'un service et ses éventuelles concrétisations via des implémentations par différents agents.

La spécification du W3C des services web est divisée en trois couches. La couche des protocoles de communication, la couche de description de services et la couche de service de découverte. Des spécifications sont en cours d'élaboration pour chacune. Actuellement les spécifications stables dans chaque couche sont, le protocole *SOAP* qui permet la communication entre les services web [Walsh, 2002], le langage de description de services web *WSDL* [Chinnici et al., 2007] et l'annuaire *UDDI* qui représente un registre des descriptions de services web [Walsh, 2002].

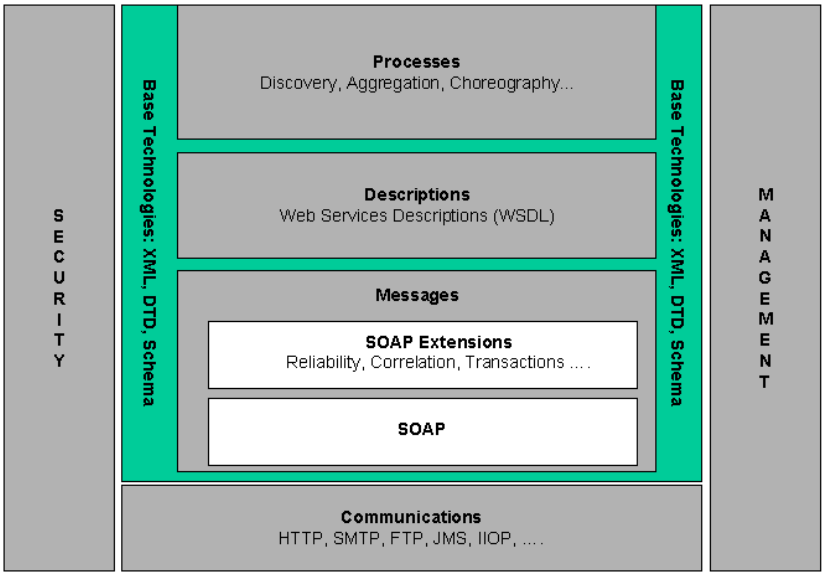


Figure 2.4: La pile d'architecture des services web [Booth et al., 2004]

Cette plateforme à couches basée sur le langage XML permet l'interaction de services via l'échange des messages SOAP.

La description des services web SOAP selon W3C se fait via le langage de description de service WSDL. Ce dernier permet de décrire la signature d'un service. Il donne, en plus de la description abstraite du service, une description concrète des points d'accès au service et du protocole préconisé pour son utilisation. La description abstraite décrit les capacités du service web ainsi que les messages SOAP de façon totalement indépendante de toute plateforme et de tout langage. Cela facilite la définition d'un service abstrait qui possède plusieurs implémentations par les différents domaines.

Pour simplifier la mise en œuvre des services web un style d'architecture a été proposé par Roy Fielding [Fielding, 2000]. Ainsi l'architecture d'un système est considérée comme étant un ensemble de ressources accessibles à des services web. Ces derniers offrent quatre opérations principales qui sont la lecture, la création, la modification et la suppression d'une ressource. Les services REST

sont implémentés et sont accessibles par le biais d'une simple requête HTTP contenant l'URL correspondant à cette opération.

2.2.4 Découverte de services

La découverte de services peut se faire soit en mode distribué ou centralisé [Bromberg, 2006]. Dans le premier cas, les fournisseurs de services participent à la découverte de services suivant un modèle de découverte de service passive ou active. La découverte passive consiste à écouter les annonces de services diffusées dans l'environnement par les fournisseurs de services, tandis que la découverte active correspond à la diffusion de requêtes décrivant les caractéristiques des services recherchés par les consommateurs. Toutefois, afin d'éviter d'engorger le réseau avec la diffusion des annonces et des requêtes on privilégie le mode de découverte centralisé par le biais d'annuaires qui centralisent l'enregistrement des annonces et le traitement des requêtes de recherche de services émanant des consommateurs. La Figure 2.2 présente le protocole d'interaction pour la découverte de services dans les architectures SOA. Dans ce protocole, la découverte est mise en oeuvre en plusieurs étapes appelées "*publish*", "*find*" et "*bind*". Dans la première étape, le fournisseur de services publie la description de ses services auprès du service de découverte. Le consommateur de services, dans une deuxième étape, interroge le service de découverte en lui soumettant la description (partielle) du ou des services requis et ce dernier lui renvoie le contrat du service et la référence d'une ou plusieurs instances de services correspondant. Enfin, dans la dernière étape, le consommateur de services initie les interactions avec le fournisseur de service suivant les termes du contrat du service. La découverte d'un service dans un environnement ubiquitaire peut être effectuée de plusieurs façons : soit de maintenir une liste d'adresses de service, ou de la configuration des applications. Seuls certains critères de sélection sont inscrits par configuration dans l'application. L'application recherche les entités réseaux répondant à ces critères durant l'exécution à l'aide d'un protocole de découverte [Bromberg, 2006].

2.2.5 Bilan

Dans les architectures SOA, les services web sont passifs jusqu'à ce qu'ils soient invoqués. Selon les différents standards proposés par le W3C, un service web possède seulement des connaissances le concernant et ne possède aucune connaissance concernant les consommateurs. De plus, il ne dispose d'aucune fonction cognitive lui permettant de s'adapter à la dynamique de l'environnement ni au contexte des consommateurs. Initialement les technologies de services web SOAP et REST n'offrent pas de langage de définition des interfaces suffisamment expressif pour décrire la sémantique de leur fonctionnalité. Ceci rend les tâches de composition et de découverte plus complexes, d'où l'intérêt des services web sémantiques, qui se basent sur les technologies du web sémantique et des ontologies.

Plusieurs langages ont été proposés et d'autres ont été étendus (e.g. SA-WSDL) pour enrichir la sémantique de la description des services dans les environnements ubiquitaires. Cette sémantique est apportée aux descriptions de services WSDL par intermédiaires des annotations ou des références vers des concepts définis dans des ontologies. Ces annotations sémantiques exprimées dans des langages formels peuvent aider à lever l'ambiguïté de la description des services web lors de la découverte automatique et la composition de services web. Dans la suite, on présente les services web sémantique qui représentent une sur-couche et non pas une technologie indépendante des services web.

2.3 Services Web Sémantiques

L'un des défis importants des systèmes à intelligence ambiante et de la robotique ubiquitaire, est de permettre à des agents de communiquer et de partager une compréhension commune des entités de l'environnement ainsi que de leurs interactions. Les ontologies pour le web sémantique constituent aujourd'hui une solution incontournable à la représentation, au raisonnement et au partage des connaissances dans le domaine de l'informatique ubiquitaire, notamment pour la mise en œuvre des modèles sémantiques pour la description de services, la description des connaissances contextuelles et des architectures d'applications. Le terme "ontologie" désigne l'étude des propriétés générales de ce qui existe. Empruntée à la philosophie, ce terme est employé dans le domaine informatique pour la première fois par John McCarthy dans les années 80. Ce dernier met en évidence le recouvrement qui existe entre les travaux sur les ontologies en philosophie et ceux liés à la définition des théories logiques en intelligence artificielle. La définition la plus adoptée du terme ontologie est celle de Gruber [Gruber, 1995]. Il définit une ontologie comme une spécification explicite d'une conceptualisation, qui correspond à une vue abstraite et simplifiée du monde que l'on veut représenter. Selon Staab and Studer [Staab and Studer, 2009], une ontologie est une spécification formelle compréhensible par la machine et explicite d'une conceptualisation partagée par un groupe d'individus.

L'expression web sémantique, due à Tim Berners-Lee [Berners-Lee et al., 2001] au sein du W3C, fait d'abord référence à la vision du web comme étant un vaste espace d'échange de ressources entre êtres humains et machines. Ce dernier permet une exploitation efficace de grands volumes d'informations par des machines. Il offre aussi des services d'inférence de haut niveau sur ces informations. Le web sémantique permet de remplacer les méta-données décrivant les ressources web avec des mots clés, par des ontologies offrant une description sémantique, formelle et intelligible de ces ressources. Le web sémantique est aussi une plateforme de partage de connaissance sémantique, de sens communs entre des systèmes informatiques hétérogènes.

Le web sémantique offre une pile de standards pour la représentation, le traitement et le raisonnement sur les ontologies. RDFS et OWL sont les deux langages phares permettant d'exprimer des ontologies web sémantique. RDF (Resource Description Framework) permet de représenter toute information sous forme d'assertions de type sujet-prédicat-objet (appelés triplets et notés $< s, p, o >$). C'est un sous-ensemble du calcul des prédicats. On peut aussi voir un tel ensemble de triplets comme un multi-graphe étiqueté et ainsi faire le lien avec le formalisme des graphes conceptuels [Chein and Mugnier, 1992]. La force de RDF est que les noms des entités (qu'ils soient sujets, prédicats ou objets) sont des URI (les identificateurs du web, que l'on peut voir comme une généralisation des URL [11] : http://www.w3.org/sw). Il est ainsi possible dans plusieurs documents

[11] Uniform Resource Locator

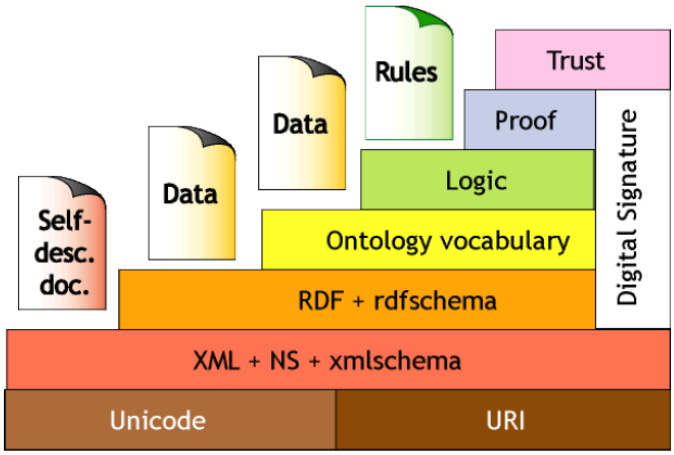


Figure 2.5: Architecture du Web Sémantique

RDF de faire référence à une même entité avec certitude. Un URI [12] dénote la même chose quel que soit son utilisateur.

L'approche fondatrice dans le développement du langage OWL est de combiner les deux principaux domaines de recherche de la représentation des connaissances, les logiques de description et les langages de règles, selon différentes notations, RDF, N3, Triple [Allemang and Hendler, 2008]. L'objectif d'usage des ontologies dans les applications de l'informatique ubiquitaire orientée services concerne la mise à disposition d'outils permettant de décrire et de partager des connaissances sémantiques relatives aux consommateurs, aux services ainsi qu'à leurs fournisseurs. Le développement des technologies du web sémantique permet d'enrichir celles des services web avec des fonctionnalités fournies par les technologies du web sémantique. Cette amélioration a donnée naissance à une classe de services dite *Services Web Semantique*. Cette classe permettra l'usage intelligent et l'automatisation de certaines tâches (e.g. la découverte, la sélection, la composition). Dans ce qui suit, on présente les frameworks de représentation de services web sémantiques. Plusieurs extensions du langage OWL ont été proposées récemment pour décrire les services web sémantiques, en l'occurrence, OWL-S, SA-WSDL, SA-REST.

Plusieurs travaux récents ont abordé l'utilisation des ontologies pour mettre en œuvre des plateformes de robotique ubiquitaire offrant des fonctions de planification de tâches et de composition de services robotiques dans des environnements à intelligence ambiante [Ha et al., 2005a, Yachir et al., 2009, Saffiotti et al., 2008]. Les avancés réalisées dans le domaine de la représentation

[12] Universal Resource Identifier

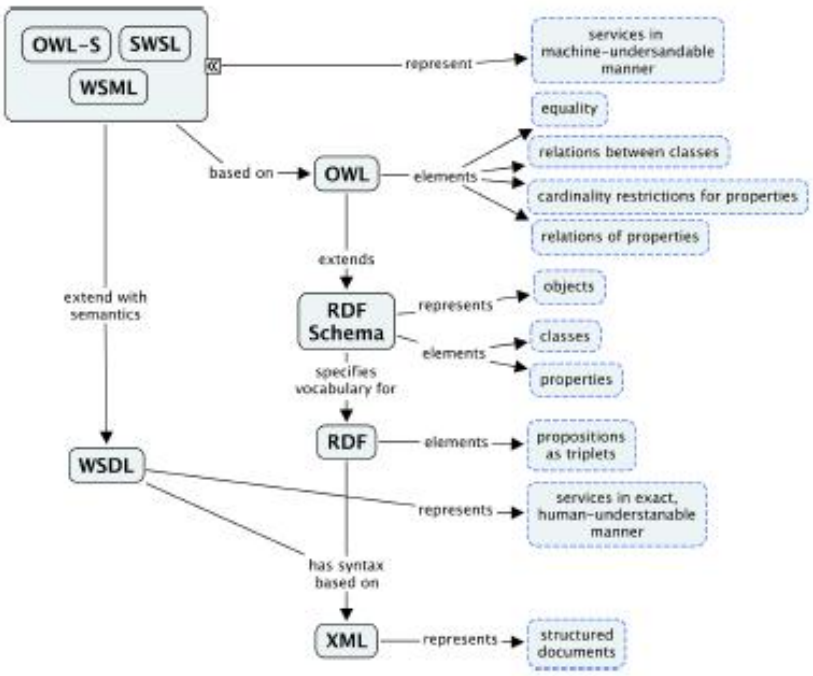


Figure 2.6: Langages de spécification des services web sémantiques [Törmä et al., 2008]

sémantique des connaissances grâce aux apports du web sémantique, ont encouragé la communauté robotique à s'intéresser aux approches sémantiques, notamment, l'utilisation des ontologies pour représenter les connaissances et les services des robots.

2.3.1 Approches basées sur des langages sémantiques : OWL-S

Le langage OWL-S [Martin et al., 2007] précédemment appelé DAML-S [Ankolekar et al., 2002] est une ontologie de haut niveau qui permet de décrire sémantiquement des services web qui sont facilement interprétables par des systèmes informatiques. Les descriptions de services produites par OWL-S peuvent être publiées dans des annuaires, et elles peuvent être découvertes à l'aide des techniques de matching sémantique d'ontologies. La description d'un service web dans OWL-S est basée sur trois ontologies, à savoir, *Service Profile*, *Service Model* et *Service Grounding*. L'ontologie *Serice Profil* définit la classe principale (Service). Cette dernière décrit ce que le service fait (les descriptions, les limites et les exigences du service). L'ontologie *Service Model* définit comment utiliser le service et l'ontologie *Service Grounding* décrit comment y accéder (détails sur la façon de communiquer avec le service, comme le format des messages, les protocoles, etc).

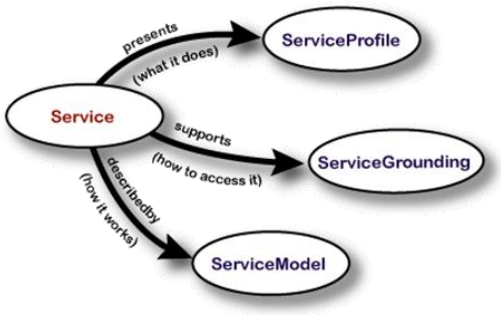


Figure 2.7: OWL-S [Burstein et al., 2004]

2.3.2 Approche de description à base d'annotations : SA-WSDL

L'approche SA-WSDL [Kopecký et al., 2007] définit un ensemble d'attributs d'extensions aux langages WSDL et XSD [13] qui permet de décrire une sémantique supplémentaire des composants WSDL. La spécification de cette extension définit comment l'annotation sémantique est effectuée en utilisant des références à des modèles sémantiques, par exemple, les ontologies. Les annotations sémantiques pour WSDL et les Schémas XML (SA-WSDL) ne spécifient pas un langage pour représenter les modèles sémantiques. Elles fournissent des mécanismes par lesquels les concepts des modèles sémantiques, généralement définies en dehors du document WSDL, peuvent être référencés

[13] XML Schema Definition Language

dans des composants de schémas XML et WSDL en utilisant les annotations.

2.4 Composition de services ubiquitaires multi-domaine

La composition de services vise à construire des services composites de plus haut niveau. Il s'agit d'un processus dans lequel l'expert humain part d'une spécification de besoins ou d'objectifs correspondant à des tâches complexes, souvent exprimées dans un langage de haut niveau ou dans un langage naturel afin d'en définir un protocole de communication permettant aux services ou agents logiciels d'échanger et de coordonner leur exécution pour atteindre les objectifs définis. Les approches proposées pour la composition de services peuvent être classifiées selon deux classes. Dans la première classe les approches sont classées en fonction du degré de participation de l'utilisateur dans la définition du schéma de composition, dans ce cas ces approches peuvent être manuelles, semi-automatiques ou automatiques (Section 2.4.1). Dans le second cas, le classement est effectué selon que des services et la spécification du workflow soit faite a priori ou non, dans ce cas une approche sera dite statique ou dynamique (Section 2.4.2).

2.4.1 Composition manuelle, semi-automatique et automatique

Pour la première classe d'approches [Charif, 2007], la composition de services peut être classée en trois catégories : manuelle, semi-automatique et automatique. Pour la composition manuelle, un expert s'occupe de la spécification des inter-connections entre les services et se charge de sélectionner les services les mieux adaptés. Dons ce cas, c'est l'expert qui va générer les workflows en utilisant un outil dédié [Majithia et al., 2004]. L'inconvénient des approches de composition manuelle réside, premièrement, dans la découverte et la sélection qui pose des problèmes de mise à l'échelle dans le cas d'un grand nombre de services. Deuxièmement, ces systèmes demandent un savoir faire à l'utilisateur. Les techniques de composition semi-automatique de services se présentent comme des outils d'aide aux utilisateurs dans le processus de découverte et de sélection basés sur la sémantique. Cependant, le rôle de l'utilisateur reste central dans la définition des workflows et dans le choix définitif des services. Malgré que ces systèmes résolvent quelques problèmes des frameworks de composition manuels, cependant ils ne permettent pas la mise à l'échelle dans le cas où le processus de filtrage fournit un grand nombre de services à l'utilisateur pour qu'il fasse son choix parmi eux. Les techniques de composition automatiques automatisent entièrement le processus de composition. Les services sont sélectionnés et composés à la volée en fonction de la requête et du contexte de l'utilisateur et en prenant en compte ses préférences. Cette composition peut se faire avant ou pendant l'exécution des services. Ce type de composition est favorable dans un environnement tel que le web et l'informatique ubiquitaire où les composants disponibles sont dynamiques et les attentes des utilisateurs sont variables et personnalisées.

2.4.2 Composition statique et dynamique

Pour la deuxième classe d'approches de composition [Charif, 2007], les différentes techniques de composition de services peuvent être classées en deux grandes catégories : les techniques de composition statique et les techniques de composition dynamique. Les techniques de composition statiques sont définies à l'aide de processus métier. Dans ce cas, le service composite est défini par un ensemble de services atomiques et par la façon dont ils communiquent entre eux. L'orchestration et la chorégraphie sont des compositions statiques de services web permettant de créer ce processus métier (workflow). Dans les méthodes de composition statique, les services à composer sont présélectionnés et le flot de contrôle préalablement spécifié. Ainsi, si l'un des services participant à une composition, n'est plus disponible, le schéma de composition n'est plus valide. Par ailleurs, dans ces approches de composition les besoins de l'utilisateur sont supposés être communs et peuvent être connus à l'avance. Cependant, l'utilisateur peut demander des services personnalisés et totalement imprévisibles par un expert. En conséquence, dans les environnements dynamiques tels que le web ou l'informatique ubiquitaire, où la prédiction des composants qui seront disponibles au moment même de la composition et les intervalles de temps pendant lesquels ces composants resteront disponibles est impossible. A cette caractéristique s'ajoute l'imperfection des composants et le fait qu'ils peuvent toujours tomber en panne alors que le processus de composition vient de réussir. Par ailleurs, les besoins des utilisateurs varient et ne peuvent pas tous être prévus à l'avance par un expert. La découverte et la composition des services doivent se faire à la volée, au lieu d'être préalablement fournies à travers un schéma de composition, doit s'effectuer de manière dynamique en fonction des services disponibles et des besoins énoncés par l'utilisateur. C'est dans ce contexte que les approches de composition dynamiques de services ont été proposées. Ces dernières prennent en compte les services disponibles, leurs fonctionnalités et le but à atteindre que ce soit avant ou pendant l'exécution des services. Elles offrent le potentiel de réaliser des compositions de services flexibles et adaptables en sélectionnant et en combinant les services de manière appropriée selon la requête et le contexte de l'utilisateur. On distingue deux axes pour la composition dynamique de services :

- Le premier se base sur le fait qu'un service composite est défini par un ensemble de services atomiques et par la façon dont ils communiquent entre eux. Dans ce cas l'idée de base est d'adapter les méthodes d'orchestration et de chorégraphie afin de les rendre dynamique.
- Dans le second courant, la composition est vue comme la génération automatique d'un plan d'exécution des services web (planification). Ces approches sont basées sur le fait qu'un service peut être spécifié par ses pré-conditions et ses effets. Les techniques de composition dynamique de services sont donc classées en deux grandes familles, à savoir, les techniques utilisant une approche basée sur les workflows et celles basées sur des techniques d'intelligence artificielle

et plus particulièrement la planification.

2.4.3 Principaux défis

La composition de services ubiquitaires implique des services appartenant en général à des domaines différents, on parle alors d'une composition de services multi-domaine. La mise en œuvre et l'exploitation de services composites pose plusieurs défis, à savoir, l'assurance de l'interopérabilité sémantique et comportementale des services utilisés dans la composition [Hilia et al., 2013a], la sécurisation des communications inter-services et la préservation de la vie privée quand les services appartiennent à des domaines hétérogènes et indépendants [Hilia et al., 2011]. La validation formelle des services composites est un défis de taille notamment quand il s'agit de mettre en œuvre des outils de preuve automatique [Hilia et al., 2012, Hilia et al., 2013b]. Finalement, la supervision dynamique de l'exécution des services composites tient compte des différents évènements et changements qui peuvent se produire dans l'environnement.

Afin d'étudier ces problématiques, on présente en détail chacune et on donne les solutions proposées.

Interopérabilité sémantique et comportementale Les services utilisés dans une composition peuvent être exprimés avec des termes ou des concepts différents, ce qui pose un problème d'hétérogénéité sémantique entre les services. Ces services doivent obligatoirement partager des caractéristiques communes. Ce partage est souvent modélisé par la définition ou la fusion des données dans une ontologie commune [Chiarugi et al., 2006]. La spécification sémantique d'une composition de services multi-domaine définit une ontologie commune, qu'on peut appeler *ontologie de coopération*, permettant de décrire formellement les objets échangés entre le services ainsi que les différentes opérations permettant de traiter ces objets au sein de chaque domaine [Tenorth et al., 2012, Kunze et al., 2011].

Vérification et validation formelle d'une composition Dans un environnement multi-domaine une coopération à base de services doit être vérifiée d'un point de vue composabilité et que les processus coopératifs sont corrects dès leur conception -design-time-. Afin de pouvoir vérifier cette propriété de correction [14], il faut utiliser des méthodes formelles pour la spécification de la sémantique et du comportement d'une composition pour pouvoir ensuite valider formellement sa consistance, sa validation sémantique et sa sureté de fonctionnement dans les domaines cibles. Bien que l'état de l'art regorge des travaux et projets de recherche visant à développer des techniques de composition de services, peu d'entre eux ont adressé la problématique de la modélisation logique et la validation sémantique des services composés en utilisant des outils formels. Dans certains domaines critiques

[14] Correctness

comme la santé, les systèmes de coopération intelligents (i.e. les environnements d'assistance, les systèmes de rééducations), l'assurance de la propriété de correction [15] de cette composition est largement justifiée. Cette vérification permet de voir que les services disponible dans un environnement ont la capacité de fournir un service assurant un objectif bien précis et de donner une preuve de la capacité de réalisation de cette tâche.

Re-configuration des compositions de services Il s'agit ici de définir des mécanismes d'adaptation permettant de reconfigurer le schéma d'une composition pour répondre aux changements qui peuvent se produire d'une part sur les besoins des utilisateurs et d'autre part des changements concernant l'état de l'environnement. Le défi est de définir des mécanismes intelligents garantissant la conformité des exécutions d'une part avec les objectifs des utilisateurs et d'autre part avec les politiques et régulation en vigueur dans chaque domaine. Ces règles se sont basées sur l'ontologie de coopération pour le suivi d'exécution et le comportement du système multi-domaine [Hansen et al., 2008].

Sécurité et préservation de la vie privée Les services d'un domaine donné communiquent avec des services d'autres domaines d'une manière sécurisée toute en préservant le caractère privé des objets échangés. La mise en œuvre de politiques pour gérer l'authentification et les autorisations d'accès qui tienne compte du caractère multi-domaine d'une composition est plus qu'important. Il s'agit ici de ne pas seulement spécifier les politiques de contrôles d'accès et d'usage de services selon des paramètres de qualité négociés aux préalables, mais il s'agit aussi d'associer à ces politiques des services configuration permettant les mettre en œuvre concrètement dans les infrastructures correspondantes comme par exemple créer un compte pour un utilisateur dans une base de données, configurer un lien réseau avec une quantité donnée de bande passante, ouvrir des ports dans certains par-feux, etc [Sheng et al., 2009]. Les services de configuration deviennent donc une partie intégrante de n'importe quel schéma de composition de services fonctionnels [Belqasmi et al., 2011]. Dans certains cas, la composition de services ne peut se faire quand certains domaines appliquent des politiques de sécurité trop restrictives pour éviter de divulguer les mécanismes d'implémentation des services de configuration. Le défi dans ce cas concerne d'une part comment transformer les politiques de contrôle interne à chaque domaine en politique multi-domaine afin d'assurer le bon fonctionnement d'une composition sans baisser le niveau de sécurité ni entraver les règles de vie privée en vigueur. D'autre part, comment mettre en place des mécanismes d'abstraction permettant de cacher tout détail inutile concernant les services internes notamment les services de configuration. Prenons l'exemple suivant, un service atomique qui permet de lister les dossiers médicaux. Ce service fait partie d'une composition locale qui elle même fait partie du schéma de composition mutli-domaine. Le service en question ne doit pas être exposé directement pour des consommateurs

[15] Correctness

appartenant à des domaines tiers.

Mise en correspondance entre la description abstraite d'une composition et des services concrets partiellement décrits Il s'agit ici d'un défi technique concernant la création de passerelles permettant d'établir des correspondances entre les vues d'abstractions utilisées dans les compositions avec les services concrets ainsi que les règles des politiques internes à chaque domaine. Ce défi peut s'accroître quand il s'agit d'établir les correspondances d'une manière automatique en générant et déployant des services web correspondants aux vues d'abstraction avec un minimum de connaissances sur le domaine local.

Par rapport à ces défis et verrous, notre étude portera sur les approches proposées dans l'état de l'art pour la composition de services dans les environnements ambiants en mettant l'accent sur celles qui proposent la validation formelle de la composition de services. Dans la suite de ce chapitre, on présente ces approches et discuter leurs avantages par rapport au développement des applications ubiquitaires multi-domaine.

2.5 Principaux techniques de composition de services

La plupart des approches de composition de services proposées dans l'état de l'art sont inspirées par les recherches menées dans le domaine de l'organisation et la gestion des flux et des processus métiers [16] (workflows, Business Process Management) ainsi que des techniques de planification de l'intelligence artificielle[17] tel que le démontre les différentes études de l'état l'art [Yue et al., 2004, Zeng et al., 2004, Dustdar and Schreiner, 2005, Antonio Bucchiarone, 2006, Peer, 2005, Rao and Su, 2005, Rao and Su, 2005, Urbieta et al., 2008b, Stavropoulos et al., 2011b, Feenstra et al., 2007]. Les techniques proposées sont appliquées dans la majorité soit pour composer des services web ou composer des services web sémantiques. La figure 2.8 présente une classification de ces techniques.

2.5.1 Composition basée sur les techniques de workflows

La composition de services sous forme de flux des tâches (Workflow) consiste à définir des processus qui peuvent être automatisés. Ces derniers impliquent l'exécution des activités, des services web et/ou des actions élémentaires d'une manière structurée, dont certaines peuvent être réalisées par les usagers humains. Les approches de composition de services à base de workflows requièrent la disponibilité d'un moteur qui permet d'orchestrer l'exécution des différentes tâches définies dans chaque workflow comme Bonita[18]. Plusieurs langages de haut niveau ont été proposés pour pouvoir définir des workflows avec un maximum d'expressivité. WS-BPEL (Web Services Business Process

[16] Workflow-based
[17] AI-Planning
[18] http://fr.bonitasoft.com/

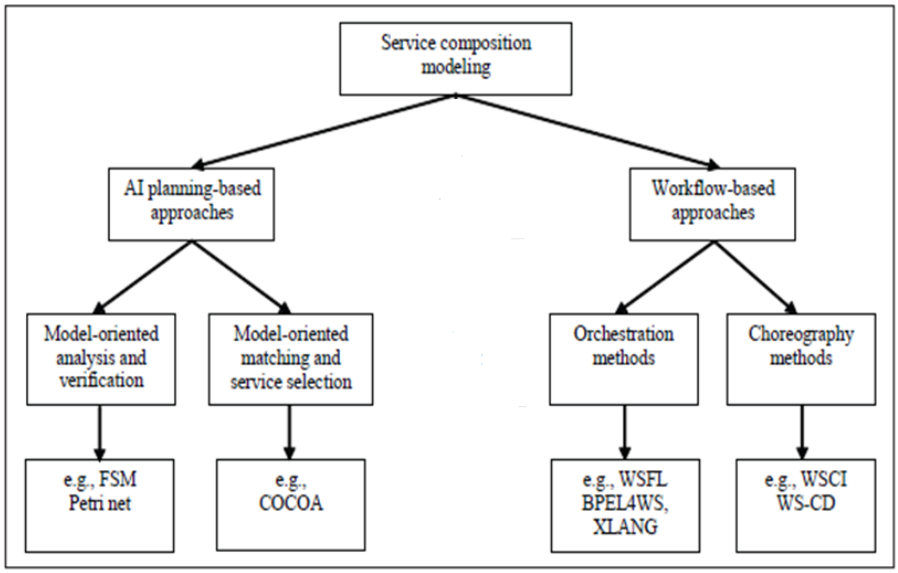


Figure 2.8: Classification des approches de composition

Execution Language) [OASIS, 2007] est le langage d'orchestration le plus utilisé dans les applications de l'informatique ubiquitaire. Le langage WS-BPEL permet de définir la manière d'orchestrer les services web et permet de décrire les interactions entre services en identifiant les messages échangés, les branchements logiques et les séquences d'invocation. Il existe néanmoins d'autres langages qui ont une capacité d'expressivité similaire à WS-BPEL, en l'occurrence Biztalk [19] et XPDL [Coalition, 2005]. Dans les méthodes de composition de services basées sur les workflows, on doit distinguer quelle est la méthode utilisée pour créer le schéma de composition (i.e. modèle de processus). La plupart des approches sont basées sur la création manuelle du schéma de composition par des experts. Ces derniers définissent des modèles abstraits de services élémentaires nécessaires pour la composition. On obtient dans ce cas un schéma abstrait de la composition. Ensuite, l'expert doit sélectionner les services concrets correspondant à chaque modèle abstrait. Tandis que dans le cas de la génération automatique des schémas de compositions, le système remplace l'expert en analysant les possibilités de combinaison des modèles abstraits pour satisfaire des contraintes ou des exigences formulées par les utilisateurs. En outre, le système prend en charge la sélection des services concrets qui correspond à chaque modèle abstrait. Les premières approches de composition basées sur les workflows telles que Rosetta, EFlow [Casati et al., 2000] sont basées sur des définitions de schémas statiques représentés sous la forme de graphes. Ces derniers peuvent être modifiés dynamiquement lors de l'exécution. Ces approches ne prennent pas en charge la possibilité de

[19] http://www.microsoft.com/france/serveur-cloud/biztalk/default.aspx

décrire sémantiquement les services composites pour pouvoir les réutiliser et n'offrent aucun outil pour valider ces services et manquent d'interopérabilité. Des travaux plus récents se sont focalisés sur la problématique d'interopérabilité sémantique des compositions de services, en l'occurrence, dans [Osman et al., 2009] les auteurs proposent d'utiliser des ontologies DAML-S afin de restreindre les choix des services web disponibles et faciliter la sélection de services selon des critères de haut niveau. Les services web sont regroupés en catégories métiers suivant les fonctionnalités attendues et les objectifs du workflow. Les travaux concernant la génération automatique des workflows sont mis en œuvre par le biais des techniques de planification proposées dans le domaine de l'IA que nous présentons dans la section suivante.

2.5.2 Composition basée sur les techniques de planification

Les techniques de planification figurent parmi les approches les plus appropriées à la fois pour la génération automatique des schémas de composition et leur exécution dynamique. L'hypothèse générale de ce genre de techniques est que chaque service peut être spécifié avec plusieurs paramètres concernant les pré-conditions d'invocation, les types de données d'entrées/sorties et les post-conditions (effets escomptés) après l'exécution du service. En partant de ces paramètres, le planificateur doit être capable de générer automatiquement le plan (i.e. schéma de composition). La mise en œuvre des planificateurs peut être basée sur différentes approches logiques comme par exemple le calcul des situations, le calcul des événements, les règles de production, des techniques de preuve de théorèmes logiques ou biens des techniques basées sur les chaines de Markov cachées ou les réseaux bayésiens. En effet, le formalisme *Situation Calculus* proposé par MCarthy figure parmi les premières approches de la planification de tâches [McCarthy, 1963]. Une situation est décrite par un ensemble de faits exprimés dans un langage de calcul d'une logique du premier ordre. Ce formalisme a été repris dans plusieurs projets pour construire des techniques de composition de service dynamique telles que [McIlraith and Son, 2002]. L'approche proposée permet de représenter la requête de l'utilisateur (procédure générique) et les contraintes des services en termes de prédicats du premier ordre dans le langage de calcul situationnel Golog [Narayanan and McIlraith, 2002]. Les services sont transformés en actions (primitives ou complexes) dans le langage de calcul situationnel. Les actions primitives sont soit des actions qui changent l'état d'un environnement ou bien des actions qui collectent des informations et changeant ainsi l'état de connaissance d'un agent. Tandis que les actions complexes sont des compositions d'actions individuelles. Ensuite, à l'aide de règles de déduction et de contraintes, l'agent génère les modèles des services composites qui sont instanciés à l'exécution à partir des préférences de l'utilisateur. L'inconvénient de ce framework est qu'il suppose l'existence d'une procédure générique. En cas de son absence, la composition ne peut pas être réalisée. En plus de cela, si l'agent ne trouve pas de service qui correspond, l'exécution échoue. Cette approche manque de flexibilité et demande la création des procédures génériques de composition pour arriver à com-

poser les services. Un des avantages de cette approche est l'usage des ontologies pour le stockage des procédures génériques. Dans d'autres travaux, la similarité entre les descriptions des ontologies de services web sémantique DAML-S, OWL-S, WSML, SA-* et les langages de planification tel que PDDL, STRIPS a été exploitée [Peer, 2004, Sheshagiri et al., 2003]. En effet, les concepts d'OWL-S sont très fortement inspirés de ceux de la planification. Ainsi une description OWL-S peut être facilement traduite en une spécification d'un plan, ce qui permet d'exploiter différents planificateurs pour produire un plan qui correspond à la composition des services.

D'autres approches utilisent la planification hiérarchique, telle que la méthode proposée par [Sirin et al., 2004]. Cette méthode utilise le planificateur SHOP2 [20] pour la composition automatique de services. SHOP2 permet de générer les étapes de chaque plan dans le même ordre que ces étapes vont être exécutées, donc l'état actuel du monde est connu à chaque étape du processus de planification. Cela réduit la complexité de raisonnement en éliminant beaucoup d'incertitudes à propos de l'état d'un environnement. SHOP2 est un planificateur HTN [21] indépendant de l'environnement. HTN est une méthode de planification de l'IA qui construit des plans en décomposant récursivement la tâche en sous tâches, jusqu'à ce que les tâches primitives sont trouvées et peuvent être directement exécutées. D'après les auteurs le concept de la décomposition de la tâche dans HTN est très similaire au concept de la décomposition de processus dans DAML-S [Sirin et al., 2004]. Ce qui rend le système de planification HTN un très bon candidat pour la composition automatique des services web. Les auteurs ont décrit une approche pour traduire les modèles de processus des services web en un ensemble de méthodes et opérateurs SHOP2, ainsi SHOP2 peut être utilisé avec les descriptions sémantiques avec l'ontologie DAML-S des services web pour composer automatiquement les services web. L'inconvénient majeur de cette approche est le fait de supposer que les informations concernant les services ne changent pas durant la planification.

Récemment, plusieurs techniques de planification basées sur des logiques formelles ont été proposées afin de vérifier que le processus de composition de service fonctionne correctement. Ces techniques, appelées méthodes formelles, sont généralement utilisés pour la spécification et la vérification des systèmes complexes. Parmi ces dernières, une variété d'approches basées sur des modèles d'état-transition (e.g. automates temporisées, les réseaux de Petri), les modèles de processus (e.g. π-calcul) et la preuve formelle de théorème sont utilisées pour décrire et raisonner sur les services web et la composition [Ter Beek et al., 2006]. Cependant, les méthodes formelles permettent de simuler et de vérifier le comportement de la composition de services web au moment de la conception. Ainsi, la vérification permet la détection et la correction des erreurs le plus tôt possible, mais ils ne s'attaquent pas à la deuxième exigence liée à la composition dynamique, car ils ne traitent pas de la localisation des services, la reconnaissance ou la sélection de ceux qui correspondent au service

[20] Simple Hierarchical Ordered Planner 2

[21] Hierarchical Task Network

demandés. Par ailleurs, des approches utilisant des ontologies et augmentées avec des techniques formelles ont été développées spécifiquement pour décrire le flux de composition de service. Ces approches permettent la description de l'interaction de services web, qui est une condition importante pour atteindre une composition dynamique de services.

2.5.3 Composition basée sur la preuve de théorèmes

Dans cette section, on présente les approches basées sur la preuve de théorème pour la composition de services. Ces techniques appliquent des règles de déduction sur une spécification d'un objectif sous forme de théorème mathématique dans le but de le démontrer. En général, une technique de preuve de théorème consiste en un ensemble d'étapes d'inférence qui peuvent être utilisées pour réduire un objectif de preuve à une liste de simples sous-objectifs. Ces derniers peuvent être prouvés automatiquement par les preuves primitives proposées par un assistant à la preuve (e.g. Coq, HOL4, Isabelle/HOL). Par ailleurs, ces techniques ont été peu utilisées dans le domaine de la composition de services [Rao and Su, 2005]. La majorité des approches proposées se sont basées sur la représentation sémantique (i.e. DAML-S ou OWL-S) qui ne donne aucun moyen de vérifier la correction. En conséquence, d'autres approches ont utilisé la transformation vers des représentations formelles différentes comme π-calcul afin de permettre la démonstration automatique en utilisant la preuve par des règles d'inférence et l'applicabilité des tactiques de raisonnement telles que la logique linéaire [Rao et al., 2006], la logique classique LL [Papapanagiotou and Fleuriot, 2011], [Traverso and Pistore, 2004], et ensuite interpréter ces preuves pour générer les schémas des compositions.

Ces techniques sont considérés comme faisant partie des techniques de planification. Il y a eu plusieurs tentatives d'utiliser les démonstrateurs (i.e. prouveurs) automatiques dans ce contexte. Le travail de Waldinger est basé sur la déduction automatique et la synthèse de programmes [Waldinger, 2001]. Il a utilisé le prouveur de théorème SNARK pour fournir des preuves aux problèmes de composition des services décrits dans une logique classique du premier ordre. Lämmermann a travaillé sur la synthèse structurée de programme (SSP)[22], une approche déductive qui utilise la logique propositionnelle intuitionniste [Lämmermann, 2002].

Lämmermann a appliqué la "SSP" pour la composition de services automatisée. Une SSP est une approche déductive permettant la synthèse de programmes à partir d'une spécification. La spécification de services inclut seulement les propriétés structurelles, c'est à dire les informations d'entrée/sortie. la SSP utilise aussi des variables propositionnelles comme identifiants des paramètres d'entrée/sortie et utilise la logique propositionnelle intuitionniste pour résoudre le problème de la composition. La composition est basée sur la propriété "proofs-as-programs" de

[22]Structural Synthesis of Program

la logique intuitionniste [Wadler, 2000]. Il assimile le programme de composition de service au problème de la recherche de preuve.

Récemment, Rao, Kungas et Matskin introduisent une méthode pour la composition automatique des services web en utilisant des preuves de théorème de la logique linéaire (LL theorem proving) [Rao and Küngas, 2004]. La méthode utilise le langage du web sémantique DAML-S pour la représentation externe des services web, tandis qu'à l'intérieur, un service est représenté par des axiomes extra logiques et des preuves exprimés dans une logique linéaire. La logique linéaire permet de définir formellement les attributs de services web (y compris les valeurs qualitatives et quantitatives des attributs non-fonctionnels). Un processus de calcul est utilisé pour représenter formellement le service composite. Ce processus de calcul est attaché aux règles d'inférences de la logique linéaire. Ainsi le modèle du processus du service composite peut être généré directement à partir de la preuve. L'idée de cette méthode est la suivante, étant donné un ensemble des services web existants, la méthode essaie de trouver une composition des services atomiques existants qui satisfait les besoins de l'utilisateur. En effet, le processus de composition est comme suit. Premièrement, la description sémantique des services en DAML-S traduite en axiomes de la logique linéaire ainsi que la requête de l'utilisateur est représentée sous forme de séquences de la logique linéaire à prouver. Deuxièmement, l'adaptateur demande au raisonneur sémantique d'analyser les relations de sous typeage entre les classes et les propriétés de l'ontologie de domaine qui décrit les services et envoie les relations sous forme d'axiomes de la logique linéaire au prouveur de théorème. Troisièmement, le prouveur de théorème vérifie si la requête peut être satisfaite en combinant les services atomiques existants. Si la séquence correspondante au service composite requis a été prouvée et la preuve a été générée, le modèle de processus du service composite est construit automatiquement à partir de la preuve (si la preuve existe). En outre, la logique linéaire a des relations étroites avec π-calcul, qui est le fondement formel de nombreux langages de composition de services web. L'idée de ce travail a motivé le travail présenté récemment dans [Papapanagiotou and Fleuriot, 2011]. Cette approche est basée sur le paradigme "*proofs-as-processus*" introduit à l'origine par Abramsky, Bellin et Scott [Papapanagiotou and Fleuriot, 2011]. En comparaison avec le travail de Rao, Papapanagiotou a essayé de rester fidèle à la théorie originale de Bellin et Scott en utilisant la logique linéaire classique (CLL) en conjonction avec la syntaxe standard de π-calcul polyadique.

La vue d'une preuve de la logique linéaire comme un processus π-calcul a d'abord été reprise officiellement par Abramsky [Abramsky, 1994], et développée par Bellin et Scott [Bellin and Scott, 1994]. Les auteurs attachent le π-calcul aux règles d'inférence de la logique linéaire dans le style de la théorie des types, donc le modèle de processus pour un service composite présenté par π-calcul peut être généré directement à partir de la preuve. Cependant, Papapanagiotou a soigneusement analysé le travail de Rao. Il a détecté un certain nombre d'incohérences. Par exemple, le calcul de processus utilisé est une extension du π-calcul. Cependant, aucune garantie n'a été donnée que les deux cal-

culs sont équivalents ou que les preuves de Bellin et Scott sont toujours valables pour le calcul de processus étendu. En outre, Rao utilise la logique linéaire intuitionniste dans le calcul des séquents à deux côtés "two-sides", ce qui n'est pas non plus garantie équivalente à l'approche de CLL à un seul côté "one-side" de Bellin et Scott. En outre, un certain nombre de soi-disant "règles de congruence structurelle" qui contiennent les termes CLL et les termes de la preuve de calcul de processus ont été introduites. Ces règles n'ont pas été formellement dérivées et leur syntaxe peuvent facilement conduire à une interprétation incorrecte. Enfin, même si le système prend en charge théoriquement les propriétés non fonctionnelles et exponentielles, la preuve de l'exemple de "Ski" présenté dans l'article [Rao et al., 2006] les ignore. Les propriétés non fonctionnelles sont indispensables pour une composition basée sur la qualité de services, alors que l'addition des exponentielles ferait la logique indécidable. Après cet intérêt apporté à la méthode proposée par Rao, d'autres représentations se sont intéressées à d'autres types d'interprétation sémantique en utilisant les logiques constructives ou intuitionnistes afin d'améliorer les techniques de modélisation du système, et pouvoir également exploiter ses capacités de calcul.

Récemment, la recherche sur d'autres interprétations de la représentation formelle classique des logiques de description a été motivée ainsi que la nécessité d'étudier ce domaine, ce qui peut améliorer les techniques de modélisation du système, et on peut également exploiter ses capacités de calcul [de Paiva, 2006, Mendler and Scheele, 2010]. Rappelons que le principal problème avec toutes ces approches composition et celles proposées dans le cadre industriel est la vérification de la correction [Ter Beek et al., 2006]. Par ailleurs, une logique de description constructive ou intuitionniste dite $\mathcal{BCDL}$ a été proposée [Ferrari et al., 2010]. Bozzato a appliqué cette logique constructive dans la composition de services web sémantique [Bozzato and Ferrari, 2010a]. Cette approche traite de la composition des services web sémantique, basée sur un sous-système de la logique de base de description constructive, appelé $\mathcal{BCDL}_0$. Bozzato propose un calcul pour la composition de service qui assure que la spécification du service composite (le profil de service) découle directement de preuves de la composition. La correction des compositions peut être vérifiée directement en vérifiant les conditions d'applicabilité des règles utilisées dans la composition, d'ailleurs, ces règles représentent directement les structures de contrôle de flux dans le domaine des workflows.

2.6 Discussion et objectif de la thèse

Durant l'étude de ces approches de composition basées sur la description sémantique de services. Ces approches passent par une transformation des descriptions sémantique vers un formalisme capable de réaliser des preuves formelles de certaines propriétés, en particulier, la correction. Parmi ces approches on se focalise principalement sur celles basées sur la preuve de théorème. Dans ces approches, les spécifications de services et les règles de composition sont représentées dans un formalisme basé sur la logique. Ces règles permettent de prouver la fiabilité et la correction des compositions [Bozzato and Ferrari, 2010b, Hilia et al., 2012]. L'approche proposée par Bozzato est intéressante. Cependant, elle présente des limitations d'utilisation et de preuve sur un prouveur de théorème ainsi que les règles de composition ne disposent pas des deux opérateurs la synchronisation des tâches multiples et l'opérateur de contrôle de fusion simple. Ces travaux de thèse ont été initiés par se poser les questions sur l'usage et la spécification des compositions de services dans un environnement multi-domaine. La logique proposée n'a pas été prouvée qu'elle soit fiable "Sound" ainsi que le langage de calcul. D'où l'idée de construire un canevas logiciel complet qui permet d'étendre l'expressivité des concepts spécifiques à un domaine et de construire des services d'approvisionnement et des coopérations formellement correctes dans un contexte ambiant multi-domaine. Cette thèse propose un canevas intégré pour la description des services web sémantiques, un modèle de composition et un module de preuve de correction. La partie contrôle d'accès et usage, le provisioning inter-domaine sont aussi pris en compte. Cette proposition peut être utilisée dans un contexte générale multi-domaine. Mais dans cette thèse on s'intéresse au domaine d'application ubiquitaire multi-domaine critique dont l'utilisateur est le centre d'intérêt. Dans le chapitre suivant, on présente la première contribution de cette thèse représentée par le canevas sémantique pour la composition de services multi-domaine.

L'objectif de la thèse est la proposition d'un canevas logiciel permettant la spécification des compositions de services ubiquitaires fiables et sémantiquement corrects pour fonctionner correctement dans des environnements multi-domaine. Le canevas répond aux défis de composition énumérés précédemment, à savoir, la représentation sémantique partagée par les différents domaines participants à la coopération, la préservation du savoir-faire par chaque domaine en exposant des vues sur les services internes, l'utilisation d'un langage formel pour la spécification des vues et la démonstration sa fiabilité [23] ainsi que la vérification formelle de la propriété de correction des compositions multi-domaine. La mise en œuvre d'une étude de cas dans le cadre de la santé pour démontrer la faisabilité du canevas proposé. La coopération via une approche hybride a été proposée. Cette dernière permet la réutilisation des services existants ainsi que la facilité de la spécification de nouveaux services. Parmi les avantages de cette approche, on cite la préservation

[23] Soundness

de savoir-faire par l'intermédiaire d'une approche à base de vue sur les structures des processus internes. La spécification formelle des vues d'abstraction sur les services offerts. La spécification du comportement des services. Enfin, la mise en correspondance avec les infrastructures cibles des domaines participants.

3

Vers un canevas sémantique pour la Composition de services multi-domaine

Sommaire du chapitre

Introduction

Un canevas sémantique est proposé pour répondre aux besoins de coopération entre plusieurs domaines ambiants. Ces domaines construisent ce qu'on appelle un *environnement multi-domaine*. Dans cet environnement les domaines sont de nature autonome, hétérogène et distribuée. L'intégration de ces domaines via des coopérations à base de services nécessite des efforts supplémentaires et une assurance de certaines propriétés d'interopérabilité sémantique et comportementale, ainsi que la vérification de la correction des processus de coopération. Plusieurs méthodes de composition de services dans les environnements d'intelligence ambiante ont été proposées dans la littérature. Cependant, peu d'entre elles ont été proposées pour l'intégration des systèmes d'intelligence ambiante multi-domaine. L'objectif de ce travail de thèse est de proposer un canevas intégré pour la description sémantique de la composition de services et la preuve de correction des processus d'intégration entre des domaines d'intelligence ambiante.

A l'origine, les domaines ne sont pas conçus dans l'objectif de collaborer, de coopérer, ou de communiquer les uns avec les autres. Cependant, une coopération entre ces différents domaines peut être construite à partir des services offerts par chacun des domaines afin de créer de nouveaux services plus complexes et de plus haut niveau. Par conséquent, le canevas proposé, en plus de répondre aux problématiques des systèmes distribués, doit assurer certaines propriétés relatives aux environnements ambiants multi-domaine, à savoir, représenter de manière sémantique les environnements, spécifier la sémantique comportementale associée à l'ensemble des événements produits et aux actions menées au sein des domaines coopérants, garantir la confidentialité, assurer la flexibilité, la réutilisation ainsi que la modélisation/spécification des accords et des engagements inter-domaines. Ces exigences justifient amplement le besoin de disposer des techniques et des approches permettant l'intégration de plusieurs domaines ambiants.

La problématique d'intégration des systèmes d'information n'est pas nouvelle. Elle a été traitée dans plusieurs domaines de recherche, par exemple, l'intégration et la médiation dans les bases de données, les systèmes de gestion de workflows, les entreprises virtuelles, etc. Cependant, aucune de ces approches n'a été proposée pour les environnements multi-domaine ambiants. Par conséquent, on a besoin d'approches permettant l'intégration de ce type de domaines (d'applications). Les approches proposées dans la littérature ont été classées dans trois grandes catégories, à savoir, les approches ascendantes, les approches descendantes et les approches à base de contrat. L'avantage des approches ascendantes réside dans l'utilisation de vues fonctionnelles sur les services et les processus internes d'un domaine (d'une organisation). Ces approches construisent une abstraction préservant la confidentialité, le savoir-faire des organisations et la conservation des processus existants. En effet, ces vues sont obtenues par une projection restreinte du processus interne par rapport aux fonctionnalités offertes et les participations dans le workflow inter-organisationnel. Par

opposition aux approches ascendantes, les approches descendantes considèrent que les processus existants ne sont pas toujours capables de répondre à des objectifs multi-domaine. Ces approches modélisent le workflow inter-organisationnel global, pour ensuite, générer les processus locaux avec les politiques métiers de contrôle. Dans les approches à base de contrat les participants expriment de manière informelle (e.g XML) des exigences, des engagements et des règles permettant le contrôle des interactions entre workflows, dans un contrat qui explicite les liaisons et les échanges entre les organisations.

Dans ce chapitre, les concepts clés utilisés tout au long de ce manuscrit sont définis, avant de donner une liste non exhaustive des exigences auxquelles un canevas de coopération dans un environnement multi-domaine doit répondre. On se focalisera particulièrement sur les exigences d'interopérabilité sémantique et comportementale, la préservation de l'étanchéité et de la confidentialité relatives au savoir-faire d'un domaine, la flexibilité, la réutilisation et la preuve de la propriété de correction par rapport à la spécification de la coopération multi-domaine. Une approche hybride est proposée par la suite, combinant les avantages des trois catégories d'approches discutées ci-dessus. Premièrement, cette proposition consiste en un modèle sémantique utilisé dans la spécification des processus de coopération. Cette spécification se base sur un système formel et une ontologie de coopération de haut niveau. Deuxièmement, elle permet de vérifier formellement les processus coopératifs ainsi spécifiés. Finalement, ce canevas présente l'avantage de modéliser et de séparer, au niveau du modèle qu'il propose, les deux types d'actions qu'on considère indispensables dans un environnement multi-domaine, à savoir, les *actions métiers* ou fonctionnelles et les actions non-fonctionnelles, ou les actions de configuration dites aussi *actions de provisioning*.

3.1 Définitions et préliminaires

Cette section définit les différents concepts dont on se sert dans la conception et la modélisation des différentes composantes de notre canevas.

Un domaine est défini comme étant un sous-système soumis à une politique de contrôle. Un tel sous-système agit sur des ressources via des services, ou des processus afin de récupérer des données, ou de configurer des paramètres nécessaires à leur fonctionnement. Par exemple, la Figure 3.1 présente une maison intelligente avec un capteur de température, considéré comme une ressource fournissant le service de mesure et de notification de la température. Pour le bien être des personnes âgées, un service de configuration du paramètre seuil est configuré à une valeur de 25 °C par le domaine hôpital. Le domaine représenté par l'hôpital peut donc appliquer cette règle de contrôle inter-domaines.

Un système multi-domaine peut être considéré comme un ensemble de systèmes d'un seul do-

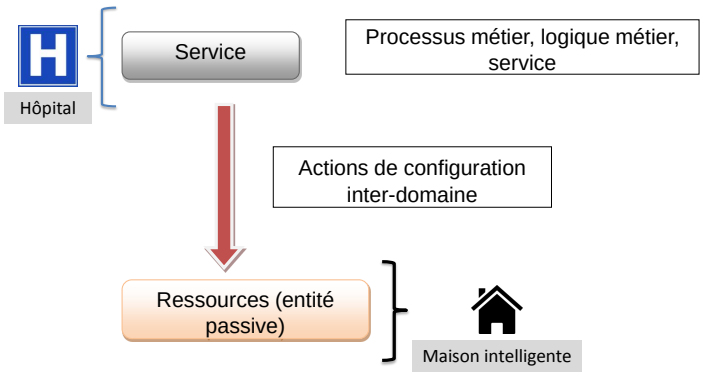


Figure 3.1: Service, Processus, Action de configuration

maine coopérant qui sont éventuellement autonomes et hétérogènes. L'hétérogénéité dans un environnement multi-domaine peut exister sous différentes formes, telles que les infrastructures logicielles, les systèmes d'exploitation, les bases de données, etc. Notons que chacun de ces domaines possède sa propre politique de gouvernance.

Un service ubiquitaire est une encapsulation d'une activité, ou d'une ressource afin de l'exposer pour utilisation ou pour livraison à un tiers. Dans un système ubiquitaire qui est souvent caractérisé par son hétérogénéité, la plupart des approches de description de services ne gèrent pas ces aspects de configuration de ressources et considèrent que toutes les ressources sont prêtes à l'usage ou opérationnelles. Dans le contexte multi-domaine ces aspects jouent un rôle très important afin de permettre la coordination et la coopération entre les sous-systèmes/domaines participants. Cela motive notre distinction entre deux classes de services, à savoir, les services métiers ou fonctionnels, et les services de configuration ou non-fonctionnels. Ces services sont décrits comme suit :

- Un *service fonctionnel* est un service ayant une interface purement fonctionnelle, ou métier au sens où il ne se préoccupe pas des aspects non-fonctionnels. Ce service correspond à l'implémentation des actions relatives à la logique de l'application, par exemple, le service de déplacement d'un robot.
- Un *service de configuration* est un service contenant la description des paramètres qualitatives, comme la confiance, et quantitatives comme le débit, le nombre d'image par second, l'adressage IP ainsi que les opérations de configuration associées. Ces opérations permettent de modifier ces paramètres, et peuvent contrôler le comportement des services

à travers une supervision de ces derniers.

Un processus est un ensemble de services combinés dans une composition qui opère sur une ou plusieurs ressources. Cette combinaison peut impliquer des services de différents domaines, pour satisfaire des objectifs définis au niveau global inter-domaines.

L'interopérabilité est définie comme étant la capacité de deux ou plusieurs systèmes ou composants d'échanger et d'exploiter des informations [Ali, 2010, Geraci, 1991]. Pour parvenir à une véritable interopérabilité, les systèmes doivent être en mesure non seulement d'échanger des informations en utilisant des architectures ou des méthodes de communication communes, mais aussi de pouvoir les interpréter et les utiliser correctement en utilisant les types de données partagées. La majorité des travaux utilisent des terminologies partagées (i.e. les ontologies), ou des schémas de document XML (i.e. XSD).

Une composition de services multi-domaine est l'intégration de services afin d'approvisionner de nouveaux services plus riches en termes de fonctionnalités. Dans les environnements d'intelligence ambiante le but des compositions de services est de fournir des services de plus haut niveau garantissant le bien être des utilisateurs et leur sécurité. Dans ce contexte, où différent domaines décident de construire des services de plus haut niveau, on parle de *composition multi-domaine*. Dans une composition de services, à la différence de la plupart des approches proposées dans la littérature, les services de configuration doivent être intégrés dans le processus de composition, et ne pas se focaliser seulement sur la logique métier. Dans ce contexte, multi-domaine, les services de configuration fournissent la capacité de modifier les paramètres d'un service. Par conséquence, la considération de ce type de service permettra un usage inter-domaines, et peut donc mener à des coopérations et des usages en dehors des frontières de contrôle des domaines.

Dans la Figure 3.2, un schéma illustratif d'une composition de services multi-domaine est donné. Cette composition combinant les deux types de services, à savoir, les services de configuration et les services fonctionnels. Le rôle des premiers est de modifier les paramètres d'un service pour le rendre accessible et prêt à l'usage. Cette configuration n'est pas toujours évidente et nécessite parfois d'avoir les droits dans les domaines cibles. Ces services de configuration peuvent aussi solliciter des processus de configuration internes au domaine cible. Les services fonctionnels auront une infrastructure opérationnelle en termes d'accès et d'usage des services concrets impliqués dans la coopération. Ces services construisent la logique métier des objectifs attendus par la coopération multi-domaine.

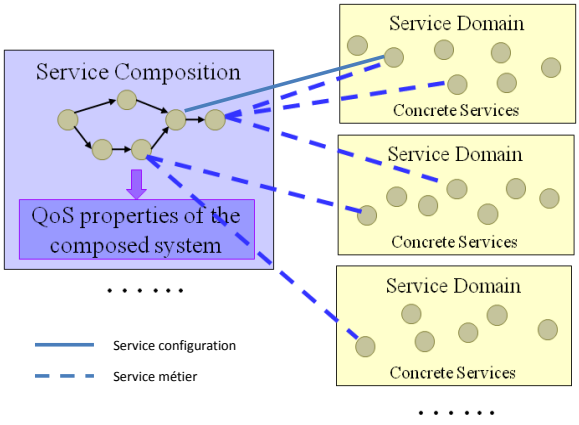


Figure 3.2: Composition de service multi-domaine

3.2 Proposition d'un canevas sémantique pour la coopération multi-domaine

Dans le contexte d'une coopération dans un environnement multi-domaine, chacun des participants offre des services et ou des ressources pour un accès et un usage inter-domaine. Par conséquent, la mise à disposition de certains de ces services à des tiers peut être accompagnée de contraintes supplémentaires d'accès et d'usage. Par ailleurs, les services mis à disposition par un domaine sont généralement soumis à des règles de configuration préalables à chaque utilisation. Dans les systèmes actuels, ces domaines représentent différemment leurs connaissances, et donnent souvent des sémantiques différentes aux mêmes concepts, en particulier aux concepts reliés aux règles. Cela engendre ce qu'on appelle une *hétérogénéité sémantique*. Les services d'un domaine sont souvent décrits de manière aléatoire et selon le domaine de compétence de leurs concepteurs et de leurs développeurs. Ces descriptions hétérogènes peuvent avoir des comportements similaires ou identiques, ce qui entraîne des problèmes d'interopérabilité dits *hétérogénéité comportementale*. Cette dernière n'exclut pas les cas ou des descriptions de services similaires exécutent des actions à effets opposés. Prenons l'exemple des deux services suivants : " Programmer une activité de rééducation", par un hôpital, et " Définir une entrée dans le calendrier quotidien d'un patient ", pour le robot compagnon à domicile. Ces deux services ont le même effet représenté par une attribution d'un créneau libre au patient. Donc on conclut qu'il est nécessaire d'attribuer une sémantique bien définie et bien déterminée (formelle) aux comportements de ces services. Comme on l'a mentionné, l'objectif est de faire collaborer plusieurs domaines avec des bases de connaissances et des services avec des comportements hétérogènes. Par ailleurs, chaque domaine exige un certain niveau d'étanchéité et de confidentialité de ces propres processus internes et de son savoir-faire.

Dans la section suivante, on présente les exigences auxquelles un cadre logiciel rigoureux dans le contexte de l'intelligence ambiante doit répondre.

3.2.1 Propriétés du canevas proposé

On part des caractéristiques d'environnement d'intelligence ambiante coopératif multi-domaine et les besoins de la coopération afin de déterminer les exigences à satisfaire dans un cadre général, ainsi que les exigences spécifiques aux domaines d'applications. Récemment, dans [Preuveneers and Novais, 2012] les auteurs présentent une étude sur les principes de développement d'applications dans le contexte de l'intelligence ambiante. Les exigences comme la description sémantique de l'environnement, les interactions dans un système ambiant, la réutilisation ainsi que la vérification formelle ont été soigneusement motivées. Dans [Liu et al., 2009] les auteurs présentent les différents défis relatifs au développement des processus métier collaboratifs. Ils présentent aussi l'intérêt de définir des contrats à base de sémantique formelle, compréhensible et interprétée par les machines. Les auteurs recommandent l'intégration de l'architecture à base de services pour assurer un faible couplage et une facilité d'évolution des plate-formes collaboratifs. L'ensemble des exigences couvertes par les travaux existant sera présenté et discuté en conséquence.

3.2.1.1 L'interopérabilité

L'interopérabilité de processus est un sujet largement étudié depuis le milieu des années 1990. L'avènement des systèmes de gestion de workflow et les services web pour l'intégration métier a donné naissance à de nouveaux concepts, par exemple, les interfaces comportementales, les processus abstraits et les organisations virtuelles. Concernant les environnements d'intelligence ambiante et les processus de développement des applications intelligentes, l'interopérabilité est essentielle pour réussir la communication d'un large nombre d'équipements disponibles dans de tels environnements et ayant des rôles différents (actionneurs, capteurs), afin de pouvoir interpréter correctement les informations fournies et les actions offertes. On distingue deux catégories d'interopérabilité; l'interopérabilité sémantique et l'interopérabilité comportementale [Ali, 2010].

L'interopérabilité sémantique Les domaines se basent sur des terminologies et des descriptions différentes pour représenter l'environnement, le modèle de données manipulé, et les services au sein d'un domaine. Cette représentation donne une sémantique implicite ou explicite à travers la terminologie utilisée. Dans le cadre multi-domaine, tous les domaines doivent se mettre d'accord sur une taxonomie commune de concepts. Les interprétations de ces concepts dépendent souvent de leurs domaines sources respectifs. Par conséquent, un canevas sémantique doit éliminer cette hétérogénéité sémantique. Les techniques à base d'ontologie ainsi que des techniques de mapping et d'alignement des représentations locales peuvent être utilisées afin de restreindre les interprétations et par conséquent permettre la coopération.

L'interopérabilité comportementale Dans un environnement multi-domaine, on qualifie une coopération réussie par une coopération exigeant une coordination mutuelle entre les processus propres à chaque domaine. Cette coopération implique que chaque domaine possède les informations nécessaires et suffisantes sur les processus et les services avec lesquels il va interagir. Avoir représenté et formalisé cette information n'est souvent pas évident dans un contexte muti-domaine. Cela exige un format d'échange de descriptions de fonctionnalités qui soit d'une part compréhensible par les domaines coopérant et interprétable par les machines afin de pouvoir automatiser certaines tâches, comme, la découverte, la sélection ou la composition.

En considérant l'exemple donnée dans les figures 3.3 et 3.4 relatif à la gestion d'une alerte par deux domaine, on montre la nécessité d'avoir une description sémantique des comportements de processus inter-domaine. Il définit deux processus appartenant respectivement à une maison intelligente, et à un centre d'urgence. La description des comportements des processus et leur coopération n'est pas simple, et une mise en place des mécanismes pour résoudre l'hétérogénéité comportementale des processus est extrêmement importante pour leur coopération. Le processus Figure 3.3 commence par une action de détection de chute notifiée par un capteur de chutes. Cette notification " Détection de Chute " est interceptée par le robot compagnon situé dans la maison intelligente. Le robot va "Se déplacer" à l'endroit indiqué pour s'assurer de l'état de la personne et "Envoie une alerte" si la personne ne réagit pas à ces questions.

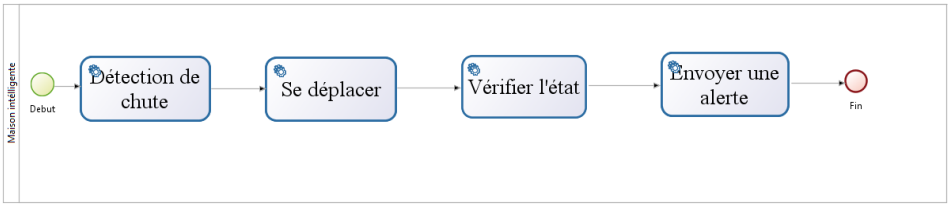


Figure 3.3: Processus de la maison intelligente

Si on regarde de près le processus du côté centre d'urgence Figure 3.4. Le centre reçoit des messages d'urgence sur des équipements de surveillance, PDA, des écrans, ou des SMS. Puis, l'agent de surveillance donne un ordre d'interversion en déclenchant l'action "Se déplacer". Imaginons le cas où sur le centre plusieurs alertes sont parvenues et qu'une coopération entre les processus des deux domaines est à mettre en place. Cette coopération doit permettre au centre de contrôler le robot et vérifier par le biais de sa camera qu'il y a vraiment urgence et qu'une intervention doit avoir lieu.

On remarque que des actions avec les mêmes labels se présentent dans les deux processus (Figures 3.3 et 3.4), et que l'action de "Se déplacer" vers la maison de la personne nécessite une action de configuration d'accès à la caméra du robot, de pourvoir lui donner des ordres de déplacement afin de prendre la décision de faire intervenir une équipe.

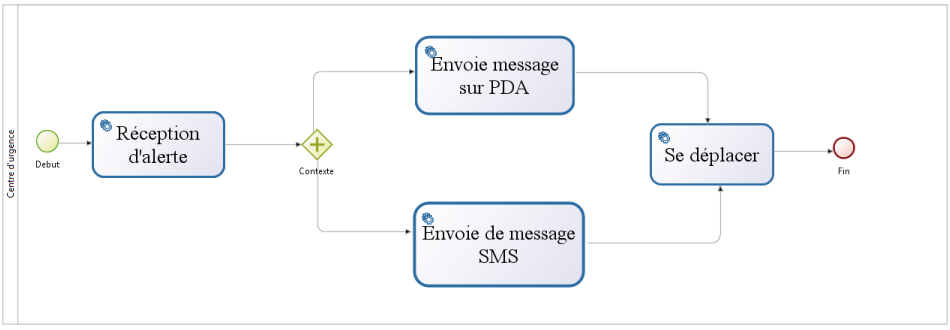


Figure 3.4: Processus du centre d'urgence

Interconnecter les deux processus nécessite une description sémantique, formelle, pour garantir le comportement du processus coopératif multi-domaine. Cette description formelle permettra de garantir la sémantique comportementale des processus et des services impliqués dans les processus coopératifs multi-domaine, à savoir, les objectifs, les actions de configuration et les règles de contrôle. Pour cela, on s'inspire des modes de communication des agents, soit les actes de langages [Moore, 1996].

3.2.1.2 L'étanchéité de processus interne et confidentialité

Les processus de gestion de chaque domaine manipulent des informations et des données confidentielles. Le comportement d'un processus inter-domaines peut impliquer la divulgation des renseignements personnels sur les règles de sécurité ou un savoir-faire spécifique, qui selon certains domaines doivent être protégés contre les concurrents [Lin and Ishida, 2008]. L'activation de la coopération multi-domaine nécessite des mécanismes permettant d'une part, une description formelle du comportement des processus, et d'autres part le contrôle approprié de la divulgation de l'information essentielle à d'autres partenaires qu'on peut assurer via des mécanismes d'abstraction pour masquer les détails inutiles concernant l'exécution des processus de coopération [Chiu et al., 2004a, Chebbi et al., 2006].

3.2.1.3 Le e-contrat de coopération et de contrôle

Les processus abstraits utilisent et manipulent des sémantiques internes et chacun des domaines présente les vues publiques de son workflow suivant leurs compréhensions et suivant leurs rôles dans la coopération. Cette coopération est la représentation d'un ensemble d'engagements mutuels d'intégration des processus de workflows. Ces engagements définissant un ensemble d'actions permettant d'atteindre un objectif commun. Dans ce contexte, la définition d'une liste d'engagements représentées par un contrat est nécessaire pour une coopération réussie [Weigand and van den Heuvel, 2002]. Le contrat exprime l'ensemble des engagements et les objectifs communs des domaines participants.

Il explicite les actions et construit une référence pour les exécutions des futurs engagements.

3.2.1.4 La flexibilité

La flexibilité désigne la capacité du canevas de coopération à adapter les processus de coopération afin de faire face aux changements qui peuvent survenir sur les politiques locales des domaines ou sur les règles de gouvernance. La séparation de la définition de processus de coopération de la partie commande est une approche intéressante pour faciliter la re-configuration des processus de coopération ou pour adapter les règles de la coopération selon les changements du contrat [Hilia et al., 2012].

3.2.1.5 La réutilisation des processus existants

Les auteurs dans [Preuveneers and Novais, 2012] présentent la propriété de réutilisation comme étant primordiale pour les méthodes de développement des applications intelligentes dans le contexte de l'intelligence ambiante. Par ailleurs, la modélisation des processus de coopération est dans la plupart des cas, une tâche fastidieuse. Pour éviter une telle contrainte, la réutilisation des processus existants est extrêmement recommandée pour accélérer la modélisation des processus, éviter la redondance des processus dans des domaines locaux ainsi que réduire le coût de développement. La réutilisation est motivée également par le fait que les processus locaux de gestion de la sécurité sont tout le temps bien conçus et hautement sécurisés. Pour illustrer l'avantage de cette exigence, prenons l'exemple du robot compagnon qui dispose d'une liste de services prêts à l'utilisation par des tiers. Par ailleurs, ces services doivent donc être accompagnés des services de provisioning pour un accès et un usage multi-domaine. Une autre forme de réutilisation est assurée par le mapping des descriptions de services abstraites vers des services concrets et des processus existants.

3.2.1.6 La vérification formelle du contrat et la preuve de "*Correctness*"

Les applications dans un environnement d'intelligence ambiante multi-domaine sont considérées comme des systèmes critiques [Coronato and Pietro, 2010], où la sécurité de l'utilisateur est une exigence très importante [Benghazi et al., 2012]. En général, l'intégration des services dans un environnement AmI est basée sur la composition multi-domaine de services, par exemple dans l'approche proposée dans [Mokarizadeh et al., 2009] une intégration des services fournis par plusieurs robots est présentée mais aucune preuve sur la correction des processus de coopération n'a été faite. A cet effet, plusieurs défis concernant la vérification formelle des services composites, la correction de son comportement, la fiabilité et la sécurité des utilisateurs [1] ont été soulevés [Bakhouya et al., 2012]. Les efforts de recherche se concentrent sur la façon de décrire sémantiquement les services (de

[1] Safety

manière formelle et expressive), comment les composer automatiquement, la façon de les découvrir et comment garantir leur exactitude [Ter Beek et al., 2006, Ter Beek et al., 2007].

Plusieurs méthodes formelles ont été proposées pour garantir l'exactitude de la composition de service. La plupart d'entre elles se sont basées sur la sémantique d'état-transition telle que les réseaux de Petri. Par ailleurs, très peu de contributions se sont basées sur la preuve de théorèmes [Papapanagiotou and Fleuriot, 2011, Rao et al., 2006]. Les deux méthodes ont pour but de vérifier les propriétés d'une spécification d'un système. Ils ont été largement utilisées dans la littérature dans différentes disciplines pour définir/spécifier les modèles et leurs propriétés requises.

Une liste non exhaustive de challenges, et de défis de la coopération ont été présentés. Dans la section suivante, une vue globale du canevas proposé pour résoudre les problématiques motivées ci-dessus sera donné ainsi qu'une méthodologie accompagnant son utilisation dans la mise en œuvre des coopérations dans un environnement ambiant multi-domaine.

3.2.2 Vue d'ensemble du canevas de coopération sémantique

Dans cette section, on présente une vue globale du canevas sémantique proposé pour la composition de services multi-domaine. Le canevas est conçu dans l'objectif d'assurer des propriétés d'interopérabilité sémantique et comportementale entre les différents domaines et de donner la preuve de correction des services composites construits. La Figure 3.5, montre l'idée de base de notre canevas qui se résume en la formalisation des services et des processus de coopération de chaque domaine et la construction de processus composites de plus haut niveau à partir de ces formalisations via la création des messages de communication entre services en utilisant la théorie des actes de langages [2] [Moore, 1996]. Le canevas proposé se compose principalement d'un modèle sémantique intégré, d'une méthodologie d'intégration des systèmes ambiants, d'un système formel de spécification d'abstraction et de preuve ainsi que d'un module de génération d'interfaces et de mapping.

Dans cette section, on présente les composantes principales et les contributions majeures de notre canevas donné Figure 3.5 :

Le modèle sémantique décrit les éléments nécessaires permettant d'établir une coopération dans un environnement ambiant multi-domaine. Ce modèle est organisé en se basant sur les fonctionnalités de ces différents modules, à savoir, le ***module générique***, qui décrit les concepts génériques et les propriétés nécessaires pour la représentation d'un environnement ambiant multi-domaine. Ce module contient l'ensemble des concepts représentant les actionneurs, les capteurs, les informations échangées entre eux, les domaines, etc. Le second module, le ***module de provisioning***, qui décrit les actions de configuration inter-domaine. Le troisième

[2] Speech act theory

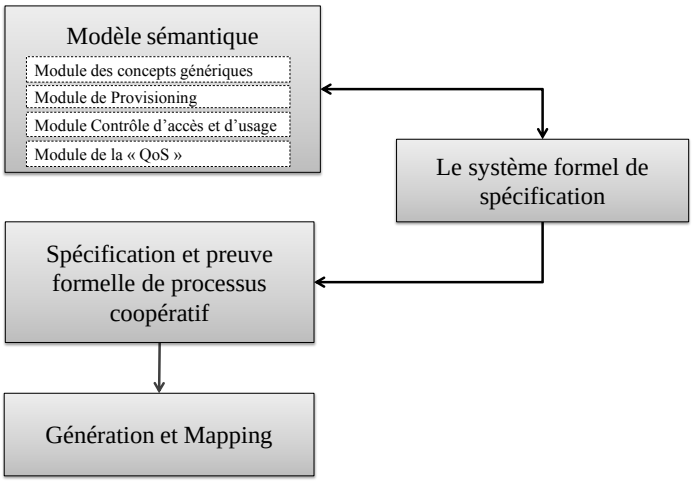


Figure 3.5: Schéma global : Les composants du canevas

module, le ***module de contrôle d'accès et d'usage***, qui représente les concepts et les propriétés pour exprimer des règles d'accès et d'usage de manière sémantique dans un cadre multi-domaine. Enfin, le dernier module, le ***module QoS***, qui décrit les paramètres de la qualité de services. On précise que ces modules ne sont pas indépendants. Cependant, pour des raisons de clarté, on a opté pour une présentation séparée de chaque module. Les avantages de ces modules par rapport aux ontologies existantes seront détaillés dans la section 4.3.

Le système formel se base sur une logique de description constructive [Bozzato and Ferrari, 2010b]. Il permet la spécification formelle du modèle sémantique, des abstractions de services coopératifs ainsi que de leurs compositions. Ces abstractions sont modélisées comme dans la majorité des travaux de description de services dans les environnement ambiants. Cette abstraction de service ou de processus modélise les pré-conditions et les post-conditions (i.e. les effets) suivant une ontologie donnée [Ghallab et al., 2004, Russell and Norvig, 2009]. On note que la spécification de ces processus qu'on appellera par la suite des vues d'abstraction se fasse au niveau des domaines, et chacun est libre de définir des actions suivant ses objectifs et suivant ses besoins en interne.

Spécification et preuve formelle des processus coopératifs ce composant formalise les structures de contrôle de flux (i.e. les règles de composition) inspirées des patterns de contrôle de workflows [van Der Aalst et al., 2003]. Ces structures permettent de construire des processus coopératifs complexes. Ce composant définit un environnement de composition dans lequel

les vues d'abstraction vont être liées entre elles via des règles de composition. Il permet aussi la génération des théorèmes nécessaires à la preuve de la propriété de correction par rapport à la spécification du processus coopératif. Ces théorèmes seront intégrés dans l'assistant Isabelle/HOL pour la preuve.

Le module de génération et d'établissement de correspondances après la spécification et la preuve des processus coopératifs qui correspondent à des plans abstraits correctes, le canevas doit être capable d'assurer son engagement envers l'approche hybride qui revient à créer des services qui n'existent pas auparavant, et de créer des correspondances avec les services et les processus existants. Ce composant permet d'analyser les processus coopératifs et de générer les messages de communication entre les différents services concrets en SOAP[3], ainsi que les ordres de configuration inter-domaine par le biais du standard SPML[4]. Les formats des processus coopératifs, seront effectués par sérialisation des processus des différents domaines participant à la coopération en XPDL[5].

Dans cette section, un aperçu sur les composants du canevas a été présenté. L'utilisation de ces composants est orchestré par une liste d'étapes qui sera présentée dans la section suivante.

3.3 Méthodologie de développement

On propose une méthodologie de mise en place de la coopération dans un environnement ambiant multi-domaine [Hilia et al., 2012]. Elle consiste en une liste d'étapes permettant de mener à bien une coopération multi-domaine. Ces étapes sont étiquetées par des numéros allant de 1 à 6. On note que chaque numéro référence un composant logiciel, et que l'ensemble intégré de ces composants logiciels correspond au canevas sémantique (Figure 3.6). On présente ci-dessous les fonctionnalités et les liaisons existantes entre ces composants, ainsi que leurs rôles respectifs dans l'ensemble du canevas sémantique.

Cette méthodologie se base sur le modèle sémantique présenté dans la section 3.4. La spécification formelle de ce modèle constitue la première étape de cette méthodologie. Dans une deuxième étape on formalise les éventuels raffinements et extensions de l'ontologie de coopération via le système formel et on vérifie sa consistance (voir Figure 3.6 : *Consistency Checking*). Ce modèle sémantique sera ensuite utilisé dans l'étape 3, qui concerne la spécification des vues d'abstraction sur les services et les processus impliqués dans la coopération multi-domaine. Chaque domaine définit les vues d'abstraction sur ses services et ses processus internes en termes de pré-conditions et post-conditions (i.e les effets). Une vue d'abstraction représente la description de l'interface et le point d'exécution

[3]Simple Object Access Protocol
[4]Service Provisioning Markup Language
[5]eXtensible Process Definition Language

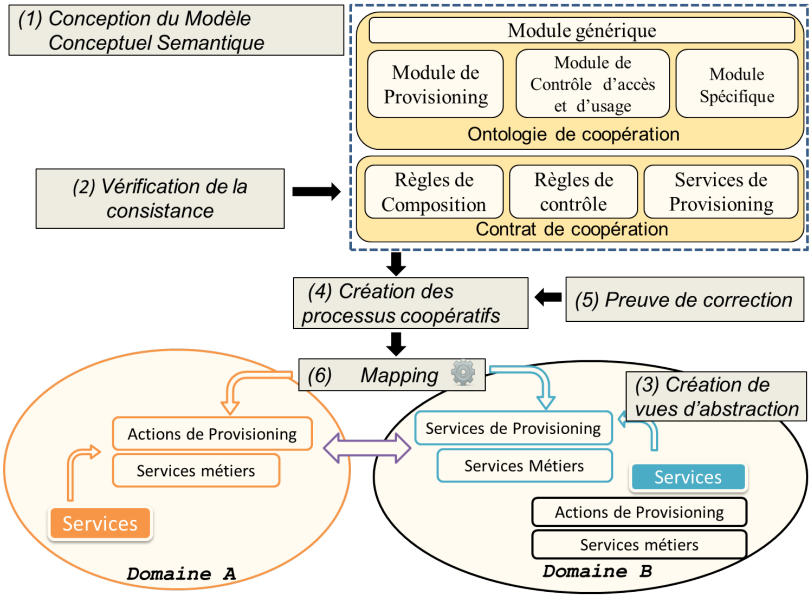


Figure 3.6: La méthodologie et les composants du canevas proposé

qui permet l'invocation d'un service concret dans un domaine donné. Ces vues d'abstraction sont formalisées suivant le modèle sémantique et le langage de composition détaillé dans le chapitre 4. L'avantage de cette abstraction est la préservation de la confidentialité de données et des structures des processus internes (voir Figure 3.6 : *Abstraction*). Dans la quatrième étape, les domaines participants définissent les processus coopératifs en tenant compte des vues abstraites disponibles. Les domaines ont la capacité de définir de nouvelles vues abstraites jugées nécessaires à la coopération et qui n'existaient pas parmi les abstractions produites dans l'étape précédente. La spécification des processus coopératifs est accompagnée de la définition des règles de contrôle, par exemple, les obligations, les permissions et les autorisations qui concernent l'accès et l'usage de chacune des vues d'abstraction par les processus coopératifs ainsi définis. La cinquième étape concerne la vérification de la propriété de correction des processus coopératifs par rapport à leurs spécifications dans l'assistant à la preuve Isabelle/HOL. La dernière étape, consiste en la génération d'interface et l'établissement des correspondances (Mapping). Dans cette étape on analyse les structures des vues d'abstraction et on génère des interfaces d'invocation des services et de processus concrets d'un domaine donné. Ces interfaces représentent souvent des actions de configuration qui ne sont pas prévues pour la gestion des accès et des usages dans un contexte multi-domaine. Le canevas établit des correspondances entre les vues abstraites créées dans les étapes précédentes et les services et les processus concrets de chacun des domaines participant à la coopération, permettant la réutilisation des processus existants.

Dans la suite de ce chapitre, on présente le modèle sémantique proposé ainsi que son utilisation dans la méthodologie présentée.

3.4 Modèle sémantique

Dans cette section, on présente le modèle sémantique permettant la description de l'environnement ambiant multi-domaine. Cette description permet la création des abstractions de services et des processus, la définition des formats de données échangées entre ces différentes vues dans l'environnement ambiant ainsi que la définition des règles de contrôle d'accès et d'usage.

Le concept de domaine précédemment introduit est essentiel pour la spécification du modèle sémantique de données. La Figure 3.7 représente le modèle sémantique qui se compose de deux grandes parties : la partie ontologie de coopération multi-domaine et la partie contrat de coopération.

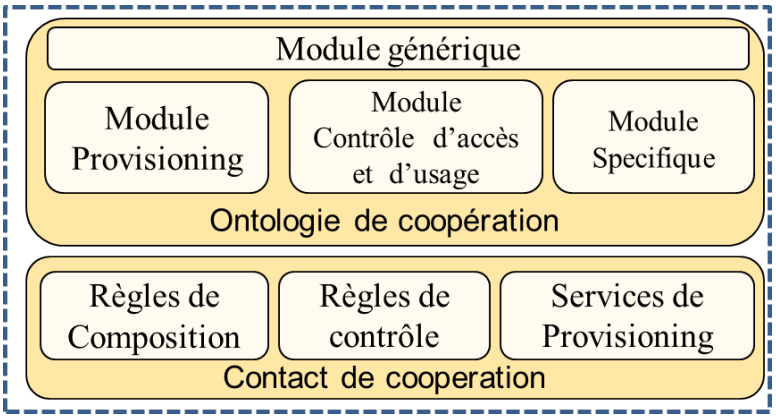


Figure 3.7: Architecture du modèle sémantique

L'ontologie de coopération multi-domaine est une ontologie de haut niveau qui explicite les connaissances relatives au domaine d'application, détaille les concepts de base pour la définition des contraintes de contrôle d'accès et d'usage dans le cadre multi-domaine ainsi que les concepts et les propriétés permettant la modélisation de services de provisioning inter-domaine. Cette ontologie est détaillée dans la section 3.4.1. Le contrat de coopération modélise les processus orchestrant les invocations de services et les règles de contrôle de ces invocations. On considère deux types de règles: les règles de composition de processus et les règles de contrôle d'accès et d'usage (y compris les règles sur la QoS). Chaque invocation de service est formalisée comme un message échangé entre deux domaines. Ce message est représenté dans une logique constructive en fonction du quadruplet suivant : (Acte de langage, Abstraction de service , domaine source, domaine de destination). Le contenu de chaque message est spécifié selon les concepts de l'ontologie de coopération multi-domaine. Le contrat est détaillé dans la section 3.4.2.

D'un point de vue technique, le modèle sémantique est une ontologie décrite dans une logique de description constructive $\mathcal{BCDL}_0$ [Bozzato and Ferrari, 2010b] ainsi que d'une liste de processus de configuration dont tout système d'intelligence ambiante doit y avoir accès pour faciliter les processus et les mécanismes de configuration entre domaines, comme la modification des paramètres, le rajout de correspondances entre des ressources et les usagers, etc. Ce modèle sémantique est ensuite utilisé

pour la description des processus de coopération. Cette description servira dans l'étape de la validation formelle de cette coopération par l'assistant à la preuve Isabelle/HOL. Le langage de spécification de l'ontologie et le contrat de coopération sont décrits dans le chapitre 4.

3.4.1 Modélisation de l'ontologie de coopération multi-domaine

Dans cette section, on présente les concepts relatifs aux entités principales nécessaires à la modélisation d'un système d'intelligence ambiante. L'ontologie qu'on propose fournit une description unifiée de la représentation des connaissances et fournit un ensemble commun de termes et de définitions. A cet effet, le transfert de connaissances se passe sans aucune ambiguïté dans des environnements composés des humains et des agents intelligents (i.e. robots) ainsi que de plusieurs domaines. Cette base de connaissances permettant aux entités hétérogènes de mieux comprendre leurs environnements via la description formelle et d'effectuer des missions plus complexes.

L'ontologie ainsi proposée est représentée par des modules (i.e. ontologies) dont les rôles, ainsi que les concepts clés et les propriétés associées seront décrits plus loin.

3.4.1.1 Module générique

Le module générique décrit les concepts de base que tout système d'intelligence ambiante doit modéliser. Les systèmes d'intelligence ambiante sont conçus à base de services. Le concept de *Service* présente donc une position centrale dans ce module [Santofimia et al., 2011]. D'autres concepts pertinents sont modélisés via ce module générique. On se focalise principalement sur les trois concepts nécessaires au domaine d'assistance ambiante, à savoir, les capteurs, les actionneurs en particulier les robots, les utilisateurs ainsi que les services associés.

Les systèmes d'intelligence ambiante sont des systèmes centrés autour de l'utilisateur. Une modélisation de ce dernier a été prise en compte dans notre modèle sémantique. Dans le schéma de la Figure 3.8, on présente le concept générique "*Person*", qui se spécialise par le concept *Patient* dans le cas d'applications dans le domaine médical. Une personne bien évidement dépend du domaine. Par exemple, dans une maison intelligente, un patient dépend de celle-ci. Son déplacement dans son domaine doit être reconnu en tout instant. Cela permet d'une part, de lui proposer des services qui peuvent améliorer son bien être. Les aspects de localisation du patient sont importants. On distingue donc deux endroits de haut niveau : l'intérieur représenté par le concept "*IndoorLocation*", et l'extérieur représenté par le concept "*OutdoorLocation*". Le concept "*Domain*" est spécialisé par les domaines introduits dans notre scénario de santé, à savoir, la maison intelligente, l'hôpital intelligent et le centre de kinésithérapie. Dans la Figure 3.8, on présente dans un schéma la modélisation de ces différentes entités. Un tableau récapitulatif sur leur définitions est présentée dans la Table 3.1.

La Figure 3.9 présente d'autres concepts de ce module, ainsi que les différentes propriétés (i.e.

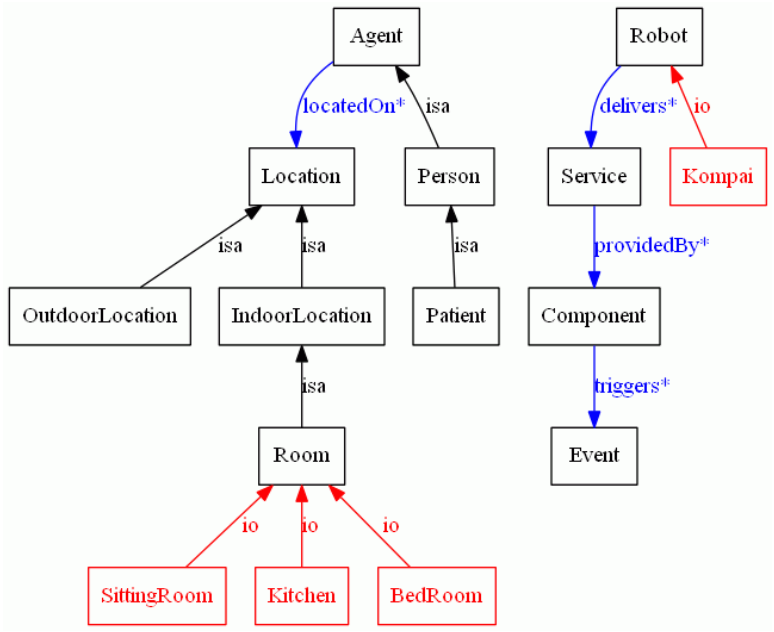


Figure 3.8: Les concepts Robot et Patient

Concept	Définition
Robot	C'est une entité physique effectuant des actions et qui interagisse avec l'environnement physique à l'aide des capteurs et actionneurs
Service	Représente une description abstraite d'une fonctionnalité offerte par un composant
Component	Désigne un composant logiciel (e.g programme) ou matériel (capteurs et actionneurs)
Event	Représente les événements, comme les notifications
IntervalTime	Représente un intervalle de temps
PreCondiction	Conditions a priori nécessaires pour exécuter une actions ou un service
Effect	Représentes les effets a posteriori, après l'exécution du service
Agent	Représente une entité abstraite pour designer les agents communiquant dans un environnement ambiant

Table 3.1: Terminologie et concepts de l'ontologie *module générique*

les rôles) qui les relient.

Les robots dans les maisons intelligentes, et en particulier les robots dits Compagnon [Ha et al., 2005b], fournissent des services de bien être et d'assistance aux utilisateurs dans leurs environnements. La propriété "*provides*" modélise le fait qu'un robot fournisse des services. Chaque service est défini par le biais d'une spécification. L'exécution d'un service ou d'une action dans un environnement ambiant est soumise à des conditions préalables sur l'état du système. Ces conditions représentent se qu'on appelle des conditions à priori ou des pré-conditions reliées à une spécification

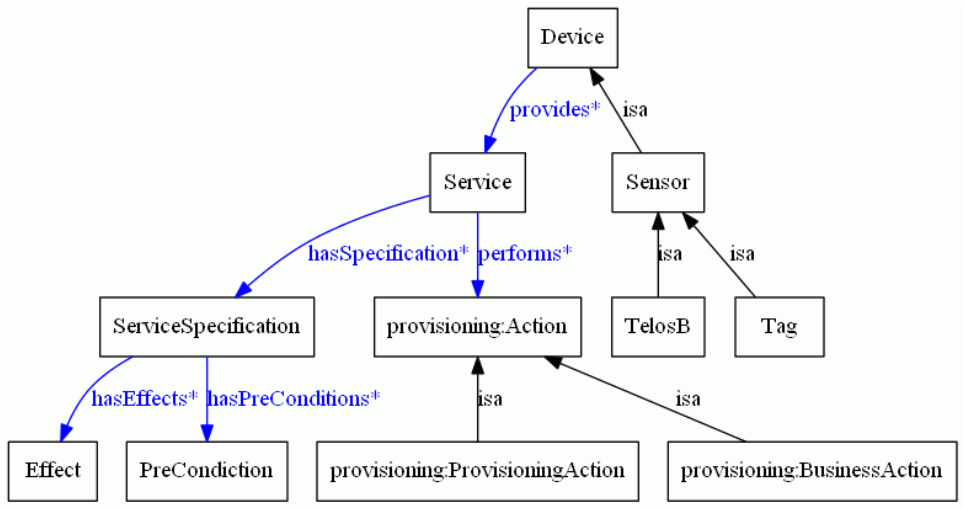


Figure 3.9: Ontologie sur l'environnement ambiant

par la propriété "*hasPreCondition*". Un service ou une action menant à des transitions entre états [Russell and Norvig, 2009]. Les nouveaux états du système sont des effets ou des post-conditions. On relie donc les effets de chaque service par la propriété "*hasPostCondition*". Un service effectue une action métier ou une action de provisioning. Ceci est modélisé par la propriété "*performs*". Ces services sont reliés à des états par la propriété *hasState*. On note que les événements se sont déclenchés dans un intervalle de temps (algèbre d'allen), propriété "*happensAt*".

3.4.1.2 Module de provisioning

Un des objectifs de l'intelligence ambiante est l'intégration des services disponibles dans l'environnement. Ces services doivent être approvisionnés (configurés) avec une intervention humaine minimale afin de pouvoir coopérer et fournir des services plus complexes. Dans cette section, on présente le modèle générique permettant la définition des requêtes de configuration de services et de ressources (voir Figure 3.10).

L'ontologie associée décrit les concepts relatifs à la partie provisioning. Le concept central "*Action*" de ce module représente une opération de configuration affectée à une tâche. Le concept "*Action*" est relié au concept "*Service*" du module générique par la propriété "*performs*". Un service effectue donc des actions. Le concept "*Action*" généralise tous les types d'actions qui peuvent être effectuées par un domaine, soit les actions de provisioning, "*ProvisioningAction*" et les actions métiers, "*BusinessAction*". Un domaine est représenté par le concept "*Domain*". Une action est exprimée par un domaine afin d'être exécutée par un autre domaine. Dans l'ontologie proposée, les actions de provisioning sont, soit acceptées et on les représente par le concept "*AcceptedAction*", soit refusées, et dans ce cas on les présente par le concept "*RefusedAction*". Ce qui correspond à la classification proposée Figure 3.10. Une action programmée par un domaine est reliée à une tâche

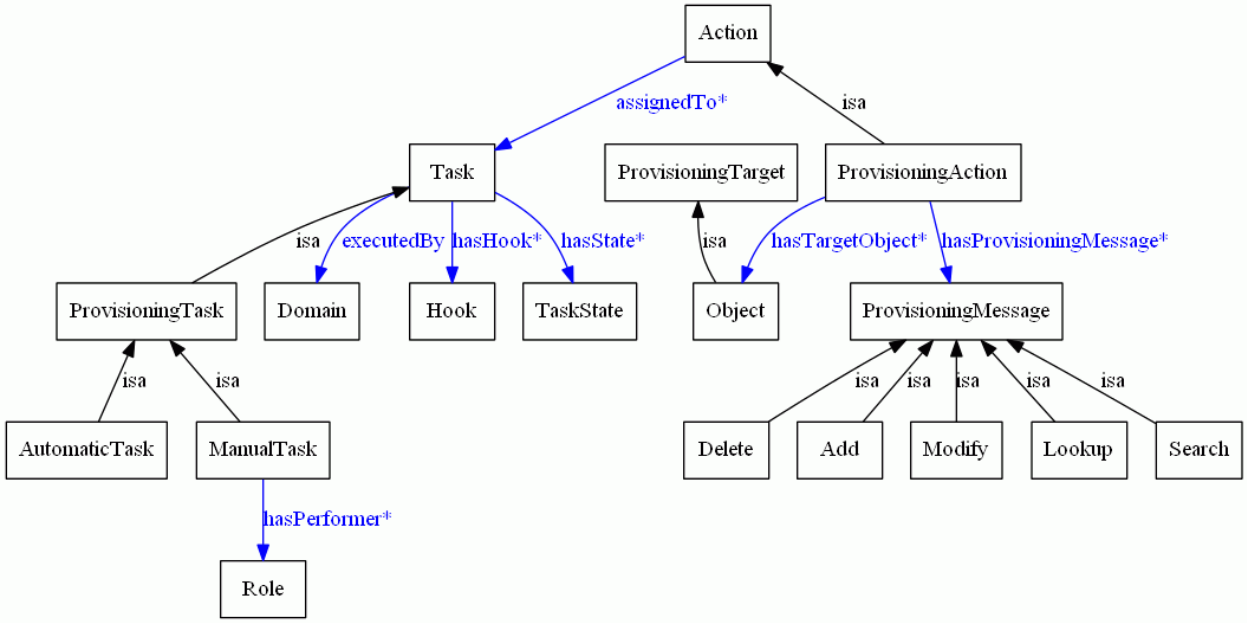


Figure 3.10: Module de l'ontology de provisioning

de configuration, par exemple, la configuration du débit du flux vidéo d'une camera de surveillance, la configuration des séances de rééducation, ou des créneaux de prise de médicaments. Ces tâches sont représentées par le concept "*ProvisioningTask*" qui représente l'unité de travail atomique dans un processus de configuration. Une tâche de provisionnement est gérée par l'exécution d'un message de configuration décrit par le concept "*ProvisioningMessage*" sur un objet spécifique (i.e. service, ressource). Ce message représente le type de la demande de configuration sur un objet spécifique. Le concept "*Objet*" représente une ressource ou un service entre les domaines participants à la coopération. Par exemple, rajouter un événement dans l'objet calendrier des activités quotidiennes d'une personne dans une maison intelligente. Le modèle proposé contient les messages de configuration suivants :

- *Add* : Ce message permet d'ajouter un objet ou une relation entre plusieurs objets.
- *Modify* : Ce message permet de mettre à jour un objet identifié.
- *Lookupt* : Ce message est utilisé pour obtenir des informations sur un objet donné.
- *Delete* : Ce message est utilisé pour supprimer un objet ou une relation entre deux objets.
- *Search* : Ce message est utilisé pour récupérer un ensemble d'objets selon certains critères.

Le concept "*Task*" représente les tâches. Il généralise la notion d'une tâche de configuration "*ProvisioningTask*". On note que chaque tâche est effectuée par une action physique atomique

représentée par un sous concept du concept action. Une tâche est liée aux trois éléments suivants : un point d'attachement, un domaine et l'état d'une tâche "*TaskState*". Le point d'attachement, "*Hook*", sert à faire un attachement ou une correspondance entre une implémentation ou à une interface spécifique et une vue abstraite. Le domaine, "*Domain*", représente le domaine d'exécution des actions. Un état d'une tâche, "*TaskState*", représente les états du cycle de vie d'une tâche donnée. On remarque que dans ce modèle, on a généralisé le concept de tâche de provisioning avant de l'associer à une action. Ceci vient du fait qu'on pourra avoir par la suite d'autres types de tâches. Cette généralisation facilite l'extension du modèle à d'autres types de tâches. Par ailleurs, une tâche de configuration est souvent automatique. Cependant, dans une coopération multi-domaine les actions sont susceptibles de passer par des approbations et des validations manuelles. On distingue donc les deux types de tâches de configuration, les tâches manuelles effectuées par une personne (agent) possédant un certain rôle dans l'environnement, et les tâches automatiquement exécutées ou déclenchées par d'autres domaines. Le concept ("*Role*") représente les rôles de coopération de haut niveau qui sont ensuite mappés sur des rôles internes des domaines participants. Les tâches de configurations suivent le cycle de vie schématisé dans la figure 3.11. On distingue les états suivants :

- Prête ("*Ready*") : Ce statue designe l'état d'une tâche disponible pour être exécuté par l'un des domaines participants.
- En exécution ("*Executing*") : représente l'état d'une tâche en cours d'exécution
- Suspendue ("*Suspended*") : cet état permet la modification d'une tâche en cours d'exécution par un domaine à un état de suspension
- Terminée ("*Finished*"): correspond à l'état de fin d'une tâche.

Le cycle de vie des tâches passe par le processus décrit dans la figure 3.11. Les tâches disponibles pour l'exécution sont désignées par l'état (Ready). Les tâches de l'état (Ready) sont affectées à un agent pour faire partie de sa liste, appelée, "TODO list". Cette liste référence la liste de tâches associées à cet agent dans le standard de la WfMC [6]. Une tâche passe de l'état "*Ready*" à l'exécution une fois sélectionnée de cette liste. A ce stade, une tâche peut être suspendue, en lui changeant dans le temps son état d'exécution à l'état suspendu avec un retour possible de l'état "*Suspended*" vers l'état "*Executing*'. Enfin, la tâche passe à l'état "*Finished*" lorsque l'agent a terminé l'exécution de la tâche.

L'ontologie dispose d'un ensemble de rôles afin d'établir les relations entre les instances des concepts de l'ontologie. Dans ce qui suit, on présente une brève description des rôles modélisées dans cette ontologie:

[6] Workflows Management Coalision

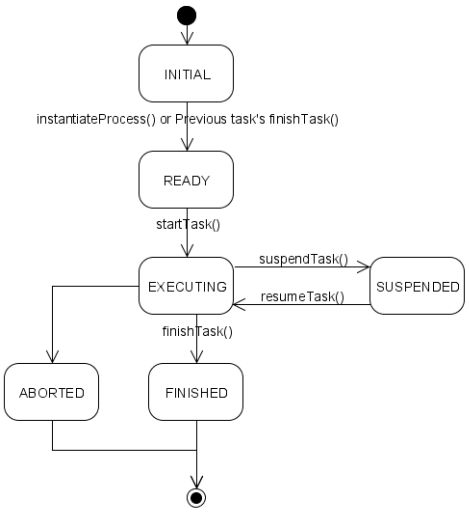


Figure 3.11: Cycle de vie de l'exécution d'une tâche

- *hasPrvgMessage* : spécifie les messages de provisionnement associées à l'action de provisionnement.
- *assignedTo* : associe l'action physique à une tâche spécifique.
- *hasPerformer* : spécifie le rôle de haut niveau qui peut effectuer la tâche manuelle.
- *hasTargetObject* : spécifie l'objet sur lequel l'action de provisionnement est appliquée.
- *executedBy* : spécifie le domaine cible qui exécute la tâche.
- *hasHook* : spécifie le lien entre la tâche et l'interface appelée.
- *hasState* : Il détermine la relation entre une tâche et son état.

L'ontologie contient également quelques propriétés des types de données "DataType property" pour déterminer des valeurs telles que le nom, les codes, etc. Cette ontologie modélise les aspects relatifs aux actions de configuration de haut niveau dans un environnement d'intelligence ambiante. Dans la section 3.4.6, on détaille comment des actions abstraites sont reliées à des tâches concrètes dans les architectures et les environnements cibles.

3.4.1.3 Module de contrôle d'accès et d'usage

Ce module est en charge du contrôle d'accès et d'usage. L'ontologie associée a été initialement développée dans le cadre du projet européen Multipol [ITEA2, 2010]. Dans les travaux récemment publiés [Hilia et al., 2011, Hilia et al., 2012]. On a proposé des extensions de cette ontologie afin

de faire face aux contraintes de gestion de contrôle d'accès et d'usage dans les environnements ambiants.

Dans les ontologies développées dans le cadre du "Pervasive Compusting" [Ye et al., 2007]. Les aspects de contrôle d'accès et d'usage n'ont pas été pris en considération. L'incorporation des concepts de contrôle d'accès et d'usage dans les environnements ambiants permet d'assurer, d'une part, la confidentialité des données manipulées, en particulier, dans un contexte multi-domaine et d'autre part, assurer la sécurité des personnes via le contrôle des actions permises dans des contextes différents. Prenons l'exemple du contrôle d'accès à des endroits ou à des localisations interdites à certaines personnes mais autorisées à d'autres dans des cas critiques ou suivant des situations d'urgence. Cela motive l'existence de ce module uniforme et extensible dans notre modèle sémantique.

La description de ce module via une ontologie de haut niveau facilitera son exploitation et son interrogation par des langages de requêtes type SPARQL [Garlik et al., 2013]. Cette description permet aussi l'utilisation des algorithmes d'alignement des politiques de sécurité locales avec ce modèle proposé. Les règles de contrôle et d'usage sont modélisées en utilisant la combinaison des concepts et des propriétés de la figure 3.12. Cette ontologie décrit une politique de contrôle et d'usage sous forme d'un ensemble de règles. Cette modélisation s'inspire du standard XACML et des logiques déontiques pour exprimer les obligations d'actions dans le cadre de la coopération multi-domaine. Dans ce qui suit, on présentera les détails sur les principaux concepts et les propriétés de cette l'ontologie.

Le concept "*Policy*", représente une politique de contrôle d'accès unique, exprimée à travers un ensemble de règles "*Rule*". Le concept "*Policy*" a un effet "*Effect*". Le concept "*Effect*" définit la décision de la règle lorsque sa condition est satisfaite. Cet effet est soit une autorisation représentée par le concept "*Permit*" ou une interdiction représentée par le concept "*Deny*". Un accès concerne une cible représentée par le concept "*Target*". Le concept "*Rule*" est donc le plus élémentaire d'une politique. Il précise l'engagement et la décision d'un domaine. Une règle est un concept générique de la notion déontique associée à un effet sur une cible spécifique. Elle exprime soit des obligations, des autorisations ou des interdictions. Le concept "*Obligation*" représente la règle qui exprime l'obligation d'exécuter l'action. Le concept "*Target*" définit la ressource cible, un service ou une action à laquelle la règle est destinée à être appliquée. Avant d'appliquer une règle une formule propositionnelle, représentée par le concept "*State*", doit être satisfaite. La propriété "*hasEffect*" spécifie la relation entre la règle et son effet. La propriété "*hasTarget*" spécifie l'entité sur laquelle la règle sera appliquée. L'ensemble des règles de la politique sont représentées par la propriété "*hasRules*". La propriété "*hasCondition*" pose la condition nécessaire à l'application d'une règle donnée. La propriété "*definedBy*" indique le domaine source de la règle définie.

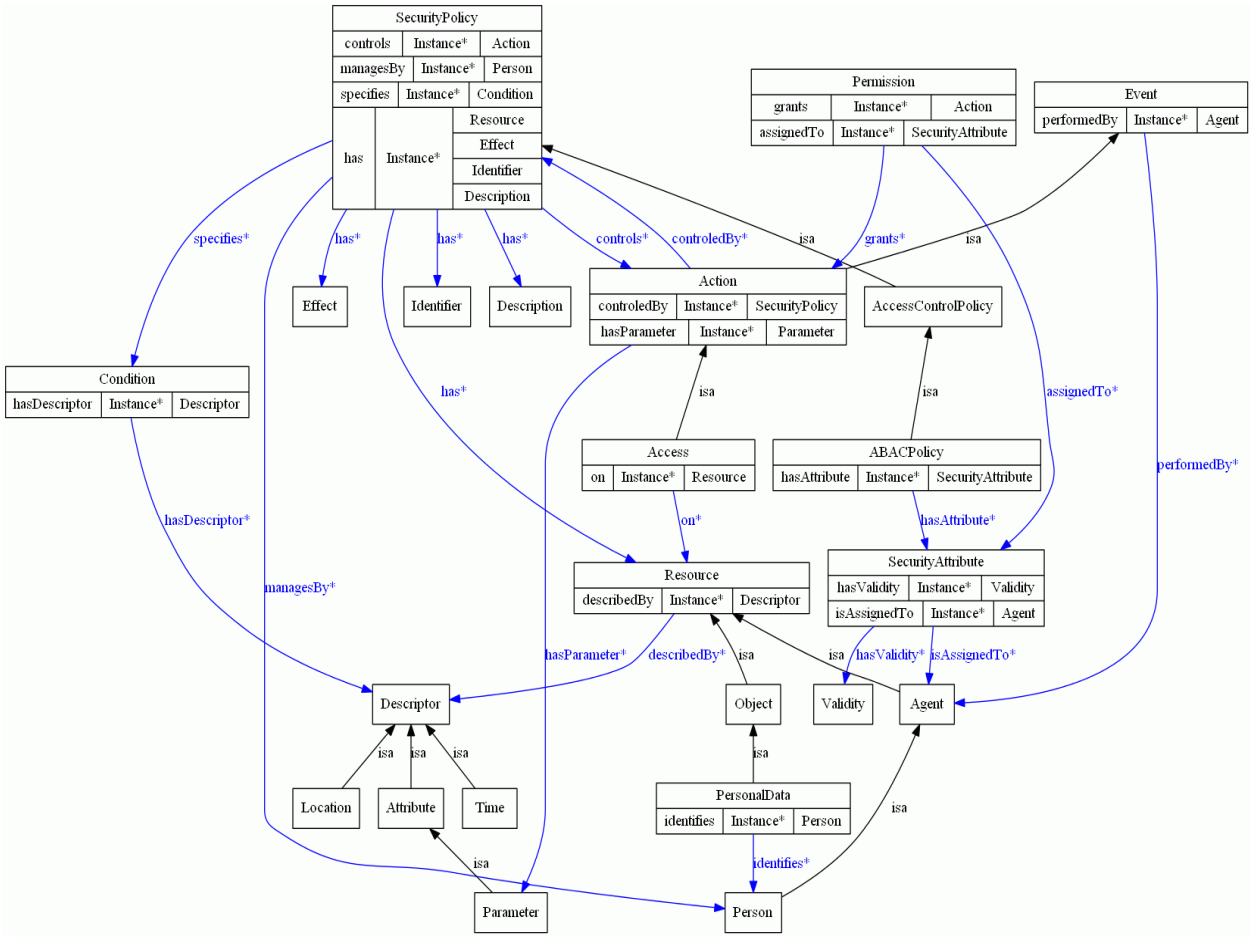


Figure 3.12: Modèle de l'ontologie de contrôle d'accès et d'usage

3.4.1.4 Module de qualité de services (QoS)

Différentes ontologies ont été proposées dans la littérature pour la gestion de la qualité de services [Stein et al., 2006, Tondello and Siqueira, 2008]. Ces ontologies ont été importées dans le module de l'ontologie de coopération en charge du suivi de la qualité de services.

3.4.1.5 Synthèse

Dans la littérature, plusieurs langages de composition et de spécification de services dans les milieux industriels et de recherche ont été proposés. Ces langages ont été classés en deux catégories, à savoir, les langages syntaxiques (e.g. BPEL et WS-CDL) et les langages sémantiques (e.g. OWL-S, WSMO) [Ter Beek et al., 2007, Ter Beek et al., 2006]. L'inconvénient majeur de ces langages est lié au manque de sémantique formelle, nécessaire à toute analyse et preuve formelle. En comparaison avec les modèles existants, comme OWL-S. Ces langages ne proposent pas des spécifications de services reliées à des considérations multi-domaine. Dans [Ter Beek et al., 2007] l'auteur précise que OWL-S est plus mature dans certain aspects (comme la chorégraphie), alors

que le WSMO fournit un modèle conceptuel plus complet qui adresse des aspects comme les buts et les médiateurs. Dans le tableau (Table 3.2) on liste quelques propriétés de comparaison pour justifier nos choix conceptuels. La première propriété concerne les séparations des préoccupations. En séparant les préoccupations de modèle, on augmente systématiquement sa modularité. Cela permettra d'accélérer les développements des applications ambiantes et d'améliorer la réutilisation des modèles par d'autres modèles équivalents ou plus efficaces [Preuveneers and Novais, 2012].

Dans ce modèle on propose les quatre ontologies suivantes : l'ontologie de la représentation de l'environnement ambiant et le domaine d'application est présentée (section 3.4.1.1), l'ontologie de configuration (section 3.4.1.2), l'ontologie de spécification des politiques de contrôle d'accès et d'usages dans le contexte multi-domaine (section 3.4.1.3), et l'ontologie de qualité de service (section 3.4.1.4).

	OWL-S	WSMO	Modèle Sémantique
Séparation des préoccupation	-	+	+
Propriété non-fonctionnelle	-	+	+
Basée sur modèle formel	-	-	+
Propriété de correction	-	-	+

Table 3.2: Comparaison avec les modèles sémantiques de composition de services

Dans la section suivante, on présente la seconde partie du modèle sémantique, à savoir, le contrat de coopération.

3.4.2 Contrat de coopération et de contrôle

Le contrat de coopération exprime les engagements en termes de processus de coopération, en d'autres termes, la spécification de la chorégraphie des vues de services impliquées dans le processus coopératif multi-domaine. Ce contrat présente une liste de modèles de processus prédéfinis pour la gestion des processus de sécurité et de configuration, à savoir, Demander, Approuver, Déléguer, et Notifier. Chacune de ces opérations est formalisée via un acte de langage [Zhang et al., 2010].

Le contrat implique donc les différentes vues créées et leur composition, les modèles de processus prédéfinis ainsi que les règles de contrôle d'accès et d'usage. Dans cette section, on présente les modèles de processus et les opérateurs permettant leur organisation dans des chorégraphies. Ensuite, on présente les différentes règles de contrôle qu'on peut exprimer via le canevas sémantique proposé. Ces règles doivent assurer un certain niveau de contrôle qui, en même temps, n'affecte pas le fonctionnement de la coopération.

3.4.2.1 Modèles de processus

En général, les processus de gestion de la sécurité amènent une ou plusieurs actions locales ou distantes pour réaliser un objectif. Ces processus sont traditionnellement exprimés sous la forme d'une autorisation accordée à un sujet d'exercer une action sur un objet. Dans ce cas, le processus amène une action atomique, par exemple, l'arrivée d'une nouvelle entité, l'assignation de rôle, l'assignation d'une permission. Ce type de processus désigne un processus simple. Dans d'autres situations, ces processus sont plus complexes, car ils traitent des actions qui nécessitent plus d'actions et de transitions conditionnelles afin de les exécuter. Les processus dits simples comme l'approbation, la délégation, etc. peuvent toujours être combinés afin d'exprimer des processus complexes. On présente dans la suite le modèle générique et une liste non exhaustive de processus d'actions de gestion de la sécurité.

Les interactions entre les domaines dans une coopération consiste à échanger des ordres et des demandes qu'on appelle des *énoncés*. Ces énoncés sont souvent sous la forme d'un ordre qui s'annonce généralement par un verbe et un contenu. Par exemple, déléguer une tâche, approuver une action, assigner un rôle, etc. Formellement, et conformément aux hypothèses de la théorie des actes de langages [Moore, 1996]. Les énoncés sont de la forme $F(P)$ avec F la force illocutoire du message porté par un verbe typé, un prédicat, (i.e. Informer, Demander) et P le contenu propositionnel du message (ce qui est informé, a demandé). Un modèle de processus correspond parfaitement à un acte de langage F(X), où le contenu X est exprimé dans un langage L et une ontologie N. On donne ci-dessous la définition des patrons de processus de base du canevas sémantique proposé.

Approbation Le processus d'approbation est une autorisation par l'approbateur, le domaine qui donne son avis comme une acceptation ou un refus, sur l'autorisation du demandeur à exécuter certaines actions. Ce processus implique un agent d'un domaine source (agent 1) qui demande une action qui nécessite une approbation d'un autre domaine cible. Le cycle d'approbation est déclenché suite à cette demande. L'approbateur reçoit la demande et prend une décision d'acceptation ou de refus, qui fera ensuite l'objet d'une notification au demandeur. L'acceptation permet donc au demandeur d'exercer l'action avec l'accord de l'approbateur.

Demande Le processus de demande est une requête d'une action qui nécessite obligatoirement une réponse par le domaine cible. Un demandeur crée sa requête de demande d'une action, type autorisation, accès, ou autre. L'action est émise vers un domaine destinataire afin de traiter la demande. Suite à ce traitement le demandeur recevra la réponse.

Délégation La délégation est une opération par laquelle un domaine, le délégué, fait ou s'oblige à faire une prestation à un autre, le délégataire, qui l'accepte sur ordre d'un troisième domaine, le délégant. La délégation réalise un véritable transfert de pouvoir à une autorité inférieur.

Ce type de délégation a pour effet de dessaisir l'autorité délégante. Celui-ci ne peut plus exercer la compétence qui a été déléguée pendant tout le temps de la délégation, le délégué doit s'assurer que le délégataire a les compétences et les moyens nécessaires pour accomplir sa mission, suivant le type de la délégation, le délégué confit les responsabilités au délégataire, donne les autorisations aux actions et aux ressources permettant d'accomplir la tâche déléguée. Le délégataire à son tour répond par une acceptation ou un refus.

Notification Ce processus s'occupe de notifier des destinataires de l'occurrence d'un événement suite à leur abonnement à celui-ci. Un domaine s'enregistre auprès de la liste des notifiés de l'occurrence d'un événement. Après le déclenchement de cette événement, le message est envoyé à tout les domaines de la liste.

Dans la suite, on présente les structures de contrôle de flux permettant la construction de ces processus coopératifs.

Structures de contrôle de flux sont proposées pour constuire des processus coopératifs plus riches, plus complexes, et d'un haut niveau d'abstraction. Des structures de contrôle de flux dites aussi des règles de composition permettent de composer les vues de processus et les patrons de processus afin de répondre aux exigences et aux objectifs de la coopération. Ces règles de composition sont décrites dans la Table 3.3. La formalisation de ces règles ainsi que l'étude de leur fiabilité à construire des processus coopératifs "corrects" seront présentés dans le chapitre 4. Les règles de composition sont conformes aux patterns basiques de contrôle dans les workflows [van Der Aalst et al., 2003].

3.4.2.2 Règles de contrôle d'usage

Les règles de contrôle englobent les règles d'usage de services et les contraintes imposées sur l'utilisation des services mis à disposition de la coopération par un domaine. Pour des raisons de modularité et de clarté, on propose de scinder les règles de contrôle en deux classes. Les obligations applicatives ou règles métiers, et les règles de contrôle d'accès et d'usages.

Les obligations Ces règles sont définies en se basant sur le modèle de la logique déontique, qui permet d'exprimer les obligations aussi bien que les permissions et les interdictions. Par exemple, une personne est obligée de quitter une pièce en cas de détection d'incendie. Concrètement, les obligations sont des indicateurs déclenchés ou retournés par un domaine demandant d'exécuter une action. Par exemple, l'obligation de s'authentifier avant de pouvoir exécuter une action. Ce type de règles ne dépend pas d'une réponse favorable ou d'une interdiction. Un autre exemple d'obligation dans notre étude de cas correspond à la situation suivante : Dans un programme d'auto-rééducation

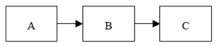
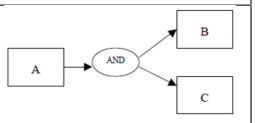
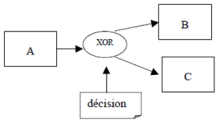
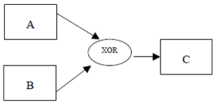
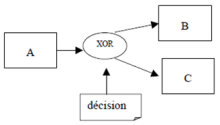
Règle	Description	Schéma
Séquence	Cette règle exprime le fait qu'un processus est exécuté après la terminaison d'un autre processus.	
Parallélisme (Parallel plit)	Cette règle représente une étape dans l'exécution du processus où une liaison de contrôle simple est divisée en plusieurs liaisons de contrôle qui s'exécutent en parallèle .	
Choix Exclusif (Exclusive choice)	Cette règle maintient une liaison dans le processus. Cette liaison est choisie en se basant sur la décision prise au moment de l'exécution. Cette décision permet le choix entre les différentes liaisons disponibles.	
Synchronisation (Synchronization)	Cette règle représente la convergence de deux ou plusieurs tâches dans un point de synchronisation unique. La tâche sortante est activée après l'exécution de toutes les tâches entrantes.	
Fusion simple (Simple merge)	Cette règle représente la convergence de deux ou plusieurs tâches en une seule tâche. Les tâches sortantes sont effectuées lorsqu'une tâche entrant est déclenchée.	

Table 3.3: Les règles de composition [van Der Aalst et al., 2003]

le patient ne peut pas faire ses activités de rééducation n'importe où dans sa maison. Une salle est prévue pour ce type d'activités. A cet effet, la personne est obligée d'être présente dans cet endroit avant toute interaction avec les services de l'hôpital.

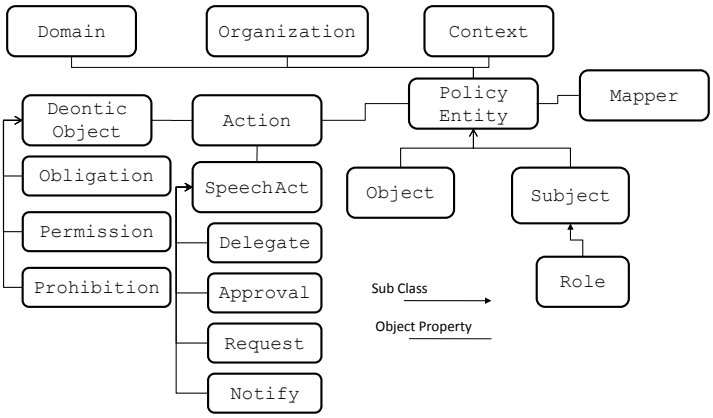


Figure 3.13: Modèle déontique pour les obligations

Règles de contrôle d'accès et d'usage Ces règles constituent la politique de coopération muti-domaine. Contrairement aux obligations, ce type de règles a besoin d'une réponse qui correspond à une autorisation d'accès par le domaine cible, un (Permit), ou un refus d'accès qui correspond à un (Deny). Ces règles de décision peuvent être écrites selon la forme suivante: Nom de la règle (*Suject* (*Role / Identity*), *Action*, *Objec* (Ressource, Service), *DeonticObject*). Un des exemple de ces règles : En cas de chute du patient une notification est envoyée au service de secours pour une intervention urgente. Une règle d'interdiction d'ouverture d'une porte le soir : *NotificationDeChute (Patient, ServiceDetectionChute, Notifier , Obligation)*

3.4.3 Vérification de la consistance de l'ontologie / cohérence

Dans la section précédente, on a développé un modèle d'ontologie de coopération de haut niveau sans donner aucune garantie sur sa consistance. Par ailleurs, ce modèle est conçu dans le but d'être étendu est adapté afin de répondre à des objectifs multi-domaine. A cet effet, les domaines peuvent rajouter des concepts et des rôles à une condition prêt qui consiste à maintenir l'ontologie dans un état consistent. Ce qui veut dire que le modèle ne doit en aucun cas contenir des incohérences ni des contradictions. Par ailleurs, l'utilisation efficace des langages sémantiques nécessite l'utilisation des raisonneurs sémantiques comme Pellet, FaCT++, RacerPro et KAON2. Ces raisonneurs ont la capacité de déduire des informations implicites, et ont des mécanismes de déduction qui leurs perme-

ttent d'inférer de nouveaux faits à partir des faits explicitement encodés dans le modèle sémantique. En revanche, la capacité de ces raisonneurs ne se limitent pas à la déduction d'informations mais, vérifient aussi la cohérence du modèle sémantique.

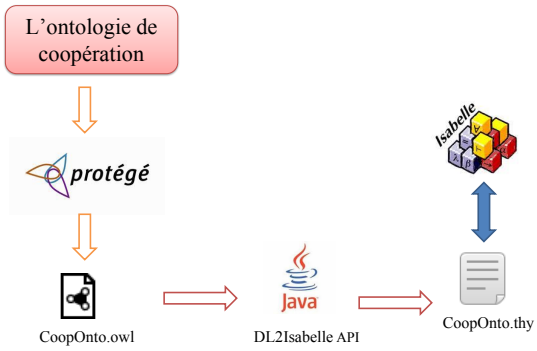


Figure 3.14: Modélisation et transformation de l'ontologie de coopération

Techniquement parlant le modèle de l'ontologie de coopération a été développé sous protégé version 3.4.8. Cette dernière dispose d'une version Pellet intégré dans sa version 1.5.2 pour détecter d'éventuelles incohérence de conception après tout raffinement, ou rajout de nouveaux concepts ou de nouveaux rôles.

3.4.4 Abstraction des services et processus existants et leur modélisation

Dans une coopération, il est nécessaire que chacun des partenaires participe d'une partie du processus multi-domaine. Cette partie collaborative est visible par tous les autres partenaires, ce qui donne un point d'accès et une vision sur les activités des workflows impliqués dans une telle coopération. Par conséquent, cette partie doit être construite soigneusement, et efficacement afin d'être le plus expressive possible, tout en protégeant la structure et le comportement internes des workflows.

Dans les précédents travaux sur la collaboration de workflows les vues sur les workflows ont été choisies comme le moyen efficace permettant de préserver le savoir-faire, et la conservation de l'autonomie des workflows internes. La vue est une abstraction du workflow interne où seulement les parties collaboratives du workflow sont exposées aux partenaires. Dans une coopération muti-domaine, le processus de coopération nécessite une connaissance minimale pour pouvoir invoquer les services. Ces abstractions sont construites en se basant essentiellement sur les concepts et les propriétés définies dans l'ontologie de coopération.

Dans la littérature plusieurs approches ont été proposées pour faire coopérer des systèmes autonomes [Chiu et al., 2004b, Lin and Ishida, 2008]. La plupart de ces approches utilisent des techniques d'abstraction de service à base de vues [Chebbi et al., 2006, Klai et al., 2006]. Les travaux à base de vues constituent une approche adéquate pour préserver le savoir-faire des or-

ganisations et la conservation des workflows existants. Ces résultats sont obtenus par une projection restreinte du workflow interne par rapport aux fonctionnalités offertes et les participations dans les processus coopératifs. La construction de ces vues a été proposée dans les travaux de Chebbi [Chebbi et al., 2006] et formalisé par la suite dans les travaux Nomane [Klai et al., 2006]. L'autonomie de ces workflows se maintient à travers des représentations abstraites, les modifications internes restent invisibles, et les processus métiers sont cachés au workflow inter-organisationnel.

L'adoption de cette approche dans le contexte de l'intelligence ambiante et sa description sémantique assure le faible couplage entre les domaines coopératifs et la facilité de découverte des services. Dans [Urbieta et al., 2008a] les auteurs justifie le choix de représentation des services dans un environnement ubiquitaire sous la forme de pré-condition et d'effets comme la représentation adéquate dans ces types d'environnement. Ceci a été justifié dans d'autres travaux comme celui de [Russell and Norvig, 2009] qui montre que les techniques de planification dans les environnements ambiants représentent les services comme étant des actions sous la forme $IOPE$, où I représente les données en entrée (Input), O les données en sortie (Output), P représente l'ensemble des pré-conditions du service (Precondition), et E les effets ou les post-conditions (Effect). Ce format est celui considéré par le langage du représentation des services web sémantique OWL-S.

Ci-dessous on donne l'exemple des abstractions dans l'étude de cas sur laquelle on va étudier la faisabilité du canevas sémantique proposé. Dans le schéma (Figure 3.15), on donnera les exemples des vues d'abstraction. On prends l'exemple de la première vue *SetProgram* qui consiste à l'initialisation d'un programme de rééducation *reeducation_epaul_1* pour une personne *patient_1*

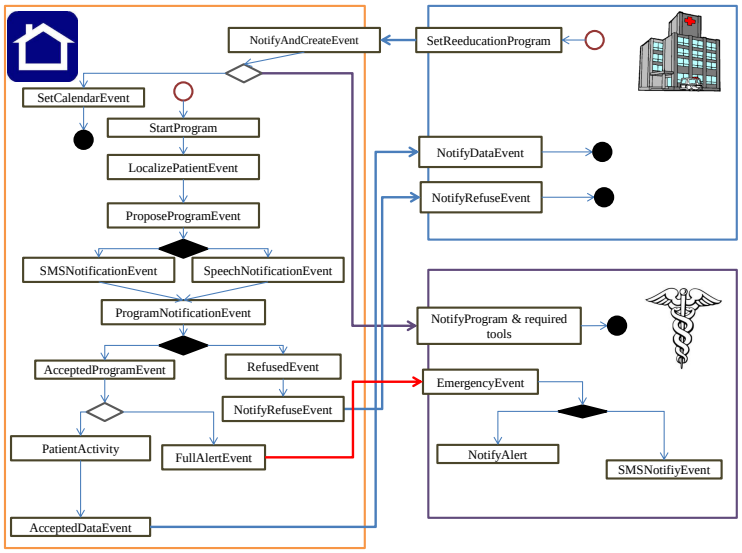


Figure 3.15: L'abstraction des vues dans l'étude de cas de télé-rééducation

Les vues sont décrites en termes de pré-conditions et des effets. Par exemple, les vues d'abstraction *ProgrammerSéance* et *Localiser* créées respectivement par les domaines de l'hôpital et de la maison

intelligente sont décrites de manière informelle comme suit :

Programmer une séance Cette vue consiste en une pré-condition représentée par un événement de configuration du calendrier par un programme de rééducation affecté à une personne destinataire. L'effet de cette action est soit un refus du le programme par le domaine destinataire, soit l'action est acceptée et le message associé à l'action de *provisioning* est déclenché. Le message correspond à un rajout dans le calendrier *Add*. On précise que cette vue correspond à une vue sur l'action de configuration du calendrier, et que ce service n'existait pas avant la mise en place du processus coopératif.

La formalisation de la vue est représentée dans le tableau (Table 3.4), plus de détails sur le langage de spécification et sa sémantique est présenté dans le chapitre 4.

Nom du service	$SetReeducationProgram$
Variables	Les variables d'entrées , $provisioning\_event$
Préconditions	$ProvisioningEvent \sqcap \exists hasProgram.(ReeducationProgram \sqcap \exists hasDestination.Person)$
Post-conditions	$RefusedEvent \sqcup (AcceptedProvisioningEvent \sqcap \exists hasAction.(ProvisioningAction \sqcap \exists hasMessage.ProvisioningMessage))$

Table 3.4: Formalisation de la vue d'abstraction du service : Programmer une séance

Localiser Cette vue a comme objectif de localiser un objet. Cet objet doit être localisable. Cette vue consiste en un événement de localisation comme entrée et un objet localisable. Elle produit les effets suivants. Soit refuser action si l'accès à cette information n'est pas permis, soit l'action est acceptée en fournissant les informations sur la localisation de l'objet. Au contraire de l'exemple de vue précédent, ce service correspond à une action métier offerte par la maison intelligente qui ne s'occupe pas des aspects de configuration.

3.4.5 Vérification formelle de "correctness" de processus coopératifs

Cette étape de la méthodologie proposée consiste en la vérification formelle de la propriété de correction des processus coopératifs précédemment définis. Comme précisé, ces processus coopératifs sont conçus en se basant sur des délégations, des approbations ainsi que sur les vues d'abstractions des processus métiers et de configuration. Cette composition nécessite la certification qu'elle est correcte et qu'elle réalise les objectifs définis dans le contrat de coopération. Cette propriété désigne la *correction* du processus coopératifs. Pour atteindre cet objectif, on propose un langage de spécification basé sur une logique de description constructive. Dans le chapitre suivant, on donne plus de détails sur la réalisation de la spécification des processus coopératifs et sur la vérification de leur correction. L'implémentation de cette partie est réalisée sous l'assistant à la preuve de théorèmes

Isabelle/HOL, pour spécifier et prouver la correction de la composition du processus coopératif et par conséquent vérifier que l'implémentation des processus composés répondent bien à l'objectif définit par la coopération.

3.4.6 Génération des interfaces d'échanges et Mapping vers les domaines coopératifs

Cette étape représente la dernière étape de la méthodologie proposée. Dans la Figure 3.16, les processus coopératifs avec les règles de contrôle formellement prouvés font la partie descriptif de haut niveau. Cette description fait l'objet d'une entrée, du composant de génération et de mapping afin de générer des web services, et les messages d'échanges en SPML. Le canevas génère aussi des interfaces d'appel au service web existant dans les différents domaines.

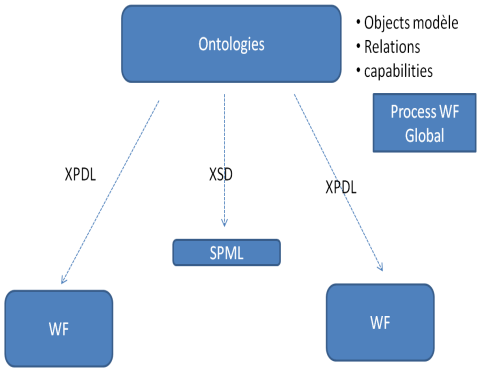


Figure 3.16: Schéma global d'architecture de Mapping

Cette étape de la méthodologie vient après la vérification que la spécification du contrat de coopération est correcte. Elle permet de générer des interfaces de communication entre les différents domaines coopératifs via des services web, décrites en WSDL. Ces interfaces représentent des points d'attachement vers des services et des processus internes. Par ailleurs elle communiquent entre elles via des messages SOAP pour les actions et les échanges sur les services fonctionnels. Les services non fonctionnels nécessitent le standard SPML afin d'échanger des ordres de configuration multi-domaine. Les processus coopératifs correspondent à des workflow multi-domaine. Ces processus sont décrits par le standard de définition de workflow proposé par la WfMC[7]. Les règles de contrôle d'accès sont sérialisés en XACML, le standard permettant de définir et exploiter des politiques de sécurité.

[7]WfMC : Workflow Management Coalition

Conclusion

Dans ce chapitre, on a présenté la première contribution qui consiste en un modèle sémantique et une méthodologie pour l'intégration d'application d'intelligence ambiante dans un contexte multi-domaine. Le grand avantage du modèle sémantique proposé est basé sur sa réalisation de la communication inter-domaine par l'établissement d'un contrat de coopération sur la capacité de réutiliser les services et les processus des domaines participant à la coopération. La modélisation du modèle sémantique via des ontologies permet de manipuler des représentations formelles de l'environnement. Ces représentations peuvent être analysées par des outils de vérification. Cette modélisation facilite aussi le partage et la réutilisation de connaissances. Dans le chapitre suivant, on présente le système formel pour spécifier ce modèle sémantique proposé. On va démontrer sa fiabilité [8], ce système formel permettra la formalisation des abstractions de service et de processus ainsi que la preuve formelle de la correction [9].

[8] Soundness
[9] Correctness

4

Spécification et validation formelle sur l'outil Isabelle/HOL

Sommaire du chapitre

Introduction

Dans ce chapitre, on s'intéresse aux langages de spécification des modèles sémantiques et à la preuve formelle. Différents langages ont été proposés pour la spécification des abstractions de services web sémantiques, et de leurs compositions, comme OWL-S ou WSMO. Cependant, les spécifications basées sur ces modèles manquent de sémantique formelle permettant leur validation. Par ailleurs, de nombreux travaux sur les logiques de description constructives ont été publiés ces dernières années [de Paiva, 2006, Mendler and Scheele, 2008], sur leurs applications à la modélisation des ontologies, ainsi que pour montrer et discuter l'avantage de leur sémantiques constructives dans la preuve formelle [Ferrari et al., 2009].

Dans ce chapitre on présente une variante des logiques de description constructives appelée *logique de description constructive basique*, notée $\mathcal{BCDL}_0$ [Ferrari et al., 2009, Bozzato and Ferrari, 2010b]. Cette logique sera employée lors de la spécification et la validation formelle du modèle sémantique introduit dans le chapitre 3. Le présent chapitre est organisé comme suit. Dans une première partie, une introduction aux logiques de description ainsi qu'un aperçu sur les méthodes formelles est présentée. Par ailleurs, les limites des sémantiques classiques par rapport à la preuve formelle de théorèmes et le besoin de passer à des sémantiques constructives sont discutés [de Paiva, 2006, Mendler and Scheele, 2008]. Cette sémantique constructive dite aussi *intuitionniste* a de nombreux avantages à savoir, le respect du paradigme *proofs-as-program*, la capacité de donner des sémantiques au calcul de preuves ainsi que le support des techniques de la preuve de théorèmes utilisées pour la vérification de la correction des spécifications d'un système. Ensuite, on se focalise sur la mise en œuvre du système formel basé sur $\mathcal{BCDL}_0$ et la preuve de sa fiabilité [1] dans l'assistant à la preuve Isabelle/HOL [Nipkow et al., 2002]. Dans une deuxième partie, le système formel sera utilisé pour la formalisation du modèle sémantique proposé dans le chapitre 3 ainsi que la formalisation des vues abstraites sur les services et les processus multi-domaine. La spécification formelle de la composition de services multi-domaine, la formalisation du langage de composition ainsi que l'implémentation et la preuve dans Isabelle/HOL sont présentés. Finalement, un scénario relatif au domaine de la santé est présenté en se basant sur l'approche formelle proposée. La preuve interactive dans Isabelle/HOL permettra d'évaluer l'impact d'un changement de la composition des processus (services) coopératifs sur la propriété de correction[2].

4.1 Langages de spécification et méthodes formelles

Un langage de spécification formelle est utilisé pour décrire un système d'une manière mathématique sans ambiguïté. Par ailleurs, des méthodes et techniques de vérification formelle ont été définies sur

[1] Soundness
[2] Correctness

la base de ces spécifications afin d'assurer la correction du système résultant. L'approche formelle adoptée pour la spécification du modèle sémantique proposé se base sur la logique de description $\mathcal{BCDL}_0$, avec une sémantique constructive permettant de donner un sens aux preuves obtenues avec ce langage de spécification.

Dans cette section, une présentation des logiques de description considérées comme des langages de spécification avec des sémantiques formelles ainsi qu'un aperçu sur les méthodes formelles permettant de garantir des propriétés attendus des systèmes spécifiés seront données. L'étude de la relation existante entre les logiques de description et les méthodes formelles en particulier la méthode de vérification formelle permettra de clarifier la relation existante entre les sémantiques classiques, les sémantiques constructives ainsi que la théorie de la preuve.

4.1.1 Logiques de description

Les logiques de description (LDs) sont une famille de langages de représentation des connaissances [Baader, 2003]. L'objectif principal des LDs consiste à représenter formellement ces connaissances et de pouvoir raisonner efficacement sur celles-ci. Les logiques de description forment des fragments décidables de la logique du premier ordre. La recherche sur les logiques de description a développée des raisonneurs fiables [3] et complets. Ces raisonneurs ont permis la croissance de l'efficacité des différentes variantes des logiques de description [M. Birna van Riemsdijk et al., 2008]. Le fait que les logiques de description sont livrées avec ces raisonneurs est un avantage important de leur utilisation pour la spécification de services de manière sémantique et formelle. Les LDs reposent principalement sur trois notions de base, à savoir, les concepts, les rôles et les individus. *Les concepts* représentent des classes d'objets. *Les rôles* constituent les relations binaires entre deux objets, alors que *les individus* représentent les objets, appelés aussi *instances*.

Pour décrire ces éléments, deux structures d'ensembles sont utilisées : la $TBox$ et la $ABox$.

Boîte terminologique (Tbox) cet ensemble comprend la description des concepts et des rôles. Deux concepts particuliers figurent au minimum dans la $TBox$, le concept le plus générique ($\top$, le concept universel), et le concept le plus spécifique ($\bot$, le concept vide). La $TBox$ est un ensemble fini de formules de la forme $A \sqsubseteq C$ (La subsumption), où A et C sont des concepts.

Boîte d'assertions (Abox) cet ensemble est constitué des individus, de leurs descriptions et des règles qui leurs sont attachées. La $ABox$ est un ensemble d'assertions de rôles et de concepts telles que :

- Une *assertion de rôle* est une formule de type (c,d) : R, avec c, d $\in$ NI et $R \in$ NR;
- Une *assertion de concept* est une formule de type t : C, avec $t \in$ NI et $C \in$ NC

[3] Sound

Où, NR, NC et NI représentent respectivement les ensembles des noms de rôles, des noms de concepts et des noms d'individus.

Dans ce qui suit, on présente la formalisation de ces différentes entités représentatives de la syntaxe de la logique de description $\mathcal{BCDL}_0$

4.1.1.1 Syntaxe

Les logiques de description se sont basées sur les ensembles dénombrables suivants : l'ensemble de noms de concepts NC, l'ensemble de noms de rôles NR, l'ensemble de noms d'individus NI et VAR l'ensemble des noms de variables d'individus. Le concept est l'élément syntaxique central de $\mathcal{ALC}$. Un concept est une formule de type suivant :

$$C, D ::= A \ \ |\neg\ C\ |C\ \sqcup\ D\ |C\ \sqcap\ D\ |\exists\ R.\ C\ |\forall\ R.\ C$$

Où $C,\ D \in$ NC et $R \in$ NR

Table 4.1: Grammaire de la logique de description $\mathcal{BCDL}_0$

Dans cette formulation conforme à la forme de BNF[4], on remarque la correspondance établie entre une formule et un type. Cette correspondance est connue comme étant l'isomorphisme de Curry-Howard [Troelstra and Schwichtenberg, 2000]. Les concepts représentent les entités du domaine à modéliser. Un concept est construit à partir des concepts dits atomiques, noté par A. La syntaxe de $\mathcal{BCDL}_0$ présente aussi des constructeurs ($\neg$, $\sqcup$, $\sqcap$, $\exists$, $\forall$) permettant de bâtir des concepts plus complexes en partant des concepts atomiques. La description de ces constructeurs est donnée Table 4.2.

Concept	Symbole	Description
La négation	$\neg\ C$	Non C
La disjonction	$C\ \sqcup\ D$	C ou D
La conjonction	$C\ \sqcap\ D$	C et D
Le quantificateur existentiel	$\forall\ R.\ C$	Il existe un R-successeur qui est dans C
Le quantificateur universel	$\exists\ R.\ C$	Tous les R-successeurs sont dans C

Table 4.2: Les constructeurs de la logique $\mathcal{BCDL}_0$

La formulation représentée Table 4.3 définit les types de formules K engendrées par rapport à un sous ensemble de noms d'individus $\mathcal{N}$ de NI. L'ensemble des formules générées sont désignées

[4]Backus–Naur

par le langage $\mathcal{L}_{\mathcal{N}}$. Les formules K sont définies par la grammaire suivante :

$K ::= \ \bot \mid \ \mathrm{t} \ : \ C \mid A \ \sqsubseteq \ C \mid \ \ (s,t) : R$
Où A est un concept atomique et s,t $\in$ NI $\cup$ VAR

Table 4.3: Les formules générées dans $\mathcal{BCDL}_0$

Le langage $\mathcal{L}_{\mathcal{N}}$ est une représentation syntaxique (vocabulaire) des connaissances d'un domaine donné, ce vocabulaire n'est pas compréhensible par les machines. Par conséquent, aucune tâche de raisonnement automatique n'est possible au-dessus, comme la déduction, la construction de nouveaux concepts, ou bien la détection des incohérences de modélisation. Aussi, une sémantique doit être attribuée aux différentes entités de la logique (i.e. concepts, rôles, individus et constructeurs). En générale, une sémantique d'une entité est définie par une fonction dite une fonction d'interprétation $(.^I)$ qui assigne à chacune des entités un sens ou une signification (i.e. une sémantique) dans un domaine cible dit domaine d'interprétation (Δ^I).

Définition 4.1.1 *(Une interprétation) Une interprétation I consiste en un ensemble non vide Δ^I appelé son domaine et une fonction d'interprétation $.^I$ qui assigne à :*

- *Chaque nom d'individu un élément $a^I \in \Delta^I$*
- *Chaque nom de concept un ensemble $C^I \subseteq \Delta^I$*
- *Chaque nom de rôle R une relation binaire $R^I \subseteq \Delta^I \times \Delta^I$*

Cette définition formalise simplement le fait que les individus soient des objets, les concepts des ensembles d'objets et les rôles des relations entre objets. Du moment où on dispose de cette interprétation du vocabulaire, la sémantique des formules de concepts et des rôles peut être calculée. Un concept non atomique est interprété par le sous ensemble C^I de Δ^I tel que :

On note que cette sémantique explicite aussi la sémantique des constructeurs de la logique dans le domine d'interprétation Δ^I.

Rappel Une formule de type K est valide dans un modèle [5] $\mathcal{M}$, et on écrit $\mathcal{M} \vDash K$, si $K \not\equiv \bot$ et si K a une valeur dans Δ^I. L'application de la fonction d'interprétation vérifie si K a une valeur dans Δ^I ou non. La notation de validité est définie comme suit :

$$\begin{array}{lll} \mathcal{M} \vDash (s,t) : R & \text{si et seulement si} & \left(s^I, t^I\right) \in R^I \\ \mathcal{M} \vDash t : C & \text{si et seulement si} & t^I \in C^I \\ \mathcal{M} \vDash A \sqsubseteq C & \text{si et seulement si} & A^I \subseteq C^I \end{array}$$

[5] Un modèle est définit comme une interprétation avec un domaine non vide

$(\neg\ C\ )^I$	$= \Delta^I \setminus C^I$
$(C\ \sqcup\ D)^I$	$= C^I \cup D^I$
$(C\ \sqcap\ D)^I$	$= C^I \cap D^I$
$(\exists\ R.\ C)^I$	$= \{c \in \Delta^I \mid$ Il existe d $\in \Delta^I$ tel que $(c,d) \in R^I$ *et* $d \in C^I\}$
$(\forall\ R.\ C)^I$	$= \{c \in \Delta^I \mid$ Pour tout d $\in \Delta^I, (c,d) \in R^I$ implique $d \in C^I\}$

Table 4.4: Interprétation d'un concept

Exemple illustratif L'exemple présenté dans la figure (Figure 4.1) modélise une partie du modèle sémantique proposé dans le chapitre précédent. On considère une partie de sa formalisation dans les tableaux (Table 4.5 et 4.6).

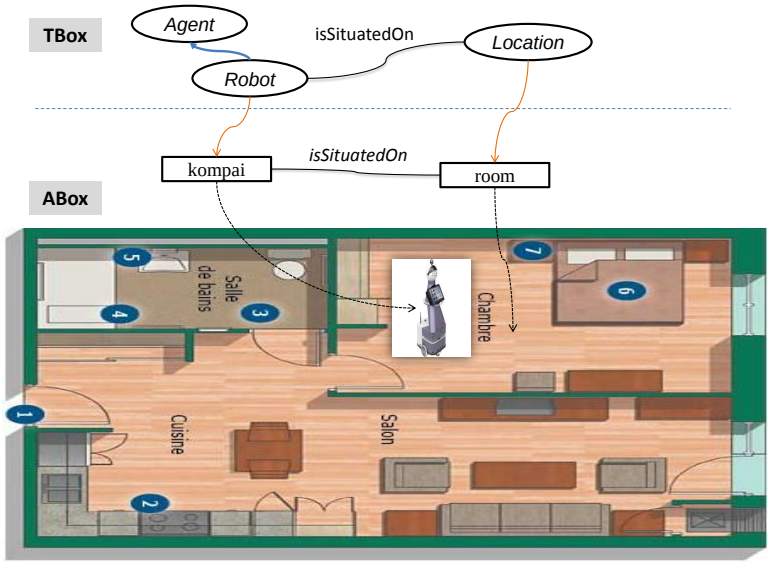


Figure 4.1: Exemple d'une description logique d'un environnement ambiant

$(Ax1)$: $NotificationAction \sqsubseteq \exists\ isPerformedBy.Robot$

$(Ax2)$: $Robot \sqsubseteq \forall\ isSituatedOn.Location$

Table 4.5: TBox : Formalisation de l'ontologie de Robot

Ces formules correspondent à des axiomes qui établissent le fait que le robot est un concept atomique représenté par le concept *Robot*. Un robot désigne une spécialisation du concept *Agent*. Le concept *Agent* modélise une entité abstraite de tout les agents de l'environnement. Un robot est associé à une position représentée par le concept *Location* via le rôle *isSituatedOn*. Le robot peut

ABox	
$(ax1)$: $kompai : Robot$	
$(ax2)$: $alert1 : NotificationAction$	$(ax3)$: $alert2 : NotificationAction$
$(ax4)$: $kitchen : Location$	$(ax5)$: $room : Location$
$(ax6)$: $(alert2, kompai) : isPerformedBy$	$(ax7)$: $(kompai, room) : isSituatedOn$

Table 4.6: Abox : Formalisation de l'ontologie de Robot

aussi exécuter des actions comme la notification, sachant que dans notre modèle sémantique un robot fournit des services diverses, comme la notification, le déplacement, la localisation d'un objet ou d'une personne, etc. L'action de notification est représentée par le concept $NotificationAction$, cette fonctionnalité est associée au robot par le rôle $isPerformedBy$.

Cette conceptualisation du domaine Table 4.5 schématise les types d'information et les concepts qui appartiennent au domaine d'application. Les concepts sont organisés dans des hiérarchies. L'ensemble ABox Table 4.6 instancie cette conceptualisation afin de décrire l'état de l'environnement et du système ainsi conçu. En effet, il donne les noms d'individus de l'environnement qu'on modélise. Par exemple les axiomes de l'exemple précédent affirment que dans l'environnement, deux actions de notification sont déclenchées par le robot *kompai* [6] représenté par le nom d'individu *kompai* à deux endroits différents, à savoir, la chambre et la cuisine. Chaque notification corresponde à une localisation du robot *kompai*. Si on remarque, le modèle (interprétation) dans lequel ces formules sont évaluées est représenté implicitement par la base de connaissance $\mathcal{T}$=(TBox, ABox). Par exemple, la validité d'une assertion affirmant que le robot *kompai* est dans la cuisine. Cette formule est représentée comme suit $(kompai, kitchen) : isSituatedOn$ n'est pas valide dans ce modèle, et on écrit : $\Delta^I \nvDash (kompai, kitchen) : isSituatedOn$ car aucune axiome ne justifie ce fait.

Cette description basée sur $\mathcal{BCDL}_0$ permet la formalisation de l'ontologie de coopération multi-domaine. Dans la suite de ce travail, cette représentation de connaissance est manipulée et exploitée par les méthodes et les techniques de preuve existantes afin de vérifier que le modèle sémantique spécifié est correcte. L'objectif étant de pouvoir raisonner et prouver formellement la correction de construction des conséquences logiques ainsi que des coopérations multi-domaine. Pour spécifier formellement et prouver des propriétés reliées à une spécification, des méthodes et techniques formelles ont été proposées dans la littérature dont un aperçu est donné dans la section suivante.

4.1.2 Méthodes et techniques formelles

Comme les systèmes deviennent de plus en plus complexes, il est crucial non seulement d'assurer certaines performances, mais également de garantir l'absence de risques, et d'un éventuel dysfonc-

[6] Kompaï est un robot fabriqué par la société RobotSoft

tionnement afin de limiter leurs impacts. Pour cela, de nombreuses méthodes formelles et techniques ont été développées durant les dernières décennies pour analyser ce type de systèmes.

Les méthodes formelles, sont des techniques basées sur les mathématiques pour décrire des systèmes et leurs propriétés. Elles fournissent un cadre rigoureux pour spécifier, développer, et vérifier des systèmes informatiques d'une façon systématique, plutôt que d'une façon ad-hoc, afin de démontrer leur validité par rapport à une certaine spécification. Ces méthodes permettent, d'une part, d'obtenir une très forte assurance de l'absence des incohérences, et d'autre part, de détecter des contradictions et des failles dans la conception aussi bien qu'à déterminer la correction de l'implémentation d'un système par rapport à sa spécification. Cependant, elles sont généralement coûteuses en termes de ressources, de temps et souvent réservées aux systèmes critiques.

Les méthodes formelles reposent sur l'utilisation des langages formels pour donner une spécification d'artefact du système que l'on souhaite développer à un niveau de détail désiré. Un *langage formel* est en effet, un langage doté d'une sémantique mathématique adéquate basée sur des règles d'interprétation et des règles de déduction. Les *règles d'interprétation* garantissent l'absence d'ambiguïté dans les descriptions produites, contrairement à des descriptions en langage informel ou semi-formel (e.g. UML)[7] qui peuvent donner lieu à différentes interprétations. Alors que les *règles de déduction* permettent de raisonner sur les spécifications afin de découvrir de potentielles incomplétudes, inconsistances, ou encore prouver des propriétés attendues. Après avoir spécifié le système via un langage formel, il existe des techniques d'analyse utilisant ces règles afin de raisonner sur le système spécifié dans le but de vérifier des propriétés de ce système. Parmi ces techniques d'analyse, on trouve principalement : la vérification formelle et la validation formelle.

- *La vérification formelle (Model Checking)*: La vérification formelle représente l'ensemble des actions de revue, d'inspection, de test, de simulation, de preuve automatique, aussi bien que d'autres techniques appropriées permettant d'établir et de documenter la conformité des artéfacts du développement vis-à-vis des propriétés préétablies. La vérification formelle a été définie comme un ensemble de techniques de vérification automatique de propriétés temporelles sur des systèmes à base de transitions (systèmes réactifs). Ces techniques prennent comme entrée une spécification d'un modèle qui représente une abstraction de son comportement et une formule formalisée avec une logique temporelle, et donne en sortie si l'abstraction satisfait ou non la formule. D'où l'origine du terme en anglais - Model Checking -. En cas de satisfaction, on dit que le système de transition est un modèle de la formule.

- *La validation formelle (Theorem Proving)* : Ces techniques appliquent des règles de déduction sur une spécification d'un modèle de système afin d'en tirer de nouvelles propriétés d'intérêt. En général, une technique de preuve de théorème consiste en un ensemble d'étapes d'inférence

[7] UML : http://www.uml.org/

qui peuvent être utilisées pour réduire un objectif de preuve à une liste de simples sous-objectifs. Ces derniers peuvent être prouvés automatiquement par les preuves primitives d'un assistant à la preuve (e.g. Coq, HOL4, Isabelle/HOL).

La technique qu'on va utiliser est celle de la validation formelle. Cette décision a été prise à l'issue de la discussion de la section suivante.

Discussion

On a vu que le but des deux méthodes formelles est de vérifier des propriétés d'une spécification d'un système. Ces méthodes sont largement utilisées dans la littérature dans différentes disciplines afin de spécifier des modèles et les propriétés que ces modèles sont tenus à garantir. A partir d'une spécification d'un modèle et d'une propriété, les méthodes de vérification formelle peuvent vérifier cette propriété. Cette vérification nécessite la modélisation du système comme un système à base d'état-transition. Une méthode de (Model Checking) vérifie tout les cas possibles des états par lesquels passera le système. Malgré l'avantage de ces méthodes d'être complètement automatiques, elles ne sont pas adaptées pour vérifier les systèmes à grandes dimensions. La limitation principale est que le système de transition doit être fini, c'est-à-dire, le système ne doit manipuler que des variables dans un domaine fini. A cet effet, ces méthodes sont limitées par la taille de l'espace d'états. Un espace de taille importante engendre le phénomène de l'explosion combinatoire du nombre des états du système. Par ailleurs, les systèmes de spécification à base d'état-transition peuvent entrainer une perte de la sémantique au cours de l'encodage du système.

Contrairement aux méthodes de vérification, la preuve de théorèmes présente des avantages par rapport aux méthodes de (Model checking), par le fait qu'elles ne sont pas limitées par la taille de l'espace d'états. Les systèmes à grandes dimensions qui ne peuvent être vérifiés en utilisant les méthodes du (Model checking), peuvent encore être vérifiés par un prouveur de théorèmes. Ces techniques nécessitent une intervention des experts afin de vérifier la propriété requise, ou bien de démontrer des sous-objectifs dans lesquels la propriété est violée. La technique de preuve de théorèmes est généralement plus difficile et demande beaucoup de savoir-faire technique et la compréhension de la spécification. Cependant, elle donne à l'utilisateur une plus grande souplesse pour établir les preuves. Cela peut donner à l'utilisateur un meilleur aperçu sur la spécification. Par ailleurs, l'utilisateur n'est pas laissé à lui-même en face d'une preuve. Des outils d'assistance à la preuve ont été développés et se sont beaucoup améliorés ces dernières années, par exemple, Coq, Isabelle, HOL light. Ces outils ont été conçus pour automatiser certaines tâches de simplification de preuves et de leur réécriture afin d'être plus lisibles et compréhensibles. En plus de la simplification des tâches difficilement menées dans certains cas. Ces outils d'assistance à la preuve ont été conçus afin d'augmenter la lisibilité, la compréhension ainsi que la manipulation des preuves. Par exemple,

le langage Isabelle/Isar de preuves structurelles est très proche du langages naturel.

Ce sont les principaux avantages de l'utilisation de la preuve de théorème. La manipulation des langages de spécification à base d'ontologies est un avantage des méthodes formelles basées sur la preuve de théorèmes. Cependant, la construction de preuves est basée sur des règles de déduction et des interprétations dites constructives, que les interprétations classiques des logiques de description ne les fournissent pas. D'où le besoin de chercher dans l'archive de l'école intuitionniste afin de donner des sémantiques aux différents concepts et aux constructeurs de la logique de description. De Paiva dans [de Paiva, 2006] a étudiée l'avantage de ces sémantiques, les bénéfices de ce type d'interprétations et leurs avantages dans la spécification et la preuve des propriétés des systèmes. Après cette motivation de De Paiva, de nombreux travaux sur des interprétations constructives ont été proposés ces dernières années [Mendler and Scheele, 2008]. En particulier, $\mathcal{BCDL}_0$ qui présente la sémantique constructive de la logique de description $\mathcal{ALC}$ basée sur les types d'information et les règles de déduction naturelles pour la construction de preuve [Ferrari et al., 2010, Bozzato and Ferrari, 2010b].

En conclusion, un rappel sur les logiques de description et sur les différentes techniques utilisées pour la spécification et la validation formelle de systèmes ont été présentés ainsi que la motivation de l'utilisation des logiques de description constructive pour la spécification formelle du modèle sémantique proposé. On a choisi l'assistant à la preuve Isabelle/HOL. Parmi les avantages de la mise en œuvre sur Isabelle/HOL, on peut notamment discuter le fait que le prouveur traite des aspects de spécification de processus en même temps que des informations concernant les aspects sémantiques. par ailleurs, l'outil dispose d'une méta-logique prouvée permettant de définir des langages de spécification et de profiter des techniques de preuves et de simplification de preuves fournies par l'outil. La spécification sur Isabelle/HOL correspond au système formel basé sur la logique $\mathcal{BCDL}_0$. Les avantages et l'usage de ce système formel pour la spécification du modèle sémantique proposé dans le précédent chapitre sont présentés dans la section suivante. Dans cette section, la logique de description constructive $\mathcal{BCDL}_0$, sa formalisation ainsi que celle du théorème de la fiabilité des règles d'inférences dans Isabelle/HOL seront introduits. On présente aussi le module de calcul pour la composition de services et la validation de tout le système sur l'outil Isabelle/HOL.

4.2 Spécification du système formel

Dans cette section, on présente le système formel qui servira comme langage formel de spécification du modèle sémantique proposé et de son implémentation sur l'assistant à la preuve Isabelle/HOL. Le système formel se base principalement sur la logique de description $\mathcal{BCDL}_0$. On va détailler donc la syntaxe et la sémantique constructive de la logique de description $\mathcal{BCDL}_0$ présentée dans

[Bozzato and Ferrari, 2010b], et son implémentation dans l'assistant à la preuve Isabelle/HOL. On note que l'intérêt de définir une sémantique constructive pour la logique de description est que cette sémantique fournit une interprétation constructive pour les preuves de formules dans $\mathcal{BCDL}_0$ conforme à celle des sémantiques classiques. L'interprétation classique d'une logique de description se base sur la notion de "vérité". C'est à dire, une formule est soit vraie, soit fausse. L'attribution de cette valeur de vérité à une formule ne dépend d'aucune analyse, de compréhension, ou raisonnement. Au contraire des logiques de description constructives qui jugent une formule par l'existence d'une construction appelée aussi preuve ou dérivation [Troelstra and Schwichtenberg, 2000]. Ces sémantiques constructives trouvent leur origine dans les résultats obtenus durant les tentatives de la formalisation des mathématiques. Vers la fin du 19 ^{eme} siècle, ces nouvelles interprétations ont vues le jour, principalement par les "constructivistes" [Miglioli et al., 1988]. Cette nouvelle vision des mathématiques considère que les objets mathématiques doivent être construits [Troelstra, 1999], que ce soit d'une manière mentale ou via un calcul. Par conséquent, les théorèmes affirmant l'existence de certains objets doivent via leurs preuve donner le moyen de construire ces objets dont l'existence avait été affirmée.

4.2.1 Vers une logique de description constructive

La sémantique constructive de la logique trouve ces racines dans la théorie des types λ-calcul inventé par Alain Church dans sa thèse en 1930. Les formules de cette logique sont considérés comme des types. Plus précisément, le type d'une formule K peut être vu comme une caractéristique d'une information dont on a besoin pour justifier sa valeur de vérité au sens classique. Cette sémantique est initialement introduite dans [Miglioli et al., 1988] où des types d'information sont associés à des formules afin de justifier leur validité. Bozzato [Bozzato and Ferrari, 2010b] a repris cette notion sous le nom de terme d'information pour représenter les états d'une formule. Le terme d'information construit la notion de base de la sémantique constructive de $\mathcal{BCDL}_0$[Bozzato et al., 2007]. Cette sémantique permettra de donner une interprétation constructive pour le calcul de preuves de déduction naturelle du système formel $\mathcal{BCDL}_0$.

Les règles d'inférence de ce système formel sont fournies par la liste de règles de déduction naturelle. Alors que les règles d'interprétation se basent sur la notion d'état encodé par un objet mathématique appelé *terme d'information* en anglais "Information terms" [Bozzato and Ferrari, 2010b]. Ces objets représentent la sémantique donnée aux formules afin de justifier leur validité dans un modèle classique. L'objet en question représente une structure mathématique qui contient les informations nécessaires permettant de juger la validité et par la suite la vérité d'une formule donnée dans un modèle classique. On précise que cet objet est calculé de manière inductive sur la structure des formules.

La sémantique des termes d'information est présentée plus loin ainsi que sa relation avec celle

des logiques classiques. Dans la suite de ce chapitre, on va présenter aussi, les règles de calcul de déduction naturelle et démontrer qu'elles sont fiables (Sound) et qu'elles respectent bien la sémantique des termes d'information. L'algorithme de calcul de cette sémantique est spécifié et implémenté sous Isabelle/HOL. Il permet le calcul mathématique des termes d'informations associés aux formules de la logique.

4.2.2 Syntaxe et Sémantique de la logique $\mathcal{BCDL}_0$

La syntaxe de $\mathcal{BCDL}_0$ est identique à celle des logiques de description $\mathcal{ALC}$. Elle se base sur les mêmes éléments et les mêmes constructeurs. Cependant, la sémantique donnée à ces entités ne se base pas sur une interprétation classique mais plutôt sur une interprétation constructive donnée par les termes d'information. Ainsi la logique $\mathcal{BCDL}_0$ fournit une interprétation constructive de la logique $\mathcal{ALC}$. Cette sémantique est présentée dans la section suivante.

L'implémentation sur Isabelle/HOL : Dans (Table 4.7) on présente la spécification sous Isabelle/HOL correspondant à la définition des différents ensembles de la syntaxe de $\mathcal{BCDL}_0$ et les formules représentées dans le tableau (Table 4.3).

```
datatype 'nr role = AtomR 'nr
datatype 'ni Individu = AtomN 'ni
datatype ('nr,'nc) Concept = AtomC 'nc
datatype ('nr,'nc) Concept = AtomC 'nc
        | NotC "('nr,'nc) Concept"
        | OrC "('nr,'nc) Concept" "('nr,'nc) Concept"
        | AndC "('nr,'nc) Concept" "('nr,'nc) Concept"
        | SomC "('nr) role" "('nr,'nc) Concept"
        | AllC "('nr) role" "('nr,'nc) Concept"
```

Table 4.7: La spécification de la syntaxe

```
datatype ('nr,'nc,'ni) Kformulas = Bottom
        | RoleF "'ni *'ni " "'nr role"
        | ConceptF "'ni" "('nr,'nc) Concept"
        | AConceptF "'nc" "('nr,'nc) Concept"
```

Table 4.8: La spécification des formules K

La sémantique est donnée par une interprétation. Dans le cas de la logique $\mathcal{BCDL}_0$. Cette interprétation est associée à chacune des formules via une fonction notée IT (voir Table 4.9). Cette fonction à comme ensemble de départ un sous ensemble $\mathcal{N}$ de NI. Elle associe donc à chaque nom d'individu de $\mathcal{N}$ des valeurs dans l'ensemble $it_{\mathcal{N}}(K)$. Cet ensemble désigne les termes d'information d'une formule K par rapport à $\mathcal{N}$. On peut faire l'analogie avec les types et la théorie des ensembles, et dire que $it_N(K)$ correspond à un type des éléments qu'il contienne. Autrement dit, un terme d'information noté note $it(K)$ d'une formule K est un objet mathématique structuré qui donne un témoin justifiant la "vérité" de la formule K. Le terme d'information, défini par rapport à un sous ensemble $\mathcal{N}$ tel que $N \subseteq NI$ est noté $IT_N(K)$. Les termes d'information sont calculés par rapport à l'état du système représenté par la Abox et la Tbox. On définit $\mathcal{L}_N$ comme étant le langage composé de la liste de formules générées K (voir Table 4.3) où les noms de variables d'individus ont des valeurs dans l'ensemble $\mathcal{N} \subseteq$ NI.

Formellement, si $\mathcal{N} \subseteq NI$ et K est *une formule close*[8], la liste des termes d'information $IT_{\mathcal{N}}(K)$ est définie par induction sur la structure de K comme suit :

$IT_{\mathcal{N}}(K)$	$=$	$\{tt\}$, ssi K est une formule close ou formule atomique
$IT_{\mathcal{N}}(c : C_1 \sqcap C_2)$	$=$	$\{(\alpha, \beta) \mid \alpha \in IT_{\mathcal{N}}(c : C_1)$ et $\beta \in IT_{\mathcal{N}}(c : C_2)\}$
$IT_{\mathcal{N}}(c : C_1 \sqcup C_2)$	$=$	$\{(k, \alpha) \mid$ et $\alpha \in IT_{\mathcal{N}}(c : C_k), k = 1, 2\}$
$IT_{\mathcal{N}}(c : \exists\ R.C)$	$=$	$\{(d, \alpha) \mid d \in \mathcal{N}$ et $\alpha \in IT_{\mathcal{N}}(d : C)\}$
$IT_{\mathcal{N}}(c : \forall\ R.C)$	$=$	$\{\phi : \mathcal{N} \to \bigcup_{d \in \mathcal{N}} IT_{\mathcal{N}}(d : C) \mid \phi(d) \in IT_{\mathcal{N}}(d : C)\}$
$IT_{\mathcal{N}}(A \sqsubseteq C)$	$=$	$\{\phi : \mathcal{N} \to \bigcup_{d \in \mathcal{N}} IT_{\mathcal{N}}(d : C) \mid \phi(d) \in IT_{\mathcal{N}}(d : C)\}$

Table 4.9: L'algorithme d'extraction des termes d'information

L'implémentation sur Isabelle/HOL : Dans l'annexe 4.10, on présente l'automatisation de l'algorithme d'extraction automatique des termes d'information. On note que Bozzato n'a pas proposé de calcul pour l'extraction des termes d'informations ni d'implémentation sur un "Theorem Prover".

[8]Une formule est dite close si elle ne contienne aucun nom de variable d'individu

```
fun IT :: "'ni set ⇒ ('nr,'nc,'ni) Kformulas ⇒ 'ni It set"
where
"IT (N) (ConceptF c A) = ITc N c A"
| "IT (N) (AConceptF c A)= ∪ { ITc N d (OrC (NotC (AtomC c)) A) | d. d ∈ N }"
| "IT (N) _ = {}"

fun ITc :: "'ni set ⇒ 'ni ⇒ ('nr,'nc) Concept ⇒ 'ni It set"
where
"ITc (N) c (AndC A B)= ( α,β) | α β. α ∈ ITc (N) c A ∧ β ∈ ITc (N) c B"
| "ITc (N) c (OrC A B) = { (k,α) | k α. ((k = 1) ∧ (α ∈ ITc (N) c A)) ∨
                    ((k = 2) ∧ (α ∈ ITc (N) c B)) } "
| "ITc (N) c (SomC R A)={Ex (d,α)|α d.d ∈N∧ α ∈ ITc (N) d A }"
| "ITc (N) c (AllC R A)= ∪  ITc N d A | d. d ∈ N "
| "ITc (N) c (NotC A) = tt"
| "ITc (N) c (AtomC A) = tt"
```

Table 4.10: La spécification de l'algorithme d'extraction des termes d'information

Dans la suite, on considère que l'ensemble $\mathcal{N}$ contienne tous les noms d'individus de la base de connaissance. La formalisation de la Table 4.9 est conforme à l'interprétation de BHK. Elle associe à chaque formule close ou atomique un type atomique, noté tt. Par exemple, $IT_{\mathcal{N}}(c : Robot) = \{tt\}$, où c est un élément de $\mathcal{N}$. Cet objet peut être simplement vu comme étant une référence d'un objet java de la classe $Robot.java$, ou d'une entrée dans un système de gestion de base de données contenant la table $Robot$. Si on considère la formule suivante, telle que $K \equiv Robot \sqcap \exists isLocatedOn.Location$ qui associe un robot à une localisation dans la maison intelligente. Le calcul de l'interprétation des termes d'information de K est le suivant :

Un exemple d'un élément $IT_{\mathcal{N}}(Ax1)$ est une fonction ϕ qui associe à chaque élément c de $\mathcal{N}$ un élément $\gamma \in IT_N(c : Robot \sqcap \exists isLocatedOn.Location)$. Ci-dessous on trouve le détail du calcul :

$$\begin{aligned} IT_N(K) &= IT_N(c : Robot \sqcap \exists isLocatedOn.Location) \\ &= (IT_N(c : (Robot)), IT_N(c : \exists isLocatedOn.Location)) \\ &= (tt, (d, tt)) \end{aligned}$$

Ce terme d'information exprimé par l'expression $(tt, (d, tt))$ signifie que c représente un robot et que d représente sa localisation. Dans ce cas, on a une structure qui associe un robot à un emplacement dans la maison intelligente. Maintenant, on dispose d'un objet qui justifie cette information de localisation et en plus il est obtenu de manière constructive par induction sur la structure de K.

4.2.3 Relation de réalisation

Les entités de la logique de description constructive ne sont pas des individus atomiques mais ils ont une structure interne. Ces structures sont considérées comme des témoins qui justifient la validité des formules au sens classique. Ces témoins correspondent donc à des réalisations de ces formules. Par exemple, Dans l'exemple précédent, l'expression $(tt, (d, tt))$ réalise la formule K. Ces réalisateurs (i.e IT) sont des extra-données de la Abox, c'est à dire au lieu que la validité de la formule soit basée uniquement sur les individus $\mathcal{M} \vDash K$ on déclare que cela signifie $\mathcal{M} \rhd \langle \alpha \rangle K$. On lit cette expression comme suit : α réalise la formule K dans le modèle $\mathcal{M}$, avec α représentant un terme d'information. Ceci dit, la réalisation qui donne une sémantique constructive additionnel aux concepts dans le sens où $\mathcal{M} \rhd \langle \alpha \rangle K$ implique $\mathcal{M} \vDash K$

Supposons que $\mathcal{M}$ est un modèle pour $\mathcal{L}_N$, K est une formule close et, $\eta \in IT_{\mathcal{N}}(K)$. La relation de *réalisation* est définie comme suit : $M \rhd \langle \eta \rangle K$ par induction sur la structure de K.

$\mathcal{M} \rhd \langle tt \rangle K$	ssi	$\mathcal{M} \vDash K$
$\mathcal{M} \rhd \langle (\alpha, \beta) \rangle c : C_1 \sqcap C_2$	ssi	$\mathcal{M} \vDash \langle \alpha \rangle c : C_1$ et $\mathcal{M} \vDash \langle \beta \rangle c : C_2$
$\mathcal{M} \rhd \langle (k, \alpha) \rangle c : C_1 \sqcup C_2$	ssi	$\mathcal{M} \vDash \langle \alpha \rangle c : C_k, k = 1, 2$
$\mathcal{M} \rhd \langle (d, \alpha) \rangle c : \exists\ R.C$	ssi	$\mathcal{M} \vDash (c, d) : R$ et $\mathcal{M} \vDash \langle \alpha \rangle d : C$
$\mathcal{M} \rhd \langle \phi \rangle c : \forall\ R.C$	ssi	$\mathcal{M} \vDash c : \forall\ R.C$, et $\forall\ d \in \mathcal{N}, \mathcal{M} \vDash (c, d) : R$ implique $\mathcal{M} \vDash \langle \phi(d) \rangle d : C$
$\mathcal{M} \rhd \langle \phi \rangle A \sqsubseteq\ C$	ssi	$\mathcal{M} \vDash A \sqsubseteq\ C$, et $\forall\ d \in \mathcal{N}$si $\mathcal{M} \vDash \langle tt \rangle d : A$ alors $\mathcal{M} \vDash \langle \phi(d) \rangle d : C$

Table 4.11: Relation de réalisation

L'implémentation sur Isabelle/HOL : La Table 4.11 présente la fonction qui calcule si dans un modèle un terme d'information réalise une formule donnée.

```
fun realizz ::
"('nr,'nc,'ni) Interp ⇒ 'ni It ⇒ 'ni ⇒ ('nr,'nc) Concept ⇒ bool"
where
"realizz (M) (et (α,β)) (c) (AndC A B) =
...
```

Table 4.12: Spécification de la relation de réalisation

Prenons l'exemple précédent, on peut précisément déduire l'état du système qui représente le modèle $\mathcal{M}$ par cette relation de réalisation. En effet, la constante tt réalise la formule $\mathcal{M} \rhd \langle tt \rangle c :$ *Robot* ssi $\mathcal{M} \vDash c :$ *Robot* admet tt comme réalisateur, donc on conclut que effectivement c'est le cas, suivant l'assertion *kompai* : *Robot*. Pour déterminer son emplacement, on cherche dans les assertions de la base de connaissance. On trouve que $\mathcal{M} \rhd \langle (d, tt) \rangle$ *kompai* : $\exists$ *isLocatedOn.Location* ssi $\mathcal{M} \vDash$

$(kompai, room) : isLocatedOn$ et $\mathcal{M} \vDash \langle tt \rangle\, room : Location$. L'extraction des termes d'information justifiant la vérité des formules sont construits de manière constructive en utilisant des règles d'inférence. Ces règles dites de déduction naturelle sont utilisées dans la preuve de validité de ces formules. L'interprétation de ces preuves conduit à extraire les termes d'information réalisateurs de la formule prouvée. Donc, la preuve construit les objets dont elle affirme l'existence. Dans la section suivante, on présente les règles de déduction naturelle de $\mathcal{BCDL}_0$.

4.2.4 Déduction naturelle

Dans la logique classique, on construit une dérivation formelle (une preuve) en analysant les prémisses et la conclusion, afin de montrer qu'il n'existe aucun cas possible ou les prémisses soient toutes vraies et la conclusion fausse. À la différence des systèmes de la logique classique, les systèmes de la logique constructive ne considèrent comme vrai que les formules prouvables par des termes typés. Par conséquent, l'objectif d'un système de preuve est de fournir un ensemble de règles suffisant pour vérifier toutes les formules et les dérivations. Le système ainsi utilisé dit *déduction naturelle*, a été initialement proposé par Gerard Gentzen [Troelstra and Schwichtenberg, 2000]. Dans ce système d'inférence, on a essentiellement deux règles minimum qu'on doit associer à chacun des constructeurs de la logique, à savoir, la règle d'introduction et la règle d'élimination.

Définition 4.2.1 *(Conséquence logique) Supposons que* Γ *est un ensemble de formules,* $\mathcal{M}$ *est un modèle, et* K *une formule. Si* Γ *est valide dans* $\mathcal{M}$*, et si* K *est valide dans* $\mathcal{M}$*, alors, on dit que* K *est une conséquence logique de l'ensemble de formule* Γ*.*

Les conséquences logiques sont obtenues par des preuves constructives en appliquant des règles en particulier les éliminations et les introductions. Dans ce système, la notion de preuve est centrale est conforme à l'interprétation BHK[9]. Une preuve est basée sur un ensemble d'hypothèses (i.e axiomes). Les conséquences logiques sont obtenues par des preuves constructives en appliquant des règles d'inférence de déduction naturelle. La méthode de raisonnement dans le système de déduction naturelle est plus proche de celles du raisonnement des mathématiciens. Obtenir une preuve d'une conclusion, consiste en une séquence de formules, qui commence par les prémisses, qui se termine par la conclusion et dans laquelle on passe toujours d'une étape à une autre en appliquant une règle logique pré-spécifiée. Les règles d'inférence sont reliées aux constructeurs de la logique $\mathcal{BCDL}_0$. Elles sont exprimées dans le style de la déduction naturelle.

Dans la Figure 4.2, les règles sont exprimées avec un contexte explicite représenté par Γ. Ce contexte contient dans un premier temps les axiomes ou les hypothèses qui figurent dans la base de connaissance. Ensuite, il est augmenté au fur et à mesure par les formules prouvées dans l'arbre de preuve.

[9] Brouwer-Heyting-Kolmogoroff

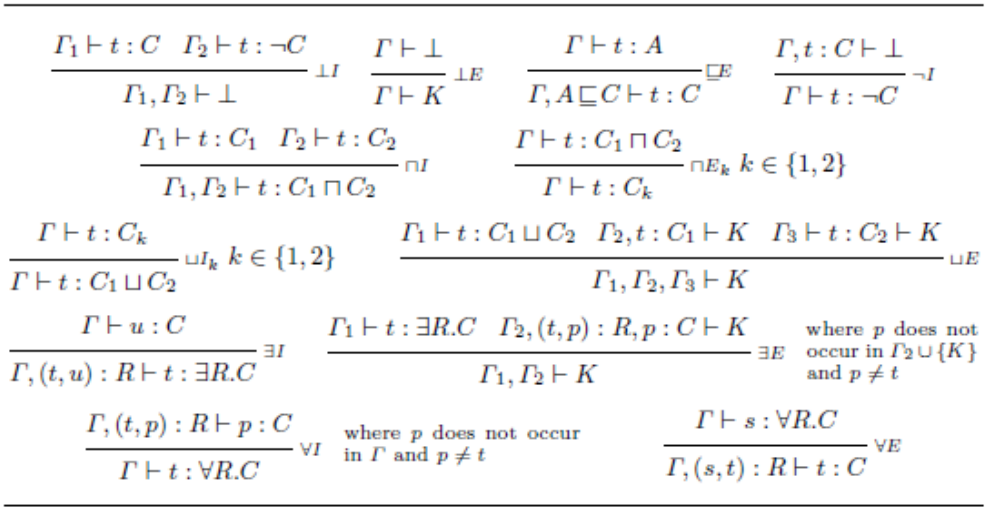


Figure 4.2: Règles de déduction naturelle [Bozzato and Ferrari, 2010b]

L'implémentation sur Isabelle/HOL : L'implémentation de ces règles de déduction sur l'outil Isabelle/HOL est détaillée dans la Table 4.13.

```
inductive deduction :: "('nr,'nc,'ni) Kformulas set
⇒ ('nr,'nc,'ni) Kformulas ⇒ bool" (infix "⊨" 28)
where
Bottom_elim: "Γ ⊨ Bottom ⟹ Γ ⊨ K"
| Not_elim: "Γ ∪ t:NotC H ⊨ Bottom ⟹ Γ ⊨ t:H"
| And_elim1: "Γ ⊨ t:AndC A B ⟹ Γ ⊨ t:A"
| And_elim2: "Γ ⊨ t:AndC A B ⟹ Γ ⊨ t:B"
...
```

Table 4.13: Formalisation des règles de déduction naturelle

Les règles de la figure (Figure 4.2) permettent de préserver la fiabilité d'une formule lors de leurs applications. Le but de l'application de ces règles est de déduire de nouveaux faits à partir d'un ensemble de formules.

L'algorithme de l'application des règles est le suivant :

- On considère un ensemble de lignes numérotées, où la première ligne contient la liste des

prémisses.

- Le passage d'une ligne à une autre ne peut se faire que par l'application d'une règle ; afin de vérifier que chaque transition est justifiée. On note la règle appliquée à droite de la ligne concernée.
- Si une formule "o" apparaît à une ligne quelconque n d'une preuve, il est toujours possible de répéter cette formule plus bas dans la preuve, à une autre ligne, en justifiant cette ligne par la règle de répétition.

La construction d'une preuve consiste à trouver une liste de dérivation qui permet de passer d'un ensemble de formules (les prémisses) à la conclusion. Pour cela, il faut trouver des règles permettant de transformer suffisamment ces prémisses de sorte que l'on parvienne finalement à écrire la conclusion. Pour chaque connecteur, on dispose de deux règles opérant ce type de transformation : une règle permettant d'éliminer le connecteur (une élimination) et une règle permettant de l'introduire (une introduction). Une preuve d'une formule K à partir d'un ensemble d'hypothèses Γ, est notée $\pi : \Gamma \vdash K$. En effet, la preuve est une séquence d'application des réglés de déduction.

Exemple d'une preuve avec la Tbox et la Abox Maintenant, on considère la définition d'un nouveau concept à partir des axiomes qui permettent d'identifier une localisation du robot en émettant une notification. Autrement dit, Le robot exécute une action de notification au moment où il se trouve dans une localisation bien déterminée. On considère la base de connaissance donnée dans les Tables 4.6 et 4.5. On peut construire la preuve suivant :

$\pi :: \mathcal{T} \vdash (NotificationAction \sqsubseteq \exists isPerformedBy.(Robot \sqcap \exists isLocatedOn.Location))$

On note que les noms d'individus ne se présente pas dans π. Supposons que $\mathcal{M}_w$ est un modèle :

$$\dfrac{\dfrac{\dfrac{Ax1}{z : \neg Notiofication \sqcup \exists isPerformedBy.Robot}\ \forall_{\sqcup}E \quad \dfrac{[y : \neg Notification]}{K}\ \sqcup I \qquad \begin{matrix} Ax2\ [y : \exists isPerformedBy.Robot] \\ \vdots\ \pi_1 \\ K \end{matrix}}{K \equiv \neg Notification \sqcup \exists isPerformedBy.(Robot \sqcap \exists isSituatedOn.Location))}\ \sqcup E}{\forall(y : \neg Notification \sqcup \exists isPerformedBy.(Robot \sqcap \exists isSituatedOn.Location)))}\ \forall_{\sqcup}I$$

où π_1 est la preuve

$$\dfrac{\dfrac{y : \exists isPerformedBy.Robot \qquad \dfrac{[(y,z) : isPerformedBy] \quad \begin{matrix} Ax2\ [z : Robot] \\ \vdots\ \pi_2 \\ z : Robot \sqcap \exists isSituatedOn.Location \end{matrix}}{y : \exists isPerformedBy.(Robot \sqcap isSituatedOn.Location)}\ \exists I}{y : \exists isPerformedBy.(Robot \sqcap \exists isSituatedOn.Location)}\ \exists E}{y : \neg Notification \sqcup \exists isPerformedBy.(Robot \sqcap \exists isSituatedOn.Location)}\ \sqcup I$$

and π_2 is the proof

$$\dfrac{\dfrac{Ax2}{z : \neg Robot \sqcup isSituatedOn.Location}\,\forall_{\sqcup}E \qquad \dfrac{\dfrac{z : Robot[z : \neg Robot]}{\dfrac{\bot}{H}\,\bot E}\,\bot I \qquad \dfrac{z : Robot[z : \exists isSituatedOn.Location]}{H}\,\sqcup E}{H \equiv Robot \sqcap \exists isSituatedOn.Location}\,\sqcup I$$

L'implémentation sur Isabelle/HOL : Dans la mise en œuvre sur Isabelle/HOL, cette preuve peut être spécifiée et prouvée dans la plupart du temps de manière automatique. Cela rend la logique utilisable car une preuve manuelle de toutes les conséquences logiques prendra beaucoup de temps. Dans ce qui suit, on montre les deux variantes de preuves (Figure 4.3 et 4.4), soit manuellement à l'aide des outils de simplification de l'assistant à la preuve Isabelle/HOL, soit d'une manière totalement et complètement automatique.

Dans les cas suivants, on va prouver le premier cas automatiquement par l'utilisation de module intégré à Isabelle/HOL, appelé *sledgehammer* [Blanchette et al., 2011], et le second cas manuellement.

- 1^{er} cas : La preuve en utilisant le module *sledgehammer*

```
lemma PROOOFf : "⟦
                Γ3 ⊢ConceptF y (OrC (NotC NotificationAction) (SomC isPerformedBy Robot));
                Γ2 ⊢ConceptF y (NotC NotificationAction); Γ1 ⊢ ConceptF y ( (SomC isPerformedBy Location))⟧ ⟹
                Γ3 ∪ Γ2 ∪ Γ1 ⊢ (ConceptF y (OrC (NotC  NotificationAction) (SomC isPerformedBy (AndC (Robot) (SomC isLocatedOn Location)))))"
apply (metis And_elim2 And_intro Or_intro1 sup.commute)
done
```

Figure 4.3: Preuve automatique en utilisant le module *sledgehammer*

- 2^{nd} cas : On procède à la preuve manuelle, comme suit :

```
lemma preuve: "⟦
          Γ1 ⊢ RoleF (y,z) isPerformedBy ;
          Γ2 ⊢(ConceptF z (AndC ( Robot) (SomC isLocatedOn Location)))⟧
          ⟹ Γ1 ∪ Γ2   ⊢ y: OrC (NotC NotificationAction) (SomC isPerformedBy (AndC Robot (SomC isLocatedOn Location)))"
apply (rule Or_intro2)
apply (rule Som_intro)
apply assumption
apply assumption
done
```

Figure 4.4: Preuve manuelle

Ces déductions ont été obtenues par l'application des règles de déduction sur les axiomes $Ax1$ et $Ax2$. Au moment où le système de déduction naturelle est utilisé dans la construction de preuve, il

est nécessaire de vérifier que l'application de n'importe quelle combinaison de ces règles ne produira jamais des déductions fausse à partir des hypothèse qu'on considère vraie à priori. Par conséquent, on a besoin de spécifier et de démontrer le théorème de la fiabilité dit en anglais "Soundness" du système formel. Dans la section suivante on présente le théorème de la fiabilité et son implémentation ainsi que sa preuve sur l'outil Isabelle/HOL. Ce théorème permet d'assurer que ces règles préservent la fiabilité d'une formule lors de leurs applications.

4.2.4.1 Théorème de la fiabilité des règles de déduction naturelle

Pour tout système formel de preuve, le théorème de fiabilité des règles d'inférence est extrêmement important. La propriété de fiabilité assure la consistance des dérivations lors de l'application des différentes règles du système d'inférence. Autrement dit, la fiabilité assure que la combinaison des règles du système d'inférence ne peut jamais conduire à une incohérence. Mathématiquement, si on considère que K est une formule de $\mathcal{BCDL}_0$ dérivée d'une liste de formule $\mathcal{BCDL}_0$ Γ suivant la preuve π. Pour tout modèle, les termes d'information réalisant l'ensemble des formules de Γ doivent être capables de justifier la réalisation de la conséquence logique et de donner le terme d'information qui la réalise.

Théorème 4.2.1 *(Fiabilité) : Soit $\mathcal{N}$ un sous ensemble fini de NI, et supposons que $\pi :: \Gamma \vdash K$ une preuve de $\mathcal{ND}$ sur $\mathcal{L}_{\mathcal{N}}$ telle que la formule K dans Γ est close, alors les lemmes suivants sont équivalents.*

1. $\Gamma \vDash K$
2. *Pour tout modèle $\mathcal{M}$ et $\gamma \in IT(\Gamma)$, $\mathcal{M} \rhd \langle\gamma\rangle\, \Gamma$ alors $\mathcal{M} \rhd \langle\phi^{\pi}_{N}(\gamma)\rangle\, K$*

L'implémentation sur Isabelle/HOL : La spécification du théorème et de sa preuve sur Isabelle/HOL se trouvent dans l'annexe Table 4.14.

Synthèse

On a présenté le système formel et son implémentation sur Isabelle/HOL. Des détails sur les aspects syntaxiques et les avantages des sémantiques constructives dans la preuve formelle de théorème ont été présentés. Le système formel développé sur Isabelle/HOL est fiable et on peut l'utiliser pour la spécification du modèle sémantique qu'on propose. Dans la suite, on procède à l'utilisation de ce langage formel pour spécifier l'ontologie de coopération multi-domaine. Dans la section suivante, on va développer l'ontologie de coopération avec le langage formel proposé. On donnera comme exemple l'ontologie de configuration des processus coopératifs dans les environnements ambiants multi-domaine.

theorem soundness: assumes "Γ ⊨ K" shows
" realiz_setF (M) (γ) (Γ) ⟶ realiz (M) (Φ (γ)) (K)"
using assms
proof induct
case (Bottom_elim Γ Bottom)then show ?case
apply auto
apply (rule realizz.induct)
...

Table 4.14: La preuve du théorème de fiabilité

4.3 Spécification de l'ontologie de coopération multi-domaine

L'ontologie de coopération présentée dans le chapitre précédent constitue la première brique du modèle sémantique proposé. Cette ontologie a été initialement conçue avec l'outil de conception des ontologies protégé[10]. Dans cette section, on procède à la spécification de cette ontologie Table 4.15. Vu la difficulté de l'encodage de toute l'ontologie sur l'outil de preuve. On a développé une API [11] en Java appelée, DL2Isabelle pour la transcription des expressions en logique de description OWL-DL conforme à $\mathcal{ALC}$ vers la syntaxe correspondante sous Isabelle/HOL. On présente ci-dessous l'ontologie de configuration/provisioning du modèle sémantique modélisée dans le chapitre précédent.

Méthodologie de transformation Dans l'outil Isabelle/HOL pour tenir compte du modèle sémantique, on le spécifie avec des formules K. Ensuite, ces formules sont utilisées pour extraire des termes d'information, et construire des preuves pour de nouveaux concepts, etc. L'ontologie coopération est encodé en utilisant le langage de modélisation d'ontologies, à savoir OWL [12]. Cette modélisation est effectuée sous l'outil *protégé* qui génère donc un fichier avec l'extension *.owl*.

Cette ontologie est ensuite analysée par l'API OWL [13]. On note que la spécification sous Isabelle/HOL peut être automatiquement générée en mettant en place une API de transformation simple, qui convertie les formules décrites dans le tableau (Table 4.15) vers la spécification sous Isabelle/HOL présentée dans l'annexe (Table 4.16). On a développé l'API DL2Isabelle qui exploite la partie d'analyse et génère un fichier qui représente la spécification en $\mathcal{BCDL}_0$ de l'ontologie de coopération. Ces fichiers ont comme extension ".*thy*". Le schéma de la figure (Figure 4.5)) illustre

[10] téléchargeable sur http://protege.stanford.edu/, version 3.4.8
[11] Application Programming Interface
[12] Ontology Web Language
[13] http://owlapi.sourceforge.net/

$Add \sqsubseteq ProvisioningMessage$	$AutomaticTask \sqsubseteq ProvisioningTask$
$AutomaticTask \sqsubseteq \neg ManualTask$	$Delete \sqsubseteq ProvisioningMessage$
$Lookup \sqsubseteq ProvisioningMessage$	$ManualTask \sqsubseteq ProvisioningTask$
$ManualTask \sqsubseteq \neg AutomaticTask$	$Modify \sqsubseteq ProvisioningMessage$
$ProvisioningAction \sqsubseteq Action$	$ProvisioningTask \sqsubseteq Task$
$ProvisioningTask \sqsubseteq \exists executedBy\ Domain$	$\exists executedBy\ Thing \sqsubseteq Task$
$Search \sqsubseteq ProvisioningMessage$	$\exists assignedTo\ Thing \sqsubseteq Action$
$\sqcap \exists hasHook\ Hook \sqcap \exists hasState\ TaskState$	$\top \sqsubseteq \forall executedBy\ Domain$
$\exists hasProvisioningMessage\ Thing \sqsubseteq ProvisioningAction$	$\exists hasPerformer\ Thing \sqsubseteq ManualTask$
$\top \sqsubseteq \forall hasPerformer\ Role$	$\exists hasState\ Thing \sqsubseteq Task$
$\top \sqsubseteq \forall hasProvisioningMessage\ ProvisioningMessage$	$\exists hasHook\ Thing \sqsubseteq Task$
$\exists hasTargetObject\ Thing \sqsubseteq ProvisioningAction$	$\top \sqsubseteq \forall hasState\ TaskState$

Table 4.15: TBox de l'ontologie de provisioning en $\mathcal{BCDL}_0$.

les entrées sorties des composants impliqués dans cette procédure de transformation. Cette API permet donc d'automatiser le passage de la spécification de cette ontologie vers une spécification syntaxique sous Isabelle/HOL.

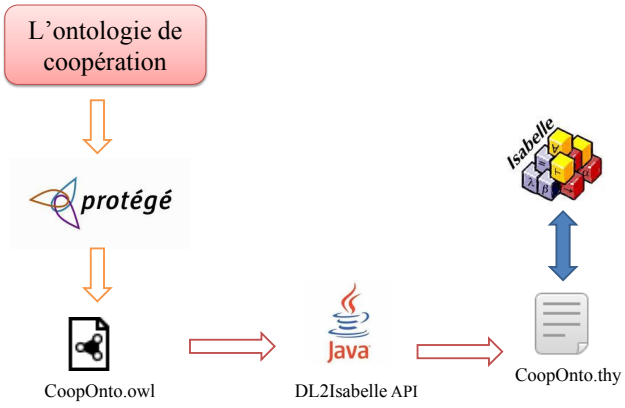


Figure 4.5: Modélisation et transformation de l'ontologie de coopération

Dans cette partie, on a donné la définition des formules qui constituent l'ontologie de coopération multi-domaine avec toutes les sous ontologies qui la compose. On a présenté le système formel qui se base sur la logique de description constructive $\mathcal{BCDL}_0$. On a également montré l'avantage de ces logiques de description dans la preuve et la technique de vérification formelle. L'implémentation du

NotifyProgramEvent	$\sqsubseteq$	(Event)
NotifyProgramEvent	$\sqsubseteq$	(AllC (hasReeducationProgram)(ReeducationProgram))
ReeducationProgram	$\sqsubseteq$	(AllC (isRelatedTo)(Person))
AcceptedProgramedEvent	$\sqsubseteq$	(AllC (hasEquipement)(MedicalEquipement))
ProgramedCalendarEvent	$\sqsubseteq$	(AllC (hasDate)(Date))
AcceptedEvent	$\sqsubseteq$	(AtomC Event)
AcceptedProvisioningEvent	$\sqsubseteq$	(AndC (ProvisioningEvent) (AcceptedEvent))
ReeducationProgram	$\sqsubseteq$	(AllC (hasDestination) (Person))
CalendarEvent	$\sqsubseteq$	(AllC (hasDescription)(Description))
ProvisioningEvent	$\sqsubseteq$	(AtomC Event)
ProvisioningEvent	$\sqsubseteq$	(AllC (hasAction)(ProvisioningAction))
ProvisioningAction	$\sqsubseteq$	(AllC (hasMessage)(ProvisioningMessage))
RefusedEvent	$\sqsubseteq$	(AndC (Event) (NotC (AcceptedEvent)))

Table 4.16: Spécification de l'ontologie avec le système formel

système formel proposé sous Isabelle/HOL permettra son utilisation et l'automatisation de plusieurs tâches de raisonnement sur les termes d'information qui seront construits en utilisant ce système. On dispose maintenant de notre modèle sémantique sur Isabelle/HOL. On rappelle l'objectif est la modélisation des processus coopératifs (services) dans un système d'intelligence ambiants multi-domaine et la preuve de leur correction. Dans la partie qui suit, on présente la description du modèle de spécification de service et de processus coopératifs, ainsi que la preuve de la correction des processus coopératifs dans un environnement d'intelligence ambiante multi-domaine.

4.4 Spécification du contrat de coopération multi-domaine

Le contrat de coopération se compose des templates de processus de provisioning, des règles de contrôle d'accès et d'usage et le modèle de composition des processus coopératifs. Ce modèle (§4.4.1) qui comprend la description des vues d'abstraction et des processus, les règles de composition et l'environnement de coopération. Chaque vue d'abstraction devrait satisfaire certaines critères dans cet environnement de coopération. Ces vues d'abstraction seront ensuite composer pour construire des processus coopératifs multi-domaines .

4.4.1 La spécification du modèle de composition

La description d'un processus coopératif (i.e. une composition de services) se base principalement sur la description des services/processus atomiques et les règles de composition (i.e règles de contrôle de flux). Dans cette partie, on présente la spécification formelle des vues d'abstraction des services/processus. Une spécification d'un processus est constituée de sa définition et de son

implémentation. Cette spécification correspond aux vues d'abstraction des services et processus concrets des différents domaines participant à la coopération dans l'environnement ambiant multi-domaine.

4.4.1.1 Définition d'une vue d'abstraction

Une spécification abstraite d'un service ou un processus est représentée par la description des pré-conditions et les post-conditions ou les effets du service. On va définir formellement différents services/processus qui opèrent dans un environnement ambiant multi-domaine. Un service/processus est définit formellement comme suit $S_{def} = (p, \Phi_p)$, où p représente la partie spécification du service, et Φ_s corresponds à l'implémentation du service s. On note par le couple $(p(x) :: P \Rightarrow Q, \Phi_p)$ (ou simplement (p, Φ_p)) constitue une définition d'un service sur $\mathcal{L}_{\mathcal{N}}$.

Spécification de la vue d'abstraction La spécification d'un service est formellement représentée par une expression de la forme $p\,(x) :: P \Rightarrow$ Q où p est une étiquette qui identifie le processus/service; x est le paramètre d'entrée du service (à être instancié avec un nom d'individus de $\mathcal{N}$), P et Q sont des concepts de $\mathcal{L}_N$. P explicite les pré-conditions du processus, notée par, $Pre(p)$ et Q représente les post-conditions (i.e. effets) du processus, notée par $Post(p)$.

Implémentation de la vue d'abstraction Une implémentation explicite le comportement d'un service/processus en terme de pre- et de post conditions. Une implémentation est modélisée par la fonction Φ_p telle que : $\Phi_p : \bigcup_{t\in\mathcal{N}} IT_{\mathcal{N}}(t : P) \rightarrow \bigcup_{t\in\mathcal{N}} IT_{\mathcal{N}}(t : Q)$. La fonction Φ_p représente une description formelle de l'implémentation de service. On note que la définition du service est basée sur l'ontologie de coopération multi-domaine.

Spécification du service
datatype ('nr,'nc,'ni) ServiceSpec =
SF "char " "'ni " "('nr,'nc)Concept" "('nr,'nc)Concept" ("_ _:: _ ⇒ _")

L'implémentation du service
definition Φ_s ::
"('nr,'nc,'ni)ServiceSpec ⇒ 'ni set ⇒ (('ni It set) ⇒ 'ni It set)"
where
"Φ_s S N = (let A =∪ { IT N (ConceptF d (Pre_s S)) | d. d ∈ N }
in (λ A. ∪ IT N (ConceptF d (Post_s S)) | d. d ∈ N)) "

Table 4.17: Définition de service

Résolution uniforme de l'implémentation d'une spécification On a besoin de vérifier la conformité de l'implémentation à la spécification. On définit $\mathcal{L}_{\mathcal{N}}$ comme un langage sur $\mathcal{N}$, une définition de processus $(p(x) :: P \Rightarrow Q, \Phi_p)$ sur $\mathcal{L}_{\mathcal{N}}$ et un modèle $\mathcal{M}$ pour $\mathcal{L}_n$. On dit que Φ_p *est une résolution uniforme de* $p(x) :: P \Rightarrow Q$, si et seulement si, pour chaque nom d'individu $t \in \mathcal{N}$, et chaque $\alpha \in IT_{\mathcal{N}}(t : P)$ tel que $\mathcal{M} \rhd \langle\alpha\rangle\, t : P$, $\mathcal{M} \rhd \langle\Phi_p(\alpha)\rangle\, t : Q$. La notion de correction de l'implémentation donnée, la spécification du service, est modélisée dans le théorème suivant :

Théorème 4.4.1 *(La résolution uniforme) : Soit un modèle* $\mathcal{M}$, Φ_s *est une résolution uniforme* $p\,(x) :: P \Rightarrow Q$ *dans* $\mathcal{M}$, *si et seulement si, pour chaque élément* $c \in \mathcal{N}$, *et chaque élément* $\alpha \in IT(c : P)$ *si* $\mathcal{M} \rhd \langle\alpha\rangle\ c : P$, *alors* $\mathcal{M} \rhd \langle\Phi_s(\alpha)\rangle\ c : Q$

Le théorème de résolution uniforme d'une implémentation d'une spécification dans un environnement représenté par le modèle $\mathcal{M}$ (i.e. l'interprétation) est implémenté sous l'outil Isabelle/HOL (Table 4.18).

```
fun Unformaly_solv :: "('nr,'nc,'ni) Interp ⇒ ('nr,'nc,'ni) ServiceDef ⇒ bool"
where
"Unformaly_solv (M) (S,Φ) =
(∀ N ::'ni set. ∀ t ∈ (N). ∀ α ∈ IT (N) (ConceptF t (Pre_s S)).
(realiz (M) (α)(ConceptF t (Pre_s S)) ⟶ realiz (M) (Φ( { α }))(ConceptF t (Post_s S)))
```

Table 4.18: Spécification du théorème de la résolution uniforme d'un service

Exemple : Modélisation d'un service de configuration Dans cet exemple, on donne l'exemple d'un processus de demande d'une action de configuration dans un domaine cible. Le nom du processus est *Request*. Il possède *action* comme paramètre d'entrée. Le processus *Request* représente une requête de configuration est une demande d'exécution d'une action. Il s'agit d'une action dans le domaine de destination *Domain*, et demande une exécution de l'action représentée par le concept *RequestAction*. Les post-conditions ou les effets de ce service/processus sont les réponses du domaine cible. Ce dernier, peut accepter l'action demandée. Cela crée une instance du concept de l'ontologie de coopération *AcceptedAction*, comme il peut la refuser, dans ce cas une instance du concept $RefusedAction$ avec le message expliquant le message de refus associé et représenté par une instance du concept $Message$ est créée. Ce processus de configuration est formalisé dans (Table 4.19).

Pour expliquer la sémantique du comportement de ce service, on suppose que, *action* est un nom de variable d'individu qui représente une action de configuration. L'implémentation du processus correspond à une fonction de correspondance de l'ensemble des termes d'information des

$Request(action)$ ::
$RequestAction \sqcap \exists\, hasTargetDomain.Domain$
$\Longrightarrow AcceptedAction \sqcup (RefusedAction \sqcup \exists\, hasMessage.Message)$

Table 4.19: Formalisation du processus de configuration *Request*.

pré-conditions vers l'ensemble des termes d'information des post-conditions. Ces fonctions formalisent le comportement de l'implémentation effective d'un service web. On considère par exemple l'implémentation $\Phi_{Request}$ du service *Request*. On suppose que *action_1* est un nom d'individu représentant une demande d'effectuer une action représentée par le concept *RequestAction*. L'entrée de ce service est tout terme d'information $\alpha \in IT_{\mathcal{N}}(action : Pre(Request))$. *action_1* peut être vu comme une référence d'une entée dans une base de données fournissant les informations requises par les pré-conditions du service et peut être vu comme une structure représentative de ce terme d'information. Supposons que α est de la forme suivante : α = (tt, (domaineA, tt)), ce terme signifie que l'information *action_1* est une demande d'action au domaine cible *domaineA*. Maintenant, on considère $\beta = \Phi_{Request}(\alpha) \in IT(action : Post(Request))$. β est l'image obtenue par l'application de la fonction sur α. Nous sommes donc devant deux cas :

- Si β = (1, tt), cette action est donc sera considérée comme acceptée
- Sinon, β pourrait être (2, (tt, (refusal_message, tt))). Cela signifie que l'action est refusée et précise qu'il y a un message $refusal\_message$ expliquant la raison du refus

Pour conclure, on remarque que pour le modèle $\mathcal{M}$ qui figure dans les formalisations précédentes utilisé dans l'évaluation de la correction du système est implicitement défini par la base de connaissance du système. En effet, $action : Pre(Request)$ est valide dans $\mathcal{M}$, si et seulement si, dans le système le nom d'individu, *action* peut codifier une demande d'action et désigne le domaine *domaineA* comme domaine cible. Dans ce cas, du moment où $\Phi_{Request}$ est une résolution uniforme de la spécification de service, alors, on a *action* : Post(Request) est valide dans $\mathcal{M}$. Ce qui implique le fait que, en analysant la base de connaissance, le domaine peut générer son acceptation.

4.4.1.2 L'environnement de coopération $E_{Cooperation}$

La composition de processus coopératifs est faite dans un environnement coopératif. Cet environnement contient toutes les vues d'abstraction des différents services et processus ainsi que les vues sur les nouveaux processus coopératifs. Ces derniers sont construis en combinant les vues d'abstraction disponibles dans l'environnement de coopération. Un environnement de coopération est représenté par une structure contenant les vues d'abstraction formalisées. Il est noté $E_{Cooperation}$. Dans le processus de la coopération, on a besoin de représenter les vues des différents processus et

services locaux afin de construire une preuve de nouveaux services et processus coopératifs. Afin d'atteindre cet objectif, on définit un environnement dit *environnement de coopération*, $E_{Cooperation}$. Cet environnement est caractérisé par sa vision sur les connaissances partagés ainsi que la liste des vues abstraites offertes par les différents domaines participant à la coopération. Formellement, on spécifie un environnement de la manière suivante: $E_{Cooperation} = \{\mathcal{L}_{\mathcal{N}}, T, \eta, (p_1, \Phi_1), \ldots, (p_n, \Phi_n)\}$ avec :

- $\mathcal{L}_{\mathcal{N}}$: La liste de formules générées ;
- T : La théorie dans $\mathcal{L}_{\mathcal{N}}$;
- $\eta \in IT_{\mathcal{N}}(T)$ l'ensemble des termes d'information de la théorie T ;
- (p_i, Φ_i) est une définition d'une vue abstraite sur un service ou un processus d'un domaine dans $\mathcal{L}_{\mathcal{N}}$, où i $\in 1, 2, \ldots, n$;

Dans cet environnement, on donne la spécification des objectifs correspondant aux processus coopératifs cibles. Ensuite on prouve la correction de cette spécification par rapport au modèle $\mathcal{M}$.

Implémentation sous Isabelle/HOL : La spécification de l'environnement de composition est comme suit :

type_synonym ('nr,'nc,'ni) Environment = "('nr,'nc,'ni) Kformulas set * 'ni It set * ('nr,'nc,'ni) ServiceDef set"

4.4.1.3 Spécification des règles de composition

Dans la méthodologie présentée dans le chapitre précédent. On a vu que le contrat de coopération établie les règles de contrôle de cette coopération et des processus coopératifs. Afin de construire les processus coopératifs, on se base sur les différentes vues des domaines participant à la coopération multi-domaine ainsi que sur un ensemble de règles de composition de flux. Dans cette partie on donnera une formalisation de la composition de service dans le framework de composition $\mathcal{BCDL}_0$ et on procède ensuite à la formalisation de chacune des règles de composition (Table 4.20). Finalement, la méthodologie pour construire un processus coopératif multi-domaine ainsi que la preuve sur sa correction.

Implémentation sur Isabelle/HOL Les règles de composition sont formalisées dans Isabelle/HOL comme dans la table 4.21. Dans cette partie on va modéliser les règles de composition de services. Ces règles nous permettent de composer manuellement des services tout en gardant la notion de correction. L'avantage principal est que notre modélisation nous permette de construire automatiquement les preuves des conditions d'applicabilités. Pour que notre service composé préserve

$\dfrac{p\,(x) :: P \Rightarrow Q}{\begin{array}{l}\Pi_1 : p_1\,(x) :: P_1 \Longrightarrow Q_1 \\ \Pi_2 : p_2\,(x) :: P_2 \Longrightarrow Q_2 \\ \vdots \\ \Pi_n : p_n\,(x) :: P_n \Longrightarrow Q_n\end{array}}$ *Sequence*	$AC = \left\{ \begin{array}{l} T, x : P \mid\!\!\overline{\;\mathcal{BCDL}_0\;} x : P_1 \\ T, x : Q_{k-1} \mid\!\!\overline{\;\mathcal{BCDL}_0\;} x : P_k, \text{for } k \in \{2, \ldots, n\} \\ T, x : Q_n \mid\!\!\overline{\;\mathcal{BCDL}_0\;} x : Q \end{array} \right.$
$\dfrac{p\,(x) :: P \Rightarrow Q}{\begin{array}{l}\Pi_1 : p_1\,(x) :: P_1 \Longrightarrow Q_1 \\ \Pi_2 : p_2\,(x) :: P_2 \Longrightarrow Q_2 \\ \vdots \\ \Pi_n : p_n\,(x) :: P_n \Longrightarrow Q_n\end{array}}$ *Parallel Split*	$AC = \left\{ \begin{array}{l} T, x : P \mid\!\!\overline{\;\mathcal{BCDL}_0\;} x : P_k, \text{for } \; k \in \{1, \ldots, n\} \\ T, x : Q_1 \sqcap \ldots \sqcap Q_n \mid\!\!\overline{\;\mathcal{BCDL}_0\;} x : Q \end{array} \right.$
$\dfrac{p\,(x) :: A \Longrightarrow B}{\begin{array}{l}\Pi_1 : p_1\,(x) :: P_1 \Longrightarrow Q_1 \\ \Pi_2 : p_2\,(x) :: P_2 \Longrightarrow Q_2 \\ \vdots \\ \Pi_n : p_n\,(x) :: P_n \Longrightarrow Q_n\end{array}}$ *Exclusive Choice*	$AC = \left\{ \begin{array}{l} T, x : P \mid\!\!\overline{\;\mathcal{BCDL}_0\;} x : P_1 \sqcup \ldots \sqcup P_n \\ T, x : Q_k \mid\!\!\overline{\;\mathcal{BCDL}_0\;} x : Q, \text{for } \; k \in \{1, \ldots, n\} \end{array} \right.$
$\dfrac{p\,(x) :: P \Longrightarrow Q}{\begin{array}{l}\Pi_1 : p_1\,(x) :: P_1 \Longrightarrow Q_1 \\ \Pi_2 : p_2\,(x) :: P_2 \Longrightarrow Q_2 \\ \vdots \\ \Pi_n : p_n\,(x) :: P_n \Longrightarrow Q_n\end{array}}$ *Synchronization*	$AC = \left\{ \begin{array}{l} T, x : P_1 \sqcap \ldots \sqcap P_n \mid\!\!\overline{\;\mathcal{BCDL}_0\;} x : P \\ T, x : Q_k \mid\!\!\overline{\;\mathcal{BCDL}_0\;} x : Q, \text{ for } \; k \in \{1, \ldots, n\} \end{array} \right.$
$\dfrac{p\,(x) :: P \Longrightarrow Q}{\begin{array}{l}\Pi_1 : p_1\,(x) :: P_1 \Longrightarrow Q_1 \\ \Pi_2 : p_2\,(x) :: P_2 \Longrightarrow Q_2 \\ \vdots \\ \Pi_n : p_n\,(x) :: P_n \Longrightarrow Q_n\end{array}}$ *Simple Merge*	$AC = \left\{ \begin{array}{l} T, x : P \mid\!\!\overline{\;\mathcal{BCDL}_0\;} x : P_k, \text{ for } k \in \{1, \ldots, n\} \\ T, x : Q_1 \sqcup \ldots \sqcup Q_n \mid\!\!\overline{\;\mathcal{BCDL}_0\;} x : Q \end{array} \right.$
$p\,(x) :: P \Longrightarrow Q \; Env$	Avec (p, Φ_p) est un processus définit dans $E_{Cooperation}$
$p\,(x) :: P \Longrightarrow Q \; AX$	$AC = \left\{ \; T, x : P \mid\!\!\overline{\;\mathcal{BCDL}_0\;} x : P_k, \text{ for } k \in \{1, \ldots, n\} \right.$

Table 4.20: Spécification formelle des règles de construction de processus coopératifs.

la notion de correction, l'environnement de composition doit être modèle. Si l'environnement est modèle cela veut dire quel que soit le service définit dans cet environnement, son implémentation est conforme à sa spécification.

les règles de compsition de services spécifié en tant que relation entre un environnement est le service compsosé, et à partir de cette spécification, l'application de ces règles garantie la correctness.

fun is_model_E :: "('nr,'nc,'ni)Interp ⇒ ('nr,'nc,'ni) Environment ⇒ bool" where
"is_model_E (M) (T,η, SETS) = ((realiz_setF (M)(η) (T)) ∧ (∀ (S,Φ) ∈ SETS. (Unformaly_solv (M) ((S,Φ)))))"
fun is_solvable:: " ('nr,'nc,'ni)Environment ⇒ bool "
where
"is_solvable (T,η, SETS) = (∀ S. ∀ M. ∃ Φ. (is_model_E (M) (T,η, (S,Φ))⟶ (Unformaly_solv (M) ((S,Φ)))))"
inductive SCER :: "('nr,'nc,'ni)Environment ⇒ ('nr,'nc,'ni)ServiceSpec ⇒ bool "
where
...
| Sequence: "SCER (E) (SF s1 x D F) ⟹ SCER (E) (SF s2 x G H)
⟹ (T∪ConceptF x A ⊢ ConceptF x (Pre_s (SF s1 x D F)))
⟹ (T∪ConceptF x (Post_s (SF s1 x D F)) ⊢ ConceptF x (Pre_s (SF s2 x G H)))
⟹ (T∪ConceptF x (Post_s (SF s2 x G H)) ⊢ ConceptF x B) ⟹ SCER (E) (SF _ x A B)"
| ParallelSplit: ...
| ExclusiveChoice: ...
| Synchronization: ...
| SimpleMerge: ...
| ENVE : ...
| AXE : ...
...

Table 4.21: Spécification de la règle de composition : La séquence

4.4.1.4 Formalisation du problème de composition

En général, une composition de services consiste à combiner une liste de services disponibles afin de construire des services et des processus plus complexes. Ces derniers ont pour but de satisfaire des requêtes dans les domaines ambiants. Une composition de services consiste à atteindre un objectif représenté par une spécification de service avec les pré- et les post- conditions. Composer les services disponibles pour satisfaire cet objectif est un problème de composition qu'on le formalise dans le contexte d'un environnement de coopération $E_{Cooperation}$ comme suit. Soit un modèle $\mathcal{M}$ de $\mathcal{L}_{\mathcal{N}}$, on dit que $\mathcal{M}$ est modèle pour l'environnement $E_{Cooperation}$, si et seulement si, $\mathcal{M} \rhd \langle\eta\rangle T$ et pour chaque service de l'environnement Φ_i est une résolution uniforme de p_i dans $\mathcal{M}$. Autrement dit, un service s_0 est dit *solvable* dans $E_{Cooperation}$, s'il existe une implémentation Φ_0 de s_0 telle que, pour chaque modèle de L_N. Si $\mathcal{M}$ est modèle pour $E_{Cooperation}$, alors Φ_0 résout uniformément s_0 dans $\mathcal{M}$.

$$\frac{p\,(x) :: P \Longrightarrow Q}{\begin{array}{l}\Pi_1 : p_1\,(x) :: P_1 \Longrightarrow Q_1 \\ \Pi_2 : p_2\,(x) :: P_2 \Longrightarrow Q_2 \\ \ldots \\ \Pi_n : p_n\,(x) :: P_n \Longrightarrow Q_n\end{array}}\ rule$$

Où

- $p\,(x) : P \Longrightarrow Q$ une spécification d'un service dans $E_{Cooperation}$
- *rule* est une des règles de composition de services
- Pour tout $i \in \{1, \ldots, n\}\,, \Pi_i : p_i(x) :: P_i \Longrightarrow Q_i$ est un processus composite dans $E_{Cooperation}$ qui satisfait les règles d'applicabilité de la règle *rule*.

4.4.2 Processus coopératifs multi-domaine et preuve de correction

On va modéliser les règles de composition de services introduites dans la section précédente en logique de description constructive. Ces règles de contrôle de flux sont détaillées ci-dessous ainsi que les conditions d'applicabilités (AC) Table 4.20. La preuve de ces conditions d'applicabilité implique l'exactitude de la composition, en d'autres termes elles assurent la correction du processus coopératif. Le modèle de composition représente le langage de composition. Il est constitué principalement des règles de contrôle de flux et l'environnement de coopération. Dans la suite, on présente la formalisation de ces différentes règles et la preuve de fiabilité correspondante. L'avantage principal de la spécification de ces règles sous Isabelle/HOL c'est la construire automatique des preuves des conditions d'applicabilités. Ces règles de composition correspondent étroitement aux définitions de concepts élémentaires de contrôle de flux fournis par le WfMC [14].

4.4.2.1 Preuve de correction d'un processus coopératifs

Le principe de la spécification des règles des composition sous Isabelle/HOL consiste en une relation entre l'environnement de coopération et le service composé. Pour qu'un service/processus composé préserve la notion de "correctness" (i.e. Fiabilité), notre environnement doit être modèle. Si notre environnement est modèle cela veut dire quel que soit le service définit dans cet environnement, son implémentation est une résolution uniforme de sa spécification. Ce qui est vérifié par la relation suivante :

l'implémentation du service la spécification de ce service. Si notre environnement est modèle, on a aussi notre modèle réalise la TBox. Donc, toutes les formules de la TBox sont prouvées valide avec la relation suivante : $\mathcal{M} \rhd \langle \eta \rangle$ T. Comme ces formules sont utilisées dans les conditions d'applicabilités, ces conditions sont prouvées justes. C'est-à-dire, n'importe quelle formule déduite à partir de ces conditions d'applicabilité est valide. Et comme les formules déduites à partir de ces conditions d'applicabilités sont la pré-condition et la post-condition de notre service composé, notre service composé préserve la notion de "Correctness". Pour confirmer cela, il suffit de calculer les conditions d'applicabilité et les comparer avec les pré-conditions et post-condition du service composé. On dit que $\mathcal{M}$ est modèle pour un environnement $E_{Cooperation}$ si et seulement si, $\mathcal{M} \rhd \langle \eta \rangle$ T et pour chaque i : 1,...,n, Φ_i uniformément résout s_i dans M. Formellement, on pose la problématique de composition de service comme suit :

une spécification de service s' est solvable dans E s'il existe une implémentation Φ' de s' telle que, pour chaque modèle $\mathcal{M}$ de $\mathcal{L}_{\mathcal{N}}$, si $\mathcal{M}$ est modèle pour E alors Φ' est une résolution uniforme de s' dans $\mathcal{M}$.

Théorème 4.4.2 *(La résolution uniforme d'un service composite) : Soit un modèle $\mathcal{M}$, Φ_s uni-*

[14] Workflow Management Coalition

formément résout $p(x) :: P \Longrightarrow Q$ *dans* $\mathcal{M}$*, si et seulement si, pour chaque élément* $c \in N$*, et chaque élément* $\alpha \in IT(c{:}P)$ *si* $\mathcal{M} \rhd \langle \alpha \rangle\rangle$ *c:P alors* $\mathcal{M} \rhd \langle \Phi_s(\alpha)) \rangle$ *c:Q*

Implémentation sous Isabelle/HOL : La spécification de ce théorème sous Isabelle/HOL est la suivante :

```
fun Unformaly_solv :: "('nr,'nc,'ni) Interp ⇒ ('nr,'nc,'ni) ServiceDef ⇒ bool"
where
"Unformaly_solv (M) (S,Φ) = (∀ N ::'ni set. ∀ t ∈ (N). ∀ α ∈ IT (N) (ConceptF t (Pre_s S)).
(realiz (M) (α)(ConceptF t (Pre_s S)) ⟶ realiz (M) (Φ( { α }))(ConceptF t (Post_s S)))
```

Table 4.22: Spécification du théorème de résolution uniforme d'une service composite

Considérant un exemple de construction d'un processus de coopération avec les deux services $NotifyService$ et $CalendarEvent$. Ces deux services sont composés via une séquence dont la preuve de correction est donnée Table 4.23.

Maintenant l'idée est comment construire une implémentation d'une spécification de service à partir de l'environnement de composition.

Ce calcul permet de composer manuellement des services en garantissant la *correctness* du service composé. Une règle est composée d'un séquent principale; des sous séquents, ainsi que des conditions d'applicabilité. Ces conditions d'applicabilité décrivent le rôle des sous séquents dans la composition. Dans la formalisation des conditions d'applicabilité de symbole $\vdash$ signifie une dérivation dans le système de déduction naturel. Pour les CI (les interprétations de calcul) des règles permettent d'associer avec la composition de service Π. Dans les règles CI, étant donnée la condition d'applicabilité (a) $\Gamma \vdash$ x:A de la règle r; on note Φ_a l'opérateur correspondant de la preuve π de $\Gamma \vdash$ x:A. On conclut la définition de la (soundness) fiabilité des règles avec le respect de la résolution uniforme [Bozzato and Ferrari, 2010a].

Théorème 4.4.3 *(Soundness). Soit* $E_{Cooperation} = (L_N, s_i)$ *un environnement et s(x):: P* $\Longrightarrow$ *Q le service composite de la composition* Π *sur E. Pour chaque modèle M, si M est un modèle pour* $E_{Cooperation}$ *alors* Φ_s *extrait de* Π *résout uniformément s dans* $\mathcal{M}$

```
theory Composition
imports NotifyService CalendarEvent
begin
lemma "⟦ SCER (E)(SF NotifyService x e g);
SCER (E)(SF S2 x h j)
; (T ∪ ConceptF x A ⊢ ConceptF x (Pre_s (SF NotifyService x e g)))
; (T ∪ ConceptF x A ⊢ ConceptF x (Pre_s (SF CalendarEvent x h j)))
; (T ∪ ConceptF x (AndC (Post_s (SF NotifyService x e g)) (Post_s (SF CalendarEvent x h j))) ⊢ ConceptF x (B))
⟧
⟹ SCER (E) (SF SS x A B) "
apply (metis AND)
done
```

Table 4.23: Compsition de service avec une règle de séquence

4.5 Impact du changement d'une composition sur la preuve

Dans cette section, on étudie l'impact de l'adaptation d'une composition de services sur la preuve de la propriété de correction. On commence par présenter les étapes par lesquelles on construit et on prouve une composition dans l'assistant Isabelle/HOL.

Algorithme de validation sur Isabelle/HOL . Cette algorithme consiste en trois étape. La première étape, a pour objectif de prouver que chaque service préserve la notion de correction. La seconde étape consiste à prouver que l'état η réalise la TBox T. C'est-à-dire prouver la relation suivante : $\mathcal{M} \rhd \langle\eta\rangle\rangle$ T. Dans la troisième étape, on construit le processus de composition comme l'exemple modélisé dans la Figure 4.6. Dans ce schéma de composition, le but est de composer le service *Process_1*. Ce dernier est composé à partir de deux autres services $S12$ et $S34$. Ces deux derniers ont été composés grâce aux services atomiques qui se trouvent dans l'environnement E. Chaque composition est réalisée avec des règles de composition. Les règles utilisées dans notre exemple sont *Parallel Split*, *Exclusive Choice* et *Sequence*. Pour chaque règle, il existe des conditions d'applicabilités, la règle ne peut pas être appliquée sans que les conditions d'applicabilités soient satisfaites.

La preuve d'une re-configuration de services Ce paragraphe présente l'impacte d'une re-configuration de la structure de composition d'un services sur la preuve de correction. Dans cette exemple, on présente l'effort nécessaire pour la preuve d'un re-configuration d'une composition. Supposons qu'on construit le service composite, *Processus_1* à partir de cet environnement E Figure 4.6. Dans cet environnement, on disponse des services atomiques S_i ou i=1,2,3 et 4. Le but est

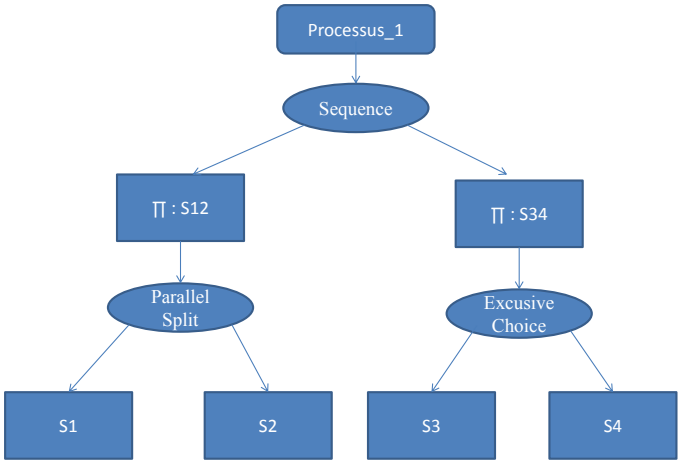


Figure 4.6: Schéma de composition du service *Process_1*

de composer le service *Processus_1* à partir de cet environnement. Ce service est une séquence des deux services $S12$ et $S34$. $S12$ représente un appel parallèle aux deux services $S1$ et $S2$. Alors que, $S34$ représente un choix exclusif entre les services $S3$ et $S4$. Maintenant, supposant qu'on va effectuer une re-configuration de la structure de composition du service composite par la modification du services $S12$ en mettant les deux services $S1$ et $S2$ dans une séquence au lieu d'une composition en parallèle. Ceci aura l'effet suivant sur la composition globale. Si on change une règle de composition, il faut juste reprouver que les conditions d'applicabilités de la règles sont satisfaites. Si on change un service il faut reprouver la notion de correction. Si on change l'environnement il faut refaire toutes les étapes de l'algorithme de validation sur Isabelle/HOL. Par conséquent, la preuve de cette nouvelle composition n'affectera que les règles de composition ont été modifié par cette re-configuration. Dans ce cas, les règles d'application associées aux *Parallel Split* et *Sequence* doivent être prouvées de nouveaux.

Conclusion

Dans ce chapitre, on a présenté le système formel utilisé dans la spécification formelle du modèle sémantique proposé. La spécification de ce système et son usage ainsi que sa fiabilité ont été implémentés sous l'outil Isabelle/HOL. Ce système formel a ensuite été utilisé pour la spécification du langage de composition. Pour ce dernier la preuve formel de sa correction et une méthodologie pour prouver les processus coopératifs ont été présentés. Finalement, un exemple pour illustrer ces deux grandes parties et la démonstration de l'impact d'une adaptation dans la composition sur la preuve formelle de tout le processus a aussi été présentée. Dans le chapitre suivant, on s'intéresse à

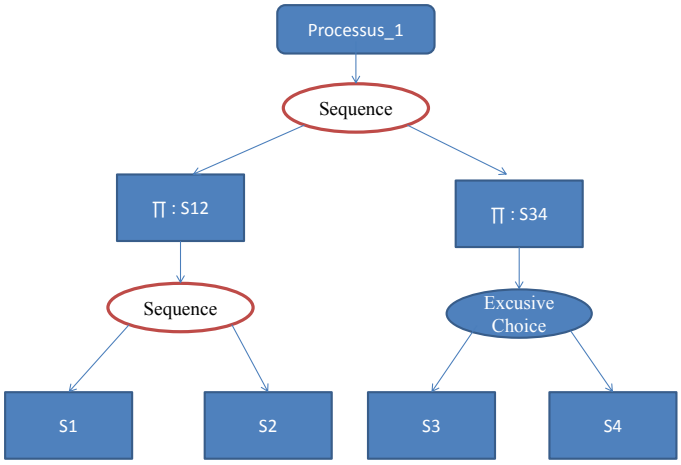


Figure 4.7: Re-configuration du Schéma de composition du service *Process_1*

la partie implémentation et expérimentation de ce canevas via la mise en œuvre d'un scénario dans le domaine de la santé. L'application concerne une application d'assistance à l'auto-rééducation à domicile des personnes âgées .

5

Implémentation et expérimentation

Sommaire du chapitre

5.1 Introduction

Dans ce chapitre, on va étudier la faisabilité de ce canevas pour le développement des applications dans un environnement d'intelligence ambiante multi-domaine. Ces applications se basent essentiellement sur la définition de services et de leur composition dans des contextes particuliers afin de répondre à des besoins utilisateurs. Ce chapitre présente la mise en œuvre de notre étude de cas développée au sein du laboratoire LISSI[1] en collaboration avec l'hôpital UPEC[2] Henry Mondor. Cette étude permettra la validation de la méthodologie et le canevas sémantique proposé dans les chapitres trois et quatre. Ce chapitre, présente aussi les aspects techniques et technologiques ainsi que les outils d'implémentation pour la mise en œuvre de cette étude de cas. Cette dernière est représentée par un scénario dans le domaine médical. Ce scénario, concerne la coopération entre plusieurs domaines ambiants pour faire réussir la gestion des programmes d'auto-rééducation des personnes âgées à domicile.

Le chapitre est organisé comme suit. Dans une première partie du chapitre, on présente le scénario de validation. Une présentation de l'ensemble des services abstraits (i.e. les vues d'abstraction) impliqués dans ce scénario sont résumés dans un démonstrateur. Le démonstrateur explicite les interactions entre ces services abstraits, et en particulier le service de provisioning multi-domaine. On présente aussi les technologies et les standards utilisés pour la mise en œuvre des services concrets correspondant aux vues d'abstraction et de leur composition. Dans une deuxième partie du chapitre, on procède à la mise en œuvre de ce scénario suivant le canevas et la méthodologie présentés dans le chapitre trois et en utilisant le système formel proposé dans le chapitre quatre. En effet, on commence par une implémentation de l'ontologie de coopération multi-domaine et sa spécification formelle. On explicite l'extension de cette ontologie pour répondre à des nouvelles exigences imposées par le scénario médical. On montrera à travers cette spécification l'agilité de l'ontologie proposée pour intégrer des adaptations et des évolutions futures. Par la suite, on va spécifier les processus de coopération et prouver leur correction dans l'assistant à la preuve Isabelle/HOL. Enfin, on va discuter de l'établissement des correspondances entre les services abstraits et les processus internes aux domaines coopérants.

5.2 Scénario de validation

L'étude de cas concerne la mise en place d'une application de rééducation des personnes âgées à domicile. Cette application est réalisée par la coopération de plusieurs domaines à intelligence ambiante, à savoir, la maison intelligente, l'hôpital intelligent et le centre de kinésithérapie (Figure 5.1). Les responsabilités de chacun de ces domaines sont réparties comme suit :

[1] Laboratoire Images, Signaux et Systèmes Intelligents
[2] Université Paris-Est Créteil

- La maison intelligente est responsable sur l'acquisition de données et d'actions, par la proposition des services ubiquitaires capteurs et actionneurs ;
- L'hôpital est responsable sur les patients en état d'auto-rééducation ;
- Le centre de kinésithérapie est responsable sur les exercices de rééducation et de l'intervention à domicile en cas de besoin.

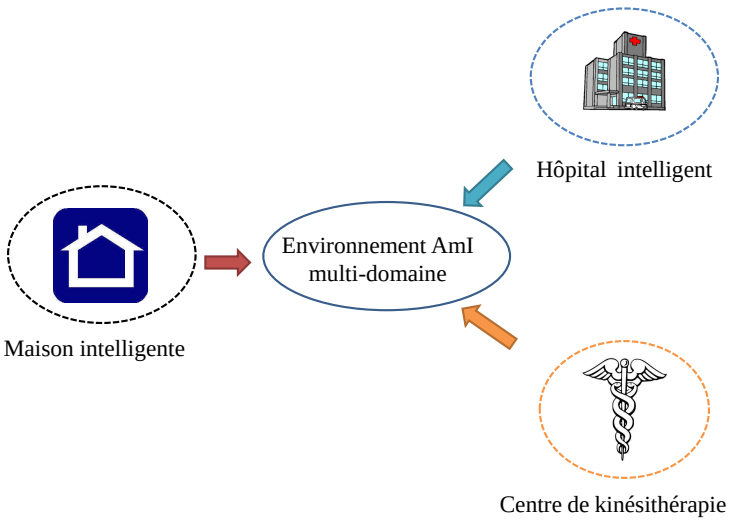


Figure 5.1: Étude de cas : Les domaines participant à l'environnement AmI multi-domaine

Dans cette étude de cas, le patient se déplace à l'hôpital pour se faire opérer ou se soigner par un médecin. Le médecin lui prescrit un programme d'auto-rééducation initial. Ce programme se compose d'une liste d'activités physiques. Le médecin de rééducation physique prescrit les programmes et maintient la mise à jour du dossier du patient. Le dossier médical est considéré comme une ressource de la maison intelligente et donc externe aux domaines, l'hôpital et le centre de kinésithérapie. Le centre de kinésithérapie après avoir reçu et accepté de suivre le patient. Il lui délivre le matériel nécessaire pour le programme prescrit, par exemple : une barre; un ballon; des tables; des bandes élastiques; etc. Le centre s'occupe aussi de la préparation de l'environnement de rééducation, des services de contrôle des conditions de confort ainsi que de la sécurité du patient. Par exemple : mettre des détecteurs de chute, configurer des rappels de conditions qui précède un programme, telles que, la prise de médicament, ou la vérification de la température ambiante de l'endroit des exercices.

Après avoir programmé les activités de rééducation. Des services de la maison intelligente s'occupent de donner l'accès pour l'ajout des horaires d'activités de rééducation au calendrier du patient en respectant certains paramètres de contexte. Afin de déclencher l'appel au programme,

ces paramètres du contexte doivent être réunis et favorables. Par conséquent, à l'instant où le contexte correspond à ce que le médecin a préconisé, la maison intelligente sollicite les services de son robot compagnon. Le robot localise la personne en utilisant le service de localisation de la maison intelligente et il se déplace vers elle afin de lui proposer le programme. Dans le cas où le patient n'est pas localisé dans la maison intelligente et que l'exercice est obligatoire à cet instant, une notification est envoyée au médecin afin d'organiser ou de proposer un autre créneau. Dans l'autre cas, le robot retrouve le patient et atteint sa position. Le robot propose via son service vocal l'exercice et projette une séquence vidéo sur son écran. Le patient se prépare et annonce son état (prêt) pour le robot. Le robot démarre son plan d'accompagnement du patient. Si un incident arrive, par exemple, une chute ou un malaise, une notification est envoyée au service du secours au centre de kinésithérapie pour une intervention urgente. Dans le cas contraire, le robot pose des questions, ou affiche un formulaire à remplir par le patient pour chacune des activités du programme de rééducation. A la fin de toutes ces étapes le patient est notifié par émail, ou via le service vocal du robot pour remplir un questionnaire global sur les conditions et le déroulement du programme. L'hôpital recevra ensuite les résultats du programme, mettra à jour le dossier médical du patient, et notifie le médecin par émail. Suivant les résultats, le médecin peut prendre des décisions, telles que, organiser une visite pour le patient, ordonner un déplacement d'un kinésithérapeute pour assister le patient, ou de lui prescrire de nouveaux programmes.

Afin de valider notre contribution et de l'évaluer, on a utilisé la plate-forme ubiquitaire de notre laboratoire qui représente un environnement d'intelligence ambiante. Dans la section suivante, on présente les interactions entre les différents domaines, et les différents processus coopératifs en particulier le processus de provisioning inter-domaines.

5.2.1 Présentation de l'infrastructure logicielle d'implémentation

La plateforme ubiquitaire est constituée d'un ensemble de capteurs, d'actionneur et un robot. Cette plate-forme fait la base de validation du système formel proposé et la méthodologie développés dans les chapitres 3 et 4. Le scénario a été implémenté en exploitant cette plateforme qui comprend différent composants :

- Un robot compagnon ;
- Des dispositifs d'affichage : téléphone portable de type Smartphone ; moniteur de PC, écran d'affichage du robot compagnon ;
- Un système de localisation "*indoor*" ;
- Un ensemble de capteurs permettant d'observer des évènements liés au contexte d'une personne âgée à domicile :

1. Un lecteur RFID longue porté, (de 1 à 8 mètres) permettant l'identification d'une entité de l'environnement. A chaque entité est associée un tag actif. Ce lecteur, embarqué sur le robot compagnon, permet aussi d'estimer la position grossière de cette entité en exploitant le système de localisation du robot ;
2. Un lecteur RFID courte portée (quelques centimètres) porté au poignet d'une personne permet d'identifier une entité et d'établir sa proximité par rapport à la personne. A chaque entité est associée un tag passif ;
3. Des capteurs de détection d'ouverture de portes, de détection de présence, de mesure d'intensité lumineuse et de température ;
4. Un bracelet permettant de détecter la chute d'une personne, de mesurer son pouls et de remonter une alarme par un appui sur un bouton d'urgence ;
5. Un actionneur de type TOR (Tout Ou Rien) permettant d'actionner une prise de courant à distance et ainsi de mettre en marche un équipement ;
6. Des actionneurs dédiés permettant d'allumer/d'éteindre des ampoules électriques à distance

Dans ce qui suit, on donne une description synthétique de chacun des composants de cette plateforme.

5.2.1.1 Le robot compagnon Kompai

Le robot Kompai développé par la société Robosoft [3], est équipé de plusieurs capteurs et actionneurs permettant d'assurer des fonctions essentielles telles que la planification de trajectoire, la navigation en environnement encombré, la cartographie, la localisation, l'interaction pour des tâches d'assistance au quotidien (Table 5.1).

Le robot Kompai embarque également une importante partie logicielle qui assure des fonctions de contrôle de bas niveau, jusqu'aux services de haut niveau à destination de l'utilisateur final. Ainsi, le robot propose des services tels que l'agenda, la messagerie Skype, la gestion des mails, la possibilité de faire ses courses en ligne, le contrôle par reconnaissance vocale, etc.

5.2.1.2 Système de localisation indoor

Cricket est un système de localisation indoor conçu à l'origine par le MIT (Figure 5.2). Il s'agit d'un réseau de capteurs sans fil basse consommation qui fournit deux informations de localisation. La première représente l'identifiant de l'espace où se trouve l'entité à localiser : Cuisine; Séjour; Chambre; etc. La seconde information représente la position courante de l'entité en coordonnées

[3] http://www.robosoft.com/

Deux caméras ;
Une tablette-PC ;
Un télémètre laser ;
Des capteurs ultra-son ;
Des capteurs infrarouges ;
Des détecteurs de contact ;
Des moteurs ;
Une batterie ;
Une unité de contrôle embarquée.

Table 5.1: Les caractéristiques du robot compagnon Kompai

cartésiennes x y z. La façon la plus courante d'utiliser le système de localisation Cricket, consiste à déployer des balises de transmission *Cricketbeacons* sur les murs ou les plafonds, et d'attacher la balise d'écoute *Cricketlistener* à un objet mobile (Personne; Robotmobile, etc.) dont la localisation doit être déterminée. La position de l'objet est ensuite estimée à partir de la mesure de la différence entre le temps de propagation des ondes radio-fréquence et des ondes ultra-sonores. A chaque fois que la balise d'écoute intercepte une information des balises de transmission, elle infère les cordonnées de sa position en se basant sur les distances par rapport aux balises de transmission (dont les positions sont connues à priori).

Figure 5.2: Le capteur de localisation indoor

5.2.1.3 Autres Capteurs/ Actionneurs

Pour les besoins de notre scénario, On a utilisé la gamme de produits CLEODE de la société CLEODE. Il s'agit d'une plate-forme matérielle et logicielle basée sur le réseau Zigbee.

Prise de courant ZPlug : La prise de courant ZigBee ZPLUG permet de commander tout type d'appareils 220V ne dépassant pas 3500W de consommation électrique. Elle fournit également

une information sur la consommation électrique de l'appareil lorsque celui-ci est activé. La prise de courant ZPLUG permet de mesurer la consommation d'un appareil électrique et de calculer la commutation de ce même appareil par l'intermédiaire d'une commande On/Off.

Bracelet-montre ZCare : ZCare est un bracelet-montre permettant d'émettre des alertes radiofréquences sur détection de défaillance de type :

- Alerte manuelle par appui sur un bouton d'urgence ;
- Détection d'un pouls hors norme : A partir de la mesure du pouls et d'une valeur moyenne fixée à l'initialisation, le bracelet peut émettre une alerte ;
- Détection d'une chute : Le bracelet permet de détecter à partir d'un profil d'activité fixé à l'initialisation ;

Détecteur ZDoor : Le détecteur ZDOOR permet de détecter l'ouverture/fermeture d'une porte/fenêtre.

- Système de commutation Le détecteur de commutation de lumière ZLight permet de commander deux lampes d'une puissance maximale de 500W chacune.
- Détecteur de présence Le détecteur de présence ZMOVE utilise un capteur infrarouge permettant de détecter les mouvements dans une pièce dans un rayon de 10 mètres maximum.

Telos-B et I-mote2 : Les deux plateforme de capteurs fournissent les données de localisation sous forme de coordonnées cartésiennes x, y, z d'un noeud mobile attaché à un objet / une personne. Plate-forme TelosB (Figure 5.3) a les mêmes fonctions que celle fournies par Imote2. Les données qu'on puisse récupérer sont la luminosité, la température et l'humidité.

Figure 5.3: Capteur Telos-B

5.2.2 Présentation de la plateforme à base de services

Dans cette section, on montre les différentes vues d'abstraction des services existants dans les domaines participant à la coopération multi-domaine. Dans le modèle sémantique proposé dans

le chapitre 3, on a motivé le fait de séparer les services de configuration des services fonctionnels ainsi que les descriptions abstraites des descriptions concrètes. Figure 5.4 détaille les interactions entre les différents domaines. Elle démontre les vues d'abstraction permettant la coopération de ces domaines. Une description de l'enchainement des interactions est présentée dans la section 5.2.2.1. Dans la section 5.2.2.2, on présente la liste des services fonctionnels abstraits disponibles par domaine (i.e. les vues d'abstraction).

5.2.2.1 Description des interactions

Le schéma de la Figure 5.4 présente deux processus coopératifs. Le premier processus coopératif concerne la création d'un programme par le service "SetReeducationProgram" du domaine, l'hôpital intelligent. Ce service envoie une demande d'ajout d'un événement programme de rééducation associé à un patient du domaine, la maison intelligente. Le service "NotifyCreateEvent". Ce service envoie une notification via le service "NotifyProgram" au domaine, le centre de kinésithérapie. Ce processus coopératif consiste donc à un appel en séquence d'un service d'organisation du programme suivi par un appel parallèle de deux services, à savoir, la création d'un événement et sa notification. Concernant les aspects de sécurité en particulier le contrôle d'accès et d'usage associé à ce processus coopératif. Le contrat coopératif impose que seul le médecin de rééducation physique ait la permission d'accéder au calendrier et d'avoir les droit d'ajout sur cette ressource externe au domaine de l'hôpital. A cet effet, un processus de provisioning pour créer une identité du médecin dans le domaine cible et lui donner les droits nécessaires d'authentification et d'autorisation doit se mettre en place.

Le second processus coopératif concerne les exercices de rééducation. Au démarrage du programme, le service "StartProgram" est appelé pour la configuration et effectue les initialisations nécessaires du système et de l'environnement. Ensuite un appel au service de localisation "LocalizePatient" permet de localiser la personne et d'atteindre sa position, le résultat étant une position que le service de déplacement du robot compagnon "ProposeProgram" le prend comme pré-condition pour son exécution. Le robot se déplace vers le patient et lui propose le programme via le service vocale "SpeechNotificaiton". Le Patient a le choix de commencer le programme et un service d'acceptation de l'événement est appelé. Sinon, un service est déclenché suite au refus du patient. Les deux services sont composés via un opérateur de choix exclusif *Exclusive Choice*, donc dans la suite un seul service est sollicité. Dans le cas de refus, le service fait appel à un service de notification pour prévenir le médecin de la décision prise par le patient. Dans l'autre cas, une initialisation des capteurs liés à l'état de la personne comme la chute, ou des informations biométriques, comme le rythme cardiaque. Cette étape démarre aussi une séquence vidéo associé au programme. À la fin, des données récoltées par les capteurs sont transmises à l'hôpital via le service "AcceptedDataEvent" pour un diagnostique par le médecin de rééducation physique. Une détection

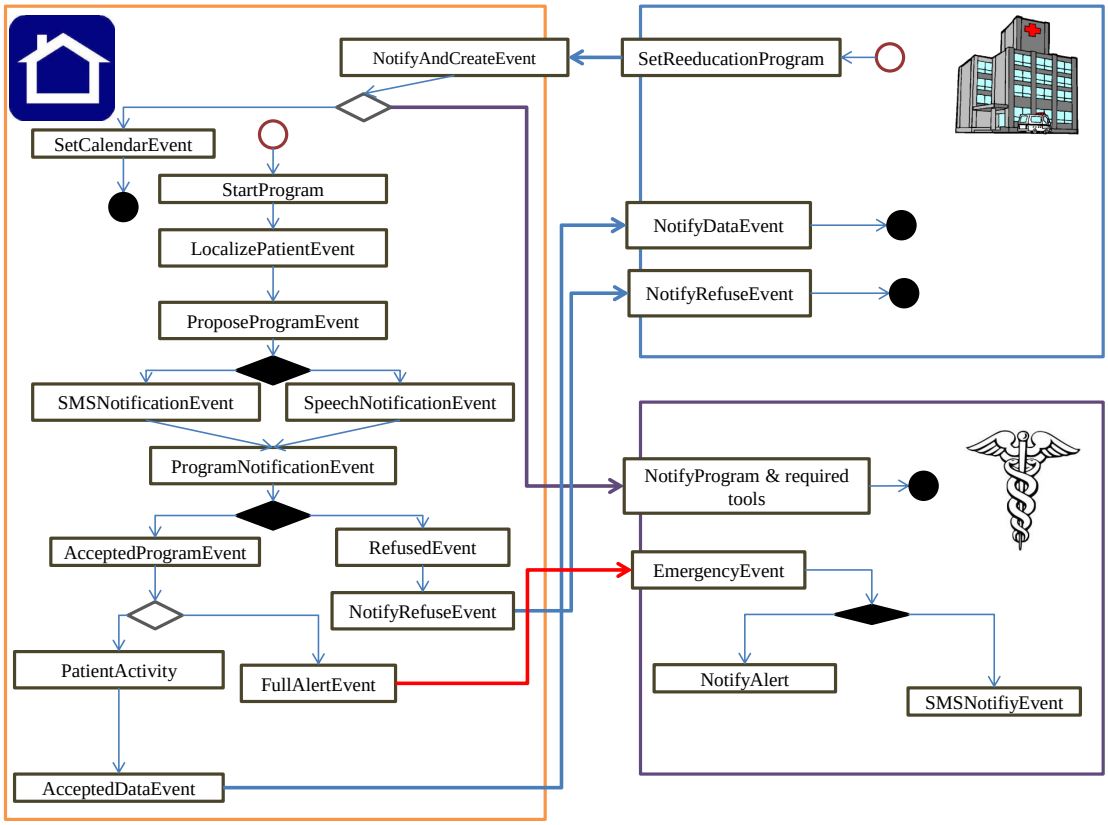


Figure 5.4: Démonstrateur du scénario de validation

de chute "FullAlertEvent" lors des exercices fera appel à un service d'urgence "EmergencyService" du centre de kinésithérapie.

5.2.2.2 Liste des services abstraits

Cette section présente les listes de services disponibles par domaine, la maison intelligente, l'hôpital intelligent et le centre de kinésithérapie.

La maison intelligente les services disponibles dans la maison intelligente sont les suivants :

- Localization : Ce service s'occupe de la localisation des agents identifiées comme localisables dans l'ontologie de coopération, comme les personnes et les robots. Ce services est une abstraction d'un service de localisation qui manipule en réalité des technologies et des mécanismes différents pour la localisation des différentes agents. Si on prends l'exemple du robot qui possède son système de localisation basé sur Karto [ref], le robot délivre un service intégré et qui envoie sa position dans un repère XYZ. Ce service de localisation des personnes dans des locaux que se soit dans la maison intelligente ou dans l'hôpital se base sur le réseaux de capteurs "Cricket". Se service permet de délivrer la position d'un capteur situé sur un objet ou

un humain et détermine sa position en fonction de XYZ. On remarque que notre abstraction est parfaite ou sens qu'elle peut être mappé sur les deux services, et suivant le contexte de localisation.

- Text to Speech : Ce service permet de transformer un texte en un flux audio. Ce service est utiliser dans l'objectif d'interagir avec les personnes dans un environnement ambiant.
- Speech To Text : Dans certaines circonstances où des contextes les interactions en audio ne sont pas très efficaces et des textes comme des SMS ou des notifications sont plus efficaces. Dans cet objectif on a créer un service qui transforme la parole d'une personne en texte. Ce service peut être utilisé pour la reconnaissance des commandes dictées ou des choix utilisateurs.
- Notification : le service de notification est très important dans cette plate-forme. Il permet de notifier les différentes entités des événements produits dans la plate-forme et de gérer les urgences. Ces notifications peuvent être diverses et variées suivant le contexte et le besoin des applications, le service que nous avons développé permet d'envoyer des mails, des SMS et des notifications sur écran de PC, et sur des équipements mobiles équipés de périphériques d'affichage.
- Service de configuration du calendrier : Ce service permet de gérer les entrées d'un calendrier.
- Service de provisionnement : Ce service permet d'encapsuler dans des messages de provisionnement des ordres de configuration dans un contexte multi-domaine.

L'hôpital intelligent Dans un premier temps l'hôpital est simulé par une partie du laboratoire. L'hôpital contient les deux services, concernant la création d'un programme de rééducation, et la gestion des demandes de configuration d'un calendrier dans un domaine cible.

- Création d'un programme : Ce service permet de créer un programme de rééducation pour un patient donné. Le programme spécifie certains critères temporels et spatiaux, comme l'endroit de l'exercice du programme. Comme il peut indiquer sous quelle température l'endroit doit être configuré avant le début de l'exercice.
- Configuration d'un calendrier : Ce service permet la gestion du calendrier quotidien d'une personne en général, et d'un patient en particulier. Dans notre étude de cas, nous avons développé ce service dans le but de créer des événements dans un calendrier. Deux connecteurs ont été mis en œuvre, l'ajout des événements dans un calendrier sous appareil mobile (e.g tablette, smartphone) et le connecteur pour les calendriers "Google Calendar".

- Service de provisionnement : Ce service permet d'exécuter et de faire des demandes d'actions de configuration, comme la demande de création d'une identité dans un domaine cible, attribution de rôle.

Le centre de kinésithérapie Le centre de kinésithérapie dispose d'un service de traitement des demandes des programmes permettant l'acceptation du suivi de certains types de programmes ou également les refus en donnant une justification, comme dans le cas de l'absence des médecins de rééducation physique. Dans notre étude de cas, les services proposés dans notre modèle sont largement suffisants pour mettre en œuvre la partie collaborative avec ce domaine. Ce qui permet de mettre en avant la généricité de notre modèle proposé.

5.3 Mise en œuvre de la méthodologie proposée

Notre approche suppose l'existence d'une liste de services par domaine. Par ailleurs, la méthodologie décrite dans la section (§3.6) consiste en un ensemble d'étapes reprises ci-dessous.

Étape 1 : Cette étape propose une extension de l'ontologie de coopération (Figure 3.7) avec un outil de conception d'ontologies, comme "protégé".

Étape 2 : Cette deuxième étape consiste en la formalisation de l'ontologie de coopération via le système formel proposé dans le chapitre 4.

Étape 3 : Concerne la spécification des vues d'abstraction sur les services et les processus impliqués dans la coopération multi-domaine. Ces vues d'abstraction sont formalisées suivant le modèle sémantique et le langage de composition détaillé dans le chapitre 4.

Étape 4 : Cette étape consiste en la création des processus coopératifs en tenant compte des vues abstraites disponibles. Les domaines ont la capacité de définir de nouvelles vues abstraites jugées nécessaires à la coopération et qui ne figurent pas parmi les abstractions produites dans l'étape précédente. La spécification des processus coopératifs est accompagnée de la définition des règles de contrôle, par exemple, les obligations, les permissions et les autorisations qui concernent l'accès et l'usage de chacune des vues d'abstraction par les processus coopératifs ainsi définis.

Étape 5 : La cinquième étape concerne la vérification de la propriété de correction des processus coopératifs par rapport à leurs spécifications dans l'assistant à la preuve Isabelle/HOL.

Étape 6 : La dernière étape, consiste en la génération d'interfaces et l'établissement des correspondances (Mapping).

Dans les sections suivantes, on appliquera la méthodologie présentée dans le chapitre 3. On présente les concepts nécessaires à notre étude de cas. Ces concepts étendent et créent des relations avec l'ontologie de coopération multi-domaine. Cette ontologie est développée par l'outil protégé en OWL-ALC. On donnera la conversion de cette ontologie vers une spécification $\mathcal{BCDL}_0$. Dans une deuxième partie, on instancie les concepts de cette ontologie afin de représenter les objets de l'environnement et son état. Ce qui correspond au modèle dans lequel les services (i.e. les vues d'abstraction) seront formalisés et dans lequel ils seront prouvés. Après la validation de ces services atomiques dans le contexte d'un environnement de composition, on va spécifier les processus coopératifs et générer les lemmes à prouver dans l'assistant à la preuve Isabelle/HOL.

Dans cette section, on procède à l'implémentation des concepts de l'ontologie qui modélisent les concepts nécessaires à la gestion des programmes de rééducation dans un environnement multi-domaine.

5.3.1 Extension AAL de l'ontologie de coopération

Dans cette section, on présente les concepts fondamentaux de l'ontologie de coopération et son extension pour la mise en œuvre de notre scénario de validation. Dans ce qui suit, on présente les principaux éléments de l'ontologie implémentée pour la validation. Cette partie concerne les extensions apportées de l'ontologie de coopération multi-domaine. Dans la section suivante, on présente la spécification de cette partie dans le système formel proposé.

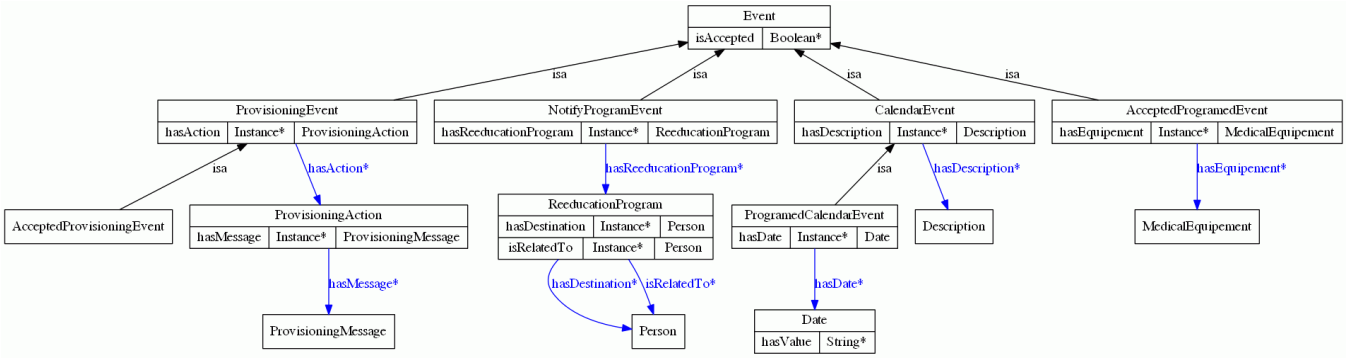


Figure 5.5: L'extension de l'ontologie de coopération

5.3.2 Formalisation de l'ontologie en $\mathcal{BCDL}_0$

La spécification en $\mathcal{BCDL}_0$ des extensions de l'ontologie de coopération multi-domaine dans Isabelle/HOL est présentée dans le tableau 5.3. Supposons que $\mathcal{T}$ est le modèle sémantique réalisé dans l'outil d'implémentation des ontologies *protégé* [4] et ensuite transformé par l'API du canevas.

[4] http://protege.stanford.edu/

Concept	Description
Date	Représente une date associé à un programme de rééducation
Description	Représente la description qui sera présente par le robot, durant la proposition du programme au patient
CalendarEvent	Représente l'événement correspondant à *Event* dans le calendrier du patient
NotifyProgramEvent	Représente un événement de notification d'un programme donné
AcceptedProgramedEvent	Représente un programme de rééducation approuvé par un domaine cible
ProgramedCalendarEvent	Représente un événement inscrit dans le calendrier du patient
MedicalEquipement	Représente un concept générique représentant des équipements nécessaires à un programme de rééducation
ReeducationProgram	Représente un programme de rééducation qui consiste en une liste d'activités
Account	Représente un compte utilisateur pour les identités du domaine. Chaque compte possède un nom d'utilisateur et un mot de passe.
AcceptedAccountEvent	Représente un événement d'acceptation pour créer un compte utilisateur
ProvisioningAccountEvent	Représente un événement de provisioning d'un compte utilisateur

Table 5.2: Description des concepts de l'extension de l'ontologie

$\mathcal{T}$ sera ensuite utilisé dans la formalisation des vues d'abstraction.

5.3.3 Spécification des services ubiquitaires abstraits

Dans cette section, on se base sur l'ontologie de coopération multi-domaine étendue afin de spécifier les services de notre scénario. Pour illustrer la méthodologie on se limitera à une sous partie du scénario. Cette partie est suffisante pour une preuve de concept et la démonstration de l'utilisation de la méthodologie et le canevas précédemment proposés. La partie à étudier concerne les deux services suivants : La notification d'un programme prescrit à destination d'un patient, *NotifyProgram*, et la création de événement par l'hôpital dans le domaine cible, *CreateEvent*. Le programme est prescrit par un médecin de rééducation et destiné pour un patient via le service *SetReeducationProgram*.

5.3.4 Spécification des services et preuve de correction

On s'intéresse ici à la spécification des services de la figure 5.6.

Le service "SetReeducationProgram" constitue le service qui produit le programme de rééducation associé au patient.

NotifyProgramEvent	$\sqsubseteq$	(Event)
NotifyProgramEvent	$\sqsubseteq$	(AllC (hasReeducationProgram)(ReeducationProgram))
ReeducationProgram	$\sqsubseteq$	(AllC (isRelatedTo)(Person))
AcceptedProgramedEvent	$\sqsubseteq$	(AllC (hasEquipement)(MedicalEquipement))
ProgramedCalendarEvent	$\sqsubseteq$	(AllC (hasDate)(Date))
AcceptedEvent	$\sqsubseteq$	(AtomC Event)
AcceptedProvisioningEvent	$\sqsubseteq$	(AndC (ProvisioningEvent) (AcceptedEvent))
ReeducationProgram	$\sqsubseteq$	(AllC (hasDestination) (Person))
CalendarEvent	$\sqsubseteq$	(AllC (hasDescription)(Description))
ProvisioningEvent	$\sqsubseteq$	(AtomC Event)
ProvisioningEvent	$\sqsubseteq$	(AllC (hasAction)(ProvisioningAction))
ProvisioningAction	$\sqsubseteq$	(AllC (hasMessage)(ProvisioningMessage))
RefusedEvent	$\sqsubseteq$	(AndC (Event) (NotC (AcceptedEvent)))
Partie provisioning d'un compte utilisateur		
PrvgAccountEvent	$\sqsubseteq$	(AtomC ProvisioningEvent)
AcceptedAccountEvent	$\sqsubseteq$	(AndC (AtomC AcceptedEvent) (SomC (hasAccount)(Accoun
ApprovedAccountEvent	$\sqsubseteq$	(AtomC AcceptedAccountEvent)
Account	$\sqsubseteq$	(AndC (Object) (AllC (hasLogin)(Login)) (AllC (hasPassword)(Password)))

Table 5.3: Spécification de l'ontologie via le système formel proposé

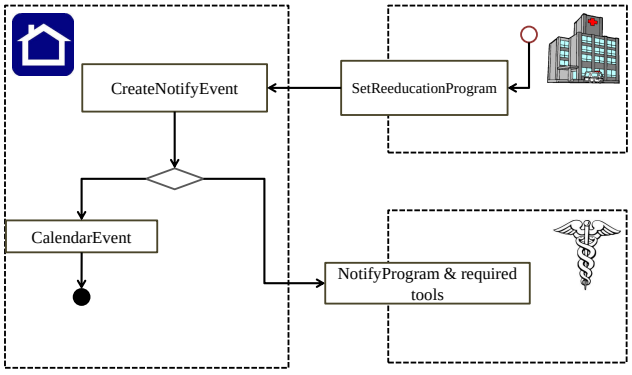


Figure 5.6: Processus coopératif : création et notification d'un programme de rééducation

Le service "CreateNotifyEvent" permet de faire un appel parallèle aux services de création et de notification.

(SP (SetReeducationProgram) (event)
(Pre-Condition)
(AtomC ReeducationProgramEvent)
(Post-Condition)
(AndC (AtomC ReeducationProgram) (SomC (isRelatedTo)(AtomC Person)
))

(SP (CreateAndNotify) (event)
(Pre-Condition)
(AndC (AndC (AtomC NotifyProgramEvent)(SomC (hasReeducationProgram) (AtomC Program))) (AndC (AtomC CalendarEvent) (SomC (hasDescription) (AtomC Description))))
(Post-Condition)
(OrC (AtomC RefusedEvent) (AndC (AndC (AtomC AcceptedProgramedEvent) (SomC (hasEquipement) (AtomC MedicalEquipement))) (AndC (AtomC ProgramedCalendarEvent) (SomC (hasDate) (AtomC Date))))))

Le service "CalendarEvent" permet de créer un événement dans le calendrier du patient.

(SP (CreateEvent) (event)
(Pre-Condition)
(AndC (AtomC CalendarEvent) (SomC (hasDescription) (AtomC Description)))
(Post-Condition)
(AndC (AtomC ProgramedCalendarEvent) (SomC (hasDate) (AtomC Date)))
)

Le service "NotifyProgram" notifie le centre de kinésithérapie du programme de rééducation de la personne concernée et des équipements nécessaires aux activités du programme.

(SP (NotifyProgram) (event)
(Pre-Condition)
(AndC (AtomC NotifyProgramEvent)(SomC (hasReeducationProgram) (AndC (AtomC ReeducationProgram) (SomC (isRelatedTo)(AtomC Person)))))
(Post-Condition)
(OrC (AtomC RefusedEvent) (AndC (AtomC AcceptedProgramedEvent)(SomC (hasEquipement) (AtomC MedicalEquipement)))))

Chacun de ces services doit être correct dans le modèle de coopération $\mathcal{T}$ qui représente l'ontologie

de coopération, donc on doit vérifier la correction de ces services avant de procéder à la composition. L'API-proofs permet de générer une théorie qui inclut la théorie du système formel ainsi que la spécification de l'ontologie et les spécifications de services et les obligations de preuve de correction à prouver. Ces preuves doivent être réalisées de manière interactive avec Isabelle/HOL.

Le service "ProvisioningService" constitue le service de provisionnement multi-domaine.

(SP (ProvisioningService) (event)
(Pre-Condition)
(AndC (AtomC ProvisioningAction) (SomC (hasSourceDomain)(AtomC Domain)
))
(Post-Condition)
(OrC (AndC (AtomC AcceptedProvisioningAction))
(AndC (AtomC RefusedProvisioningAction) (SomC (isRelatedTo)(AtomC Message)))

5.3.5 Formalisation des processus coopératifs et la preuve de correction

Après avoir prouvé la correction de ces services, on propose la composition de ces services avec un opérateur de parallélisme pour construire le service "CreateAndNotify". L'API-proofs génère les conditions d'applicabilité à prouver pour ce service composite. La preuve de ces conditions est effectuée à l'aide de la tactique "*Metis*" (Table 5.5). Le spécification de cet exemple et la preuve de correction est dans l'annexe 6.2.2.

Spécification du scénario Dans ce démonstrateur, on se limite à présenter la partie concernant le premier processus coopératif 5.6. On note que ce processus coopératif a besoin que le médecin de rééducation physique dispose des droits nécessaires et d'une identité dans le domaine cible afin de pouvoir y créer des événements (i.e. La maison intelligente).

Règles de gestion du contrôle d'accès et d'usage La composition de services de provisionnement représentée dans la figure 5.7. Elle établie la règles permettant de créer une identité du médecin dans le domaine cible. Cette identification consiste en la création d'un compte utilisateur et donner les droit d'accès basé sur l'authentification de l'utilisateur externe (le médecin de rééducation physique) via un Login/mot de passe. Ces informations permettent à des utilisateurs l'usage des applications et/ou l'accès au ressource (i.e. Dossier médical). Cependant, tout accès requiert l'approbation du gestionnaire de la maison intelligente. Pour ce scénario les règles de contrôle d'accès au dossier médical d'un utilisateur et l'usage du service de création d'un événement ont été

spécifiés. On a définit au niveau de l'ontologie de coopération les rôles globaux à savoir *Doctor-Bronze*, *DoctorGold*. Le rôle *DoctorBronze* a le droit d'utiliser le service pour créer un événement dans le calendrier du patient.

$AddCalendarEvent(DoctorBronze, CreateProgramAction, CreationService, Authorization)$;
Le *DoctorGold* peut modifier le dossier médical du patient. Les deux règles suivante sont formaliser comme suit :

$EditPatientRepository(DoctorGold, EditRepositoryAction, ModifyService, Authorization)$;

Cette composition de services de provisionnement est formalisée ainsi que sa correction est prouvée en utilisant le modèle du chapitre 4.

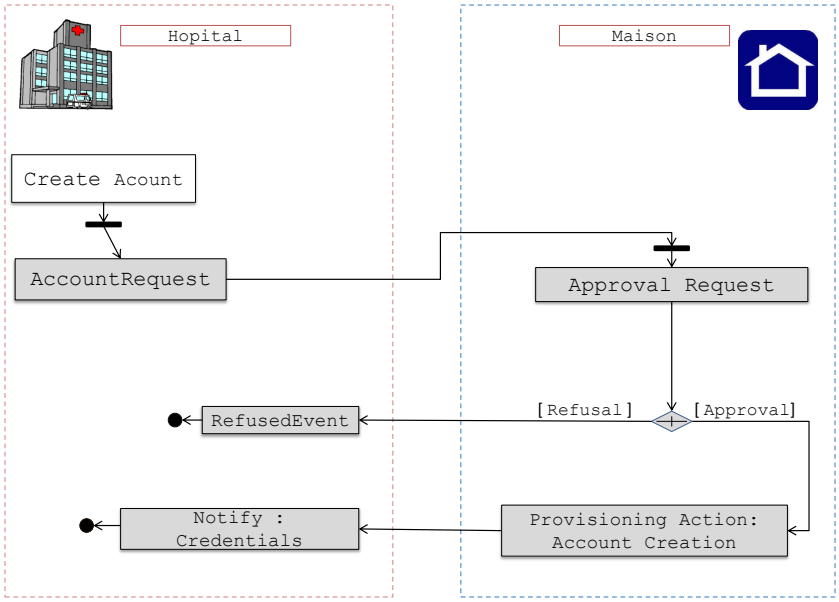


Figure 5.7: Processus coopératif de provisioning

Preuve de correction Dans le table 5.4, on présente la formalisation du processus coopératif de composition de provisioning. Ce service implique deux services de provisioning *RequestAccount* permettant la demande de création d'une identité dans le domaine cible, et le service *ApprovalRequest* qui donne un accord à travers une approbation de la précédente demande. Le résultat de l'approbation est la création du compte utilisateur et l'envoie des paramètres de connexion.

5.3.6 Mise en correspondance vers les domaines cibles

Cette partie concerne le mapping des processus vers les domaines cibles. On s'intéresse au service de provisionnement utilisé par les différents domaines pour gérer les demandes et les traitements des ordres de configration dans un contexte multi-domaine. On présente ci-dessous les processus

$ProvisioningProcess(action) :: ApprovedAction \sqcap \exists\ hasRequester.D_A \sqcap \exists\ hasApprover.D_B \sqcap$ $\exists hasRequestTask.(AccountRequest \sqcap \exists hasSourceDomain.D_A \sqcap \exists\ hasTargetDomain.D_B \sqcap$ $\exists\ hasRequestTask.PrvgAccountAction) \Longrightarrow$ $(ApprovedAccountAction \sqcap \exists hasAccount.Account) \sqcup$ $(RefusedAction \sqcap \exists toIdentity.ManagerD_A)$ — *Sequence*
$\Pi_1 : doRequest(action : PrvgAccountAction) ::$ $AccountRequest\ \sqcap\ \exists hasSourceDomain.D_A\ \sqcap \exists\ hasTargetDomain.D_B \sqcap$ $\exists\ hasRequestTask.PrvgAccountAction \Longrightarrow AcceptedAction\ \sqcup\ RefusedAction$
$\Pi_2 : doApproval(a : AccountRequest) :: (RefusedAction \sqcap \exists\ toIdentity.ManagerD_A) \sqcup$ $(ApprovedAction\ \sqcap\ \exists\ hasRequester.ManagerD_A \sqcap \exists\ hasApprover.ManagerD_B \sqcap$ $\exists\ hasApprovalAction.AccountRequest) \Longrightarrow RefusedNotifyAction \sqcup$ $(\exists\ hasResponse.\top\ \sqcap (AcceptedNotifyAction\ \sqcap \exists\ hasMessage.Message))$ — *ExclusiveChoice*
$\Pi_{2.1} : Notify(identity) :: RefusedAction\ \sqcap \exists\ toIdentity.ManagerD_A$ $\Longrightarrow\ RefusedNotifyAction$
$\Pi_{2.2} : ApprovalRequest(action) :: ApprovedAction \sqcap \exists\ hasRequester.ManagerD_A \sqcap$ $\exists\ hasApprover.ManagerD_B\ \sqcap \exists\ hasApprovalAction.AccountRequest$ $\Longrightarrow \exists\ hasResponse.\top \sqcap (AcceptedNotifyAction \sqcap \exists hasMessage.Message)$

Table 5.4: Formalisation du processus coopératif de provisioning

responsables du provisionnement d'un domaine cible à partir d'un domaine émetteur ou demaine utilisateur. Ce service est responsable des échanges d'ordre de provisionnement dans un contexte multi-domaine.

5.3.6.1 Service de provisionnement

Ce service représente une vue générique commune aux domaines coopératifs. Il est à la fois générique et extensible. Ce qui lui permet la prise en compte des ordres de provisionnement plus complexes. La définition de l'interface comprend la description globale du processus coopératif à l'ensemble des messages pouvant être échangés et le schéma des objets à provisionner. L'interface sépare la notion de processus et les opérations de provisionnement internes aux domaines. Les actions de provisionnement sont modélisées dans l'ontologie de coopération multi-domaine. Elles sont exprimées ensuite en SPML. Si des opérations plus sophistiquées sont requises, il est possible d'étendre les opérations basiques en utilisant la flexibilité du modèle sémantique et du langage SPML. Le fournisseur de service est dans le domaine cible alors que le consommateur du service est dans le domaine de

theory Composition
imports NotifyService CalendarEvent
begin
lemma "⟦ SCER (E)(SF NotifyService x e g);
SCER (E)(SF S2 x h j)
;(T ∪ ConceptF x A ⊢ ConceptF x (Pre_s (SF NotifyService x e g)))
;(T ∪ ConceptF x A ⊢ ConceptF x (Pre_s (SF CalendarEvent x h j)))
;(T ∪ ConceptF x (AndC (Post_s (SF NotifyService x e g)) (Post_s (SF CalendarEvent x h j))) ⊢ ConceptF x (B)) ⟧
⟹ SCER (E) (SF SS x A B) "
apply (metis AND)
done

Table 5.5: Preuve de correction de la composition de services "NotifyService" et "CalendarEvent"

l'utilisateur. Les objets à provisionner sont décrits par l'ontologie de coopération.

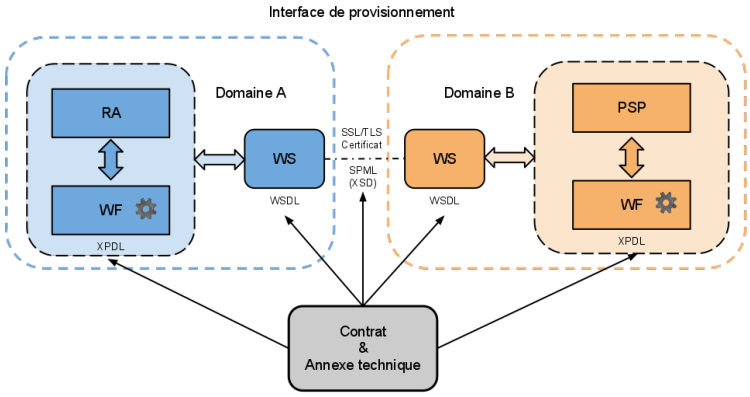


Figure 5.8: Service de provisioning inter-domaines

La couche concrète de ce service a été développée dans le cadre du projet européen *Role-ID*. L'interface est spécifiée dans un document commun correspondant à une annexe technique au contrat établi entre les deux domaines. Celui-ci doit être utilisée pour générer :

- Le fichier XPDL pour chaque workflow;
- Le fichier WSDL décrivant les échanges;
- Le fichier XSD contenant le schéma des objets à provisionner;

Cette interface implique deux entités à savoir le RA (Request Autority) qui représente le demandeur de l'action de provisionnement et le PSP (Provisioning Service Provider), responsable de l'analyse des requêtes et de leurs de exécutions. Le RA catalyse les évènements déclenchés par

l'utilisateur émetteur d'une requête et se charge d'initialiser tous les composants nécessaires à son propre fonctionnement.

L'interface est minimale et permet de déclencher ou d'annuler les opérations supportées par le Provider. Le Provider lui, ne permet pas directement la liaison avec un autre logiciel. L'interface d'exécution des requêtes du module SPML en a la charge. Le rôle du Provider est d'enclencher les processus de traitement et les workflows adéquats lors de la réception d'une requête. A l'instar du RA, il se charge d'initialiser chacun des composants similaires qu'il utilise, ainsi que le module SPML, son schéma d'objets et les paramètres de connexion aux différentes cibles (PST)[5].

Pour la sécurité du système de provisionnement, des certificats ont été déployés afin d'authentifier les domaines émetteurs. Le PSP est capable ensuite de filtrer les requêtes, les analyser afin de déclencher les workflow internes permettant d'exécuter ces ordre de provisionnement (i.e. Demande de création de permission sur une ressource).

L'utilisation de deux workflows se justifie comme suit : l'un correspond au circuit usuel de validation qui est déclenché par les actions de l'utilisateur dans son domaine, alors que l'autre est activé pour le provisionnement inter-domaine. Dans la section suivante, on présente les workflows du domaine utilisateur et du domaine cible.

Ces workflows de provisionnement reliées à des infrastructures spécifiques ont été développés afin de simuler les workflows internes.

Processus cible de provisionnement (PSP) Le workflow le plus simple est celui situé du coté du PSP. Il conserve la requête en attendant son approbation ou son refus par un intervenant.

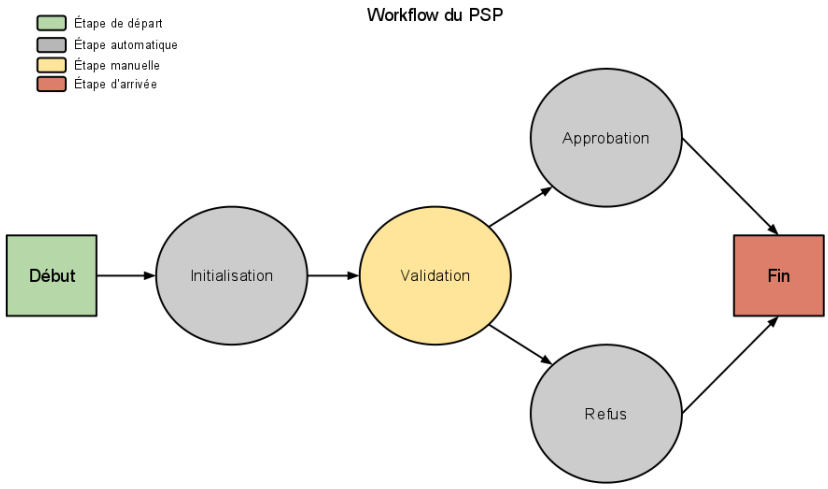


Figure 5.9: Le processus du domaine cible (PSP)

L'étape d'initialisation permet de configurer les modules annexes dont les différentes étapes

[5]Provisioning Service Target

seront amenées à utiliser. Pour le démonstrateur par exemple, une classe permettant l'envoi d'email depuis un workflow et à partir de modèles de références doit être initialisée avec l'adresse du serveur d'envoi (SMTP), etc. L'étape d'attente nécessite une intervention extérieure, afin de déterminer le prochain état dans lequel doit entrer le workflow. Au sein du démonstrateur cette étape notifie l'administrateur du domaine qu'une nouvelle requête requiert son approbation. L'étape d'approbation effectue la requête et stocke son résultat dans une variable interne, alors que l'étape de refus enregistre une réponse prévenant l'émetteur d'un échec et de sa cause.

Processus source de provisionnement (RA) Le second workflow est celui situé du coté du RA, qui commence par récupérer le compte de l'utilisateur dans le domaine distant s'il est déjà connu, ou demande la création du compte dans le cas contraire. Ensuite, le workflow demande l'assignation d'un compte utilisateur à un rôle. Ce dernier est renseigné au démarrage de ce processus.

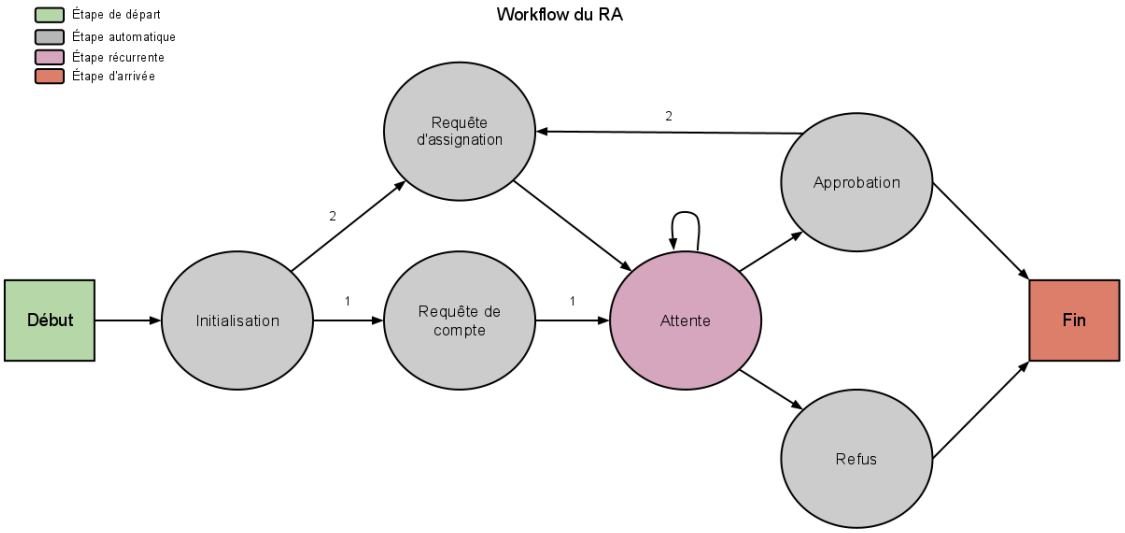


Figure 5.10: Le processus du domaine source (RA)

L'étape d'initialisation est similaire à celle du workflow du PSP. L'étape de requête de compte tente de récupérer le compte distant de l'utilisateur ayant démarré le processus. Si celui-ci existe on passe directement à l'étape requête d'assignation, sinon on attend une réponse du PSP et l'étape d'approbation si elle a lieu mettra ensuite le workflow dans l'état requête d'assignation. L'étape de requête d'assignation, émet la demande d'attribution d'un rôle du domaine distant au compte de l'utilisateur dans le second domaine. Cette état est une étape obligatoire, puisqu'elle représente le but du provisionnement. L'étape d'attente est un état récurrent. À chaque itération, celui-ci envoie une demande de statut de la requête de compte ou d'assignation. L'étape d'approbation et de refus stockent la réponse reçue dans une variable interne. Dans le démonstrateur, ces étapes envoient également un email de notification à l'utilisateur ayant fait la demande avec le résultat de

sa requête et ses identifiants le cas échéant.

5.3.6.2 Services concrets, JAX-WS et les Services Web

La concrétisation des vue d'abstraction ainsi que l'interface de provisionnement est faite à l'aide des technologies de services web. Cette section, décrit les technologies utilisées dans l'implémentation des services concrets, en particulier JAX-WS[6]. JAX-WS [7] est une implémentation de la JSR [8] 244. La spécification formelle des services d'abstraction représente en réalité les données en entrées et les paramètres de sortie d'un web service.

Du coté PSP (Figure 5.11), le comportement du service effectue une première analyse de la requête afin de déterminer si celle-ci est autorisée et si elle nécessite ou non un asynchronisme. Le cas échéant, une réponse détenant un numéro d'identification est renvoyée et permet à l'émetteur de s'enquérir de son statut dans le futur.

JAX-WS est une technologie fondamentale pour le développement de web service SOAP et RESTful (services Web qui utilisent Representational State Transfer, ou REST, outils) en java. JAX-WS est conçu pour prendre la place de l'interface JAVA-RPC (Remote Procedure Call) dans les services Web et les applications basées sur le Web. JAX-WS est également utilisé pour créer des services Web et les clients correspondants qui communiquent en utilisant XML pour envoyer des messages ou utiliser appels de procédures distantes pour échanger des données entre le client et le fournisseur de service. JAX-WS représente les appels de procédures distantes ou des messages en utilisant des protocoles basés sur XML comme SOAP. Cependant, la complexité innée de SOAP derrière une API basée sur Java est caché.

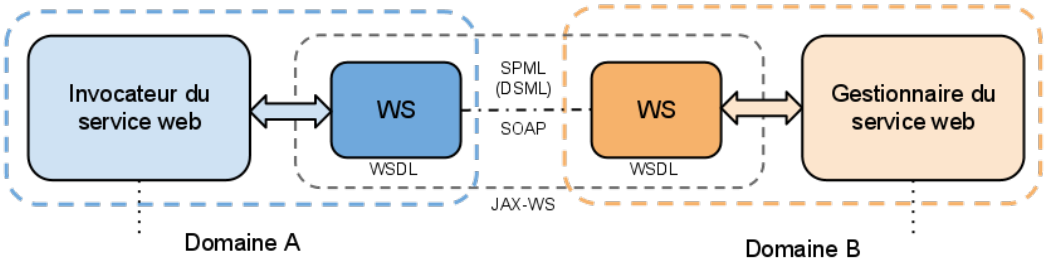


Figure 5.11: Le concept du service de provisionnement multi-domaine

[6] Java API for XML-Based Web Services
[7] http://jax-ws.java.net/
[8] Java Specification Requests

5.3.6.3 Génération automatique des services Web

La dernière étape concerne l'établissement des correspondances entre les processus internes aux domaines, et les vues d'abstraction. Cela consiste en la génération des appels aux web services des domaines participant à la coopération ainsi que l'instanciation des processus de workflows. Notre classe est maintenant compatible. Mais il nous manque le stub pour en faire un SEI. La classe d'implémentation doit également définir un constructeur sans argument avec une visibilité publique. JAX-WS est basée sur le concept de POJO (Plain Old Java Object). Pour exposer une fonction donnée, par exemple, *moveTo* dans cet exemple, il suffit d'ajouter quelques annotations à notre classe *MoveRobot*. Dans notre cas il faut ajouter @WebService (devant la déclaration class) et @WebMethod (devant la déclaration de fonction). Prenons l'exemple du service de deplacement de robot *MoveRobot*.

```
package org.upec.lissi.framework.services;
import javax.jws.WebMethod;
import javax.jws.WebService;
@WebService(name="MoveRobotService")
/* MoveRobotService : le nom de service */
public class MoveRobot {
        @WebMethod
        /* moveTo : la fonction offerte par ce service */
        /*x et y sont les pre-conditions */
        public String moveTo(String x, String y){
                /* le corps du service */
                return "(x,y) position is reached";
        }
}
```

Par la suite, on a mis en place un script XML Ant qui permet de générer directement les services web SOAP. Le template de ce script et le suivant.

```
<wsgen    sei="..."
    destdir="directory␣for␣generated␣class␣files"
    classpath="classpath" | cp="classpath"
    resourcedestdir="directory␣for␣generated␣resource␣files␣such␣as␣WSDLs"
    sourcedestdir="directory␣for␣generated␣source␣files"
    keep="true|false"
    verbose="true|false"
    genwsdl="true|false"
    protocol="soap1.1|Xsoap1.2"
    servicename="..."
    portname="...">
    extension="true|false"
    <classpath refid="..."/>
</wsgen>
```

5.4 Conclusion

Dans ce chapitre, on a réalisé une étude de cas par le canevas sémantique proposé. L'étude de cas est représentée par scénario dans le domaine médical. Notre visite à l'hôpital Henry Mondor a élargie notre perception du problème et du métier. Ceci nous a aidé à mieux spécialiser les concepts de l'ontologie et de modéliser les entités nécessaires à cette étude de cas. La réalisation a consistée en l'extension de l'ontologie de coopération par les concepts spécifiques relatifs au domaine d'application et à l'étude de cas, en l'utilisation du système formel et en le respect de la méthodologie proposée pour la spécification des processus coopératifs. Le canevas sémantique et la méthodologie facilite certaine tâches mais demande certaines compétences comme la preuve sur Isabelle/HOL. Le niveau d'automatisation de certaines tâches comme la transformation et la génération des obligations de preuve sont très importantes. Cependant, le canevas manque d'être sous un format logiciel distribuable comme des Plug-in Eclipse ou une application Desktop pour qu'il soit sous les critiques de la communauté scientifique et de pouvoir s'interfacer avec plusieurs outils que ce soit de conception d'ontologies ou des assistants à la preuve de théorème ou du model checking. Dans le chapitre suivant, on va présenter la conclusion et les perspectives proches ainsi que les perspectives à long termes.

6 Conclusion et Perspectives

Sommaire du chapitre

6.1 Conclusion

L'émergence de nouvelles technologies de communication et de l'information et le besoin de créer des applications centrées utilisateur ont conduit les recherches à traiter la problématique de sureté de ces applications et à considérer la sécurité dans les environnements de la vie quotidienne des utilisateurs.

Dans ce mémoire de thèse, on a abordé la problématique de construction des applications ubiquitaires dans un environnement à plusieurs domaines. On a considéré la description sémantique des services ubiquitaires dans ces environnements, la composition de ces services ainsi que la validation formelle de la propriété de correction des compositions de services proposées. D'abord on a présenté les différentes notions importantes des services web et leurs compositions. Ensuite, on a présenté le système formel étudié, ce qui nous a permis de situer notre travail et de le comparer par rapport à d'autres approches. La maîtrise de l'assistant de preuve Isabelle/HOL, la gestion des types relatifs à l'implémentation de l'algorithme des *information term* ainsi que l'analyse des différentes techniques de preuve, ont été des étapes nécessaires pour proposer des solutions aux problématiques considérées dans cette thèse.

Le travail réalisé consiste en la proposition d'un canevas sémantique intégré permettant de répondre à des exigences imposées par ce type d'applications. Cet outil intègre l'hétérogénéité des fournisseurs de services, la description des comportements de ces services, la composition de services ainsi que la validation formelle via une approche à base de preuve du théorème de la correction des compositions de services. Dans cette même lignée, on a conçu un langage formel de description d'une ontologie de coopération entre des fournisseurs hétérogènes dont la connaissance est utilisée pour la spécification formelle des abstractions de services. On a aussi spécifié le langage de composition avec les règles de composition, afin de valider des scénarii de composition de services. Cette spécification formelle dans l'assistant à la preuve Isabelle/HOL permet la composition de services et sa validation formelle, ce qui diminue les risques d'erreurs dès les premières étapes de conception des applications ubiquitaires considérées comme critiques.

6.2 Perspectives

Dans la suite de ces travaux, des perspectives intéressantes à cours et à long termes nécessitent d'approfondir les aspects théoriques et les choix des scénarios d'implémentation. Dans ce qui suit, on énumère quelques unes de ces perspectives.

6.2.1 Perspectives à cours termes

- L'intégration avec des techniques de planification va améliorer le comportement dynamique et la composition des actions;
- Faire une étude permettant de modéliser le langage de composition de services dans une méta-logique;
- Implémenter la modélisation de ce langage de composition de services au niveau du système d'inférence HOL et Isabelle/pure;
- Faire une étude pour avoir un module qui prouve automatiquement une composition et intègre le module dans Isabelle/HOL.

6.2.2 Perspectives à long termes

- S'interfacer avec des outils industriels de composition comme WS-BPEL ou WComp, afin de leur donner des outils de validation des compositions construites ;
- Une autre perspective se base sur l'étude des "Programs Synthesis" afin de générer des programmes à partir des preuves formelles de la vérification. Ceci nécessitera d'autres règles supplémentaires dans la logique utilisée. En conséquence, la preuve de "Soundness " et l'implémentation de tout le système dans l'assistant à la preuve Isabelle/HOL sont également à revoir.

Références

[Abramsky, 1994] Abramsky, S. (1994). Proofs as processes. *Theor. Comput. Sci.*, 135(1):5–9.

[Ali, 2010] Ali, S. (2010). *Semantic interoperability of ambient intelligent medical devices and e-health systems*. PhD thesis, Saarlandische Universitats- und Landesbibliothek, Postfach 151141, 66041 Saarbracken.

[Allemang and Hendler, 2008] Allemang, D. and Hendler, J. (2008). *Semantic web for the working ontologist : effective modeling in RDF, RDFS and OWL*. Morgan Kaufmann Publishers/Elsevier, Amsterdam ; Boston.

[Ankolekar et al., 2002] Ankolekar, A., Burstein, M., Hobbs, J., Lassila, O., Martin, D., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Payne, T., et al. (2002). DAML-S: web service description for the semantic web. *The Semantic Web ISWC 2002*, pages 348–363.

[Antonio Bucchiarone, 2006] Antonio Bucchiarone, S. G. (2006). A survey on services composition languages and models. *Proceedings of International Workshop on Web Services Modelling and Testing*, pages 51–63.

[Baader, 2003] Baader, F. (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

[Bakhouya et al., 2012] Bakhouya, M., Campbell, R., Coronato, A., Pietro, G. d., and Ranganathan, A. (2012). Introduction to special section on formal methods in pervasive computing. *ACM Trans. Auton. Adapt. Syst.*, 7(1):1–9.

[Bellin and Scott, 1994] Bellin, G. and Scott, P. J. (1994). On the pi-Calculus and Linear Logic. *Theoretical Computer Science*, 135:11–65.

[Belqasmi et al., 2011] Belqasmi, F., Glitho, R., and Fu, C. (2011). RESTful web services for service provisioning in next-generation networks: a survey. *Communications Magazine, IEEE*, 49(12):66–73.

[Benghazi et al., 2012] Benghazi, K., Hurtado, M. V., Hornos, M. J., Rodríguez, M. L., Rodríguez-Domínguez, C., Pelegrina, A. B., and Rodríguez-Fórtiz, M. J. (2012). Enabling correct design and formal analysis of ambient assisted living systems. *Journal of Systems and Software*, 85(3):498 – 510.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific american*, 284(5):28–37.

[Blanchette et al., 2011] Blanchette, J. C., Bulwahn, L., and Nipkow, T. (2011). Automatic proof and disproof in isabelle/hol. In Tinelli, C. and Sofronie-Stokkermans, V., editors, *Frontiers of Combining Systems (FroCoS 2011)*, volume 6989 of *LNCS*, pages 12–27. Springer.

[Booth et al., 2004] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. (2004). Web services architecture. World Wide Web Consortium.

[Bozzato and Ferrari, 2010a] Bozzato, L. and Ferrari, M. (2010a). Composition of semantic web services in a constructive description logic. In *Web Reasoning and Rule Systems*, pages 223—-226. Springer.

[Bozzato and Ferrari, 2010b] Bozzato, L. and Ferrari, M. (2010b). A note on semantic web services specification and composition in constructive description logics. *Journal of Syntax And Semantics*.

[Bozzato et al., 2007] Bozzato, L., Ferrari, M., Fiorentini, C., and Fiorino, G. (2007). A constructive semantics for alc. In Calvanese, D., Franconi, E., Haarslev, V., Lembo, D., Motik, B., Turhan, A.-Y., and Tessaris, S., editors, *Description Logics*, volume 250 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Bromberg, 2006] Bromberg, D. (2006). *Résolution de l'hétérogénéité des intergiciels d'un environnement ubiquitaire*. These, Université de Versailles-Saint Quentin en Yvelines.

[Burstein et al., 2004] Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K. (2004). Owl-s: Semantic markup for web services. *W3C Member Submission*.

[Casati et al., 2000] Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., and Shan, M. (2000). EFlow: a platform for developing and managing composite e-services. In *Proceedings Academia/Industry Working Conference on Research Challenges*, pages 341–348.

[Charif, 2007] Charif, Y. (2007). *Chorégraphie dynamique de services basée sur la coordination d'agents introspectifs*. PhD thesis, Université Pierre et Marie Curie (UPMC). Type : Thèse de Doctorat – Soutenue le : 2007-12-10 – Dirigée par : El fallah seghrouchni, Amal – Encadrée par : SABOURET Nicolas.

[Chebbi et al., 2006] Chebbi, I., Dustdar, S., and Tata, S. (2006). The view-based approach to dynamic inter-organizational workflow cooperation. *Data & Knowledge Engineering*, 56(2):139–173.

[Chein and Mugnier, 1992] Chein, M. and Mugnier, M.-L. (1992). Conceptual graphs: Fundamental notions. In *Revue d'intelligence artificielle*. Citeseer.

[Chiarugi et al., 2006] Chiarugi, F., Zacharioudakis, G., Tsiknakis, M., Thestrup, J., Hansen, K. M., Antolin, P., Melgosa, J. C., Rosengren, P., and Meadows, J. (2006). Ambient intelligence support for tomorrow's health care: Scenario based requirements and architectural specifications of the eu-DOMAIN platform. In *Proceedings of the International Special Topic Conference on Informational Technology in BioMedicine in Ioannina, Greec*.

[Chinnici et al., 2007] Chinnici, R., Moreau, J.-J., Ryman, A., and Weerawarana, S. (2007). Web services description language (wsdl) version 2.0 part 1: Core language. World Wide Web Consortium, Recommendation REC-wsdl20-20070626.

[Chiu et al., 2004a] Chiu, D. K., Cheung, S. C., Till, S., Karlapalem, K., Li, Q., and Kafeza, E. (2004a). Workflow view driven cross-organizational interoperability in a web service environment. *Information Technology and Management*, 5(3):221–250.

[Chiu et al., 2004b] Chiu, D. K., Cheung, S. C., Till, S., Karlapalem, K., Li, Q., and Kafeza, E. (2004b). Workflow view driven cross-organizational interoperability in a web service environment. *Information Technology and Management*, 5(3):221–250.

[Coalition, 2005] Coalition, W. M. (2005). Wfmc: Process definition language: Xpdl 2.0. Specification TC-1025, Workflow Management Coalition. The intended audience for this document is primarily vendor organizations who seek to implement the XML Process Definition Language (XPDL) of the Workflow Management Coalition (WfMC), or using it as a file format for the Business Process Modeling Notation.

[Cook et al., 2009] Cook, D. J., Augusto, J. C., and Jakkula, V. R. (2009). Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298.

[Coronato and Pietro, 2010] Coronato, A. and Pietro, G. D. E. (2010). Formal specification of wireless and pervasive healthcare applications. *ACM Transactions on Embedded Computing Systems (TECS)*, 10(1):12.

[de Paiva, 2006] de Paiva, V. (2006). Constructive description logics: what, why and how. *Context Representation and Reasoning, Riva del Garda*.

[Dustdar and Schreiner, 2005] Dustdar, S. and Schreiner, W. (2005). A survey on web services composition. *International Journal of Web and Grid Services*, 1(1):1–30.

[Feenstra et al., 2007] Feenstra, R. W., Janssen, M., and Wagenaar, R. W. (2007). Evaluating web service composition methods: the need for including multi-actor elements. *the Electronic Journal of e-Government*, 5(2):153–164.

[Ferrari et al., 2009] Ferrari, M., Fiorentini, C., and Fiorino, G. (2009). Bcdl : Basic constructive description logic. *Journal of Automated Reasoning*, 44(4):371–399.

[Ferrari et al., 2010] Ferrari, M., Fiorentini, C., and Fiorino, G. (2010). BCDL: Basic constructive description logic. *Journal of Automated Reasoning*, 44(4):371–399.

[Fielding, 2000] Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California.

[Garlik et al., 2013] Garlik, S. H., Seaborne, A., and Prud'hommeaux, E. (2013). SPARQL 1.1 Query Language. http://www.w3.org/TR/sparql11-query/.

[Geraci, 1991] Geraci, A. (1991). *IEEE Standard Computer Dictionary: Compilation of IEEE Standard Computer Glossaries*. IEEE Press, Piscataway, NJ, USA.

[Gérodolle and Bottaro, 2006] Gérodolle, A. and Bottaro, A. (2006). OSGi et le projet IST Amigo. In *Atelier de travail OSGi 2006*, Paris, France. projet IST.

[Ghallab et al., 2004] Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: Theory & Practice*. Morgan Kaufmann, 1 edition.

[Gruber, 1995] Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5):907–928.

[Ha et al., 2005a] Ha, Y.-G., Sohn, J.-C., Cho, Y.-J., and Yoon, H. (2005a). Towards a ubiquitous robotic companion: Design and implementation of ubiquitous robotic service framework. *ETRI Journal*, 27(6):666–676.

[Ha et al., 2005b] Ha, Y.-G., Sohn, J.-C., Cho, Y.-J., and Yoon, H. (2005b). Towards a ubiquitous robotic companion: Design and implementation of ubiquitous robotic service framework. *ETRI journal*, 27(6):666—676.

[Hansen et al., 2008] Hansen, K., Zang, W., Fernandes, J., and Ingstrup, M. (2008). Semantic web ontologies for ambient intelligence. In *Proceedings of the 1st International Research Workshop on the Internet of Things and Services*.

[Helal, 2002] Helal, S. (2002). Standards for service discovery and delivery. *Pervasive Computing, IEEE*, 1(3):95–100.

[Hilia et al., 2011] Hilia, M., Chibani, A., Amirat, Y., and Djouani, K. (2011). Cross-organizational cooperation framework for security management in ubiquitous computing environment. In *Proceedings of the 23nd International Conference on Tools with Artificial Intelligence*, pages 464–471.

[Hilia et al., 2013a] Hilia, M., Chibani, A., and Djouani, K. (2013a). Trends and challenges in formal specification and verification of services composition in ambient assisted living applications. In *Proceeding of the 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013)*, volume 19, pages 540 – 547. Springer.

[Hilia et al., 2012] Hilia, M., Chibani, A., Djouani, K., and Amirat, Y. (2012). Semantic service composition framework for multidomain ubiquitous computing applications. In *Service-Oriented Computing - 10th International Conference, ICSOC 2012, Shanghai, China, November 12-15*, pages 450–467. Springer.

[Hilia et al., 2013b] Hilia, M., Chibani, A., Djouani, K., and Amirat, Y. (2013b). Formal specification and verification for ubiquitous robotic services composition. In *4th Workshop on Formal Methods for Robotics and Automation in conjunction with RSS2013*, Berlin,Germany.

[Itea2, 2008] Itea2 (2008). Soda : Service-oriented device & delivery architectures.

[ITEA2, 2010] ITEA2 (2010). Multipol itea2 project portal.

[Klai et al., 2006] Klai, K., M'Bareck, N. O. A., and Tata, S. (2006). Behavioral technique for workflow abstraction and matching. In *Business Process Management*, pages 477–483.

[Kopecký et al., 2007] Kopecký, J., Vitvar, T., Bournez, C., and Farrell, J. (2007). Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, 11(6):60–67.

[Kunze et al., 2011] Kunze, L., Roehm, T., and Beetz, M. (2011). Towards semantic robot description languages. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5589–5595.

[Lämmermann, 2002] Lämmermann, S. (2002). *Runtime service composition via logic-based program synthesis*. PhD thesis, KTH.

[Lin and Ishida, 2008] Lin, D. and Ishida, T. (2008). Interorganizational workflow collaboration based on local process views. In *Asia-Pacific Services Computing Conference*, pages 789–794, Los Alamitos, CA, USA. IEEE Computer Society.

[Liu et al., 2009] Liu, C., Li, Q., and Zhao, X. (2009). Challenges and opportunities in collaborative business process management: Overview of recent advances and introduction to the special issue. *Information Systems Frontiers*, 11(3):201–209.

[M. Birna van Riemsdijk et al., 2008] M. Birna van Riemsdijk, M. B., Hennicker, R., Wirsing, M., and Schroeder, A. (2008). Service specification and matchmaking using description logic. In *12th International Conference on Algebraic Methodology and Software Technology (AMAST'08)*, volume 5140, pages 392–406. Springer-Verlag.

[Majithia et al., 2004] Majithia, S., Shields, M., Taylor, I., and Wang, I. (2004). Triana: a graphical web service composition and execution toolkit. In *Web Services, 2004. Proceedings. IEEE International Conference on*, pages 514–521.

[Martin et al., 2007] Martin, D., Burstein, M., McDermott, D., McIlraith, S., Paolucci, M., Sycara, K., McGuinness, D., Sirin, E., and Srinivasan, N. (2007). Bringing semantics to web services with OWL-S. *World Wide Web*, 10(3):243–277.

[McCarthy, 1963] McCarthy, J. (1963). Situations, actions, and causal laws. Technical report, DTIC Document.

[McIlraith and Son, 2002] McIlraith, S. and Son, T. C. (2002). Adapting golog for composition of semantic web services. *KR*, 2:482–493.

[Mendler and Scheele, 2008] Mendler, M. and Scheele, S. (2008). Towards constructive description logics for abstraction and refinement. In *Proc. 21st Int'l Workshop Description Logics*.

[Mendler and Scheele, 2010] Mendler, M. and Scheele, S. (2010). Towards constructive DL for abstraction and refinement. *Journal of Automated Reasoning*, 44:207–243.

[Miglioli et al., 1988] Miglioli, P., Moscato, U., and Ornaghi, M. (1988). Pap: a logic programming system based on a constructive logic. In Boscarol, M., Carlucci Aiello, L., and Levi, G., editors, *Foundations of Logic and Functional Programming*, volume 306 of *Lecture Notes in Computer Science*, pages 141–156. Springer Berlin / Heidelberg.

[Miguel and Roberto, 2009] Miguel, Ángel, H. and Roberto, G. (Uganda, 2009). Fast development in ami through soa. In *IST-Africa 2009*.

[Mokarizadeh et al., 2009] Mokarizadeh, S., Grosso, A., Matskin, M., Kungas, P., and Haseeb, A. (2009). Applying semantic web service composition for action planning in multi-robot systems. In *Fourth International Conference on Internet and Web Applications and Services, ICIW 2009, Venice/Mestre, Italy*, pages 370 –376.

[MonAmI, 2011] MonAmI (2011). Monami - mainstreaming on ambient intelligence.

[Moore, 1996] Moore, C. A. (1996). Testing speech act theory and its applicability to EDI and other computer-processable messages. In *Twenty-Ninth Hawaii International Conference on System Science*, pages 30–38.

[Narayanan and McIlraith, 2002] Narayanan, S. and McIlraith, S. A. (2002). Simulation, verification and automated composition of web services. In *Proceedings of the 11th international conference on World Wide Web*, pages 77–88. ACM.

[Nipkow et al., 2002] Nipkow, T., Paulson, L. C., and Wenzel, M. (2002). *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer.

[OASIS, 2007] OASIS (2007). Web services business process execution language version 2.0. Padrão.

[Osman et al., 2009] Osman, T., Thakker, D., and Al-Dabass, D. (2009). Utilisation of case-based reasoning for semantic web services composition. *International Journal of Intelligent Information Technologies (IJIIT)*, 5(1):24–42.

[Papapanagiotou and Fleuriot, 2011] Papapanagiotou, P. and Fleuriot, J. (2011). A theorem proving framework for the formal verification of web services composition. *In Proceedings of the 7th International Workshop on Automated Specification and Verification of Web Systems*, pages 1–16.

[Papazoglou, 2008] Papazoglou, M. P. (2008). *Web Services - Principles and Technology*. Prentice Hall.

[Peer, 2004] Peer, J. (2004). A pddl based tool for automatic web service composition. In Ohlbach, H. and Schaffert, S., editors, *Principles and Practice of Semantic Web Reasoning*, volume 3208 of *Lecture Notes in Computer Science*, pages 149–163. Springer Berlin Heidelberg.

[Peer, 2005] Peer, J. (2005). Web service composition as AI planning : a survey. Technical report, University of St. Gallen, Switzerland.

[Preuveneers and Novais, 2012] Preuveneers, D. and Novais, P. (2012). A survey of software engineering best practices for the development of smart applications in ambient intelligence. *Journal of Ambient Intelligence and Smart Environments*, 4(3):149–162.

[Ramos et al., 2008] Ramos, C., Augusto, J. C., and Shapiro, D. (2008). Ambient intelligence - the next step for artificial intelligence. *IEEE Intelligent Systems*, 23(2):15–18.

[Rao and Küngas, 2004] Rao, J. and Küngas, P. (2004). Logic-based web services composition: From service description to process model. *In Proceedings of the International Conference on Web Services(ICWS)*, pages 446–453.

[Rao et al., 2006] Rao, J., Küngas, P., and Matskin, M. (2006). Composition of semantic web services using linear logic theorem proving. *Information Systems*, 31(4-5):340–360.

[Rao and Su, 2005] Rao, J. and Su, X. (2005). A survey of automated web service composition methods. In *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition(SWSWPC)*, pages 43–54. Springer.

[Russell and Norvig, 2009] Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition.

[Saffiotti and Broxvall, 2005] Saffiotti, A. and Broxvall, M. (2005). Peis ecologies: Ambient intelligence meets autonomous robotics. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*, pages 277–281. ACM.

[Saffiotti et al., 2008] Saffiotti, A., Broxvall, M., Gritti, M., LeBlanc, K., Lundh, R., Rashid, M. J., Seo, B., and Cho, Y.-J. (2008). The peis-ecology project: Vision and results. In *IROS*, pages 2329–2335. IEEE.

[Santofimia et al., 2011] Santofimia, M. J., Fahlman, S. E., del Toro, X., Moya, F., and Lopez, J. C. (2011). A semantic model for actions and events in ambient intelligence. *Engineering Applications of Artificial Intelligence*, 24(8):1432–1445.

[Sheng et al., 2009] Sheng, Q., Benatallah, B., Maamar, Z., and Ngu, A. (2009). Configurable composition and adaptive provisioning of web services. *IEEE Transactions on Services Computing*, 2(1):34–49.

[Sheshagiri et al., 2003] Sheshagiri, M., desJardins, M., and Finin, T. (2003). A Planner for Composing Services Described in DAML-S. In *Proceedings of the AAMAS Workshop on Web Services and Agent-based Engineering,*.

[Sirin et al., 2004] Sirin, E., Parsia, B., Wu, D., Hendler, J., and 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.

[Staab and Studer, 2009] Staab, S. and Studer, R. (2009). *Handbook on ontologies*. Springer.

[Stavropoulos et al., 2011a] Stavropoulos, T., Vrakas, D., and Vlahavas, I. (2011a). A survey of service composition in ambient intelligence environments. *Artificial Intelligence Review*, pages 1–24.

[Stavropoulos et al., 2011b] Stavropoulos, T., Vrakas, D., and Vlahavas, I. (2011b). A survey of service composition in ambient intelligence environments. *Journal of Artificial Intelligence Review*, pages 1–24.

[Stein et al., 2006] Stein, S., Payne, T. R., and Jennings, N. R. (2006). Flexible provisioning of semantic web service workflows using a qos ontology. In *International Semantic Web Conference (ISWC)*.

[Tenorth et al., 2012] Tenorth, M., Perzylo, A., Lafrenz, R., and Beetz, M. (2012). The roboearth language: Representing and exchanging knowledge about actions, objects, and environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1284–1289.

[Ter Beek et al., 2006] Ter Beek, M., Bucchiarone, A., and Gnesi, S. (2006). A survey on service composition approaches: From industrial standards to formal methods. Technical report, Technical Report 2006-TR-15.

[Ter Beek et al., 2007] Ter Beek, M. H., Bucchiarone, A., and Gnesi, S. (2007). Formal methods for service composition. *Annals of Mathematics, Computing & Teleinformatics*, 1(5):1–10.

[Tigli et al., 2011] Tigli, J.-Y., Lavirotte, S., Rey, G., Hourdin, V., and Riveill, M. (2011). Lightweight service oriented architecture for pervasive computing. *CoRR*, abs/1102.5193.

[Tondello and Siqueira, 2008] Tondello, G. F. and Siqueira, F. (2008). The qos-mo ontology for semantic qos modeling. In *Proceedings of the 2008 ACM symposium on Applied computing*, SAC '08, pages 2336–2340, New York, NY, USA. ACM.

[Traverso and Pistore, 2004] Traverso, P. and Pistore, M. (2004). Automated composition of semantic web services into executable processes. *Journal of The Semantic Web-ISWC*, 3298:380–394.

[Törmä et al., 2008] Törmä, S., Villstedt, J., Lehtinen, V., Oliver, I., and Luukkala, V. (2008). Semantic web services : A survey. Technical Report KK-TKO-B158, Laboratory of Software Technology.

[Troelstra and Schwichtenberg, 2000] Troelstra, A. and Schwichtenberg, H. (2000). *Basic proof theory*, volume 43. Cambridge Univ Pr.

[Troelstra, 1999] Troelstra, A. S. (1999). From constructivism to computer science. *Theoretical Computer Science*, 211(1):233–252.

[Urbieta et al., 2008a] Urbieta, A., Azketa, E., Gomez, I., Parra, J., and Arana, N. (2008a). Analysis of effects- and Preconditions-Based service representation in ubiquitous computing environments. In *Proceedings of the International Conference on Semantic Computing*, pages 378–385.

[Urbieta et al., 2008b] Urbieta, A., Barrutieta, G., Parra, J., and Uribarren, A. (2008b). A survey of dynamic service composition approaches for ambient systems. In *Proceedings of the 2008 Ambi-Sys workshop on Software Organisation and MonIToring of Ambient Systems*, SOMITAS '08, pages 1:1–1:8, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[van Der Aalst et al., 2003] van Der Aalst, W. M. P., Ter Hofstede, A. H. M., Kiepuszewski, B., and Barros, A. P. (2003). Workflow patterns. *Distributed and parallel databases*, 14(1):5–51.

[Wadler, 2000] Wadler, P. (2000). Proofs are programs: 19th century logic and 21st century computing. *Dr Dobb's Journal*, page 313. Cited by 0007.

[Waldinger, 2001] Waldinger, R. J. (2001). Web agents cooperating deductively. In *Proceedings of the First International Workshop on Formal Approaches to Agent-Based Systems-Revised Papers*, FAABS '00, pages 250–262, London, UK, UK. Springer-Verlag.

[Walsh, 2002] Walsh, A. E. (2002). *UDDI, SOAP, and WSDL: The Web Services Specification Reference Book*. Prentice Hall Professional Technical Reference.

[Weigand and van den Heuvel, 2002] Weigand, H. and van den Heuvel, W. J. (2002). Cross-organizational workflow integration using contracts. *Decision Support Systems*, 33(3):247–265.

[Weiser, 1991] Weiser, M. (1991). The Computer for the Twenty-First Century. *Scientific American*, 265(3):94–104.

[Yachir et al., 2009] Yachir, A., Tari, K., Amirat, Y., Chibani, A., and Badache, N. (2009). Qos based framework for ubiquitous robotic services composition. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2019–2026.

[Ye et al., 2007] Ye, J., Coyle, L., Dobson, S., and Nixon, P. (2007). Ontology-based models in pervasive computing systems. *Knowl. Eng. Rev.*, 22(4):315–347.

[Yue et al., 2004] Yue, K., Wang, X.-L., and Zhou, A.-Y. (2004). Underlying techniques for web services: A survey. *Journal of software*, 15(3):428–442. Cited by 0336.

[Zeng et al., 2004] Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., and Chang, H. (2004). Qos-aware middleware for web services composition. *Software Engineering, IEEE Transactions on*, 30(5):311–327.

[Zhang et al., 2010] Zhang, B., Shi, Y., and Xiao, X. (2010). A policy-driven service composition method for adaptation in pervasive computing environment. *The Computer Journal*, 53(2):152–165.

Annexe

L'exemple de preuve notifie un programme de rééducation programmé le *10-10-2012*. Cet événement est instancié par *event-1* pour une personne instancié par l'instance *personne*. La personne exerce le programme *reeducation-program* avec l'equipement *equipement-ball*. Les deux services de création d'événement et de notification sont prouvés corrects dans l'environnement de composition et ensuite sont composés par la règle de composition séquence en un service composite appelé Composition prouvé correct dans l'environnement de composition multi-domaine.

theory *NotifyService*
imports *biell*
begin
term (*SP* (*NotifyProgram*)
(*event*)
(∗ *Pre−Condition* ∗)(*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*)))))
(∗ *post−Condition* ∗)(*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))))

lemma (*IT* ({*reeducation-program*,*person*})
(*ConceptF* *event* (*Pre-s* (*SF* (*NotifyProgram*)
(*event*)
(*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*)))))
(*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*))))
))))
⊆ *P*
apply(*auto*)
oops

lemma {*et* (*tt*, *It.Ex* (*reeducation-program*, *et* (*tt*, *It.Ex* (*person*, *tt*)))) } ⊆
(*IT*
{*reeducation-program*,*person*}
(*ConceptF*
event-1 (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*)))))))
apply *auto*
done

lemma *realiz M* (*et* (*tt*, *It.Ex* (*reeducation-program*, *et* (*tt*, *It.Ex* (*person*, *tt*))))) (*ConceptF event-1* (*Pre-s* (*SF* (*NotifyProgram*) (*event*) (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*))))) (*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))))))
apply(*auto*)
oops

lemma ⟦
interp-i M event-1 ∈ *interp-c M NotifyProgramEvent*;
(*interp-i M event-1*, *interp-i M reeducation-program*) ∈ *InterpR M hasReeducationProgram*;
interp-i M reeducation-program ∈ *interp-c M ReeducationProgram*;
(*interp-i M reeducation-program*, *interp-i M person*) ∈ *InterpR M isRelatedTo*;
interp-i M person ∈ *interp-c M Person*
⟧ ⟹*realiz M* (*et* (*tt*, *It.Ex* (*reeducation-program*, *et* (*tt*, *It.Ex* (*person*, *tt*))))) (*ConceptF event-1* (*Pre-s* (*SF* (*NotifyProgram*) (*event*) (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*)))))
(*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))))))
apply *auto*
done

lemma (*IT* ({*equipment-ball*})
(*ConceptF event* (*Post-s* (*SF* (*NotifyProgram*)
(*event*)
(*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*)))))
(*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))))))) ⊆ *P*
apply(*auto*)
oops

lemma {*ou* (*p2*, *et* (*tt*, *It.Ex* (*equipment-ball*, *tt*))) } ⊆ (*IT* {*equipment-ball*}
(*ConceptF*
event-1 (*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*)
(*AtomC MedicalEquipement*))))))
apply *auto*
done

lemma *realiz M* (*ou* (*p2*, *et* (*tt*, *It.Ex* (*equipment-ball*, *tt*)))) (*ConceptF event-1* (*Post-s* (*SF* (*NotifyProgram*)
(*event*) (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*)))))
(*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))))))
apply(*auto*)
oops

lemma

$[\![$

interp-i M event-1 $\in$ *interp-c M AcceptedProgramedEvent*;

(*interp-i M event-1*, *interp-i M equipment-ball*) $\in$ *InterpR M hasEquipement*;

interp-i M equipment-ball $\in$ *interp-c M MedicalEquipement*

$]\!] \Longrightarrow$

realiz M (*ou* (*p2*, *et* (*tt*, *It.Ex* (*equipment-ball*, *tt*)))) (*ConceptF event-1* (*Post-s* (*SF* (*NotifyProgram*) (*event*) (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*)))))

(*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))))))

apply *auto*

done

lemma (*IT-setF* (*N*) ({(*NotifyProgramEvent*) $\sqsubseteq$ (*AtomC Event*),

(*NotifyProgramEvent*) $\sqsubseteq$ (*AllC*(*hasReeducationProgram*) (*AtomC ReeducationProgram*)),

(*ReeducationProgram*) $\sqsubseteq$ (*AllC*(*isRelatedTo*)(*AtomC Person*)),

(*AcceptedProgramedEvent*) $\sqsubseteq$ (*AllC*(*hasEquipement*)(*AtomC MedicalEquipement*)),

(*ProgramedCalendarEvent*) $\sqsubseteq$ (*AllC*(*hasDate*)(*AtomC Date*)),

(*AcceptedEvent*) $\sqsubseteq$ (*AtomC Event*),

(*AcceptedProvisioningEvent*) $\sqsubseteq$ (*AndC* (*AtomC ProvisioningEvent*) (*AtomC AcceptedEvent*)),

(*ReeducationProgram*) $\sqsubseteq$ (*AllC* (*hasDestination*) (*AtomC Person*)),

(*CalendarEvent*) $\sqsubseteq$ (*AllC*(*hasDescription*)(*AtomC Description*)),

(*ProvisioningEvent*) $\sqsubseteq$ (*AtomC Event*),

(*ProvisioningEvent*) $\sqsubseteq$ (*AllC*(*hasAction*)(*AtomC ProvisioningAction*)),

(*ProvisioningAction*) $\sqsubseteq$ (*AllC* (*hasMessage*)(*AtomC ProvisioningMessage*)),

(*RefusedEvent*) $\sqsubseteq$ (*AndC* (*AtomC Event*) (*NotC* (*AtomC AcceptedEvent*))) })) $\subseteq$ *P*

apply (*auto*)

oops

term *is-model-E* (*M*) ({(*NotifyProgramEvent*) $\sqsubseteq$ (*AtomC Event*),(*NotifyProgramEvent*) $\sqsubseteq$ (*AllC*(*hasReeducationProgram*)

(*AtomC ReeducationProgram*)),

(*ReeducationProgram*) $\sqsubseteq$ (*AllC* (*isRelatedTo*)(*AtomC Person*)),

(*AcceptedProgramedEvent*) $\sqsubseteq$ (*AllC*(*hasEquipement*)(*AtomC MedicalEquipement*)),

(*ProgramedCalendarEvent*) $\sqsubseteq$ (*AllC*(*hasDate*)(*AtomC Date*)),(*AcceptedEvent*) $\sqsubseteq$ (*AtomC Event*),
(*AcceptedProvisioningEvent*) $\sqsubseteq$ (*AndC* (*AtomC ProvisioningEvent*) (*AtomC AcceptedEvent*)),(*ReeducationProgram*)
$\sqsubseteq$ (*AllC* (*hasDestination*) (*AtomC Person*)),
(*CalendarEvent*) $\sqsubseteq$ (*AllC*(*hasDescription*)(*AtomC Description*)),(*ProvisioningEvent*) $\sqsubseteq$ (*AtomC Event*),
(*ProvisioningEvent*) $\sqsubseteq$ (*AllC*(*hasAction*)(*AtomC ProvisioningAction*)),(*ProvisioningAction*) $\sqsubseteq$ (*AllC* (*hasMessage*)(*AtomC ProvisioningMessage*)),
(*RefusedEvent*) $\sqsubseteq$ (*AndC* (*AtomC Event*) (*NotC* (*AtomC AcceptedEvent*)))},
{*ou* (*p1*, *tt*),*ou* (*p2*, *tt*)},{
(*SF* (*NotifyProgram*) (*event*) (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*)
(*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*)))))
(*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))) , Φ*NotifyProgram*)})

lemma *is-model-E* (*M*) ({(*NotifyProgramEvent*) $\sqsubseteq$ (*AtomC Event*),
(*NotifyProgramEvent*) $\sqsubseteq$ (*AllC*(*hasReeducationProgram*) (*AtomC ReeducationProgram*)),
(*ReeducationProgram*) $\sqsubseteq$ (*AllC*(*isRelatedTo*)(*AtomC Person*)),
(*AcceptedProgramedEvent*) $\sqsubseteq$ (*AllC*(*hasEquipement*)(*AtomC MedicalEquipement*)),
(*ProgramedCalendarEvent*) $\sqsubseteq$ (*AllC*(*hasDate*)(*AtomC Date*)),
(*AcceptedEvent*) $\sqsubseteq$ (*AtomC Event*),
(*AcceptedProvisioningEvent*) $\sqsubseteq$ (*AndC* (*AtomC ProvisioningEvent*) (*AtomC AcceptedEvent*)),
(*ReeducationProgram*) $\sqsubseteq$ (*AllC* (*hasDestination*) (*AtomC Person*)),
(*CalendarEvent*) $\sqsubseteq$ (*AllC*(*hasDescription*)(*AtomC Description*)),
(*ProvisioningEvent*) $\sqsubseteq$ (*AtomC Event*),
(*ProvisioningEvent*) $\sqsubseteq$ (*AllC*(*hasAction*)(*AtomC ProvisioningAction*)),
(*ProvisioningAction*) $\sqsubseteq$ (*AllC* (*hasMessage*)(*AtomC ProvisioningMessage*)),
(*RefusedEvent*) $\sqsubseteq$ (*AndC* (*AtomC Event*) (*NotC* (*AtomC AcceptedEvent*))) },{*ou* (*p1*, *tt*),*ou* (*p2*, *tt*)},
{(*SF* (*NotifyProgram*) (*event*) (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*)
(*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*)))))
(*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))) , Φ*NotifyProgram*)})
$\Longrightarrow$ *SCER*({
(*NotifyProgramEvent*) $\sqsubseteq$ (*AtomC Event*),
(*NotifyProgramEvent*) $\sqsubseteq$ (*AllC*(*hasReeducationProgram*) (*AtomC ReeducationProgram*)),
(*ReeducationProgram*) $\sqsubseteq$ (*AllC*(*isRelatedTo*)(*AtomC Person*)),
(*AcceptedProgramedEvent*) $\sqsubseteq$ (*AllC*(*hasEquipement*)(*AtomC MedicalEquipement*)),
(*ProgramedCalendarEvent*) $\sqsubseteq$ (*AllC*(*hasDate*)(*AtomC Date*)),
(*AcceptedEvent*) $\sqsubseteq$ (*AtomC Event*),
(*AcceptedProvisioningEvent*) $\sqsubseteq$ (*AndC* (*AtomC ProvisioningEvent*) (*AtomC AcceptedEvent*)),
(*ReeducationProgram*) $\sqsubseteq$ (*AllC* (*hasDestination*) (*AtomC Person*)),

(*CalendarEvent*) $\sqsubseteq$ (*AllC*(*hasDescription*)(*AtomC Description*)),
(*ProvisioningEvent*) $\sqsubseteq$ (*AtomC Event*),
(*ProvisioningEvent*) $\sqsubseteq$ (*AllC*(*hasAction*)(*AtomC ProvisioningAction*)),
(*ProvisioningAction*) $\sqsubseteq$ (*AllC* (*hasMessage*)(*AtomC ProvisioningMessage*)),
(*RefusedEvent*) $\sqsubseteq$ (*AndC* (*AtomC Event*) (*NotC* (*AtomC AcceptedEvent*))) },{ *ou* (*p1*, *tt*),*ou* (*p2*, *tt*)},
{(*SF* (*NotifyProgram*) (*event*) (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*)
(*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*)))))
(*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC
MedicalEquipement)))) , Φ*NotifyProgram*)})

(*SF DD x* (*Pre-s* (*SF* (*NotifyProgram*) (*event*) (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*)
(*AndC* (*AtomC ReeducationProgram*) (*SomC* (*hasDestination*)(*AtomC Person*)))))
(*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC
MedicalEquipement))))))

(*Post-s* (*SF* (*NotifyProgram*) (*event*) (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*)
(*AndC* (*AtomC ReeducationProgram*) (*SomC* (*hasDestination*)(*AtomC Person*)))))
(*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC
MedicalEquipement)))))))

apply(*rule ENVE*)
apply *assumption*
done

end

theory *CalendarEvent*
imports *biell*
begin

term (*SP* (*CreateEvent*)
(*event*)
(∗ *Pre−Condition* ∗)(*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*)))
(∗ *post−Condition* ∗)(*AndC* (*AtomC ProgramedCalendarEvent*) (*SomC* (*hasDate*) (*AtomC Date*))))

lemma (*IT* ({*reeducation-event*})
(*ConceptF* *event* (*Pre-s* (*SF* (*CreateEvent*)
(*event*)
(*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*)))
(*AndC* (*AtomC ProgramedCalendarEvent*) (*SomC* (*hasDate*) (*AtomC Date*)))
))))
⊆ *P*
apply(*auto*)
oops

lemma {*et* (*tt*, *It.Ex* (*reeducation-event*, *tt*)) } ⊆
(*IT*
{*reeducation-event*}
(*ConceptF*
event-1(*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*)))))
apply *auto*
done

lemma *realiz M*(*et* (*tt*, *It.Ex* (*reeducation-event*, *tt*))) (*ConceptF event-1* (*Pre-s* (*SF* (*CreateEvent*)
(*event*) (*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*)))
(*AndC* (*AtomC ProgramedCalendarEvent*) (*SomC* (*hasDate*) (*AtomC Date*))))))
apply(*auto*)
oops

lemma ⟦
interp-i M event-1 ∈ *interp-c M CalendarEvent*;
(*interp-i M event-1*, *interp-i M reeducation-event*) ∈ *InterpR M hasDescription*;
interp-i M reeducation-event ∈ *interp-c M Description*

$\rrbracket \Longrightarrow$*realiz M (et (tt, It.Ex (reeducation-event, tt))) (ConceptF event-1 (Pre-s (SF (CreateEvent) (event)*
(AndC (AtomC CalendarEvent) (SomC (hasDescription) (AtomC Description)))
(AndC (AtomC ProgramedCalendarEvent) (SomC (hasDate) (AtomC Date))))))
apply *auto*
done

lemma *(IT ({date-10-10-2012})*
(ConceptF event (Post-s (SF (CreateEvent)
(event)
(AndC (AtomC CalendarEvent) (SomC (hasDescription) (AtomC Description)))
(AndC (AtomC ProgramedCalendarEvent) (SomC (hasDate) (AtomC Date)))))))
$\subseteq P$
apply(*auto*)
oops

lemma *{et (tt, It.Ex (date-10-10-2012, tt)) } ⊆*
(IT
{date-10-10-2012}
(ConceptF
event-1(AndC (AtomC ProgramedCalendarEvent) (SomC (hasDate) (AtomC Date)))))
apply *auto*
done

lemma *realiz M(et (tt, It.Ex (date-10-10-2012, tt))) (ConceptF event-1 (Post-s (SF (CreateEvent)*
(event) (AndC (AtomC CalendarEvent) (SomC (hasDescription) (AtomC Description)))
(AndC (AtomC ProgramedCalendarEvent) (SomC (hasDate) (AtomC Date))))))
apply(*auto*)
oops

lemma
$\llbracket$
interp-i M event-1 ∈ interp-c M ProgramedCalendarEvent;
(interp-i M event-1, interp-i M date-10-10-2012) ∈ InterpR M hasDate;
interp-i M date-10-10-2012 ∈ interp-c M Date
$\rrbracket \Longrightarrow$
realiz M (et (tt, It.Ex (date-10-10-2012, tt))) (ConceptF event-1 (Post-s (SF (CreateEvent) (event)
(AndC (AtomC CalendarEvent) (SomC (hasDescription) (AtomC Description)))
(AndC (AtomC ProgramedCalendarEvent) (SomC (hasDate) (AtomC Date))))))
apply *auto*

done

lemma (*IT-setF* (*N*) ({(*NotifyProgramEvent*) ⊑ (*AtomC Event*),
(*NotifyProgramEvent*) ⊑ (*AllC*(*hasReeducationProgram*) (*AtomC ReeducationProgram*)),
(*ReeducationProgram*) ⊑ (*AllC*(*isRelatedTo*)(*AtomC Person*)),
(*AcceptedProgramedEvent*) ⊑ (*AllC*(*hasEquipement*)(*AtomC MedicalEquipement*)),
(*ProgramedCalendarEvent*) ⊑ (*AllC*(*hasDate*)(*AtomC Date*)),
(*AcceptedEvent*) ⊑ (*AtomC Event*),
(*AcceptedProvisioningEvent*) ⊑ (*AndC* (*AtomC ProvisioningEvent*) (*AtomC AcceptedEvent*)),
(*ReeducationProgram*) ⊑ (*AllC* (*hasDestination*) (*AtomC Person*)),
(*CalendarEvent*) ⊑ (*AllC*(*hasDescription*)(*AtomC Description*)),
(*ProvisioningEvent*) ⊑ (*AtomC Event*),
(*ProvisioningEvent*) ⊑ (*AllC*(*hasAction*)(*AtomC ProvisioningAction*)),
(*ProvisioningAction*) ⊑ (*AllC* (*hasMessage*)(*AtomC ProvisioningMessage*)),
(*RefusedEvent*) ⊑ (*AndC* (*AtomC Event*) (*NotC* (*AtomC AcceptedEvent*))) })) ⊆ *P*
apply (*auto*)
oops

term *is-model-E* (*M*)({(*NotifyProgramEvent*) ⊑ (*AtomC Event*),
(*NotifyProgramEvent*) ⊑ (*AllC*(*hasReeducationProgram*) (*AtomC ReeducationProgram*)),
(*ReeducationProgram*) ⊑ (*AllC*(*isRelatedTo*)(*AtomC Person*)),
(*AcceptedProgramedEvent*) ⊑ (*AllC*(*hasEquipement*)(*AtomC MedicalEquipement*)),
(*ProgramedCalendarEvent*) ⊑ (*AllC*(*hasDate*)(*AtomC Date*)),
(*AcceptedEvent*) ⊑ (*AtomC Event*),
(*AcceptedProvisioningEvent*) ⊑ (*AndC* (*AtomC ProvisioningEvent*) (*AtomC AcceptedEvent*)),
(*ReeducationProgram*) ⊑ (*AllC* (*hasDestination*) (*AtomC Person*)),
(*CalendarEvent*) ⊑ (*AllC*(*hasDescription*)(*AtomC Description*)),
(*ProvisioningEvent*) ⊑ (*AtomC Event*),
(*ProvisioningEvent*) ⊑ (*AllC*(*hasAction*)(*AtomC ProvisioningAction*)),
(*ProvisioningAction*) ⊑ (*AllC* (*hasMessage*)(*AtomC ProvisioningMessage*)),
(*RefusedEvent*) ⊑ (*AndC* (*AtomC Event*) (*NotC* (*AtomC AcceptedEvent*))) },{ *ou* (*p1*, *tt*),*ou* (*p2*, *tt*)},

{(*SF* (*CreateEvent*) (*event*) (*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*)))
(*AndC* (*AtomC ProgramedCalendarEvent*) (*SomC* (*hasDate*) (*AtomC Date*))) , Φ*CreateEvent*)})

lemma *is-model-E* (*M*)({(*NotifyProgramEvent*) ⊑ (*AtomC Event*),
(*NotifyProgramEvent*) ⊑ (*AllC*(*hasReeducationProgram*) (*AtomC ReeducationProgram*)),
(*ReeducationProgram*) ⊑ (*AllC*(*isRelatedTo*)(*AtomC Person*)),
(*AcceptedProgramedEvent*) ⊑ (*AllC*(*hasEquipement*)(*AtomC MedicalEquipement*)),

(*ProgramedCalendarEvent*) $\sqsubseteq$ (*AllC*(*hasDate*)(*AtomC Date*)),
(*AcceptedEvent*) $\sqsubseteq$ (*AtomC Event*),
(*AcceptedProvisioningEvent*) $\sqsubseteq$ (*AndC* (*AtomC ProvisioningEvent*) (*AtomC AcceptedEvent*)),
(*ReeducationProgram*) $\sqsubseteq$ (*AllC* (*hasDestination*) (*AtomC Person*)),
(*CalendarEvent*) $\sqsubseteq$ (*AllC*(*hasDescription*)(*AtomC Description*)),
(*ProvisioningEvent*) $\sqsubseteq$ (*AtomC Event*),
(*ProvisioningEvent*) $\sqsubseteq$ (*AllC*(*hasAction*)(*AtomC ProvisioningAction*)),
(*ProvisioningAction*) $\sqsubseteq$ (*AllC* (*hasMessage*)(*AtomC ProvisioningMessage*)),
(*RefusedEvent*) $\sqsubseteq$ (*AndC* (*AtomC Event*) (*NotC* (*AtomC AcceptedEvent*))) },{ *ou* (*p1*, *tt*),*ou* (*p2*, *tt*)},{(*SF*
(*CreateEvent*) (*event*) (*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*)))
(*AndC* (*AtomC ProgramedCalendarEvent*) (*SomC* (*hasDate*) (*AtomC Date*))) , Φ*CreateEvent*)})$\Longrightarrow$*SCER*({(*NotifyProgram*
$\sqsubseteq$ (*AtomC Event*),
(*NotifyProgramEvent*) $\sqsubseteq$ (*AllC*(*hasReeducationProgram*) (*AtomC ReeducationProgram*)),
(*ReeducationProgram*) $\sqsubseteq$ (*AllC*(*isRelatedTo*)(*AtomC Person*)),
(*AcceptedProgramedEvent*) $\sqsubseteq$ (*AllC*(*hasEquipement*)(*AtomC MedicalEquipement*)),
(*ProgramedCalendarEvent*) $\sqsubseteq$ (*AllC*(*hasDate*)(*AtomC Date*)),
(*AcceptedEvent*) $\sqsubseteq$ (*AtomC Event*),
(*AcceptedProvisioningEvent*) $\sqsubseteq$ (*AndC* (*AtomC ProvisioningEvent*) (*AtomC AcceptedEvent*)),
(*ReeducationProgram*) $\sqsubseteq$ (*AllC* (*hasDestination*) (*AtomC Person*)),
(*CalendarEvent*) $\sqsubseteq$ (*AllC*(*hasDescription*)(*AtomC Description*)),
(*ProvisioningEvent*) $\sqsubseteq$ (*AtomC Event*),
(*ProvisioningEvent*) $\sqsubseteq$ (*AllC*(*hasAction*)(*AtomC ProvisioningAction*)),
(*ProvisioningAction*) $\sqsubseteq$ (*AllC* (*hasMessage*)(*AtomC ProvisioningMessage*)),
(*RefusedEvent*) $\sqsubseteq$ (*AndC* (*AtomC Event*) (*NotC* (*AtomC AcceptedEvent*))) },{ *ou* (*p1*, *tt*),*ou* (*p2*, *tt*)},{(*SF*
(*CreateEvent*) (*event*) (*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*)))
(*AndC* (*AtomC ProgramedCalendarEvent*) (*SomC* (*hasDate*) (*AtomC Date*))) , Φ*CreateEvent*)})
(*SF SS x* (*Pre-s* (*SF* (*CreateEvent*) (*event*) (*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*)
(*AtomC Description*)))
(*AndC* (*AtomC ProgramedCalendarEvent*) (*SomC* (*hasDate*) (*AtomC Date*)))))
(*Post-s*(*SF* (*CreateEvent*) (*event*) (*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC*
Description)))
(*AndC* (*AtomC ProgramedCalendarEvent*) (*SomC* (*hasDate*) (*AtomC Date*))))))
apply (*rule ENVE*)
apply *assumption*
done

end

theory *Composition*

imports *NotifyService CalendarEvent*

begin

term *is-model-E* (*M*) ({(*NotifyProgramEvent*) ⊑ (*AtomC Event*),(*NotifyProgramEvent*) ⊑ (*AllC*(*hasReeducationProgram*) (*AtomC ReeducationProgram*)),
(*ReeducationProgram*) ⊑ (*AllC* (*isRelatedTo*)(*AtomC Person*)),
(*AcceptedProgramedEvent*) ⊑ (*AllC*(*hasEquipement*)(*AtomC MedicalEquipement*)),
(*ProgramedCalendarEvent*) ⊑ (*AllC*(*hasDate*)(*AtomC Date*)),(*AcceptedEvent*) ⊑ (*AtomC Event*),
(*AcceptedProvisioningEvent*) ⊑ (*AndC* (*AtomC ProvisioningEvent*) (*AtomC AcceptedEvent*)),(*ReeducationProgram*)
⊑ (*AllC* (*hasDestination*) (*AtomC Person*)),
(*CalendarEvent*) ⊑ (*AllC*(*hasDescription*)(*AtomC Description*)),(*ProvisioningEvent*) ⊑ (*AtomC Event*),
(*ProvisioningEvent*) ⊑ (*AllC*(*hasAction*)(*AtomC ProvisioningAction*)),(*ProvisioningAction*) ⊑ (*AllC* (*hasMessage*)(*AtomC ProvisioningMessage*)),
(*RefusedEvent*) ⊑ (*AndC* (*AtomC Event*) (*NotC* (*AtomC AcceptedEvent*)))},
{*ou* (*p1*, *tt*),*ou* (*p2*, *tt*)},
{
(*SF* (*NotifyProgram*) (*event*) (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*)
(*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*)))))
(*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))) , Φ*NotifyProgram*),
(*SF* (*CreateEvent*) (*event*) (*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*)))
(*AndC* (*AtomC ProgramedCalendarEvent*) (*SomC* (*hasDate*) (*AtomC Date*))) , Φ*CreateEvent*)})

term (*SP* (*CreateAndNotify*)
(*event*)
(∗ *Pre*−*Condition* ∗)
(*AndC*
(*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*)))))
(*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*))))

(∗ *post*−*Condition* ∗)
(*OrC* (*AtomC RefusedEvent*)
(*AndC*
(*AndC* (*AtomC AcceptedProgramedEvent*) (*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))
(*AndC* (*AtomC ProgramedCalendarEvent*) (*SomC* (*hasDate*) (*AtomC Date*)))

)
)
)

lemma ⟦*SCER* (*E*)
(*SF* (*NotifyProgram*) *x* (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*))))) (*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))));
SCER (*E*)
(*SF* (*CreateEvent*) *x* (*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*))) (*AndC* (*AtomC ProgramedCalendarEvent*) (*SomC* (*hasDate*) (*AtomC Date*))));

(*T* ∪ {*ConceptF x* (*AndC* (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*))))(*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*))))}
⊢ *ConceptF x* (*Pre-s* (*SF* (*NotifyProgram*) *x* (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*))))) (*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))))));

(*T* ∪ {*ConceptF x* (*AndC* (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*))))(*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*))))}
⊢ *ConceptF x* (*Pre-s* (*SF* (*CreateEvent*) *x* (*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*))) (*AndC* (*AtomC ProgramedCalendarEvent*) (*SomC* (*hasDate*) (*AtomC Date*))))));

(*T* ∪ {*ConceptF x* (*AndC* (*Post-s* (*SF* (*NotifyProgram*) *x* (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*)))))) (*OrC* (*AtomC RefusedEvent*) (*AndC* (*AtomC AcceptedProgramedEvent*)(*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))))) (*Post-s* (*SF* (*CreateEvent*) *x* (*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*))) (*AndC* (*AtomC ProgramedCalendarEvent*) (*SomC* (*hasDate*) (*AtomC Date*))))))}
⊢ *ConceptF x* ((*OrC* (*AtomC RefusedEvent*)(*AndC*(*AndC* (*AtomC AcceptedProgramedEvent*) (*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))(*AndC* (*AtomC ProgramedCalendarEvent*) (*SomC* (*hasDate*) (*AtomC Date*)))))))
⟧
⟹ *SCER* (*E*) (*SF CreateAndNotify x* (*AndC* (*AndC* (*AtomC NotifyProgramEvent*)(*SomC* (*hasReeducationProgram*) (*AndC* (*AtomC ReeducationProgram*) (*SomC* (*isRelatedTo*)(*AtomC Person*))))(*AndC* (*AtomC CalendarEvent*) (*SomC* (*hasDescription*) (*AtomC Description*)))) (*OrC* (*AtomC RefusedEvent*)(*AndC*(*AndC* (*AtomC AcceptedProgramedEvent*) (*SomC* (*hasEquipement*) (*AtomC MedicalEquipement*)))(*AndC* (*AtomC Programed-

CalendarEvent) (*SomC* (*hasDate*) (*AtomC Date*))))))

apply (*metis AND*)
done

end