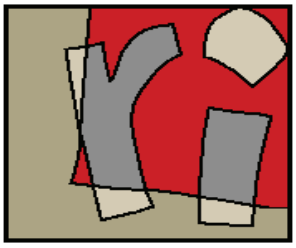
Informations personnelles sensibles aux contextes : modélisation, interrogation et composition

Rania Khéfifi



UNIVERSITÉ PARIS-SUD

ÉCOLE DOCTORALE 427 :
INFORMATIQUE PARIS SUD

Laboratoire de Recherche en Informatique

THÈSE DE DOCTORAT EN INFORMATIQUE

par

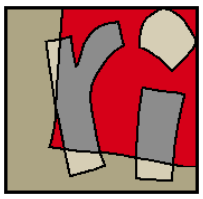
Rania KHÉFIFI

Informations personnelles sensibles aux contextes : modélisation, interrogation et composition

Date de soutenance : 26/09/2014

Composition du jury :

Directeur de thèse :	Pascal POIZAT	Professeur (Université Paris Ouest)
Rapporteurs :	Salima BENBERNOU	Professeur (Université Paris Descartes)
	Mohand-Saïd HACID	Professeur (Université Lyon 1)
Examinateurs :	Anne VILNAT	Professeur (Université Paris Sud)
	Kais KLAI	Maître de conférences (Université Paris 13)
	Fatiha SAÏS	Maître de conférences (Université Paris Sud)



Université Paris Sud

École Doctorale Informatique Paris Sud

Laboratoire de Recherche en Informatique

Thèse de Doctorat en Informatique

Informations personnelles sensibles aux contextes : modélisation, interrogation et composition

Présentée par :
Rania KHÉFIFI
pour l'obtention du
Doctorat de l'université Paris-Sud

Jury

Pr. Salima Benbernou	Université Paris Descartes	Rapporteur
Pr. Mohand-Saïd Hacid	Université Lyon 1	Rapporteur
Pr. Anne Vilnat	Université Paris Sud	Examinatrice
Dr. Kais Klai	Université Paris 13	Examinateur
Pr. Pascal Poizat	Université Paris Ouest	Directeur de thèse
Dr. Fatiha Saïs	Université Paris Sud	Co-directrice de thèse

soutenue publiquement le 26 Septembre 2014

Remerciements

À l'issue de la rédaction de ce manuscrit, je suis convaincue que la thèse est loin d'être un travail solitaire. En effet, je n'aurais jamais pu réaliser ce travail de recherche sans le soutien d'un grand nombre de personnes dont la générosité, la bonne humeur et la confiance m'ont permis de progresser dans cette phase délicate qui est "l'apprentissage de la recherche". Il me sera donc très difficile de nommer tout le monde dans ces modestes lignes.

Je souhaite remercier en premier lieu mon directeur de thèse, Pascal POIZAT, professeur des universités au Laboratoire d'Informatique de Paris 6 (LIP6). Je lui suis également reconnaissante pour le temps conséquent qu'il m'a accordé, ses qualités pédagogiques et scientifiques, sa franchise et sa sympathie. J'ai beaucoup appris à ses côtés et je lui adresse ma gratitude pour tout cela.

J'adresse également mes chaleureux remerciements à ma co-directrice de thèse, Fatiha SAÏS, Maître de conférences au Laboratoire de Recherche en Informatique (LRI), pour son attention de tout instant sur mes travaux, pour ses conseils avisés et son écoute qui ont été prépondérants pour la réussite de cette thèse, sa confiance et sa gentillesse. J'ai pris un grand plaisir à travailler avec elle.

Je tiens à remercier tous les membres du jury qui m'ont fait l'honneur d'avoir accepter d'examiner mon travail de thèse. Je remercie particulièrement Mme Salima BENBERNOU, professeur des universités à l'Université Paris Descartes et M. Mohand-Saïd HACID professeur des universités au Laboratoire d'Informatique en Image et Systèmes d'information (LIRIS) d'avoir accepté de rapporter mon travail de thèse, d'en avoir fait une lecture détaillée, et d'avoir émis des commentaires pertinents. Je remercie également Mme Anne VILNAT professeur des universités au Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur (LIMSI) et M. Kais KLAI Maître de conférence au Laboratoire d'Informatique de l'Université Paris Nord (LIPN) d'avoir accepté d'examiner ce travail.

J'adresse mes remerciements les plus sincères à Bouba pour les maux de tête que je lui causé lors de la relecture et de la correction de mon rapport de thèse.

Je tiens également à remercier les membres des équipes administratives et techniques de l'Université Paris Sud et du Laboratoire de Recherche en Informatique pour leur disponibilité et leur gentillesse. Je remercie également les membres des équipes de recherche auxquelles j'ai pu appartenir durant ces années de thèse LaHDAK et VALS (ancien ForTesSE) en particulier Nathalie PERNELLE pour ces encouragements et sa gentillesse et Brigitte SAFAR pour avoir relu et corrigé mon manuscrit.

Je remercie énormément tous les fêtards qui m'ont permis d'oublier momentanément le travail dans des soirées, repas, foot, basket, bowling, discussion ou autre : Boubacar DIOUF, Lina BENTAKOUK, Abderrahmane FELIACHI, Mohamed Amine BAAZIZI, Vincent ARMANT, Mouad BAHI, Taj Muhammad KHAN, Nadjet Zemirline et Fayçal HAMDI.

Enfin, les mots les plus simples étant les plus forts, j'adresse toute mon affection à ma famille, notamment mon père Hédi, ma mère Neziha, ma sœur Sondes et mon frère Nidhal. Malgré la distance qui nous sépare depuis quelques années, leur générosité, leur confiance, leur tendresse et leur amour me portent et me guident tous les jours. Merci pour avoir fait de moi ce que je suis aujourd'hui.

Un grand merci à tout ceux qui, un jour ou un autre, ont croisé ma route, m'ont aidé à grandir et à évoluer ne serait ce qu'avec un sourire.

Resumé

Cette thèse a été réalisée dans le cadre du projet PIMI, financé par l'Agence Nationale de la Recherche (ANR). Elle porte sur la modélisation et l'utilisation d'informations personnelles dont la validité ou l'utilisabilité dépend du contexte. Plus particulièrement, elle a pour but d'aider l'utilisateur à réaliser des procédures en ligne, administratives ou personnelles. Elle aborde les problématiques de la représentation d'informations hétérogènes, d'interrogation d'espaces d'informations personnelles (PIS) contextualisées, de remplissage automatique de formulaires et de réalisation automatique de procédures définies à un haut niveau d'abstraction par composition de services disponibles en ligne.

Pour répondre à ces problématiques, nous avons proposé plusieurs contributions. La première contribution porte sur la gestion d'un espace d'informations personnelles (PIMS). Nous avons proposé une modélisation permettant la description des informations personnelles en utilisant plusieurs ontologies de domaine. Ces informations personnelles sont ainsi instanciées avec différentes valeurs dont l'utilisabilité dépend du contexte (e.g, utilisation du numéro de téléphone personnel ou professionnel selon le contexte correspondant) et d'un degré d'utilisabilité. Nous avons également proposé deux algorithmes d'interrogation contextuelles SQE et FQE qui permettent la recherche par des requêtes sur des informations personnelles stockées. La seconde contribution porte sur l'utilisation de ces informations par de différents services en ligne, et ce dans deux cas. Dans le cas du remplissage automatique de formulaires, nous avons proposé un algorithme permettant de générer une requête sémantique à partir de la représentation annotée d'un formulaire. Cette requête est évaluée en utilisant les deux algorithmes d'interrogation SQE et FQE. Dans le cas de la réalisation d'un objectif utilisateur (une procédure abstraite) par composition de services, nous avons étendu l'algorithme de Graphplan pour prendre en compte la contextualisation des données et des politiques de sécurité spécifiées par l'utilisateur. Ces dernières permettent ainsi à l'utilisateur d'augmenter le contrôle sur ses informations et de limiter leur divulgation.

Mots-clés : Informations personnelles, contextes, ontologies, composition de services, Planification, Graphplan, politiques de sécurité.

Abstract

This thesis was conducted within the PIMI project, financed by the National Agency of the Research (ANR). It concerns the modeling, the querying and the composition of personal information. We considered that the use and the access to personal information is context dependent (***e.g.,*** social, geographical). More particularly, it aims to support the user when realising online, administrative or personal procedures. In this setting, the tackled problems are the representation of heterogeneous information, the context-aware personal information spaces querying, the automatic form-filling and the automatic realization of procedures defined at a high level of abstraction by composition of online available services.

To solve these problems, we have developped several contributions. The first one concerns the management of the personal information space (PIMS). We have defined a model allowing the description of personal information using several domain ontologies. Our model can be instantiated on the user's personal information with several usability values depending on the context (e.g., the office phone number can be used in the professional context while the mobile phone number can be used in a personal context) and with a usability degree. We have also proposed two contextual querying algorithms SQE and FQE which allow to query the recorded information. The second contribution concerns the use of these information by several online services. It presents two use cases. In the case of the automatic forms-filling, we have proposed an algorithm allowing to generate a semantic query from an annotated form representation. This query is evaluated by using both querying algorithms SQE and FQE. Then, in the case of the user objective realization (an abstract procedure) by service composition, we have extended the Graphplan algorithm to take into account the contextualization of the data and the access policy rules specified by the user. The latter allows the user to increase the control of its information and to limit their leaking.

All these contributions have been implemented and led to several software prototypes.

Keywords: Personal Information, contexts, ontologies, services composition, Planning, Grpahplan, policies.

Table des matières

Introduction

Les systèmes d'information (SI) occupent une place majeure au sein des entreprises. Leur complexité a été accrue par la volonté d'informatiser le maximum de tâches (***e.g.,*** gestion des salaires, gestion des congés). Les avancées dans le domaine du génie logiciel ont permis de réduire cette complexité. S'appuyer sur les principes méthodologiques définis dans le domaine du génie logiciel [AB04] tels que la modularité et la réutilisation de modèles et de composants logiciels, rendrait possible la conception de systèmes d'informations plus robustes, plus flexibles et plus facilement adaptables aux nouveaux besoins et exigences de l'environnement socio-économique de l'entreprise. Ces principes sont au cœur de travaux de recherche ayant abouti à des paradigmes tels que les approches modulaires [Par72], orientées objets [Mey88, Tay97], orientées composants [Szy02] ou encore orientées services [Erl05, KBS05, Erl08]. Ces approches préconisent la construction de systèmes logiciels par la réutilisation et l'intégration de briques logicielles existantes.

Les approches orientées services promeuvent une nouvelle vision dans laquelle l'ensemble des traitements informatisés dans un SI est réalisé grâce à un ensemble de services élémentaires ou composites. Ces derniers sont le résultat d'un processus nommé ***composition de services*** [RS05]. La composition de services est un nouveau paradigme de développement logiciel permettant de construire des applications plus ou moins complexes par assemblage d'entités logicielles existantes. Ce paradigme contribue à un développement plus rapide des applications, à moindre coût, et en maximisant la réutilisation de l'existant.

Comme de nombreuses entités logicielles, les services prennent des données en entrée et en fournissent d'autres en sortie. Ces données peuvent être en accès libre, comme par exemple des données géographiques ou météorologiques, ou sujettes à des contraintes d'accès et de divulgation plus restrictives. Respecter ces contraintes est particulièrement crucial lorsqu'il s'agit d'informations personnelles.

1 Composition de services

Le nombre de services logiciels disponibles aujourd'hui via le Web est en constante augmentation. Ces services sont conçus de façon à être utilisés par tout type d'utilisateur sans se soucier ni de la technologie, ni du langage utilisé pour leur implémentation. La plupart de ces services se limitent à des fonctionnalités simples, ce qui est suffisant pour résoudre un grand nombre de problèmes. Toutefois, pour des besoins plus complexes, il est souvent nécessaire d'utiliser plusieurs services. Un exemple de besoin complexe pourrait être la planification d'un voyage culturel en Australie pour lequel il serait nécessaire de réserver un billet d'avion, de demander un visa touristique et de réserver deux places pour un concert à l'opéra *House* de Sydney. Vu la particularité de ce besoin, il n'existe très probablement pas de service accomplissant à lui tout seul toutes les fonctionnalités requises.

La composition de services apporte une solution à ce type de problèmes. Elle permet de développer rapidement et à la demande, des applications à valeur ajoutée en réutilisant des services existants. Le processus de composition de services consiste à sélectionner des services disponibles, à les combiner puis à les exécuter dans un certain ordre afin de réaliser un objectif donné.

La composition de services, lorsqu'elle est réalisée au sein d'une architecture orientée services (SOA), permet de profiter des avantages majeurs de cette architecture qui sont les suivants :

- la **réutilisation** qui offre la possibilité de développer de nouveaux services en réutilisant des services existants.
- le **couplage faible** qui représente un degré de dépendance faible entre les différents services et qui permet de réduire l'impact causé par un changement apporté à un service utilisé dans une composition.
- l'**évolutivité et la souplesse** qui offrent la possibilité de modifier ou de substituer, à l'exécution, les services utilisés par une composition pour satisfaire de nouveaux besoins, confortant ainsi le dynamisme au sein des applications.

Dans la littérature, la composition de services est souvent étudiée dans une architecture SOA particulièrement conçue pour les services Web [NM02, HB03, WSH+06, SPM09]. Les services Web « *prennent leurs origines dans l'informatique distribuée* » [Afe09]. L'utilisation des services Web permet à toute organisation d'exporter à travers le réseau Internet ses compétences et son savoir-faire. Pour réaliser une composition de services Web de qualité il est important de considérer les critères suivants :

- **automatisation :** une grande quantité de services Web sont disponibles sur le Web. Cette quantité est au-delà des capacités humaines d'analyse et de

traitement ce qui rend impossible une composition manuelle.
— **passage à l'échelle :** dans le contexte du Web, il est nécessaire que les approches de composition fonctionnent en un temps raisonnable sur un nombre important de services.
— **tolérance aux pannes :** les services utilisés dans une composition peuvent être supprimés ou devenir défaillants de manière imprévisible. Cet environnement très dynamique peut rendre nécessaire le remplacement de services pendant l'exécution voir le recalcul des compositions.

La composition de services peut être aussi informée par des informations et des contraintes supplémentaires telles que des préférences utilisateur, la qualité de service, des politiques de sécurité ou encore des informations qui relèvent des données manipulées. Dans ce cas, des approches de composition considérant ces informations et contraintes supplémentaires doivent être développées.

2 Informations personnelles

Le terme *information personnelle* a été abordé pour la première fois dans la littérature scientifique par Vannevar Bush en 1945. Dans [Bus45], il définit une information personnelle comme pouvant être un livre, un rapport ou une communication. De la même façon, en 1988, Lansdale [Lan88a] définit une information personnelle comme étant une note, un enregistrement personnel, un dossier ou toute information qu'un individu pourrait conserver pour une utilisation personnelle ultérieure. Des années plus tard, Boardman et al. [BS04] distinguent deux interprétations possibles à donner aux informations personnelles. La première interprétation considère qu'il s'agit d'informations concernant un seul individu. Dans ce cas les informations personnelles correspondent aux informations données par une institution (***e.g.,*** numéro de sécurité sociale, numéro de carte de crédit). La seconde interprétation, considère qu'il s'agit de toute information enregistrée et gérée par un logiciel de gestion d'informations personnelles. De plus, Boardmane et al. définissent une information personnelle comme étant « *toute information appartenant à un individu et sous son contrôle direct* ». Dans nos travaux, nous avons considéré la seconde interprétation des informations personnelles.

Actuellement la quantité des informations personnelles dont dispose un individu est énorme et en constante augmentation. Ces informations sont disponibles sur différents équipements (***e.g.,*** ordinateurs, tablettes, téléphones), se présentent sous différents formats (***e.g.,*** textuel, structuré, binaire), et sont gérées par différents types d'applications (***e.g.,*** agenda, gestion de courriers). De plus, une information personnelle peut comme tout type d'information avoir une durée de

vie déterminée. Un passeport a par exemple une durée de validité de 10 ans et une carte bancaire a une durée de validité de 3 ans. Les informations personnelles peuvent par ailleurs être multi-valuées. Les personnes ayant une double nationalité peuvent ainsi posséder deux passeports et donc deux numéros de passeport et deux photos scannées.

L'ensemble des informations personnelles d'une personne constitue son ***espace d'informations personnelles*** (ou PIS pour Personal Information Space). Le système permettant d'organiser, de gérer et de rechercher ces informations quant à lui est appelé ***Système de Gestion d'Informations Personnelles*** (ou PIMS pour Personal Information Management System) [BN95].

Compte tenu des caractéristiques parfois contraignantes et exigeantes, des informations personnelles disponibles aujourd'hui et des environnement dans lesquels elles sont manipulées, il est important de développer des systèmes de gestion d'informations personnelles (PIMS) permettant de répondre à ces contraintes.

3 Problématiques abordées

Cette thèse a été réalisée dans le cadre du projet PIMI [1], financé par l'Agence Nationale de la Recherche (ANR). Le projet avait pour objectif de définir un environnement de développement et de déploiement d'une plate-forme pour la gestion d'informations personnelles.

L'architecture mise en place dans le projet PIMI est donnée en figure 1. Le système (PIMS) doit permettre l'accès et l'utilisation des informations personnelles d'un utilisateur pour réaliser ses besoins (e-procédure). Le besoin de l'utilisateur est exprimé sous la forme de procédures administratives (***e.g.,*** déclaration de naissance) ou de procédures personnelles (***e.g.,*** planification de voyage). Le PIMS du demandeur peut aussi accéder à certaines informations personnelles de ses contacts. Par exemple dans le cas d'un parent souhaitant effectuer une déclaration de naissance, le système devra pouvoir utiliser ses informations ainsi que celles de son conjoint.
Dans le projet PIMI, les informations personnelles d'un utilisateur sont soit déclarées par l'utilisateur lui même [KPS12b, KPS12a] soit collectées à partir des réseaux sociaux [LEHP14].

Notre implication dans le projet porte sur la réalisation des objectifs (e-procédures) de l'utilisateur en réaction à des événements de vie, (comme montré

1. Personal Information Management through Internet (PIMI-ANR-2010-VERS-0014-03)

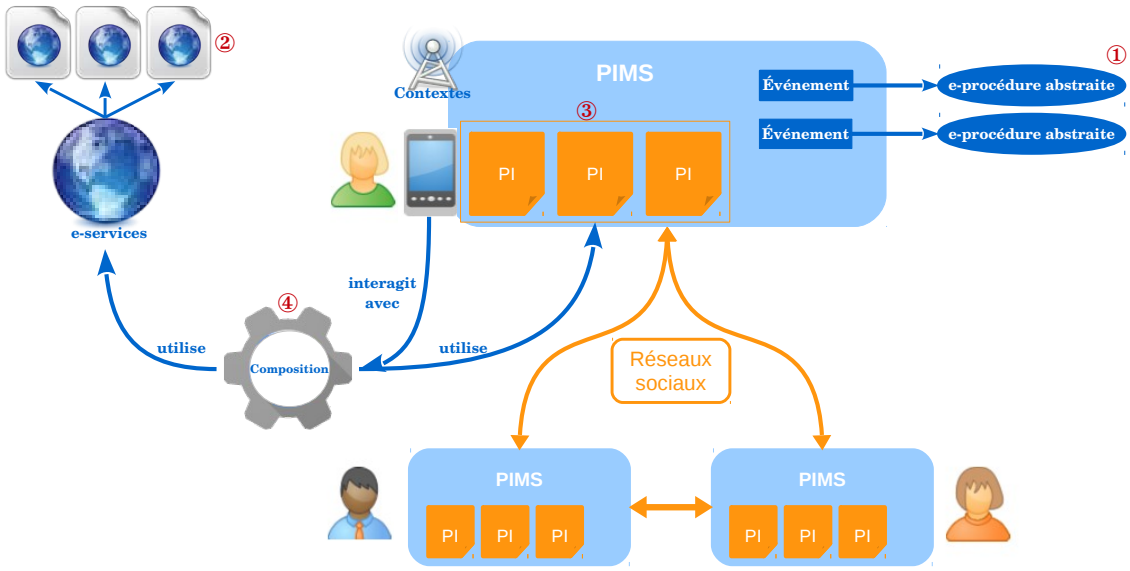


FIGURE 1: Architecture de PIMI

en ① dans la figure 1). Sur la base de services en ligne (e-services) référencés dans l'annuaire de l'infrastructure PIMI, (comme montré en ② dans la figure 1), et des informations personnelles de l'utilisateur utilisables par ces e-services, (comme montré en ③ dans la figure 1), il s'agit de générer une composition de e-services, (comme montré en ④ dans la figure 1), avec laquelle l'utilisateur pourra interagir pour réaliser ses objectifs.

Une fois la composition de services calculée, l'utilisateur devra interagir avec les e-services pour renseigner les données d'entrée nécessaires à l'exécution des services. Cette interaction s'effectue via des formulaires de saisie. Souvent, les mêmes informations sont demandées par différents formulaires. Afin d'éviter la saisie répétitive et fastidieuse des informations personnelles, nous avons aussi été conduits à étudier et développer une méthode de remplissage automatique de formulaires.

Compte tenu de l'hétérogénéité à la fois syntaxique et sémantique des informations personnelles, le besoin de représenter les informations sous un modèle uniforme et permettant de représenter différents points de vue s'est avéré prépondérant.

Pour les trois problématiques mentionnées ci-dessus, c'est-à-dire la composition automatique de e-services, le remplissage automatique de formulaires et la modélisation uniforme des informations personnelles, nous avons tenu compte de l'aspect contextuel des informations personnelles. En effet, l'utilisation de ces dernières peut être dépendante du contexte social, géographique ou encore temporel. Par exemple si une étudiante possède deux adresses électroniques (*alice.bernard@u-psud.fr,*

alice21@gmail.com), elle pourrait utiliser l'adresse *u-psud.fr* pour envoyer un projet à son professeur et l'adresse *gmail.com* pour inviter ses amis à une soirée.

Enfin, l'utilisation des informations personnelles par des services en ligne présente un risque important d'atteinte à la vie privée de l'utilisateur. Gérer la façon dont ces informations peuvent être utilisées et manipulées par les services est un enjeu majeur. En effet, il est important d'étudier le problème de modélisation des politiques de sécurité et de leur prise en compte lors de l'exploitation des informations personnelles.

Les objectifs de cette thèse sont donc les suivants :

1. Représentation uniforme d'informations personnelles
   — modélisation des informations personnelles selon différents points de vue
   — représentation des contextes d'utilisation des informations personnelles
2. Interrogation d'informations personnelles
   — exploitation des contextes d'utilisation lors de l'interrogation du PIS
3. Remplissage automatique de formulaires
   — traduction de formulaires de e-services en requêtes sémantiques sur les informations personnelles
4. Composition de services exploitant les contextes d'utilisation des informations personnelles et respectant les politiques de sécurité.

4 Contributions

Pour traiter les différentes problématiques énoncées dans la section précédente, nous présentons dans cette thèse les contributions suivantes.

Modélisation contextuelle et multi-points de vue des informations personnelles. Pour pallier le problème d'hétérogénéité des informations personnelles, nous avons proposé une modélisation fondée sur l'utilisation des ontologies. En effet, les ontologies offrent un cadre conceptuel et formel permettant de définir des modèles sémantiques communs pour la représentation des connaissances. De plus, comme nous avons voulu exploiter le caractère multi-points de vue des informations personnelles, c'est-à-dire des descriptions différentes des mêmes informations, le choix des ontologies s'est avéré particulièrement judicieux. Ainsi, des ontologies existantes exprimant les différents points de vue peuvent être utilisées par un utilisateur pour concevoir son ontologie de types d'informations personnelles (PITO) servant de modèle commun pour la représentation de ses informations. De plus, le caractère sensible aux contextes des informations personnelles a été

pris en compte par, d'une part, la modélisation des contextes via des ontologies de contextes (***e.g.,*** géographique, social, temporel) et d'autre part, par l'utilisation de ces ontologies pour associer des contextes aux différentes valeurs des informations personnelles gérées dans le PIS de l'utilisateur. Enfin, pour prendre en compte l'aspect confidentiel des informations personnelles, nous avons utilisé une catégorisation par thème (***e.g.,*** impôt, banque, santé) des informations personnelles. Ces catégories sont représentées dans une ontologie de catégories servant à classer les informations personnelles et ainsi faciliter la gestion des politiques de sécurité lors de leur exploitation.

Nos travaux sur la modélisation des informations personnelles ont été publiés dans trois conférences :

— Khéfifi, R., Poizat, P., Saïs, F. : Modeling and Querying Context-Aware Personal Information Spaces. DEXA(2012)
— Khéfifi, R., Poizat, P., Saïs, F. : Modélisation et interrogation d'espaces d'informations personnelles sensibles au contexte. EGC(2012)
— Khéfifi, R., Poizat, P., Saïs, F. : Modeling and Querying Context-Aware Personal Information Spaces. SOS-DLWD(2012)

Interrogation contextuelle des informations personnelles. Pour exploiter les informations personnelles stockées dans le PIS, il est possible de poser des requêtes simples ou conjonctives. Compte tenu du fait que les informations personnelles sont contextuelles, les requêtes le sont également. En d'autres termes, nous considérons des requêtes conjonctives auxquelles un contexte et une fonction sont associés. Cette fonction, pouvant être soit l'égalité, la similarité ou la disjonction, permet de comparer les contextes associés aux informations personnelles avec le contexte de la requête. Ainsi, seules les réponses satisfaisant cette fonction sont renvoyées à l'utilisateur. Pour l'évaluation de ces requêtes contextuelles nous avons développé deux algorithmes *SQE* et *FQE* où la satisfiabilité de la fonction de comparaison est plus ou moins stricte. Ces deux algorithmes calculent un score de pertinence de chaque réponse en fonction des scores élémentaires calculés pour chaque valeur de propriété. Ces scores sont utilisés pour ordonner les réponses renvoyées à l'utilisateur.

De plus, dans le cadre d'un PIS pouvant être exploité par des e-services pour qui les entrées sont fournies via des formulaires, les requêtes contextuelles sont automatiquement générées pour chaque formulaire. Pour générer ces requêtes, nous avons considéré une représentation annotée des formulaires par le vocabulaire de l'ontologie PITO. Ainsi, en utilisant ces requêtes il est possible de remplir automatiquement les formulaires à partir des informations personnelles stockées dans le PIS.

Le travail sur l'interrogation contextuelle a été publié dans DEXA 2012, EGC 2012 et SOS-DLWD 2012.

Notre travail concernant le remplissage automatique de formulaires a été présenté dans un article :

— Khéfifi, R., Poizat, P., Saïs, F. : Vers une utilisation automatique des informations personnelles pour la réalisation de e-procédures. Revue RNTI 2014.

Composition de e-services sensible aux contextes. Les objectifs des utilisateurs sont souvent complexes dans le sens où ils nécessitent la réalisation ordonnée de plusieurs fonctionnalités. Dans le cadre du projet PIMI, les procédures représentent les besoins de l'utilisateur sous la forme de *workflows*. Nous avons considéré que la réalisation d'une procédure peut être effectuée dans un contexte prédéfini par l'utilisateur.

Afin de répondre à la problématique de la réalisation de procédures par l'utilisateur, nous avons proposé une approche de composition de services dont les originalités sont :

— la description des procédures (objectifs de composition) à l'aide de *workflows*, exprimés en termes des fonctionnalités attendues ;
— la prise en compte d'un contexte de référence dans lequel la procédure doit être réalisée ;
— l'exploitation des politiques de sécurité sur les informations personnelles ;
— l'utilisation d'une technique de planification efficace fondée sur l'algorithme *GraphPlan*. Cette technique a été étendue pour prendre en compte les contextes et leurs degrés de pertinence ainsi que pour exploiter des politiques de sécurité.

Nos travaux sur la composition ont donné lieu aux publications suivantes :

— Khéfifi, R., Poizat, P., Saïs, F. : Automatic Composition of Form-Based Services in a Context-Aware Personal Information Space. ICSOC(2013)
— Khéfifi, R., Poizat, P., Saïs, F. : Data-Flow Oriented Service Composition : AI-Planning or Petri Net ? SOS-DLWD(2013)

5 Plan du manuscrit

Outre cette introduction, ce manuscrit est organisé en deux parties principales. La première partie intitulée « ***État de l'art*** », introduit plus en détail les domaines dans lesquels se place cette thèse. La seconde partie, intitulée « ***Contributions*** », présente nos travaux relatifs à la gestion d'informations personnelles et à la composition de services.

Première Partie : « *État de l'art* »
Cette première partie du manuscrit est organisée en deux chapitres :

— Le chapitre 1 « ***Gestion des informations personnelles*** » introduit le besoin de la gestion d'informations personnelles et propose une classification des approches existantes

— Le chapitre 2 « ***Composition de services*** » présente le processus de composition et élabore une comparaison des approches de composition existantes vis-à-vis de notre problématique.

Deuxième Partie : « *Contributions* »
Cette seconde partie du manuscrit est organisée en quatre chapitres :

— Le chapitre 3 « ***Préliminaires*** » présente un ensemble de notions utilisées dans les chapitres de contribution (chapitre 4, 5 et 6). Ce chapitre donne une introduction générale aux ontologies. Il présente par la suite les langages du Web sémantique utilisés pour la représentation des connaissances (OWL et RDF) ainsi qu'un langage de requête utilisé pour l'interrogation de données RDF et d'ontologies OWL. Finalement, nous y introduisons une mesure de similarité sémantique utilisée dans les chapitres suivants.

— Le chapitre 4 « ***Modélisation contextuelle des informations personnelles*** » introduit le modèle de représentation des informations personnelles que nous proposons. Ce modèle exploite des ontologies existantes et permet à l'utilisateur de participer à la construction et à la description de son espace d'informations personnelles. Il permet également de décrire les informations personnelles en fonction de contextes. Nous présentons à la fin de ce chapitre, le prototype CoPIMS développé pour réaliser la description de l'espace d'informations personnelles et l'instanciation contextuelle de ces dernières.

— Le chapitre 5 « ***Interrogation contextuelle des informations personnelles*** » présente l'interrogation des informations personnelles en utilisant des requêtes contextuelles. Nous montrons également comment ces dernières sont générées et évaluées pour le remplissage automatique de formulaires.

— Le chapitre 6 « ***CoCliCo : une approche de composition de services Web sensible aux contextes*** » présente d'une part le processus de composition de services défini pour la réalisation automatique d'objectifs utilisateurs et d'autre part, l'outil développé pour tester notre approche. Ce chapitre présente par ailleurs un outil de génération de jeux de données que

nous avons réalisé pour pouvoir évaluer notre approche ainsi qu'un ensemble de résultats expérimentaux concernant cette dernière.

Enfin, suite à l'ensemble de ces chapitres nous concluons et présentons des perspectives à nos travaux.

Première partie

État de l'art

CHAPITRE 1

GESTION D'INFORMATIONS PERSONNELLES

Sommaire

La gestion d'informations personnelles est la pratique et l'étude des activités que tout individu accomplit dans le but d'acquérir, d'organiser, de maintenir et de rechercher des informations personnelles au quotidien. Une bonne gestion des informations personnelles selon William Jones [Jon04] « *is that we always have the right information in the right place, in the right form, and of sufficient completeness and quality to meet our current need* ». Cet idéal est loin d'être atteint malgré la disponibilité de nombreux outils et techniques permettant la gestion d'informations

personnelles. De plus, cette diversité d'outils et de techniques contribue en partie au problème d'hétérogénéité des informations personnelles. Ces dernières sont disponibles sur différents équipements, se présentent sous différents formats, et sont gérées par différents types d'applications. Cette répartition et cette hétérogénéité de l'information rend la tâche de recherche plus difficile. C'est pour cette raison que nous avons besoin de modèles de gestion d'informations personnelles capables de gérer cette hétérogénéité et cette distribution de façon transparente.

L'intérêt porté à la gestion d'informations personnelles a débuté avec Vannevar Bush en 1945 dans son système MEMEX [Bus45]. Ce dernier a été conçu pour une utilisation individuelle. Le terme *gestion d'information personnelles* a été utilisé pour la première fois en 1980 par [Lan88b, TJB06] et depuis, des systèmes de gestion d'informations personnelles n'ont cessé de voir le jour. Barreau et al. [BN95] ont conçu et développé un système de gestion d'informations personnelles (PIMS) en faisant apparaître quatre fonctions : (*i*) l'acquisition, qui permet de collecter de nouvelles informations personnelles, (*ii*) l'organisation, qui permet de structurer ces informations et de les classer en catégories, (*iii*) la maintenance, qui assure la mise à jour, et (*iv*) la recherche, qui se charge de retrouver et/ou d'accéder aux informations pré-enregistrées.

Dans ce chapitre, nous présentons les résultats de notre étude des travaux de recherche dans le domaine de la gestion des informations personnelles. Tout d'abord, nous introduisons dans la section 1.1 la terminologie utilisée. La section 1.2 présente quelques critères de comparaison des systèmes de gestion d'informations personnelles. Nous présentons en section 1.3 les différents modèles utilisés pour l'organisation d'informations personnelles et en section 1.4 les stratégies de recherche utilisées dans ces différents systèmes et approches. Finalement, en fournissant une étude comparative des approches, nous concluons ce chapitre dans la section 1.5.

1.1 Terminologie

Donnée vs Information. La frontière entre les termes *donnée* et *information* est très mince. En effet, il est assez facile de les confondre et d'utiliser l'un ou l'autre. Une donnée est considérée, généralement, comme un fait brut destiné à être traité. Elle peut déjà être présente dans le système, ou bien être traitée par un outil de supervision ou de saisie, ou par une personne. Une information est une donnée qui possède un sens et une interprétation, et qui peut être utilisée pour une prise de décision.

Selon Orna [Orn04], les informations sont « *ce que les êtres humains transforment en connaissances quand ils veulent les communiquer à autrui. C'est la connaissance*

rendue visible ou audible dans des mots écrits ou imprimés ou dans un discours ».

Une information est statique et périssable, sa valeur se dégrade au fil du temps. Selon le contexte, une donnée peut devenir une information. Prenons l'exemple d'une donnée brute « *17* ». Ce nombre, en soi, n'a aucune signification particulière sans la présence d'un contexte. En effet, « *17* » peut être une note, une température, une heure ou un âge. La donnée « *17* » ne peut pas être utilisée telle quelle. Cependant, en associant ce nombre à l'âge que Bob aura en Mars 2015, cette donnée prend un sens et devient une information. Nous utilisons le terme *informations personnelles* et non pas *données personnelles* puisque que les données à considérer sont assignées à une description de personne et ont un sens.

Information personnelle (PI). Selon la CNIL (Commission Nationale de l'Informatique et des Libertés www.cnil.fr), une information personnelle est toute donnée permettant d'identifier directement ou indirectement une personne physique. Elle peut correspondre à un nom, un prénom, une adresse, un courriel, un numéro de téléphone, un numéro de sécurité sociale, un numéro de carte de payement ou une plaque d'immatriculation d'un véhicule.

Une information personnelle en informatique correspond à toute ressource possédée par une personne [BS04]. Il s'agit d'informations personnelles définies selon la CNIL ou de tout document stocké sur un équipement numérique (***e.g.,*** ordinateur, tablette ou téléphone).

1.2 Critères de comparaison des systèmes de gestion d'informations personnelles

De nombreux travaux ont été proposés pour la gestion d'informations personnelles. Pour comparer ces approches, différents critères peuvent être utilisés. Dans cette thèse, nous avons choisi de les comparer en terme d'utilisation de modèles sémantiques et de prise en compte de contextes.

Utilisation de modèles sémantiques. La répartition et l'hétérogénéité des informations personnelles rendent leur gestion difficile. Une modélisation sémantique permet d'une part de représenter les informations de manière uniforme et d'autre part de se doter de capacités de raisonnement qui rendent leur gestion plus efficace. Utiliser une modélisation sémantique commune facilite la recherche, la maintenance mais aussi la détection d'incohérences, comme les données redondantes.

Contexte. Comme mentionné en introduction, l'utilisation des informations personnelles peut dépendre des contextes (***e.g.,*** social, géographique, temporel). Le

contexte représente un environnement ou une situation [ST94, SDA99]. Schilit et al. [SAW94] spécifient que les aspects importants concernant les contextes sont : où vous êtes ?, qui vous êtes ?, avec qui vous êtes ? et quelles sont les équipements à proximité ?. Dey [Dey01] définit le contexte comme étant toute information qui pourrait être utilisée pour caractériser une situation ou une entité (***e.g.,*** une personne, un lieu). Tenir compte des contextes favorise l'obtention de l'information la plus adaptée à une situation.

1.3 Modélisation d'informations personnelles

La modélisation est au cœur de tout système de gestion d'informations personnelles. Dans notre étude des systèmes et des approches de gestion d'informations personnelles existants, il apparaît que les informations personnelles peuvent être soit des documents soit des méta-données. Dans les deux cas, différents modèles de représentation ont été proposés.

En ce qui concerne la modélisation des informations personnelles, nous classons les approches selon deux catégories : modélisation non-sémantique et modélisation sémantique.

1.3.1 Modélisation non-sémantique

La plupart des travaux sur la gestion d'informations personnelles considèrent qu'une information personnelle est tout document ou ressource possédé par un utilisateur, en général, stocké sur un poste de travail. La modélisation de ces informations s'appuie traditionnellement sur l'organisation hiérarchique.

Organisation hiérarchique. Une première façon naturelle de modéliser et d'organiser les informations est le stockage hiérarchique dans un système de fichiers. Plusieurs systèmes de gestion d'informations personnelles s'appuient sur ce type d'organisation. Ces systèmes permettent d'organiser les ressources suivant une ou plusieurs dimensions. LifeStreams [FG96] permet d'enregistrer chaque document reçu ou créé en suivant un ordre chronologique. TimeScape [Rek99], TimeSpace [KJ05], UMEA [Kap03] et Stuff I've Seen [DCC+03] permettent d'organiser les informations personnelles selon différentes dimensions (***e.g.,*** activité, temps, localisation). Lamming et Flynn [LF94] ont proposé un système d'aide mémoire Forget-me-not. Ces systèmes collectent des informations concernant les activités de l'utilisateur et les enregistrent sous la forme de méta-données dans une base de données appelée biographie.

Ce type d'organisation ne permet d'utiliser qu'un ensemble prédéfini de catégories alors que dans certaines situations l'utilisateur pourrait être intéressé par l'uti-

lisation de ses propres catégories. L'annotation pourrait être une solution à cette limite.

Annotation. L'annotation est à mi-chemin entre l'organisation hiérarchique non-sémantique et la modélisation sémantique. En effet, il s'agit de garder l'organisation hiérarchique et d'annoter les documents et les ressources par un ensemble de termes permettant ainsi d'assigner une ou plusieurs catégories à chaque information. Dans ce cadre, PLACELESS DOCUMENTS [DEL+00], MYLIFEBITS [GBL+02] sont des systèmes qui utilisent l'annotation des ressources pour modéliser les informations personnelles.

PLACELESS DOCUMENTS [DEL+00] présente un modèle de gestion de documents essentiellement des e-mails, des feuilles de calcul, des images et des fichiers texte. Ces documents sont annotés par des propriétés universelles (***e.g.,*** auteur, date de création, date de la dernière modification) et des propriétés définies par l'utilisateur. MYLIFEBITS [GBL+02] est une base de données contenant des documents média et des liens entre ressources. Les annotations sont effectuées entre les ressources et spécifiées par les liens. Par exemple, une annotation pourrait être un lien entre une photo de Pharel William et son clip.

Après cette présentation succincte des systèmes s'appuyant sur une modélisation non-sémantique des informations personnelles, nous décrivons dans la section suivante les systèmes proposant une modélisation sémantique des informations personnelles.

1.3.2 Modélisation sémantique

La plupart des PIMS organisent les informations personnelles sous une forme hiérarchique et imposent aux utilisateurs le respect de cette modélisation. Cependant, cette structuration hiérarchique implique un temps important lors de la recherche des informations. En effet, cette modélisation n'exploite pas les relations sémantiques qu'il peut y avoir entre ces informations. Comme alternative à la modélisation hiérarchique, certains systèmes ont adopté les technologies du Web sémantique pour représenter leurs informations. Parmi ces systèmes nous citons SEMEX, PIMO, *OntoPIM*, *iMecho*, Beagle++ et iMeMex.

SEMEX [CDH+05] est un système permettant de gérer les documents sur le poste de travail selon leurs méta-données. Il permet l'extraction d'instances d'objets (***e.g.,*** personne, message, publication) et d'associations (***e.g.,*** authoredBy, attachedTo) à partir de plusieurs sources de données hétérogènes (***e.g.,*** bibtex, courriel, contact, site Web) conformes à une ontologie générique. SEMEX permet ensuite de réconcilier ces différentes instances pour supprimer les redondances avant de les enregistrer dans une base de données.

Xiao et Cruz [XC05] quant à eux proposent une approche de modélisation d'informations personnelles en plusieurs couches et en utilisant plusieurs ontologies. L'approche est composée de quatre niveaux : *espace d'informations personnelles*, *niveau applications*, *niveau domaine* et *niveau ressources*. L'espace d'informations personnelles contient tous les documents et les données enregistrés sur le poste de travail de l'utilisateur. Ils peuvent être structurés dans une base de données, semi-structurés (XML) ou non-structurés (***e.g.,*** texte, vidéo, photo). Le niveau applications représente des ontologies de différentes applications (***e.g.,*** iCal, Thunderbird) utilisées par le système. Le niveau domaine contient les ontologies représentant des connaissances de plusieurs domaines publiées sur le Web tels que des conférences, des personnes ou des courriels. Ce niveau est conçu pour être faiblement couplé afin de supporter la dynamicité du Web avec l'arrivée ou le départ des ontologies. Enfin, le niveau ressources qui contient toutes les ressources des informations personnelles représentées par des *URI* (Uniform resource identifier), et des méta-données. Ce niveau permet de gérer la correspondance entre les données hétérogènes de l'espace d'informations personnelles et les ontologies de domaine.

Une approche similaire à la précédente est l'approche appelée PIMO, qui a été proposée par Sauermann et al. [SBD05, SVED07]. Les auteurs proposent une plateforme avec plusieurs ontologies pour représenter les documents dont dispose un utilisateur. Le modèle d'informations personnelles PIMO est conçu en utilisant une ontologie à différents niveaux comme suit : (*i*) *PIMO-Basic* présente la racine de l'ontologie, (*ii*) *PIMO-Upper* présente les classes abstraites de l'ontologie (***e.g.,*** *PersonConcept*, *OrganizationConcept, Document*), (*iii*) *PIMO-Mid* est le niveau qui sert à intégrer plusieurs ontologies et fournit des classes concrètes (***e.g.,*** *Person*, *Project*), (*iv*)*ontologies de domaine* est un ensemble d'ontologies où chacune décrit un domaine d'intérêt de l'utilisateur et (*v*) *PIMO-User* est une extension de modèle créée par l'utilisateur pour une utilisation personnelle.

Une autre approche proposée par Katifori et al. [KPS+05] nommée *OntoPIM* s'appuie également sur l'utilisation de plusieurs ontologies. Le but de cette approche et de représenter sémantiquement le poste de travail de l'utilisateur offrant ainsi un moyen efficace d'interrogation des données de l'utilisateur. *OntoPIM* est constitué de trois niveaux : (*i*) le niveau physique qui contient tous les documents stockés sur l'ordinateur de l'utilisateur, (*ii*) le niveau DS (*Domain Specific*) qui représente les différents objets du niveau physique tels que les e-mails, les documents ou les photos avec leurs méta-données (***e.g.,*** auteur, date de création, date de modification) et (*iii*) le niveau DI (*Domain Independant*) qui représente les instances d'objets spécifiées dans le niveau DS. Dans le même type d'approche, Chen et al. [CGWW09] propose une approche de représentation sémantique du poste de travail nommée *iMecho*. Cette approche s'appuie sur l'utilisation et la création d'associations entre les différentes ressources (documents, e-mails, pièces jointes d'e-mails, pages Web enregistrées) en analysant les activités de l'utilisateur (***e.g.,*** réception/envoi de

mail) ainsi que son comportement au travail. *iMecho* s'appuie sur une ontologie pour lier tous les objets et les associations créés.

Beagle++ [CGG+05] modélise un ensemble d'informations contextualisées par rapport à l'activité effectuée sur le poste de travail. Il exploite trois types d'activités concernant la gestion : des e-mails, de la hiérarchie de fichiers et du cache Web. Chacune de ces activités est modélisée par une ontologie servant à extraire des instances d'informations correspondantes. Par exemple lors de l'envoi d'un e-mail, les informations pouvant être collectées sont l'objet, le destinataire et la date d'envoi de l'e-mail. Ces informations seront par la suite enregistrées sous la forme de triplets RDF.

Finalement, le système iMeMex [DS06] permet de représenter les informations personnelles stockées dans des formats hétérogènes (*i.e.* structurées, semi-structurées et non-structurées) de façon uniforme. Pour ce faire, une représentation graphique est créée pour représenter les informations personnelles ainsi que leurs liens. Ces derniers peuvent être externes, représentant les liens entre documents (***e.g.,*** doc1 est un raccourci de doc2) ou internes représentant la structure du document (***e.g.,*** section, sous-section).

1.4 Recherche d'informations personnelles

La recherche d'informations a été définie par Manning et al. [MRS08] comme étant la sélection de documents ou d'informations satisfaisant un besoin, à partir d'une grande collection d'informations, éventuellement stockées sur un poste de travail. Une étude sur le comportement de l'utilisateur pour la recherche d'informations [Kim12] a montré que les humains utilisent différents types d'interaction pour accéder à l'information. Ceci dépend du niveau de connaissance et du comportement de chacun.

Teevan et al. [TAAK04] présentent deux méthodes de recherche d'informations qui sont *l'orientation* et *la téléportation*. L'orientation consiste à chercher une information ou un document en utilisant un contexte et des connaissances antérieures concernant l'objet de la recherche. Par exemple, pour rechercher le numéro de téléphone d'une personne nommée Alice travaillant au LRI, l'utilisateur se dirigerait naturellement vers le site Web www.lri.fr ensuite il consulterait l'annuaire du LRI en cherchant le numéro de téléphone de la personne. La téléportation permet d'accéder directement à l'information en utilisant une requête précise. Par exemple, pour trouver le numéro de téléphone d'Alice du LRI, il faudrait saisir dans la barre de recherche d'un moteur de recherche « numéro de téléphone d'Alice au LRI » et choisir une réponse parmi celles obtenues. Une classification similaire a été proposée par Mackinlay et Zellweger [MZ95]. Il s'agit de *la navigation* et *des requêtes*. Elles correspondent respectivement à l'orientation et à la téléportation.

Nous présentons dans ce qui suit les systèmes de gestion d'informations personnelles selon le type de recherche proposé : navigation ou requêtes.

1.4.1 Stratégie orientée navigation

La navigation permet l'exploration des liens qu'ils soient arborescents ou sémantiques entre les informations personnelles, dans le but de rechercher une information. L'utilisation de cette stratégie nécessite des connaissances par rapport à l'information recherchée et au modèle adopté pour leur organisation.

Selon des études réalisées par Boardman et al. [BS04] et par Bergman et al. [BBMN+08], les utilisateurs préfèrent la navigation à la recherche par requêtes. Les systèmes utilisant la navigation sont, généralement, inclus dans les systèmes d'exploitation comme par exemple Mac Finder pour Mac OS et Windows Explorer pour Windows. Ces derniers supportent la navigation hiérarchique des fichiers. Parmi les approches de gestion d'informations personnelles qui utilisent la navigation pour la recherche d'informations, nous trouvons, Forget-me-not [LF94], LifeStreams [FG96], UMEA [Kap03], TimeScape [Rek99], TimeSpace [KJ05], et Xiao et Cruz [XC05].

Forget-me-not [LF94] présente une navigation ascendante et descendante de la biographie de l'utilisateur, permettant ainsi de visualiser la liste des événements impliquant un contact (par défaut il s'agit de l'utilisateur lui même). LifeStreams [FG96] et TimeScape [Rek99] permettent de naviguer entre les documents de façon chronologique du passé vers le présent. UMEA [Kap03] permet de naviguer suivant la hiérarchie d'activités proposées et TimeSpace [KJ05] effectue une navigation combinant la chronologie et les catégorisations. Enfin, Xiao et Cruz [XC05] proposent trois types de navigation : (*i*) la navigation verticale qui permet de parcourir les informations depuis les ontologies d'application vers les fichiers enregistrés et vice-versa, (*ii*) la navigation horizontale permet notamment de suivre les liens entre les concepts de différentes ontologies du même niveau application-application ou domaine-domaine, et (*iii*) la navigation temporelle permet de parcourir les instances d'un objet par ordre chronologique par exemple voir les différentes versions d'un article de recherche.

1.4.2 Stratégie orientée requête

La stratégie de recherche orientée requête offre à l'utilisateur une façon plus rapide et flexible de rechercher des informations suivant différents critères. Ce type de recherche peut être de deux types : la recherche par mots-clés et la recherche en utilisant un langage d'interrogation évolué.

La recherche mots-clés. Un mot-clé désigne, dans le vocabulaire des documentalistes, un mot ou un groupe de mots permettant de caractériser le contenu

d'un document. Dans les moteurs de recherche, il sert à sélectionner un ensemble de documents contenant ce mot-clé.

Actuellement, la majorité des systèmes d'exploitation intègrent ce mode de recherche. La recherche est assurée par des outils de recherche permettant d'indexer toutes les informations disponibles sur l'ordinateur [Evé10] et d'évaluer les requêtes mots-clés sur cette représentation indexée des informations. Parmi ces outils nous trouvons *Google Desktop* [TBS+06] et *Spotlight* [Spo09] d'*Apple*. Ces outils considèrent uniquement la structure hiérarchique du système de fichiers et ses méta-données (***e.g.,*** date, type de fichier) comme condition de filtrage. *Apple Spotlight* utilise la stratégie de recherche itérative par mot-clé. En effet, il affine les résultats de la recherche au fur et à mesure que l'utilisateur complète la saisie des mots-clés et les résultats sont présentés par type de ressources. *Google Desktop* est aussi un outil de recherche local dans les documents situés sur un poste de travail. Il offre, comme *Spotlight*, la recherche par mots-clés. Récemment, il permet de filtrer la recherche en fonction du type de fichier.

D'autres systèmes de gestion d'informations personnelles ont également utilisé cette stratégie. MyLifeBits [GBL+02] permet une recherche par mots-clés sur la description des ressources de l'espace d'informations personnelles. La recherche s'étend aux annotations associées aux ressources. Par exemple en cherchant le mot *anniversaire*, la requête retourne toutes les ressources (***e.g.,*** photo, vidéo, document) ayant le mot anniversaire dans leurs méta-données ou leurs annotations. Beagle++ [CGG+05] lui aussi utilise la recherche par mots-clés exploitant ainsi les ontologies d'activité utilisées.

Interrogation par requêtes complexes. La recherche par requêtes complexes est effectuée sur la base de l'expression de requêtes caractérisant un besoin de recherche. Ce besoin est dans la plupart des cas complexe. En effet, l'utilisation unique de mots-clés ne permet pas de le satisfaire. La requête est formulée en utilisant la structure de données utilisée. Il existe plusieurs langages d'interrogation comme SQL pour les bases de données relationnelles, XQuery pour les bases XML ou SPARQL pour les bases RDF. Face au besoin d'avoir des moyens flexibles de recherche d'informations personnelles, certains systèmes de gestion d'informations personnelles ont eu recours à cette stratégie dans le but de rendre possible une spécification plus complexe de la recherche à effectuer.

OntoPIM [KPS+05] utilise des requêtes conjonctives posées sur l'ontologie de l'utilisateur. La résolution de ces requêtes est effectuée en exploitant un ensemble de *mappings* entre les différentes ontologies. Ces mappings sont créés afin de retrouver les documents correspondant dans le niveau physique. Semex [CDH+05] offre trois façons d'explorer l'espace d'informations personnelles : (*i*) la recherche par mots-clés (*ii*) la recherche par type d'information en sélectionnant une classe ou une association de l'ontologie et (*iii*) en utilisant une requête conjonctive exprimée en fonction

du vocabulaire de l'ontologie. Ce type de requêtes est destiné à être utilisé par des utilisateurs initiés aux langages de requêtes. Xiao et Cruz [XC05] dans leur modèle proposent un moteur de requêtes fondé sur deux techniques complémentaires : d'une part, la recherche par mots-clés qui permet de faire correspondre la liste des mots-clés à un ensemble de documents candidats en calculant un score de similarité pour chaque correspondance, puis de retourner les résultats de façon ordonnée. D'autre part, un langage plus formel pour écrire des requêtes plus complexes en utilisant le langage RDQL (RDF Data Query Language) [RDQ04]. Quant à PIMO [SVED07], il utilise le langage SPARQL pour écrire les requêtes.

1.4.3 Stratégie orientée facette

Un dernier type de stratégie de recherche proposé dans la littérature est la stratégie de recherche par facette. Il s'agit d'une technique de recherche d'informations s'appuyant sur une classification par facettes et permet de filtrer une collection d'informations en choisissant un ou plusieurs critères. Ce type de stratégie a été utilisé dans iMecho [CGWW09] où les auteurs raffinent les résultats de recherche par mots-clés en sélectionnant les catégories correspondant à la recherche. Cette stratégie a été également utilisée dans Stuff I've seen [DCC+03] qui propose un système d'interrogation itératif. L'utilisateur spécifie le mot-clé à chercher et le système retourne toutes les ressources possibles correspondantes. Ensuite, l'utilisateur peut affiner sa recherche en spécifiant le type de ressources (***e.g.,*** courriel, page Web), la date d'enregistrement ou l'auteur.

1.5 Conclusion et discussion

Nous avons présenté dans ce chapitre un certains nombre de systèmes et d'approches de gestion d'informations personnelles. Nous avons étudié d'une part leur modèles de représentation en les classifiant selon leurs degrés de sémantisation. D'autre part, nous avons étudié leurs stratégies de recherche. Un résumé de cette étude comparative est donné en tableau 1.1.

En nous appuyant sur ce tableau (tableau 1.1), il apparaît que :

— Les approches de gestion d'informations personnelles se sont intéressées à la représentation des documents stockés sur les postes de travail en les considérant comme étant des informations personnelles. Seuls Forget-me-not et Semex se sont intéressés aux méta-données.

— L'utilisation de langages de requêtes est effectuée uniquement avec les approches sémantiques. Cependant, la navigation et la recherche par

TABLEAU 1.1: Systèmes de gestion d'informations personnelles

					Recherche			
	Approches	*Document*	*Méta-donnée*	*Contexte*	*Navigation*	*Mot-clé*	*Requête*	*Facette*
Non-sémantique	FORGET-ME-NOT [LF94]		✓		✓			
	LIFESTREAMS [FG96]	✓		Temps	✓	✓		
	TIMESCAPES [Rek99]	✓		Temps	✓	✓		
	PLACELESS DOCUMENTS [DEL+00]	✓				✓		
	MYLIFEBITS [GBL+02]	✓				✓		
	STUFF I'VE SEEN [DCC+03]	✓				✓		
	UMEA [Kap03]	✓			✓			
	TIMESPACE [KJ05]				✓			
Sémantique	SEMEX [CDH+05]	✓	✓			✓	✓	
	Xiao et Cruz [XC05]	✓			✓			
	ONTOPIM [KPS+05]	✓					✓	
	Beagle++[CGG+05]	✓		Activité		✓		
	iMEMEX [DSK06]	✓					✓	
	PIMO [SBD05, SVED07]	✓					✓	
	iMECHO [CGWW09]	✓						✓

mots-clés ont été utilisées par les différents types de systèmes. Ceci pourrait s'expliquer par le niveau d'abstraction suffisant du modèle utilisé grâce à la représentation sémantique.

— Certains de ces systèmes ont utilisé le terme contexte soit pour délimiter le domaine de leur application comme Beagle++[CGG+05] soit comme une façon d'organiser les informations comme avec LIFESTREAM [FG96] ou TIMESCAPE[Rek99].

Nous rappelons que l'un des objectifs de cette thèse porte sur la réalisation automatique des besoins utilisateur par utilisation des informations personnelles. L'une des problématiques posées concerne la représentation uniforme des informations personnelles et de leurs interrogations. Ces informations sont des méta-données fournies par l'utilisateur ou extraites à partir de ses documents numériques.

Suite à l'analyse présentée ci-dessus, les approches sémantiques présentent un niveau d'expressivité plus élevé que celui des approches non-sémantiques. Elles offrent ainsi un vocabulaire riche pour l'interrogation des informations.

La représentation des contextes est un atout majeur lors de l'automatisation et de la réalisation des tâches. Nous choisissons alors de représenter les informations personnelles en fonction d'un certain nombre de contextes d'utilisation. Nous

décidons de les représenter aussi par un modèle sémantique, permettant ainsi de pallier l'absence de certaines valeurs.

En conséquence du choix que nous avons effectué concernant la représentation contextuelle des informations personnelles, il est important que la recherche et la sélection des informations soient aussi sensibles aux contextes. En effet, sélectionner les informations personnelles en prenant en compte un contexte d'utilisation favorise l'obtention de résultats plus pertinents.

Chapitre 2

Composition de services

Sommaire

La composition de services est le processus de construction d'applications à valeur ajoutée par réutilisation de services disponibles. En se basant sur ces services comme éléments atomiques, la composition de services favorise le faible couplage entre les entités logicielles constituant une application et rend ainsi plus aisée leur évolution : les services peuvent être plus facilement remplacés par d'autres en cas de défaut ou de disponibilité de nouveaux services présentant une meilleure qualité.

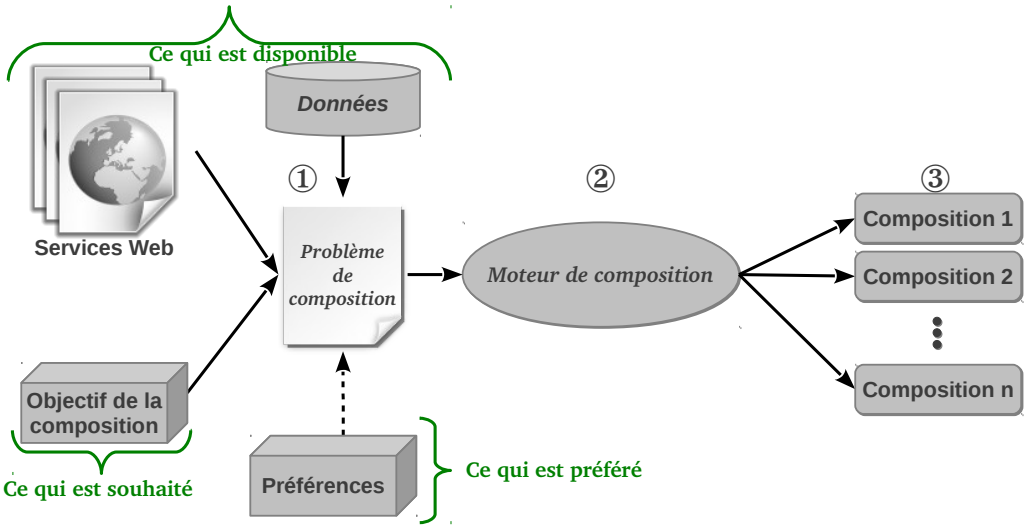


FIGURE 2.1: Description du processus de composition

Le problème auquel nous nous intéressons consiste en la réalisation d'objectifs utilisateur de haut niveau par utilisation d'applications Web et d'informations personnelles. La composition de services est le candidat naturel pour résoudre ce problème. Cependant, la spécificité de ce dernier (*i.e.* caractère sensible des informations personnelles, importance de l'utilisation de contextes dans le besoin de l'utilisateur et dans la valuation des données) nécessite d'étendre les approches existantes de composition de services.

Dans ce chapitre, nous faisons un état de l'art de la composition de services et des techniques automatiques utilisées. Sur la base des critères liés à notre problème, nous étudions aussi l'adéquation de ces techniques à notre problème. En section 2.1, nous présentons le processus de composition et les éléments qui le composent. Nous listons ensuite, en section 2.2, un ensemble de critères d'évaluation de différentes approches de composition en vue de notre problématique. En section 2.3, nous détaillons les différentes techniques automatiques de composition. Enfin, nous terminons en section 2.4 par une conclusion et une discussion.

2.1 Processus de composition de services

La composition de service est effectuée pour répondre à un objectif donné. Ce dernier peut être exprimé en fonction de différents éléments : les éléments dont l'utilisateur dispose, ce qu'il cherche à faire et ce qu'il préfère.

Après une étude de différentes synthèses [RS05, DS05, MP09, BSD14] et d'ap-

proches de composition proposées dans la littérature, il apparaît que le processus de composition de services s'articule autour de trois éléments, comme le montre la figure 2.1 : les entrées du processus de composition aussi appelées problème de composition (①), le moteur de composition (②) et le résultat de la composition (③).

2.1.1 Problème de composition

Les entrées du problème de composition représentent la traduction du besoin de l'utilisateur et les éléments aidant à son accomplissement. Elles sont exprimées en fonction de quatre composants : *les services disponibles, les données de l'utilisateur, l'objectif de la composition* et les éventuelles *préférences* à satisfaire par les compositions produites.

Dans la littérature, la composition de services a principalement été étudiée dans le contexte de services Web [HB03, WSH+06, SPM09] qui sont la principale implantation des SOA. Nous ferons donc de même dans cette étude.

2.1.1.1 Services

Un service Web est une entité logicielle modulaire qui permet de fournir une fonctionnalité à d'autres applications à travers le Web. Les interactions entre les applications sont effectuées à travers l'envoi et la réception de messages et en utilisant des langages standards : WSDL [CMRW07] pour la description de l'interface publique d'accès aux services et SOAP [BEK+00] pour la communication à base de messages entre client et service. L'interface d'un service Web est composée d'une ou de plusieurs opérations. Pour faciliter l'utilisation des services et automatiser leur découverte, leur sélection et leur composition, il est important de fournir une description explicite pour chaque opération, ce qu'elle prend en entrée, ce qu'elle réalise en terme de fonctionnalités et ce qu'elle produit en sortie.

Plusieurs définitions de « *service* » ont été données. Nous pouvons retenir tout d'abord celle du W3C :

> *A service is an abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of providers entities and requesters entities. To be used, a service must be realized by a concrete provider agent.* [W3C14]

Cette définition spécifie qu'un service est décrit par une description abstraite précisant sa fonctionnalité. Cette description abstraite ne suffit pas pour exécuter

un service qui doit être réalisé par un fournisseur concret.
Une définition complémentaire a été donnée par Arsanjani.

> *A service is a software resource with an externalized service description. This service description is available for searching, binding, and invocation by a service consumer.* [Ars04]

Cette définition spécifie qu'un service a une description externalisée. Cette description permet aux utilisateurs de le trouver, de l'associer avec d'autres services et de l'invoquer.

Un service peut être décrit de diverses façons. Canal et al. [CMP+06] le décrivent en fonction de quatre couches : *la couche signature*, *la couche comportement*, *la couche sémantique* et *la couche non-fonctionnelle*. Arsanjani [Ars04] décrit un service en deux niveaux : *fonctionnel* et *non-fonctionnel*. Ces deux modèles sont en fait équivalents car le niveau fonctionnel du deuxième modèle contient implicitement la signature, le comportement et la sémantique du premier.

Le ***niveau fonctionnel*** qui est souvent appelé simplement « *interface* » permet de décrire un service par *des attributs en entrée* (spécifiant les données nécessaires à son invocation), *des attributs en sortie* (représentant les données produites par le service), et *des fonctionnalités* explicitant les tâches élémentaires réalisées par le service. Un service peut avoir zéro ou plusieurs fonctionnalités. Dans le cas où il est décrit par plus d'une fonctionnalité, leur ordonnancement peut être représenté par une conversation. Un exemple de conversation est donné en figure 2.2. Elle présente le comportement d'un service typique de payement en ligne décrit par plusieurs fonctionnalités. Ce service permet de se connecter, d'ajouter du crédit à son compte puis soit de demander un ticket et de se déconnecter ensuite, soit de se déconnecter directement. Un service qui ne comporte aucune fonctionnalité peut être utilisé pour des besoins techniques ou théoriques [HB03].

Le ***niveau non-fonctionnel*** permet la prise en compte de propriétés non-fonctionnelles qui sont importantes lors de la sélection et de la composition de

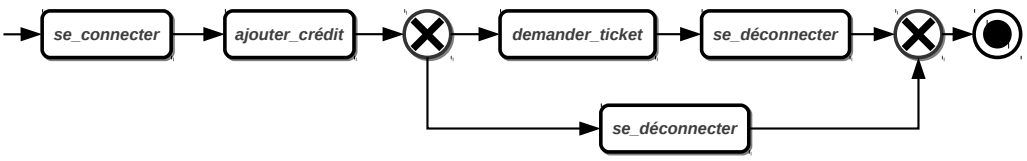


FIGURE 2.2: Exemple de conversation de service [PY10]

services, telles que la sécurité, la fiabilité ou le coût. Ces propriétés sont utilisées afin de favoriser l'automatisation de la prise de décision quant au choix d'un service particulier parmi un ensemble, dans le cas où ils sont tous équivalents du point de vue purement fonctionnel et accomplissent les mêmes fonctionnalités.

2.1.1.2 Données

Dans tout système d'information, les données manipulées par les programmes jouent un rôle important. Il en va de même dans le cadre de la composition de services. Le choix d'une composition de services est ainsi fait en fonction de l'ensemble des données que l'utilisateur est prêt à fournir, et éventuellement de l'ensemble des données qu'il souhaite obtenir grâce à la composition. Il faut noter que, lors du processus de composition, la valeur des données n'intervient pas pour sélectionner un service ou un autre. On s'intéresse essentiellement à la disponibilité de la donnée, à son type primitif (***e.g.,*** entier, chaîne de caractères) et à sa description sémantique (***e.g.,*** date de départ, date de retour). Pour cette raison, nous utiliserons aussi le terme *attribut*.

2.1.1.3 Objectif de la composition

Le troisième élément du problème de composition est son objectif. Il représente une formalisation du besoin de l'utilisateur. Sa description peut contenir différents éléments.

Une ***description orientée attributs*** est la donnée d'un ensemble d'attributs dont l'utilisateur souhaite obtenir les valeurs grâce à l'exécution de la composition.

Une ***description orientée fonctionnalité*** est la donnée d'un ensemble de fonctionnalités que l'utilisateur souhaite réaliser grâce à cette exécution. Ces fonctionnalités peuvent être spécifiées avec (conversation) ou sans ordre précis entre les fonctionnalités. Le but, dans le second cas est d'exécuter toutes les fonctionnalités pour accomplir l'objectif sans se soucier de l'ordre particulier d'exécution de ces dernières. Dans le premier cas l'utilisateur souhaite que la réalisation des fonctionnalités respecte un ordre particulier. Cet ordre peut être décrit à l'aide d'opérateurs de séquence (***e.g.,*** F1 puis F2), de choix exclusif (***e.g.,*** F1 ou F2) ou de parallélisme (***e.g.,*** F1 et F2).

Pour décrire les conversations, plusieurs langages ont été proposés : *diagrammes d'activité d'UML* [OMG13], *workflows* [HH93], *LTS* [UKM03] ou *réseaux de Petri* [GV03].

Une ***description orientée contraintes de qualité de service (QoS)*** consiste à spécifier un seuil minimal de qualité qui doit être respecté par les compositions générées. Ceci est réalisé en prenant en compte et en exploitant des critères de qualité participant à la description des services et des métriques de qualité.

2.1.1.4 Préférences

Le dernier élément du problème de composition correspond aux préférences de l'utilisateur. Elles enrichissent le processus de composition par un ensemble de propriétés additionnelles (***e.g.,*** qualité du service, politiques de sécurité, contextes). Ces propriétés sont utilisées pour quantifier la qualité des solutions possibles au problème de composition en exploitant les descriptions non-fonctionnelles des services qui composent ces solutions. L'utilisation des préférences permet également d'ordonner les solutions obtenues en fonction du score calculé pour chacune d'elles.

2.1.2 Moteur de composition

Un moteur de composition est un algorithme (et un outil ou composant logiciel lorsque l'algorithme est implanté) qui prend en entrée un problème de composition traduit dans un langage compréhensible par la machine (***e.g.,*** XML, PNML, PDDL) et produit une ou plusieurs compositions répondant aux contraintes spécifiées dans le problème de composition. Dans le cas où aucune composition n'existe le moteur de composition retourne un ensemble vide.

Les approches de composition présentées dans la littérature utilisent différents formalismes pour représenter les problèmes de composition et différentes techniques pour calculer les compositions. Les principales familles de formalismes utilisées sont les systèmes de planification, les réseaux de Petri, les automates et les règles logiques. Une étude plus détaillée des différents types d'approches est présentée dans la section 2.3.

2.1.3 Résultat du moteur de composition

Le dernier constituant du processus de composition correspond à ses résultats. Le processus de composition peut produire une seule composition, l'ensemble de toutes les compositions possibles, les meilleures [BBHB11] ou la meilleure composition [SPM09]. Dans les deux derniers cas, une sélection peut être faite en fonction d'un ensemble de critères (***e.g.,*** moins coûteuse, plus rapide) spécifiés par les préférences de la composition. Les moteurs de composition produisent en général des compositions exprimées dans le langage interne du moteur de composition (***e.g.,*** plan, trace, LTS ou arbre de preuve). Afin de présenter une composition à un utilisateur,

il est nécessaire de la traduire en un langage compréhensible par celui-ci ou dans un format exécutable tel que BPEL (Business Process Execution Language) [WS-14].

2.2 Critères d'évaluation

De nombreuses approches de composition ont été proposées dans la littérature (voir [RS05, DS05, MP09, BSD14], pour une vue d'ensemble). Pour comparer ces approches, différents critères d'évaluation peuvent être utilisés. Dans cette thèse, nous avons choisi de les comparer en terme d'expressivité et d'efficacité.

2.2.1 Expressivité

S'appuyer uniquement sur les entrées et les sorties des services ne suffit pas toujours pour produire une composition parfaitement adaptée aux besoins de l'utilisateur. Il est souvent nécessaire pour cela d'augmenter l'expressivité du problème de composition. Cela consiste à l'enrichir avec un ensemble de contraintes et d'exigences additionnelles permettant de mieux spécifier le besoin.

La prise en compte de ***conversations*** permet d'augmenter l'expressivité des interfaces des services et ainsi de prendre en compte l'ordre dans lequel leurs opérations doivent être appelées et dans lequel leurs fonctionnalités sont réalisées. De même, les conversations augmentent l'expressivité de l'objectif de la composition en permettant de spécifier l'ordre dans lequel les différentes tâches qui le composent doivent être réalisées.

La prise en compte de la ***qualité de service*** permet de favoriser la sélection d'un service par rapport à un autre service ayant la même fonctionnalité. Elle permet aussi, à fonctionnalités équivalentes, de sélectionner la composition ayant le meilleur rendu pour l'utilisateur (***e.g.,*** coût ou temps d'exécution).

Des ***politiques de sécurité*** peuvent être définies au niveau des services et au niveau de l'objectif de composition. Dans le contexte de la thèse, elles permettent de limiter la composition aux services qui sont autorisés par l'utilisateur à utiliser ses informations personnelles.

Les ***contextes***, enfin, permettent d'adapter et de personnaliser la solution d'un problème de composition à une situation donnée. Cela peut correspondre à un contexte géographique (***e.g.,*** en France vs à l'étranger), un contexte temporel (***e.g.,*** en semaine vs le *week-end*), un contexte social (***e.g.,*** pour le travail ou pour des raisons privées) ou une combinaison de ces derniers. Le contexte peut être présent au niveau

de la description des services, faire partie de l'objectif de composition ou même être associé aux valuations des données utilisables.

2.2.2 Efficacité

L'efficacité d'un processus de composition peut être définie en fonction de critères de deux natures différentes. Il peut s'agir de critères théoriques tels que la complexité en temps ou en mémoire, calculée en fonction de la taille des entrées du problème de composition. Mais en raison du caractère applicatif de la problématique de composition, il peut aussi s'agir de critères plus pratiques.

Le ***nombre de solutions calculées*** est un premier critère. Comme nous l'avons présenté dans la section 2.1.3, une approche de composition peut générer (figure 2.3) une (1), toutes ($\forall$), plusieurs (n), les n meilleures ($\lceil n \rceil$) ou la meilleure solution ($\lceil 1 \rceil$). La possibilité de générer plusieurs solutions est un atout vu qu'elle peut permettre, en cas de défaillance, d'exécuter une autre solution sans avoir à recalculer une nouvelle composition.

La capacité à générer des ***compositions avec du parallélisme*** est un autre critère important. En effet, les compositions avec du parallélisme sont plus efficaces. Par exemple, soient trois services, w_1, w_2 et w_3. Le temps d'exécution du service w_1 (resp. w_2 et w_3) est noté t_{w_1} (resp. t_{w_2} et t_{w_3}). Le temps d'exécution d'une composition constituée de w_1 et w_2 en parallèle suivie de w_3 est $max(t_{w_1}, t_{w_2}) + t_{w_3}$. En revanche le temps d'exécution d'une composition séquentielle constituée de w_1 suiv de w_2 puis de w_3 est $t_{w_1} + t_{w_2} + t_{w_3}$.

Le ***degré d'automatisation*** est un troisième critère important, le nombre de services disponibles ainsi qu'une expressivité accrue des services (avec une conversation) rendent irréalisable le fait de composer ces derniers manuellement. La composition manuelle suppose que la recherche des services, la modélisation de la composition ainsi que son implantation se fassent manuellement. Une composition automatique peut être de deux types. Le premier type consiste à modéliser la conversation du

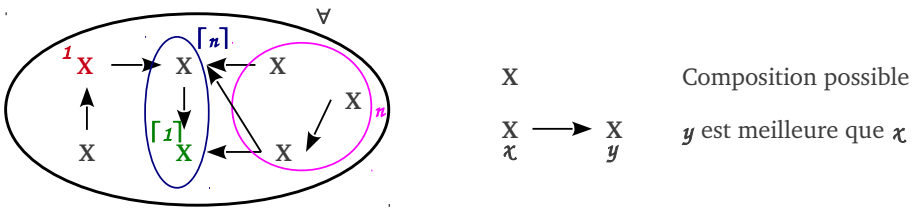


FIGURE 2.3: Espace de compositions possibles

besoin en termes de capacités et d'automatiser la découverte de la sélection des services susceptibles de remplacer chaque capacité (ds). Ceci est fait dans la plupart des cas sans se soucier des possibles correspondances à avoir entre les entrées et les sorties de chaque service. Le deuxième type consiste à automatiser toutes les étapes afin de produire des compositions cohérentes où les entrées et les sorties des services successifs s'accordent (c). Un dernier critère, important, consiste en la présence d'un outil disponible offrant la possibilité de tester et de comparer les approches (i).

2.3 Approches de composition de services

De nombreuses approches de composition de services ont été proposées dans la littérature. Afin de les comparer, différents types de classification ont été utilisés. Certains travaux se sont intéressés à l'usage de la composition (conception, exécution) [DS05, RF11], d'autres au degré de la composition [RS05] (automatique ou semi-automatique) ou encore à la technique de composition utilisée [Pee05, BP10]. C'est ce dernier choix que nous suivons car il nous guidera sur la technique à réutiliser et à étendre. Nous classifierons donc, dans la suite, les approches de composition en quatre catégories : les approches fondées sur la planification, les approches fondées sur les réseaux de Petri, les approches fondées sur les automates et les approches fondées sur le raisonnement logique.

Pour faciliter leur compréhension, nous détaillons les différents types d'approches de composition en les illustrant sur un exemple simple de problème de composition de services (Exemple 2.1).

Exemple 2.1. *Soient trois services w_1, w_2 et w_3, et une donnée disponible a. Le but de la composition de services est d'obtenir la donnée e. Chaque service w_i est décrit par l'ensemble des données qu'il requiert (entrées) et l'ensemble des données qu'il produit (sorties) sous la forme d'une paire d'ensembles ({entrées},{sorties}) :*

- $w_1 = (\{a\}, \{b\})$
- $w_2 = (\{a\}, \{c\})$
- $w_3 = (\{b, c\}, \{e\})$

2.3.1 Planification

La première technique utilisée par les approches de composition est la planification. Cette technique est définie par Ghallab et al. comme suit :

> *Planning is the reasoning side of acting. It is an abstract, explicit deliberation process that chooses and organizes actions by anticipating their expected outcomes.* [GNT04]

Selon cette définition, la planification est une technique de raisonnement sur les actions. Elle permet de les choisir et de les organiser afin de produire un objectif attendu. De nombreuses approches de composition [WSH+06, LKS08, PY10] ont utilisé la planification pour résoudre leurs problèmes de composition. En effet, il existe, comme nous allons le voir dans la suite, une certaine similarité entre les problèmes de composition et ceux résolus par la planification.

Étant donné un ensemble de propositions, un problème de planification est composé de trois éléments :

- un ensemble d'actions qui agissent sur les propositions,
- un ensemble de propositions initiales (vraies au départ) et
- un ensemble de propositions objectifs.

La planification a été appliquée à la composition en utilisant différents types d'approches telles que le chaînage [BF97] ou la planification hiérarchique [EHN94].

Le chaînage consiste en la construction d'un espace de recherche traduisant le problème de composition et la recherche d'une solution dans cet espace. La solution sera ensuite extraite en utilisant un algorithme de recherche (***e.g.,*** chaînage-arrière [McG82]). Une des techniques les plus utilisées pour le chaînage est *GraphPlan* que nous détaillons dans la suite pour illustrer l'exemple 2.1. La planification hiérarchique consiste à utiliser non seulement un ensemble d'actions mais aussi un ensemble de règles décrivant la manière dont une tâche complexe peut être décomposée en tâches plus simples. Le principe de cette technique est de décomposer récursivement l'objectif en un ensemble de sous-tâches jusqu'à atteindre les tâches primitives correspondant aux services.

Présentation de *GraphPlan*. Cette approche consiste à construire un graphe de planification, sur lequel on applique un algorithme de recherche pour extraire une solution. Le ***graphe de planification*** est un graphe orienté biparti. Il est composé d'une alternance de niveaux composés de deux types de nœuds : les propositions qui constituent les niveaux de propositions et les actions qui constituent les niveaux d'actions.

Un état est composé par toutes les propositions vraies à un instant donné, par exemple l'état initial est constitué de toutes les propositions initiales du problème.

Une action est définie par ses préconditions, ses effets négatifs et ses effets positifs. Les préconditions représentent les propositions qui doivent être satisfaites avant l'exécution d'une action et les effets négatifs (resp. les effets positifs) représentent les propositions à consommer (resp. les propositions à produire) après l'exécution de l'action.

La procédure de construction du graphe de planification, à partir d'un problème de planification est effectuée en quelques étapes détaillées ci-dessous.

$\mathbf{PL}_i$ (resp. $\mathbf{AL}_i$) représente le $i^{ème}$ niveau de propositions (resp. d'actions) du graphe de planification.

1. le premier niveau de propositions $\mathbf{PL}_1$ contient toutes les propositions initiales,
2. le $i^{ème}$ niveau d'actions $\mathbf{AL}_i$ contient toutes les actions applicables à partir du $i^{ème}$ niveau de propositions. Une action $\mathbf{a}$ est dite applicable si et seulement si toutes ses préconditions $\mathbf{Pre}(\mathbf{a})$ et ses effets négatifs $\mathbf{Eff}^-(\mathbf{a})$ sont dans $\mathbf{PL}_i$
3. le niveau suivant de propositions, $\mathbf{PL}_{i+1}$, est calculé en ajoutant à $\mathbf{PL}_i$ tous les effets positifs des actions du niveau d'actions $\mathbf{AL}_i$ ainsi qu'en éliminant tous les effets négatifs de ces actions.
4. toute action $\mathbf{a} \in \mathbf{AL}_i$ est connectée à ses préconditions (resp. ses effets négatifs) dans $\mathbf{PL}_i$ par des arcs de préconditions (resp. des arcs à effets négatifs) et avec ses effets positifs dans $\mathbf{PL}_{i+1}$ par des arcs à effets positifs.
5. des actions spécifiques (no-ops) sont utilisées afin de garder les propositions d'un niveau pour les transmettre au niveau suivant.
6. la construction du graphe de planification s'arrête avec succès lorsque l'objectif est atteint, *i.e.* le but a été généré dans le dernier niveau de propositions. La construction peut également s'arrêter avec échec, lorsque deux niveaux consécutifs de propositions sont identiques (point-fixe).

   Il a été démontré que s'il n'existe aucune solution, la construction se termine toujours par un point-fixe, ce qui garantit la terminaison du processus [BF97].

Après l'étude de nombreuses approches de composition utilisant la planification [WSH+06, LKS08, PY10], nous remarquons qu'un problème de composition de services peut être traduit en un problème de planification comme suit :

— à chaque service correspond une action dont les préconditions (resp. les effets positifs) correspondent aux attributs d'entrée du service (resp. aux attributs en sortie),
— les propositions initiales sont les données disponibles du problème de la composition et
— les propositions objectifs du problème de planification correspondent à l'objectif du problème de composition.

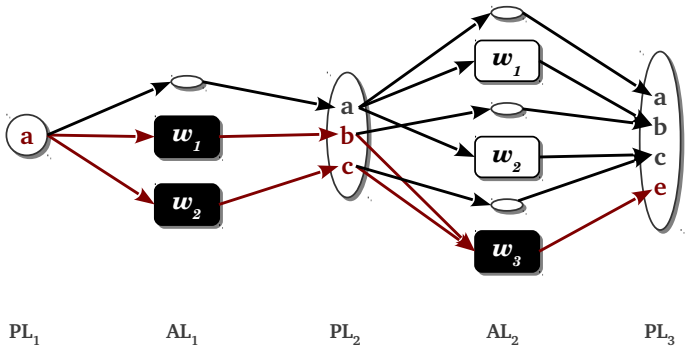


FIGURE 2.4: Graphe de planification de l'exemple 2.1

Exemple 2.2. *En nous appuyant sur la technique de GraphPlan, le problème de composition donné en exemple 2.1 est traduit en un problème de planification comme suit :*

INIT : a

BUT : e

$\star$ *Action* (w_1)	$\star$ *Action* (w_2)	$\star$ *Action* (w_3)
Pre : a	**Pre** : a	**Pre** : b, c
Eff$^+$: b	**Eff**$^+$: c	**Eff**$^+$: e

Le graphe de planification correspondant à ce problème est présenté en figure 2.4.

Le niveau de proposition $\mathbf{PL}_1$ *est composé par la proposition* a*. À partir de ce niveau deux actions peuvent être appliquées au niveau actions* $\mathbf{AL}_1$*,* w_1 *et* w_2*. Le niveau de proposition suivant* $\mathbf{PL}_2$ *contient en plus des propositions du niveau* $\mathbf{PL}_1$ *les effets positifs de toutes les actions du* $\mathbf{AL}_1$*, à savoir* b *et* c*. La construction du graphe s'arrête à l'obtention de la proposition* e*. La solution à ce problème est obtenue par parcours arrière à partir des objectifs :* $w_1 \;//\; w_2 \,;\, w_3$*, où "//" représente la concurrence d'appel et ";" la séquence*

Conversation. La prise en compte de conversations permet de préciser dans quel ordre les fonctionnalités à réaliser sont attendues dans une composition. Dans ce contexte, Kuter et al. [KSP+05] présentent une approche de composition basée sur l'utilisation de la planification hiérarchique. La conversation est prise en compte par la description hiérarchique de tâches. De plus, les auteurs prennent en considération la possibilité que les informations disponibles dans l'état initial soient incomplètes. Le but est donc de réaliser la conversation objective tout en collectant les informations manquantes. Cette approche est supportée par un prototype, ENQUIRER, qui permet de sélectionner aléatoirement un plan parmi ceux générés au cours du processus de composition. Wu et al. [WPS+03] proposent une autre approche de composition basée sur la planification hiérarchique et qui supporte la conversation de l'objectif. L'intérêt est que cette approche a été testée sur de vrais services pour

s'assurer de son passage à l'échelle. La planification hiérarchique est aussi utilisée par Klusch et al. [KGS05] qui la combine avec une technique de graphe de planification relaxée[1] (FF-Planner). Cette combinaison permet de bénéficier des avantages des deux types de planification. Elle permet d'une part de trouver une solution, s'il en existe une, grâce à l'utilisation du graphe de planification et d'autre part de décomposer les tâches complexes en des tâches simples et de gérer les conversations grâce à l'utilisation de la planification hiérarchique. L'approche proposée permet aussi de s'adapter en cas de panne et de recalculer une nouvelle composition.

Qualité de service. Plusieurs approches basées sur la planification se sont intéressées à la prise en compte de la qualité de service dans la composition de services. Parmi ces approches nous pouvons citer Yan et al. [YCY12] qui se base sur *GraphPlan*. Des critères de qualité sont pris en compte durant le processus de construction du graphe de planification en l'annotant par des scores de qualité (un poids est attribué à chaque nœud). Le meilleur plan est obtenu en appliquant l'algorithme de Dijkstra [Dij59] sur le graphe annoté.

Contextes. Un autre critère d'expressivité qui a été étudié est le contexte. Dans ce cadre, Vukovic et Robinson [VR04] proposent une approche basée sur la planification hiérarchique. Les contextes et leurs comportements attendus sont modélisés sous la forme d'actions. L'utilisateur précise pour le problème de composition les contextes à considérer comme entrées du problème de composition.

Conversation et qualité de services. D'autres approches de composition ont choisi de combiner deux ou plusieurs des critères d'expressivité, comme par exemple d'utiliser la conversation et la qualité de service en même temps. Dans ce contexte, Chen et al. [CPY14] proposent une approche de composition basée sur *GraphPlan* qui utilise la QoS pour la recherche de la meilleure solution. Cette approche combine les techniques utilisées dans les travaux de Poizat et Yan [PY10] et de Yan et al. [YCY12].

Conversation et préférences. Lin et al. [LKS08] proposent une approche de composition supportant les conversations et les préférences, basée sur la planification hiérarchique. Pour traiter les préférences, les auteurs décomposent l'objectif désiré en s'appuyant sur une ontologie de préférences de l'utilisateur et ils calculent la meilleure solution en utilisant une heuristique de recherche. Une autre approche basée sur la planification hiérarchique a été proposée par Sohrabi et McIlraith [SM10]. Cette approche permet de calculer la meilleure composition selon un score de préférence calculé en utilisant à nouveau une heuristique de recherche.

Nous avons pu voir que la planification a été utilisée avec succès pour la composition de services, y compris en présence de conversations, de qualité de services,

1. La relaxation est un principe fondamental pour la génération de fonctions heuristiques. Un graphe de planification relaxé signifie que le problème utilisé pour construire le graphe considère le moins de contraintes possible (***e.g.,*** ignorer les effets négatifs des actions).

de contextes et de préférences. Une synthèse comparative des approches présentées ci-dessus est donnée en tableau 2.1. Cette synthèse est basée sur l'utilisation de deux principaux critères d'expressivité et d'efficacité. Pour l'expressivité, nous comparons les approches en fonction de la présence ou non de conversations pour les services (S) et pour l'objectif (O), la prise en compte de critères de qualité, la prise en compte de préférences pour la sélection de la meilleure solution ou l'utilisation des contextes. En ce qui concerne les critères d'efficacité, nous nous appuyons sur le nombre de solutions (compositions) calculables et sur le degré d'automatisation de l'approche. Nous utilisons les notations suivantes pour la colonne Solutions : 1 pour une solution, $\forall$ pour toutes les solutions, n pour plusieurs solutions, $\lceil n \rceil$ pour les n meilleures solutions et $\lceil 1 \rceil$ pour la meilleure solution. Nous ajoutons le symbole $||$ pour la capacité de produire des solutions parallèles. Concernant la colonne Automatisation nous utilisons $\checkmark_{ds}$ pour désigner une approche automatique de découverte et de sélection, $\checkmark_{c}$ pour une approche automatique de composition, $\checkmark_{ds+i}$ pour une approche automatique de découverte et de sélection avec la disponibilité d'un outil et $\checkmark_{c+i}$ pour désigner une approche de composition automatique avec la disponibilité d'un outil. Nous ajoutons une colonne pour préciser la technique de composition utilisée en dénotant par GP une approche à base de *GraphPlan*, par HTN une approche basée sur la planification hiérarchique, par HP une approche heuristique, et nous ajoutons le symbole + si l'approche est une extension de ces derniers.

TABLEAU 2.1: Approches de composition fondées sur la planification

Approches	***Technique utilisée***	***Expressivité***					***Efficacité***	
		Conversation		*QoS*	*Préférence*	*Contexte*	*Solutions*	*Automatisation*
		S	O					
Wu et al. [WPS+03]	*HTN*		✓				1	$\checkmark_{c+i}$
Vukovic et Robinson [VR04]	*HTN*					✓	1	$\checkmark_{c+i}$
Kuter et al. [KSP+05]	*HTN*		✓				1	$\checkmark_{c+i}$
Klusch et al. [KGS05]	HP^{+}						1	$\checkmark_{c+i}$
Lin et al. [LKS08]	*HTN*		✓		✓		$\lceil 1 \rceil$	$\checkmark_{c+i}$
Sohrabi et McIlraith [SM10]	*HTN*		✓		✓		$\lceil 1 \rceil$	$\checkmark_{c+i}$
Poizat et Yan [PY10]	*GP*	✓	✓				1||	$\checkmark_{c+i}$
Yuhong et al. [YCY12]	GP^{+}		✓	✓			$\lceil 1 \rceil$||	$\checkmark_{c}$
Chen et al.[CPY14]	GP^{+}	✓	✓	✓			$\lceil 1 \rceil$||	$\checkmark_{c}$

Analyse (tableau 2.1). Les approches basées sur *GraphPlan* permettent de produire des compositions contenant du parallélisme, elles sont ainsi plus rapides. En revanche, la prise en compte de la conversation est difficile. Les approches utilisant la

planification hiérarchique supportent naturellement les conversations (sous la forme d'un arbre).

2.3.2 Réseaux de Petri

Une deuxième catégorie d'approches de composition est fondée sur les réseaux de Petri. Un réseau de Petri [Pet62] est un graphe biparti, composé de deux types de nœuds (places et transitions), ainsi que d'arcs reliant places et transitions. Un réseau de Petri (P/T simple) marqué présente un certain nombre de caractéristiques :

- chaque place contient un nombre entier positif ou nul de jetons.
- une transition est tirable si toutes ses places d'entrées contiennent au moins un jeton.
- le franchissement d'une transition consiste à enlever un jeton de chacune de ses places d'entrée et à ajouter un jeton à chacune de ses places de sortie.

Un marquage du réseau représente un état de sa sémantique et le graphe de marquages représente les enchaînements possibles de ces états en fonction du franchissement des transitions.

Un problème de composition peut être modélisé en un réseau P/T marqué comme suit :

- chaque donnée du problème (entrées/sorties des services) correspond à une place
- chaque service correspond à une transition
- pour chaque service w, un arc à double sens est spécifié entre les places correspondant aux entrées de w et la transition lui correspondant. De plus un arc simple est spécifié entre la transition correspondante du service et les places correspondantes à ses attributs en sortie.
- les places correspondant aux données initiales du problème sont marquées par des jetons.

Exemple 2.3. *Nous illustrons le déroulement de cette technique avec l'exemple 2.1. Le problème de composition est transformé sous la forme d'un réseau de Petri comme suit :*
Places : a, b, c, e
Transition : w_1, w_2, w_3
Marquage initial : a
Objectif : e

Le réseau de Petri correspondant à ce problème est donné en figure 2.5.

Le graphe de marquage (une sous partie, car il est infini) de ce réseau est donné en figure 2.6 :

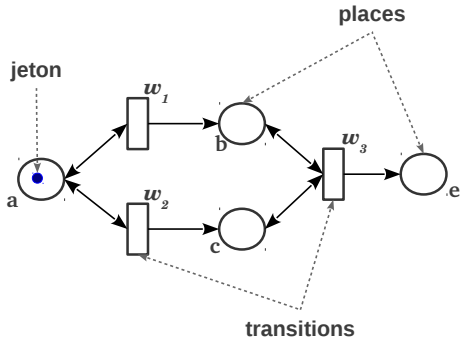


FIGURE 2.5: Réseau de Petri de l'exemple 2.1

La construction du graphe des marquages d'un réseau de Petri permet de vérifier s'il existe un état atteignable où l'ensemble des places correspondant aux données objectifs sont marquées par au moins un jeton.

Dans le but d'extraire une solution, il est possible de faire appel à un outil de *model checking* [CGP99]. Le principe du *model checking* est donné en figure 2.7. Il s'agit de vérifier si un modèle satisfait ou non une propriété (***e.g.,*** l'absence de blocage).

A l'issue de cette vérification, si la *propriété* n'est pas satisfaite, un *contre exemple* est fourni, permettant ainsi d'identifier un cas d'exécution non conforme et ainsi de localiser et de corriger la source de l'erreur.

Dans le cadre de la composition de services, il est possible d'utiliser la capacité du *model checking* à générer des contre-exemples. Pour cela, l'objectif de la composition est transformé en formule puis sa négation est vérifiée par le *model checking*. Dans le cas où cette négation est satisfaite, cela veut dire qu'il n'existe aucun état atteignable satisfaisant l'objectif de la composition. Dans le cas contraire, le *model checker* présente un contre exemple qui correspond alors à une solution du problème

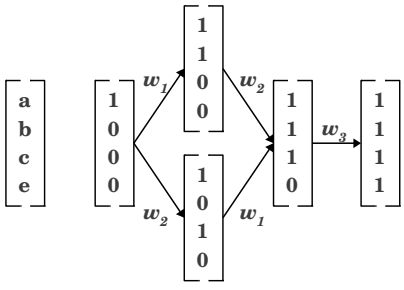


FIGURE 2.6: Graphe de marquage de l'exemple 2.1 (sous partie)

de composition.

Exemple 2.4. *En appliquant la technique du model checking sur notre exemple :*
- *la propriété à vérifier est* $\neg e$.
- *un contre exemple est ainsi donné :* w_1 ; w_2 ; w_3.

Plusieurs approches de composition de services sont basées sur un encodage en réseaux de Petri.

Hétérogénéité. Tan et al. [TFZT10] proposent une approche supportant l'hétérogénéité des vocabulaires de description entre l'objectif de la composition et les services à utiliser. Ils utilisent pour cela un ensemble de règles de mise en correspondance entre les données utilisées par les services et le besoin exprimé lui aussi en terme de données. L'approche génère ensuite un réseau de Petri correspondant à ces règles et extrait une solution en se basant sur un algorithme de décomposition.

Conversation. Hamadi et Benatallah [HB03] proposent une approche de composition de services dont les conversations sont modélisées par des réseaux de Petri. Pour obtenir des compositions, les auteurs utilisent un ensemble d'opérations algébriques (séquence, choix, synchronisation) permettant d'assembler les réseaux de Petri et par conséquence les services. Une approche similaire a été proposée par Brogi et Corfini [BC07]. Il s'agit d'une approche qui prend en compte la phase de découverte des services à composer. Les services sont modélisés par des réseaux de Petri. Le besoin de l'utilisateur est donné en fonction de l'ensemble des données qu'il est prêt à fournir et de l'ensemble des données qu'il souhaite obtenir. La solution obtenue est optimale vu qu'elle contient uniquement les services nécessaires à la réalisation du besoin de l'utilisateur. D'autres approches basées sur les réseaux de Petri

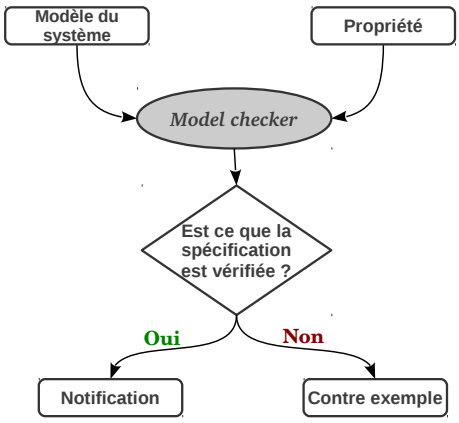


FIGURE 2.7: Principe du *model checking*

ont été proposées par Xia et al. [XZLZ13] et par Narayanan et McIlraith [NM02]. La phase de calcul des compositions est effectuée, dans les deux cas, en exploitant des outils de vérification des réseaux de Petri.

Qualité de service. L'approche de [CEHMR11] utilise les réseaux de Petri en supportant la prise en compte de la QoS dans la composition de services. Les services sont décrits en terme d'attributs en entrée et d'attributs en sortie. Un réseau de Petri est créé à partir de l'ensemble des services disponibles. La composition est faite par utilisation d'un algorithme de sélection se basant sur la maximisation de critères de qualité et le résultat est donné en terme d'un réseau de Petri coloré.

TABLEAU 2.2: Approches de composition fondées sur les réseaux de Petri

	Expressivité				***Efficacité***	
Approches	*Conversation*		*QoS*	*Préférence*	*Solutions*	*Automatisation*
	S	O				
Narayanan et McIlraith [NM02]	✓				1	$✓_{c+i}$
Hamadi et Benatallah [HB03]	✓				1	$✓_{c}$
Brogi et al. [BC07]	✓				$\lceil 1 \rceil$	$✓_{c}$
Ren et al. [RLC+08]		✓			1	$✓_{c+i}$
Tan et al. [TFZT10]					1	$✓_{c}$
Cardinale et al.[CEHMR11]			✓		$\lceil 1 \rceil$	$✓_{ds+i}$
Bao et al.[BGX+13]		✓			1	$✓_{c}$

Analyse (tableau 2.2). L'utilisation des réseaux de Petri dans la composition de services permet de supporter naturellement les conversations sans le passage par un encodage complexe du fait de la présence de transformations de langages expressifs (*workflows*, BPMN) vers les réseaux de Petri. Cependant, la capacité de passer à l'échelle reste à tester.

2.3.3 Automates

La troisième catégorie d'approches de composition est celle fondée sur l'utilisation des systèmes à base d'états et de transitions. Nous utiliserons le terme automate par abus de langage pour désigner ces différents modèles de façon générique.

Un automate est un graphe orienté composé d'un ensemble de nœuds et d'un ensemble d'arcs. Les nœuds sont appelés *états*, et les arcs sont appelés *transitions*.

L'état initial (unique) représente l'état dans lequel débute la dynamique décrite par l'automate. Les états finaux (0 ou plusieurs) représentent l'état terminal (ou objectif).

Les automates supportent différentes opérations telles que la différence, l'intersection et l'union. L'opération la plus utilisée cependant est le produit (synchrone ou asynchrone) [Arn94]. Elle permet en effet de représenter le comportement d'un système composite (éventuellement communicant) à partir des comportements (automates) de ses constituants.

Dans le cadre de la composition de services, le problème de composition est traduit sous la forme d'un automate, éventuellement résultant du produit d'un ensemble d'automates représentant les services atomiques. La recherche de solution dans un système modélisé par un automate peut être résolue par différentes techniques [CDGL+08], par exemple en transformant le problème de composition en problème de satisfiabilité [BCDG+03, BCG+05, BCDG+05]. Le problème peut également être résolu par utilisation de la logique temporelle linéaire (LTL) [PPS06] et du *model checking* [BBD03]. D'autres types d'approches récemment proposé, permettent de calculer directement une composition en se basant sur l'utilisation de variantes de la notion formelle de simulation [BCGP08, SPDG08].

Après une étude des approches de composition de services s'appuyant sur l'utilisation des automates, il apparaît qu'un problème de composition peut être traduit en un automate comme suit :

— un état représente un monde où une ou plusieurs données sont disponibles. L'état initial correspond aux données initialement disponibles. Tout état contenant les données objectifs est final.
— une transition représente un service. Elle relie un état s contenant au moins une donnée nécessaire en entrée du service à un état contenant les données de s plus les données produites par le service.

Exemple 2.5. *Nous illustrons cette technique avec l'exemple 2.1, où le problème de composition est traduit par l'automate de la figure 2.8.*

L'état initial du problème est $\{a\}$ *et l'état final est* $\{a, b, c, e\}$*. Les transitions relient les états, par exemple, la transition* w_1 *est possible depuis tout état contenant* a *et rajoute* b*. Le problème peut se résoudre en utilisant la logique LTL en ajoutant une transition FINAL à l'état final. La formule logique à résoudre est alors* $\neg\Diamond$*FINAL et les compositions obtenues peuvent être* w_1 ; w_2 ; w_3 *ou* w_2 ; w_1 ; w_3

Dans la littérature de nombreuses approches utilisent les automates pour résoudre le problème de la composition. C'est le cas des approches proposées par Berardi at al. [BCG+05, BCGP08], Melliti et al. [MPM08], Pistore et al. [PTB05] et, en prenant en compte des critères de qualité ou des contextes, Ben Mokhtar

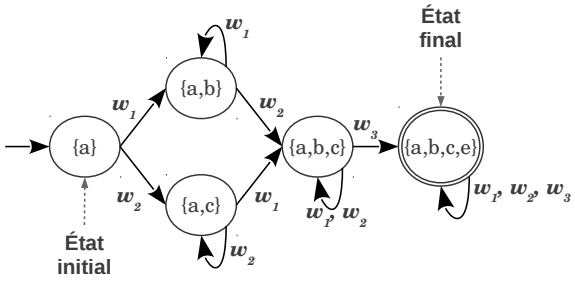


FIGURE 2.8: Automate de l'exemple 2.1

et al. [BMGI07, MFGI05]. Berardi et al. [BCG+05] présentent une approche de composition de services où l'objectif de la composition ainsi que les services sont représentés par des automates. Pour accomplir l'objectif de la composition, l'objectif et les services sont transformés en un ensemble de formules logiques en DPDL (Deterministic Propositional Dynamic Logic) [BAHP82]. Dans le cas où ces formules logiques sont satisfiables, un automate du service composite est produit. Dans un récent travail, Berardi et al. [BCGP08] remplacent les automates à états finis par des systèmes de transitions et utilisent la simulation afin de calculer toutes les compositions possibles. Melliti et al. [MPM08] proposent une autre approche où les services et les objectifs sont décrits par des automates étendus pour prendre en compte la sémantique des données échangées. La composition (et d'éventuels adaptateurs résolvant des incompatibilité comportementales) sont générés à l'aide d'un produit d'automates. Pistore et al. [PTB05] présentent une plate-forme formelle pour la composition de services Web. Le problème de la composition est modélisé avec un système états/transitions. La résolution de ce problème est faite en utilisant un *model cheker* [GT00, CPRT03]. La plate-forme développée dans ce travail est capable de gérer des objectifs complexes [DLPT02] (contraintes sur l'objectif).

Contextes. D'autres approches ont permis d'augmenter l'expressivité du problème de composition avec des contextes ou des critères de qualité de service. C'est le cas de Ben Mokhtar et al. [MFGI05, BMGI06, BMGI07] qui proposent des approches de composition orientées comportement en environnement pervasif. Les services et l'objectif de la composition sont représentés par des automates. Chaque fonctionnalité de l'objectif est remplacée par un service effectuant une fonctionnalité équivalente ou similaire en utilisant une ontologie de capacités. De plus, dans [MFGI05], les auteurs prennent aussi en compte des contextes en annotant les services et l'objectif pour obtenir une composition adéquate par rapport au contexte. Les contextes utilisés sont centrés utilisateur (***e.g.,*** profil) ou centrés services (***e.g.,*** localisation).

Qualité de service. Ben Mokhtar et al. [BMGI07] étendent [MFGI05, BMGI06] en augmentant la description des services avec des propriétés de QoS. La résolution du problème est faite en utilisant deux algorithmes de sélection. Le premier identifie

les services susceptibles de remplacer une fonctionnalité abstraite du besoin et le second permet de sélectionner le service ayant le meilleur score de QoS.

TABLEAU 2.3: Approches de composition fondées sur les automates

Approches	***Expressivité***				***Efficacité***	
	Conversation		*QoS*	*Contexte*	*Solutions*	*Automatisation*
	S	O				
Berardi at al. [BCG+05]	✓	✓			1	$\checkmark_{c+i}$
Pistore et al. [PTB05]		✓			1	$\checkmark_{c+i}$
Ben Mokhtar et al. [MFGI05]	✓	✓		✓	1	$\checkmark_{ds+i}$
Ben Mokhtar et al. [BMGI06]	✓	✓			1	$\checkmark_{ds+i}$
Ben Mokhtar et al. [BMGI07]	✓	✓	✓		$\lceil 1 \rceil$	$\checkmark_{ds+i}$
Berardi et al. [BCGP08]	✓	✓			$\forall$	$\checkmark_{c}$
Melliti et al. [MPM08]	✓	✓			1	$\checkmark_{c}$

Analyse (tableau 2.3). Les automates offrent naturellement un niveau d'expressivité permettant la prise en compte du comportement des services et de l'objectif de la composition. Cependant, l'utilisation du produit pour effectuer la composition présente un frein à ces approches en raison de leur incapacité à passer à l'échelle.

2.3.4 Raisonnement logique

La dernière catégorie d'approches de composition est celle basée sur le raisonnement logique. Le problème de composition est traduit en un ensemble de formules logiques qui vont être résolues en utilisant des méthodes et des outils de raisonnement.

Avant de présenter les mécanismes de résolution de problèmes en logique du premier ordre, nous introduisons tout d'abord les notions suivantes :

- un ***atome*** est un prédicat n-aire de la forme $A(t_1, t_2, \ldots, t_n)$, où $t_1, t_2, \ldots, t_n$ sont des termes,
- un ***littéral*** est un atome ou la négation d'un atome,
- une ***formule logique*** est définie comme suit :
  - un atome est une formule,
  - si A est une formule alors $\neg A$ est une formule,
  - si A et B sont des formules alors $A \wedge B$, $A \vee B$ et $A \rightarrow B$ sont des formules.
- une ***clause*** est une formule représentée sous la forme d'une disjonction de littéraux (***e.g.,*** $\neg A \vee B$).

Afin de résoudre un ensemble de formules logiques, diverses méthodes existent.

Résolution. La résolution est un principe qui a été proposé par Alan Robinson [Rob65] pour la démonstration automatique de théorèmes en logique propositionnelle. Ce principe consiste à déduire de nouvelles propositions à partir d'une théorie modélisée en logique propositionnelle. Le principe de résolution est appliqué aux clauses sous la Forme Normale Conjonctive (CNF) d'un ensemble de formules booléennes modélisant un théorème donné.

Dans le cas de clauses sans variables, le principe de résolution est le suivant : pour chaque paire de clauses C_1 et C_2, s'il existe un littéral L_1 (resp. un littéral L_2) dans la clause C_1 (resp. la clause C_2) tel que L_1 et $L2$ sont complémentaires (*i.e.* $L_1 = A,\ L_2 = \neg A$) alors supprimer les deux littéraux et construire par la suite la disjonction des littéraux restants. Ceci est fait en appliquant la règle d'inférence suivante.

$$\frac{C_1 : (A \vee B), C_2 : (\neg A \vee C)}{C_{1.2} : (B \vee C)}$$

Réfutation. La réfutation [Lak76] (ou raisonnement par l'absurde) est un principe de logique qui consiste à prouver la fausseté ou l'insatisfiabilité d'une proposition ou d'une formule. Dans le cas de clauses sans variables, le principe de réfutation consiste à intégrer la négation de la proposition à satisfaire dans l'ensemble de théorèmes et à prouver par la suite l'inconsistance de l'ensemble. Ceci correspond à la règle suivante :

$$\frac{S \cup \{\neg p\} \vdash \bot}{S \vdash p}$$

Dans ce qui précède, p est la proposition que l'on souhaite démontrer et S est un ensemble de théorèmes. On considère la négation de p en plus de S, on applique le principe de raisonnement, et si cela mène à une contradiction logique ($\bot$) alors on peut conclure qu'à partir des propositions de S, on peut déduire p.

Dans le contexte de la composition de services, la programmation logique est une technique largement utilisée. La modélisation du problème est faite en trois étapes. La première étape consiste à transformer chaque service en une implication $(A \Rightarrow B)$, où la *prémisse* (A) est la conjonction de propositions correspondant aux attributs d'entrée du service et la *conclusion* (B) est la conjonction des propositions correspondant aux attributs en sortie.

Exemple 2.6. *Le service w_3 a comme entrées, les propositions b et c et comme sortie e. La formule logique correspondant à ce service est $b \wedge c \Rightarrow e$*

La deuxième étape consiste à transformer les formules produites lors de l'étape précédente en un ensemble de clauses disjonctives.

Exemple 2.7. *La formule correspondant à w_3 est $\neg b \vee \neg c \vee e$.*

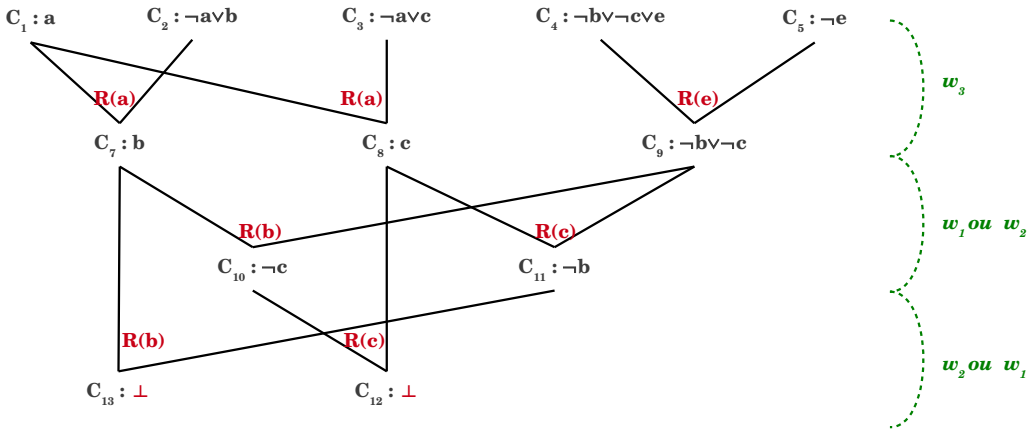


FIGURE 2.9: Arbre de résolution de l'exemple 2.1

Finalement, on cherche le but. Une première méthode consiste à appliquer les règles de résolution et à inférer toutes les propositions jusqu'à saturation (aucune nouvelle proposition ne peut être inférée). On vérifie alors l'existence ou non du but dans cet ensemble. Une autre méthode largement utilisée est celle de la réfutation, en incluant la négation du but dans l'ensemble de clauses à vérifier.

Exemple 2.8. *Nous illustrons notre exemple 2.1. Le problème de composition est traduit en un ensemble de clauses comme suit.*

Les services sont traduit en :

- $w_1 : \neg a \vee b$ (C_2)
- $w_2 : \neg a \vee c$ (C_3)
- $w_3 : \neg b \vee \neg c \vee e$ (C_4)

La donnée du problème est $(C_1) : a$
La négation du but est $(C_5) : \neg e$
Le principe de la résolution par réfutation est appliqué comme le montre la figure 2.9.
Nous dénotons par $R(x)$ *une résolution sur* x.
Une solution à ce problème est w_1 ; w_2 ; w_3 *ou* w_2 ; w_1 ; w_3.

De nombreuses approches de composition de services sont basées sur un raisonnement logique.

Composition simple. Dans SWORD [PF02], un service est décrit par ses attributs en entrée et ses attributs en sortie. L'approche traduit les services en formules logiques et utilise un moteur de règles (Jess) pour les résoudre.

Conversations. Rao et al. [RKM04] proposent une approche de composition de services supportant les conversations de services. Ces derniers sont transformés en un ensemble d'axiomes et de théorèmes. Pour résoudre cet ensemble, les auteurs

utilisent un moteur d'inférence dédié (LL theorem prover). La composition est ainsi directement traduite à partir des preuves. Une autre approche a été proposée par McIlraith et Son [MS02]. Les services sont décrits par une conversation. Ils sont aussi traduits en formules logiques. Pour les résoudre, les auteurs ont adapté et étendu le langage Golog [LRE+94].

Conversations et préférences. L'approche [MS02] a été étendue dans [SPM09] en prenant en compte des préférences de l'utilisateur spécifiées dans l'objectif de la composition. Ceci est fait, en traduisant les préférences en formules logiques et en les injectant dans le système Golog. Ce dernier permet de générer la meilleure solution, c'est-à-dire celle qui satisfait au mieux les préférences.

Contextes. Boyaci et al. [BBS10] proposent une plate-forme de composition de services sensible aux contextes. Les auteurs spécifient un ensemble de formules logiques exprimées en fonction des contextes (local, événement ou temporel). Les contextes sont aussi donnés sous la forme de règles logiques spécifiant ainsi les actions à exécuter lors d'un changement de contexte. La plate-forme permet de détecter le contexte des services et de le traduire automatiquement en formules logiques qui sont utilisées par la suite par SECE (Sense Everything, Control Everything) pour calculer une composition réalisant le besoin de l'utilisateur. Beltran et al. [BAS12] proposent une approche de composition de service combinant SECE et GloServ. Ils étendent SECE avec l'utilisation d'une ontologie de services permettant la prise en compte de plusieurs types de domaines de services découverts par GloServ [AS04], qui est un module de découverte de services.

TABLEAU 2.4: Approches de composition fondées sur le raisonnement logique

Approches	***Expressivité***				***Efficacité***	
	Conversation		*Préférence*	*Contexte*	*Solutions*	*Automatisation*
	S	O				
SWORD [PF02]					1	$\checkmark_{c+i}$
McIlraith et Son [MS02]	✓				1	$\checkmark_{c+i}$
Rao et al. [RKM04]	✓				1	$\checkmark_{c+i}$
McIlraith et Son [SPM09]	✓		✓		[1]	$\checkmark_{c+i}$
Boyaci et al. [BBS10]				✓	1	$\checkmark_{c+i}$
Beltran et al. [BAS12]				✓	1	$\checkmark_{c+i}$

Analyse (tableau 2.4). La composition de services basée sur le raisonnement logique supporte les conversations, les contextes et les préférences. De plus il existe

des moteurs d'inférence et de règles puissants permettant la résolution des formules logiques de façon efficace.

Des approches de composition ont exploité la puissance des moteurs d'inférence en combinant les techniques de planification et le raisonnement logique, par exemple SatPlan [KS92]. L'idée derrière cette approche est de transformer le problème de planification en un problème de satisfiabilité de formules booléennes (*i.e.* un ensemble de clauses CNF), pour lequel il existe plusieurs moteur Sat performants comme satz ou walksat.

2.4 Conclusion et discussion

Dans cette thèse nous abordons le problème de la réalisation automatique d'objectifs utilisateurs de haut-niveau par composition de services dans le contexte de la gestion des informations personnelles. Pour ce faire, nous avons choisi la technique de la composition de services.

Dans ce chapitre, nous avons tout d'abord présenté le processus de composition comme il a été étudié dans la littérature. Nous avons listé par la suite un ensemble de critères permettant d'évaluer et de comparer les approches de composition. Enfin, nous avons étudié les approches de composition automatiques en les classifiant en quatre catégories : les approches de composition basées sur la technique de planification, celles basées sur les réseaux de Petri, celles basées sur l'utilisation des automates et celles basées sur l'utilisation du raisonnement logique.

Une comparaison des différentes approches de composition présentées dans cet état de l'art est donnée dans le tableau 2.5.

Les conversations sont supportées par différentes techniques. Le raisonnement logique et la planification nécessitent un effort d'encodage important des conversations afin de pouvoir les intégrer dans le processus de composition. Elles sont en revanche utilisées plus simplement avec les réseaux de Petri et avec les automates. Ces derniers bénéficient par ailleurs de transformations de modèles depuis des langages standards tels que ABPEL ou BPMN permettant la modélisation des conversations par les concepteurs de services.
La qualité de service a été prise en compte par des approches à base de planification, de réseaux de Petri et d'automates. Les préférences ont été étudiées avec la planification et le raisonnement logique. Les contextes ont été pris en compte dans des approches basées sur la planification, les automates ou le raisonnement logique. Ces approches exploitent principalement les contextes lors de la sélection

des services.

Cette étude montre que la technique de planification est assez flexible et extensible pour supporter l'utilisation de nouvelles propriétés. Pour résoudre notre problématique nous avons opté pour la réutilisation de *GraphPlan*. Ce choix a été motivé par deux caractéristiques importantes de *GraphPlan* à savoir le support du parallélisme dans les solutions et une capacité importante de passage à l'échelle.

TABLEAU 2.5: Comparaison des approches de composition

		Expressivité					***Efficacité***	
	Approches	*Conversation*		*QoS*	*Préférence*	*Contexte*	*Solution*	*Automatisation*
		Service	Objectif					
Planification	Wu et al. [WPS+03]		✓				1	$✓_{c+i}$
	Vukovic et Robinson [VR04]					✓	1	$✓_{c+i}$
	Kuter et al. [KSP+05]		✓				1	$✓_{c+i}$
	Klusch et al. [KGS05]						1	$✓_{c+i}$
	Lin et al. [LKS08]		✓		✓		$\lceil 1 \rceil$	$✓_{c+i}$
	Sohrabi et McIlraith [SM10]		✓		✓		$\lceil 1 \rceil$	$✓_{c+i}$
	Poizat et Yan [PY10]	✓	✓				$1\vert\vert$	$✓_{c+i}$
	Yuhong et al. [YCY12]		✓	✓			$\lceil 1 \rceil\vert\vert$	$✓_{c}$
	Chen et al. [CPY14]	✓	✓	✓			$\lceil 1 \rceil\vert\vert$	$✓_{c}$
R. de Petri	Narayanan et McIlraith [NM02]	✓					1	$✓_{c+i}$
	Hamadi et Benatallah [HB03]	✓					1	$✓_{c}$
	Brogi et al. [BC07]	✓					$\lceil 1 \rceil$	$✓_{c}$
	Ren et al. [RLC+08]		✓				1	$✓_{c}$
	Tan et al. [TFZT10]						1	$✓_{c+i}$
	Cardinale et al. [CEHMR11]			✓			$\lceil 1 \rceil$	$✓_{ds+i}$
	Bao et al. [BGX+13]		✓				1	$✓_{c}$
Automates	Berardi at al. [BCG+05]	✓	✓				1	$✓_{c+i}$
	Pistore et al. [PTB05]		✓				1	$✓_{c+i}$
	Ben Mokhtar et al.[MFGI05]	✓	✓			✓	1	$✓_{ds+i}$
	Ben Mokhtar et al. [BMGI06]	✓	✓				1	$✓_{ds+i}$
	Ben Mokhtar et al.[BMGI07]	✓	✓	✓			$\lceil 1 \rceil$	$✓_{ds+i}$
	Berardi et al. [BCGP08]	✓	✓				$\forall$	$✓_{c}$
	Melliti et al. [MPM08]	✓	✓				1	$✓_{c}$
R. Logique	SWORD [PF02]						1	✓
	McIlraith et Son [MS02]	✓					1	$✓_{c+i}$
	Rao et al. [RKM04]	✓					1	$✓_{c+i}$
	McIlraith et Son [SPM09]	✓			✓		$\lceil 1 \rceil$	$✓_{c+i}$
	Boyaci et al. [BBS10]					✓	1	$✓_{c+i}$
	Beltran et al. [BAS12]					✓	1	$✓_{c+i}$

Deuxième partie

Contributions de la thèse

Chapitre 3

Préliminaires

Sommaire

Suite à l'étude des travaux sur la gestion d'informations personnelles présentée dans le chapitre 1, l'exploitation de la sémantique est un enjeu majeur pour la représentation des informations personnelles. L'utilisation des ontologies et des technologies du Web sémantique est la solution naturelle pour la modélisation et l'exploitation de la sémantique de ces informations.

Les travaux que nous présentons dans cette thèse s'appuient sur différentes notions que nous tâchons d'introduire dans ce chapitre. Nous commencerons par introduire la notion d'ontologie en section 3.1. Nous présentons ensuite les langages du Web sémantique que nous avons utilisés : le langage OWL en section 3.2.1 pour représenter les différentes ontologies, le langage RDF en section 3.2.2 pour la représentation des données et le langage SPARQL en section 3.2.3 pour exprimer des requêtes. Enfin, nous terminons en section 3.3 par une présentation succincte des mesures de similarité hiérarchiques.

3.1 Ontologies

Les ontologies jouent un rôle central dans de nombreux domaines de l'informatique tels que l'ingénierie des connaissances, la représentation des connaissances, la conception de base de données et la recherche d'informations. Une ontologie permet de spécifier de manière formelle des vocabulaires pour la description sémantique de contenus. Ces vocabulaires sont souvent représentés par une hiérarchie de concepts auxquels sont associées des propriétés, elles-mêmes pouvant être organisées en hiérarchie.

Thomas Gruber a proposé une définition formelle spécifiant qu'une ontologie est « *une spécification explicite et formelle d'une conceptualisation commune* » [Gru95]. L'ontologie est *explicite* dans le sens où les concepts et les contraintes doivent y être explicitement déclarés. Elle est *formelle* dans le sens où le contenu de l'ontologie doit être compréhensible et interprétable par une machine. Enfin, l'ontologie est *commune* car elle doit être construite de façon consensuelle afin de favoriser l'échange de connaissances et de données.

Une ontologie est en général composée de trois éléments principaux : (*i*) les concepts (nommé aussi les classes), (*ii*) les propriétés qui sont de deux types : les attributs ayant comme domaine un concept et comme co-domaine un type de données primitif, et les relations dont le domaine et le co-domaine sont des concepts, et (*iii*) les axiomes qui représentent l'ensemble de contraintes sémantiques que l'on peut déclarer sur l'ontologie. Il s'agit, notamment des axiomes de subsomption/partOf entre concepts et entre propriétés, des axiomes de disjonction entre concepts et des contraintes de cardinalité pour les propriétés.

Définition 3.1 (Ontologie)**.** *Une ontologie est définie par un tuple* $\mathcal{O} = \langle \mathcal{C}, \mathcal{P}, \mathcal{A} \rangle$ *où :*

- $\mathcal{C}$ *est l'ensemble des concepts de l'ontologie* $\mathcal{O}$*,*
- $\mathcal{P}$ *est l'ensemble des propriétés de l'ontologie associées aux concepts,*
- $\mathcal{A}$ *est l'ensemble des axiomes de l'ontologie.*

La figure 3.1, présente un extrait de l'ontologie FOAF (Friend-Of-A-Friend) [FOA00]. Elle inclut des concepts comme *Person*, *e-CommerceAccount* et *Group*. Chaque concept peut être décrit par des propriétés telles que *firstName*, *homepage* ou *known*. Dans cette ontologie, on trouve également des axiomes de subsomption comme par exemple celui entre les deux concepts *SocialAccount* et *OnlineAccount*.

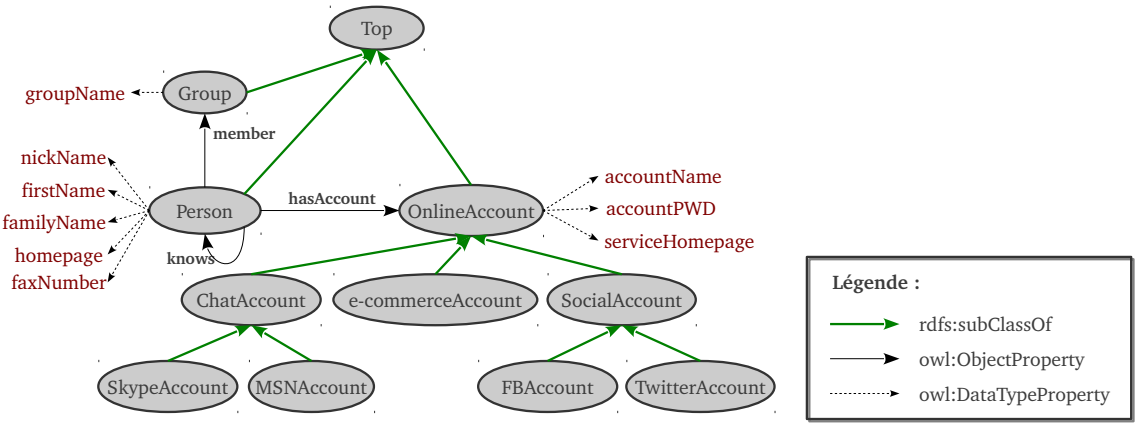


FIGURE 3.1: Extrait de l'ontologie FOAF (Friend-of-a-Friend)

3.2 Langages du Web sémantique

Plusieurs langages ont été proposés pour représenter des connaissances dans le domaine du Web sémantique. Parmi ces langages, nous pouvons citer OWL pour la représentation des ontologies, RDF pour la représentation des méta-données et SPARQL pour l'interrogation des ontologies et des données RDF.

3.2.1 Langage de représentation d'ontologie – OWL

OWL (*Web Ontology Langage*) [MVH+04] est le langage standard du W3C (Consortium World Wide Web) [1] pour représenter les ontologies. Il permet ainsi de représenter des connaissances construites sur le modèle RDF avec un pouvoir expressif important et par conséquent une complexité plus grande pour l'inférence. Il prend ses origines du langage DAML+OIL [Hor02]. OWL présente plus d'expressivité pour représenter des connaissances que le langage RDFS [RDF04b] (Resources Description Framework Schema). En effet, OWL permet, notamment, de déclarer des classes, des propriétés, des relations de subsomption entre les classes et les propriétés et des contraintes de cardinalité. De plus, OWL est fondé sur la logique de description [Baa03] ce qui lui permet de tirer profit des résultats théoriques de ce domaine.

Le langage OWL dans sa première version, offre trois sous langages avec des niveaux d'expressivité croissante [MVH+04] : OWL-Lite, OWL-DL et OWL-Full (voir figure 3.2).

1. www.w3c.org

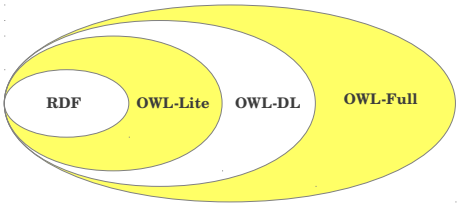


FIGURE 3.2: Hiérarchie des sous langages de OWL

— **OWL-Lite** est un sous langage ayant une faible expressivité. Il permet la déclaration de hiérarchies de classes et de propriétés ainsi que la déclaration de contraintes de typage sur les propriétés.
— **OWL-DL** offre un pouvoir d'expression supérieur à celui de OWL-Lite et des capacités de raisonnement décidable et complet.
— **OWL-Full** est le sous langage ayant le plus fort pouvoir d'expression mais ne garantit pas la décidabilité du raisonnement.

Une ontologie représentée en OWL est stockée dans un fichier texte suivant la syntaxe XML. Le contenu d'une ontologie OWL est décrit entre les balises :

```
<owl:Ontology rdf:about="Ontology_Name"> ... </owl:Ontology>
```

OWL offre des constructeurs pour déclarer notamment :

— les classes représentant les concepts de l'ontologie et définies comme un ensemble d'individus. Une classe Person peut être déclarée en OWL comme suit :

```
<owl:Class rdf:ID="#Person"/>
```

— les propriétés représentent les caractéristiques des individus. Dans le langage OWL, on distingue deux types de propriétés :
  — les *owl:DataTypeProperty* que nous avons précédemment appelé attribut, dont le domaine est une classe et le co-domaine (*range*) est un type de données primitif. À titre d'exemple, la propriété firstName peut être déclarée en OWL comme suit :

```
<owl:DataTypeProperty rdf:ID="firstName">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="&xsd;xsd:string"/>
</owl:DataTypeProperty>
```

  — les *owl:ObjectProperty* que nous avons précédemment appelé relation, dont le domaine et le co-domaine sont des classes. À titre d'exemple, la propriété hasAccount peut être déclarée comme suit :

```
  <owl:ObjectProperty rdf:ID="hasAccount">
        <rdfs:domain rdf:resource="#Person"/>
```

```
        <rdfs:range rdf:resource="#OnlineAccount"/>
</owl:ObjectProperty>
```

— une hiérarchie de classes grâce aux relations de subsomption que l'on peut déclarer entre deux classes en utilisant le constructeur *rdfs:subClassOf*
— une hiérarchie de propriétés grâce aux relations de subsomption que l'on peut déclarer entre deux propriétés en utilisant le constructeur *rdfs:subPropertyOf*
— des axiomes de disjonction que l'on peut déclarer entre deux classes en utilisant le constructeur *owl:disjointWith*.

La sémantique en Logique du Premier ordre (LP) des déclarations de subsomptions entre classes, de subsomption entre propriétés, typage des domaines et des co-domaines des propriétés et des disjonctions entre classes est donnée en tableau 3.1.

TABLEAU 3.1: Sémantique des principaux constructeurs OWL

Constructeurs	***N-Triple***	***Sémantique LP***
Subsomption entre classes	C1 rdfs:subClassOf C2	$\forall X(C1(X) \Rightarrow C2(X))$
Subsomption entre propriétés	p1 rdfs:subPropertyOf p2	$\forall X\ \forall Y(p1(X,Y) \Rightarrow p2(X,Y))$
Typage du domaine d'une propriété	p rdfs:domain C	$\forall X\ \forall Y(p(X,Y) \Rightarrow C(X))$
Typage du co-domaine d'une propriété	p rdfs:range C	$\forall X\ \forall Y(p(X,Y) \Rightarrow C(Y))$
Disjonction entre classes	C1 owl:disjointWith C2	$\forall X(C1(X) \wedge C2(X) \Rightarrow \bot)$

Le langage OWL2 [OWL12] est la dernière version de OWL, recommandée par le W3C en 2012. Cette nouvelle version a notamment permis une utilisation plus aisée des relations N-aires, la possibilité d'associer des cardinalités aux classes, la description de la réflexivité, l'irréflexibilité et l'asymétrie des propriétés ainsi que l'ajout d'annotations au contenu de l'ontologie.

Dans cette thèse nous nous restreignons à l'utilisation de la première version de OWL.

3.2.2 Langage de représentation de données – RDF

RDF(Resource description Framework) [RDF04a] est un langage de description sémantique de données (méta-données). Ces descriptions sont représentées par un ensemble de déclarations sous la forme de triplets <sujet, prédicat, objet> : où

— *sujet* représente la ressource décrite,
— *prédicat* représente la propriété associée à la ressource décrite,
— *objet* représente la valeur de la propriété.

Cette structure rappelle la structure d'une phrase en français « sujet, verbe et complément ». Prenons l'exemple de la déclaration suivante « *La chanson Happy est chantée par Pharrell Williams* ». Elle peut être représentée en RDF comme suit :

— sujet : La chanson Happy
— predicat : est chantée par

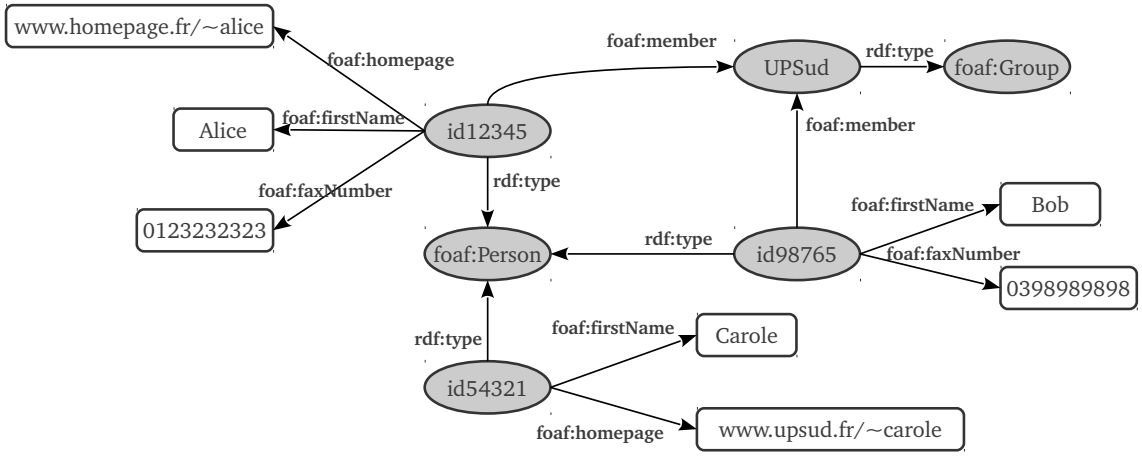


FIGURE 3.3: Exemple de graphe RDF

— object : Pharrell Williams

Le langage RDF permet ainsi une description des ressources à la fois plus aisée pour les humains et plus compréhensible pour les machines. Il existe différentes syntaxes équivalentes pour représenter des déclarations RDF : N3, N-Triples et Turtle.

Dans la suite du manuscrit, nous utiliserons la notation N-triple où chaque déclaration est exprimée sous la forme de *triplets*.

<La chanson Happy, est chantée par, Pharrell Williams>

Les déclarations RDF peuvent être également représentées par un graphe étiqueté où les nœuds représentent soit des classes (ellipses en figure 3.3) ou des valeurs (un rectangle arrondi (*bubtangle*) en figure 3.3) et les arcs étiquetés représentent les différentes propriétés.

En figure 3.3, nous donnons un exemple de graphe RDF décrivant des données personnelles conformes à l'ontologie FOAF de la figure 3.1. La personne référencée par id12345 a pour nom Alice et a comme numéro de fax 0123232323.

Il est parfois utile, en plus de la déclaration de méta-données, de pouvoir déclarer des propriétés sur les méta-données elles-mêmes. Ce type de déclaration est connu sous le terme de *réification*. En philosophie la réification est « *le processus par lequel on transforme quelque chose de mouvant, de dynamique en être fixe, statique* » [Lar]. En informatique, la réification consiste à transformer un concept en un objet informatique. Par exemple, réifier une classe en une instance d'une méta-classe. En RDF, la réification permet de considérer un triplet comme une ressource que l'on peut décrire sémantiquement, *i.e.* un triplet devient un nœud du graphe RDF. La réification d'un triplet permet ainsi de l'utiliser comme sujet ou

objet d'une propriété. En RDF, ceci reviendrait à déclarer un *rdf:Statement* comme suit :

```
<t rdf:type rdf:Statement>
<t rdf:subject sujet>
<t rdf:predicate predicat>
<t rdf:object objet>
<t ex:p valeur>
```

Ce mécanisme de réification sera utilisé dans notre travail pour la contextualisation des données personnelles.

3.2.3 Langage d'interrogation – SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) [SPA06] est le langage standard du W3C pour l'interrogation de données RDF et d'ontologies OWL. SPARQL est un langage de requêtes créé par le groupe de travail DAWG (RDF Data Access Working Group) du W3C. Il s'appuie sur la représentation RDF sous la forme de triplets et il est particulièrement adapté à la structure de graphes RDF. La syntaxe utilisée par SPARQL est inspirée de celle de SQL. SPARQL permet d'exprimer deux types de requêtes.

- les *interrogatives* qui peuvent être des requêtes de type *SELECT*, de type *ASK* ou de type *DESCRIBE*. Une requête SELECT permet d'extraire un sous graphe du graphe RDF correspondant à un ensemble de ressources vérifiant les conditions spécifiées par la requête. Une requête ASK permet de retourner vrai ou faux selon que le graphe interrogé vérifie ou non les conditions.
- les *constructives* sont représentées par des requêtes *CONSTRUCT*, permettant d'ajouter un nouveau graphe complétant le graphe interrogé.

Exemple 3.1. *Sur le graphe RDF présenté en figure 3.3, la requête SELECT suivante permet d'obtenir tous les prénoms et numéros de fax des personnes représentées dans ce graphe.*

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?faxNumber WHERE {
  ?person a foaf:Person.
  ?person foaf:firstname ?name.
  ?person foaf:faxNumber ?faxNumber.
}
```

On note que PREFIX *représente la déclaration du namespace de l'ontologie utilisée. L'évaluation de cette requête sur le graphe RDF de la figure 3.3 donnerait le résultat en tableau 3.2.*

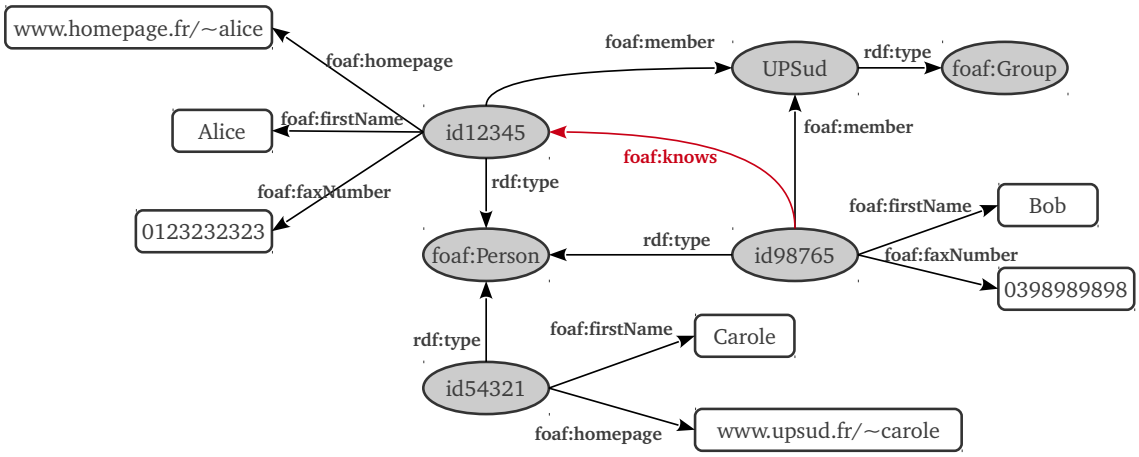


FIGURE 3.4: Résultat de la requête CONSTRUCT

En utilisant une requête CONSTRUCT, on pourrait ajouter une relation foaf:knows entre deux personnes appartenant à un même groupe.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
CONSTRUCT{ ?p1 foaf:knows ?p2 }
WHERE {
  ?p1 foaf:member ?g.
  ?p2 foaf:member ?g.
}
```

Le résultat de cette requête est donné en figure 3.4. Cette requête a permis d'ajouter une relation foaf:knows entre Bob et Alice.

3.3 Mesures de similarité hiérarchiques

Dans les travaux présentés dans cette thèse, nous avons souvent parlé de l'utilisation de mesures de similarité hiérarchique. Il s'agit notamment de la phase de comparaison de contextes à la fois pour l'évaluation des requêtes contextuelles mais aussi lors de la phase de composition de e-services.

Le calcul de similarité hiérarchique est un sujet encore très ouvert, notamment dans des applications de type liage de données, alignement d'ontologies ou encore

TABLEAU 3.2: Résultat de la requête SELECT de l'exemple 3.1

name	faxNumber
Alice	0123232323
Bob	0398989898

les moteurs de recherche. L'une des solutions concerne l'utilisation des mesures de similarité s'appuyant sur des taxonomies.

Dans la littérature plusieurs mesures de similarité hiérarchiques ont été proposées [RMBB89, WP94, Res95, Lin98]. Nous détaillons la mesure de Wu et Palmer car elle est efficace pour le calcul de similarité et simple à implémenter.

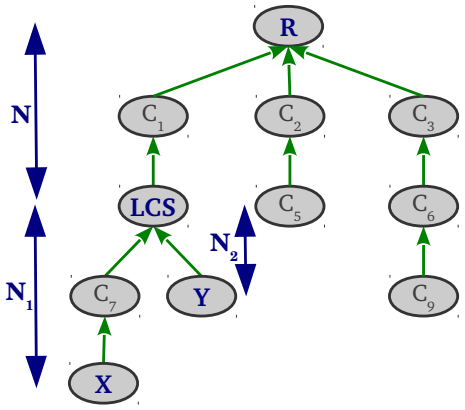


FIGURE 3.5: Exemple d'un extrait d'une ontologie $\mathcal{O}$

La mesure de Wu et Palmer est fondée sur le principe suivant : soit une ontologie $\mathcal{O}$ (voir figure 3.5) constituée d'un ensemble de nœuds tous atteignables à partir d'un nœud R. Soient X et Y deux concepts de l'ontologie dont nous souhaitons calculer la similarité hiérarchique. Le principe du calcul est fondé sur l'utilisation des distances N1 et N2 séparant les nœuds X et Y du nœud contenant leur concept subsumant commun (LCS) [2], et de la distance N qui sépare le LCS du nœud racine (R). Selon Wu et Palmer, la similarité hiérarchique entre ces deux concepts X et Y est calculée comme suit :

$$sim_{WP}(X,Y) = \frac{2 \times N}{N_1 + N_2 + 2 \times N}$$

Exemple 3.2. *En utilisant l'ontologie FOAF présenté dans la figure 3.1, la similarité entre les concepts foaf:FBAccount et foaf:SkypeAccount, où leur LCS est foaf:OnlineAccount, est la suivante :*

$$\begin{aligned} sim_{WP}(foaf{:}FBAccount, foaf{:}SkypeAccount) &= \frac{2\times1}{2+2+2\times1} \\ &= \frac{1}{3} \end{aligned}$$

2. LCS (Least Commun Subsumer) est le concept le plus spécifique subsumant commun

3.4 Conclusion

Dans ce chapitre nous avons présenté une définition formelle des ontologies. Nous avons également présenté les trois langages du Web sémantique que nous avons utilisé dans le cadre de nos travaux. Enfin nous avons terminé en présentant une mesure de similarité hiérarchique particulièrement utilisée pour comparer des contextes représentés dans des hiérarchies (ontologies).

Chapitre 4

Modélisation contextuelle d'informations personnelles

Sommaire

Le problème que nous adressons dans cette thèse concerne la construction d'espaces d'informations personnelles offrant une intégration sémantique et une exploitation sécurisée et la plus automatique possible des informations personnelles.

Dans ce chapitre, nous présentons notre première contribution concernant une approche de modélisation contextuelle d'informations personnelles.

Pour garantir une intégration sémantique des données personnelles, nous avons choisi d'utiliser les ontologies en guise de schéma pour les données. Ainsi ces

ontologies pourront être utilisées comme interface d'interrogation (*i.e.* les requêtes seront exprimées dans le vocabulaire de l'ontologie) et comme support de raisonnement. Nous avons également souhaité prendre en compte la vision que pourrait avoir l'utilisateur de la modélisation de son espace d'informations personnelles, nous conduisant ainsi à une modélisation multi-points de vue. De plus, comme mentionné dans l'introduction de cette thèse, certaines informations personnelles sont particulièrement sensibles aux contextes qu'ils soient sociaux, géographiques ou temporels. Pour répondre à ce besoin, nous avons proposé une modélisation où une donnée personnelle est associée à un contexte. Enfin, pour prendre en compte le caractère confidentiel des informations personnelles, nous avons proposé une catégorisation par thème facilitant ainsi leur exploitation dans les politiques de sécurité.

Dans la suite du chapitre, nous présentons en section 4.1 notre modèle multi-points de vue pour la représentation des informations personnelles. En section 4.2, nous introduisons la façon dont les informations personnelles sont contextualisées. Nous donnons ensuite en section 4.3, leur classification pour le respect des politiques de sécurité. En section 4.4, nous présentons le prototype CoPIM implémentant la conception d'un système d'informations personnelles suivant leur représentation contextuelle. Nous concluons ce chapitre en section 4.4.

4.1 Modélisation multi-points de vue

Lorsque l'on s'intéresse à la modélisation du monde réel en vue d'obtenir une représentation dans un système informatique, on est souvent confronté à plusieurs représentations possibles. Ces différentes représentations sont dues aux différents points de vues que l'on peut avoir de la situation modélisée. Il en est de même lorsque l'on s'intéresse à la modélisation des informations personnelles. Différentes personnes peuvent avoir des points de vue cognitifs différents sur leurs informations personnelles. Par exemple, une personne pourrait considérer que tous ses collègues sont aussi des amis et une autre que ses collègues ne peuvent pas être des amis. Cette différence de points de vue peut être également due aux divers types d'applications que l'on peut concevoir pour exploiter les informations personnelles. On peut par exemple utiliser les mêmes informations personnelles pour réaliser des procédures dans une application de santé ou dans une application gérant les titres de transport.

Afin de tenir compte de cet aspect multi-points de vue, nous proposons une modélisation permettant de gérer différents points de vue sur les mêmes informations personnelles. Ainsi, tout utilisateur souhaitant concevoir son espace d'informations personnelles (PIS) pourrait avoir la possibilité soit d'utiliser

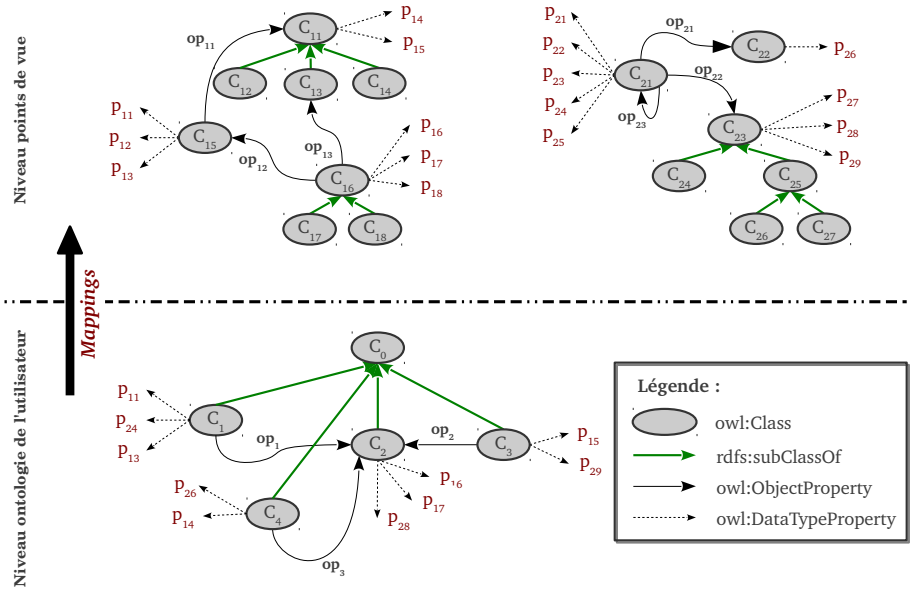


FIGURE 4.1: Schéma de la modélisation multi-points de vue des informations personnelles

une ontologie existante ou de concevoir sa propre ontologie en réutilisant des parties d'ontologies existantes lui permettant ainsi de combiner différents points de vue.

Comme montré en figure 4.1, la partie conceptuelle des informations personnelles s'articule en deux niveaux : niveau points de vue que nous détaillerons en section 4.1.1 et le niveau ontologie d'informations personnelles que nous présentons en section 4.1.2. Dans cette même section nous allons montrer comment les deux niveaux sont liés grâce à un ensemble de liens de mise en correspondance appelés aussi *mappings*.

4.1.1 Points de vue

Nous considérons qu'un point de vue est une ontologie OWL modélisant tout ou une partie d'un domaine d'application où les classes, les propriétés et leurs relations sont déclarées de façon explicite.

Définition 4.1 (Point-de-vue)**.** *Un point de vue est une ontologie définie par un tuple* $pv = \langle \mathcal{C}, \mathcal{P}, \mathcal{A} \rangle$ *où* $\mathcal{C}$ *est l'ensemble de classes du domaine modélisé par l'ontologie,* $\mathcal{P}$ *est l'ensemble de propriétés décrivant les classes et* $\mathcal{A}$ *est l'ensemble des axiomes exprimant les relations de subsomption, de disjonction ou d'équivalence sur les classes et les propriétés de l'ontologie.*

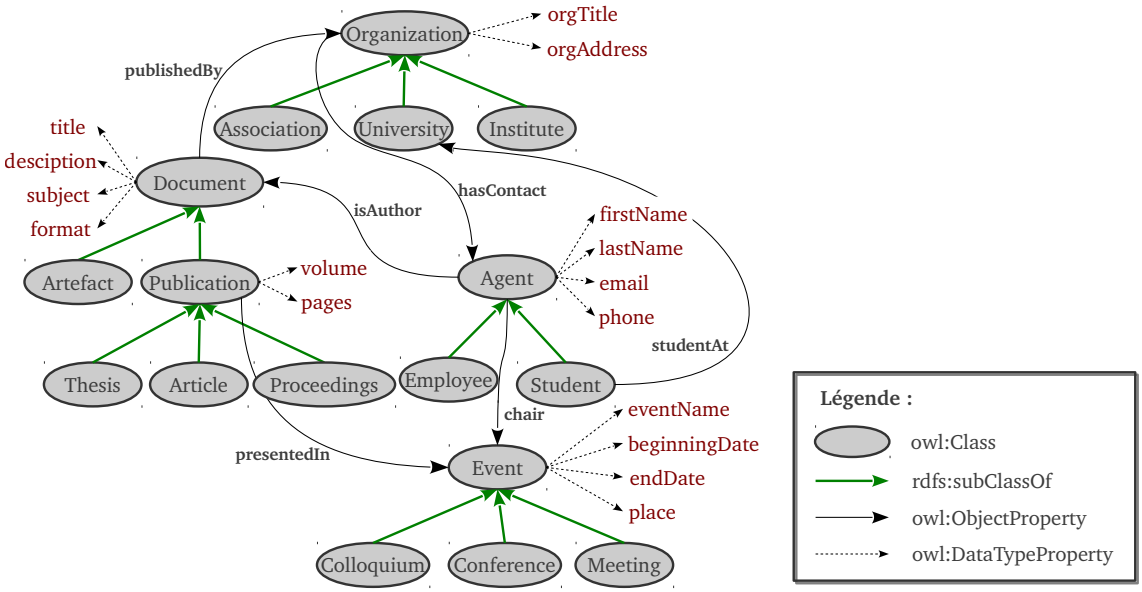


FIGURE 4.2: Extrait de l'ontologie SWC (Semantic Web Conference)

Nous considérons qu'un point de vue est représenté par la totalité ou une partie d'une ontologie de domaine d'application disponible sur le Web ou créée par un expert. En figure 4.2, nous montrons un exemple d'ontologie représentant un point de vue modélisant des connaissances sur la gestion d'événements scientifiques [SWC07]. Nous trouvons par exemple les classes *swc*:*agent*, *swc*:*organisation* représentant respectivement les personnes et les organisations impliquées de près ou de loin dans l'organisation des événements scientifiques. Les mêmes connaissances sont représentées dans l'ontologie FOAF, dont un extrait a été donné en figure 3.1, par les classes *foaf*:*Person* et *foaf*:*Group*. Chaque classe est décrite par un ensemble de propriétés, par exemple les propriétés *foaf*:*firstName* et *foaf*:*hasAccount* décrivent la classe *foaf*:*Person*. Les classes sont reliées entre elles avec des relations de subsomption par exemple dans l'ontologie SWC les classes *swc*:*Employee* et *swc*:*Student* sont subsumées par la classe *swc*:*Agent*. Ces deux ontologies se recoupent mais elles sont complémentaires. Nous n'avons par exemple pas de classe référant à la notion de document dans l'ontologie FOAF et nous n'avons pas de classe référant à la notion de compte en ligne dans l'ontologie SWC.

Comme mentionné précédemment, le premier niveau de notre modèle d'informations personnelles est le niveau points de vue. Ce niveau représente l'ensemble des ontologies pouvant être exploitées par l'utilisateur pour construire sa propre ontologie. Pour tirer profit de la richesse des connaissances représentées dans chaque ontologie, un ensemble de mappings est calculé entre les différentes ontologies. Ces mappings sont obtenus grâce aux outils d'alignement d'ontologies aujourd'hui dis-

ponibles (voir [ES13] pour plus de détails). Nous pouvons par exemple déclarer un mapping d'équivalence entre les classes *swc*:*Agent* et *foaf*:*Person* ou encore entre les propriétés *swc*:*lastName* et *foaf*:*nickName*. Nous donnons ci-dessous (définition 4.2) une définition plus formelle de ce qu'est un niveau points de vue.

Définition 4.2 (Niveau Points de Vue)**.** *Le niveau points de vue est défini par un couple* $\mathcal{PV} = \langle \mathcal{O}_{pv}, \mathcal{M}_{pv} \rangle$ *où*

- $\mathcal{O}_{pv} = \{pv_1, pv_2, \ldots, pv_n\}$ *est l'ensemble de points de vue considérés pour la modélisation des informations personnelles,*
- $\mathcal{M}_{pv}$ *est l'ensemble de mappings entre les classes et les propriétés des différents points de vue.*

Grâce aux différents mappings déclarés entre les points de vue, il est possible de raisonner plus globalement sur l'ensemble des ontologies représentant les points de vue.

4.1.2 Ontologie de types d'informations personnelles

Nous considérons que les informations personnelles peuvent être décrites de différentes façons selon le domaine d'application (***e.g.,*** santé, fiscalité, agenda).

Dans les systèmes de gestion d'informations personnelles (PIMS) existants, l'utilisateur est souvent contraint de manipuler une modélisation de ses informations personnelles imposée par le concepteur du PIMS. L'utilisation des ontologies permet d'une part d'unifier la structure des informations et d'autre part de laisser la liberté aux utilisateurs de concevoir leur propre système. Chaque utilisateur construit une Ontologie de Types d'Informations Personnelles (PITO). Pour ce faire, l'utilisateur crée son modèle d'informations personnelles en exploitant un ensemble d'ontologies existantes décrivant différents points de vue des informations personnelles.

Un Type d'Informations Personnelles (PIT) est une classe OWL qui décrit un ensemble d'instances possibles d'informations personnelles par un ensemble de propriétés. Il est créé par l'utilisateur et il est décrit par un label qui doit être unique dans l'ontologie PITO. Le PIT est décrit également par un ensemble de propriétés. Ces propriétés sont sélectionnées par l'utilisateur à partir des ontologies points de vue. Pour garder un lien entre le PIT et les points de vue, le PIT est également décrit par l'ensemble de classes à partir desquelles l'utilisateur a sélectionné les propriétés associées à la classe PIT ; ainsi qu'un ensemble de mappings reliant les propriétés du PIT à leurs propriétés sources dans les ontologies points de vue.

Définition 4.3 (Type d'informations personnelles (PIT))**.** *Il est défini par un tuple* $\langle c, label, \mathcal{P}, \mathcal{C}, M_p \rangle$ *où*

— *c est la classe représentant le type d'informations personnelles,*
— *label est le label de la classe c,*
— *$\mathcal{P}$ est un ensemble de propriétés (owl:DataTypeProperty ou owl:ObjectProperty) associées à la classe c,*
— *$\mathcal{C}$ est l'ensemble des classes des ontologies points de vue à partir desquelles les propriétés $\mathcal{P}$ ont été sélectionnées,*
— *M_p est l'ensemble de mappings d'équivalence entre les propriétés $\mathcal{P}$ du PIT et les propriétés des classes de $\mathcal{C}$. Ces mappings sont de la forme $(c.p_1 \equiv o_1 : c_1.p_2)$ où c est la classe OWL représentant le PIT, p_1 est une propriété associée à c, o_1 est une ontologie point de vue, c_1 est une classe de l'ontologie o_1 et p_2 est une propriété associée à c_1.*

Exemple 4.1. *En utilisant les deux ontologies présentées dans les figures 3.1 et 4.2, une classe PIT **Friend** pourrait être décrite comme suit :*

— *c = pito:Friend*
— *label = Friend*
— *$\mathcal{P}$ = {foaf:firstName, swc:email, swc:phone, foaf:skypeId }*
— *$\mathcal{C}$ = {foaf:Person, swc:Agent, foaf:OnlineAccount }*
— *M_p = {(pito:Friend.firstName ≡ foaf:Person.firstName), (pito:Friend.email ≡ swc:Agent.email), (pito:Friend.phone ≡ swc:Agent.phone), (pito:Friend.skypeId ≡ foaf:OnlineAccount.AccountName)}*

PITO est une ontologie décrivant des types d'informations personnelles créés par l'utilisateur. Elle est définie par un ensemble de types $\mathfrak{C}$ représentés par des classes, un ensemble de propriétés associées à ces dernières et un ensemble de mappings liant les classes et les propriétés PITO aux classes et aux propriétés des ontologies points de vue utilisées. L'ontologie PITO contient initialement une classe générique nommée *PITORoot* qui est la racine de l'ontologie. Ainsi tous les types créés par l'utilisateur seront subsumés par cette classe. Elle contient également une classe centrale appelée *Identity* à laquelle toutes les autres classes sont liées via des *owl:ObjectProperty* de la forme *hasNomClasse*.

Définition 4.4 (Ontologie de types d'informations personnelles (PITO))**.** *Il s'agit d'une ontologie OWL représentant les types d'informations personnelles, qui est définie par un tuple $\langle id, \mathfrak{C}, \mathfrak{R}, \mathfrak{M}_p \rangle$ où :*

— *id est le PIT (voir définition 4.3) représentant l'identité de l'utilisateur,*
— *$\mathfrak{C}$ est l'ensemble de classes PIT créées par l'utilisateur,*
— *$\mathfrak{R}$ est l'ensemble de relations entre les différents PIT,*

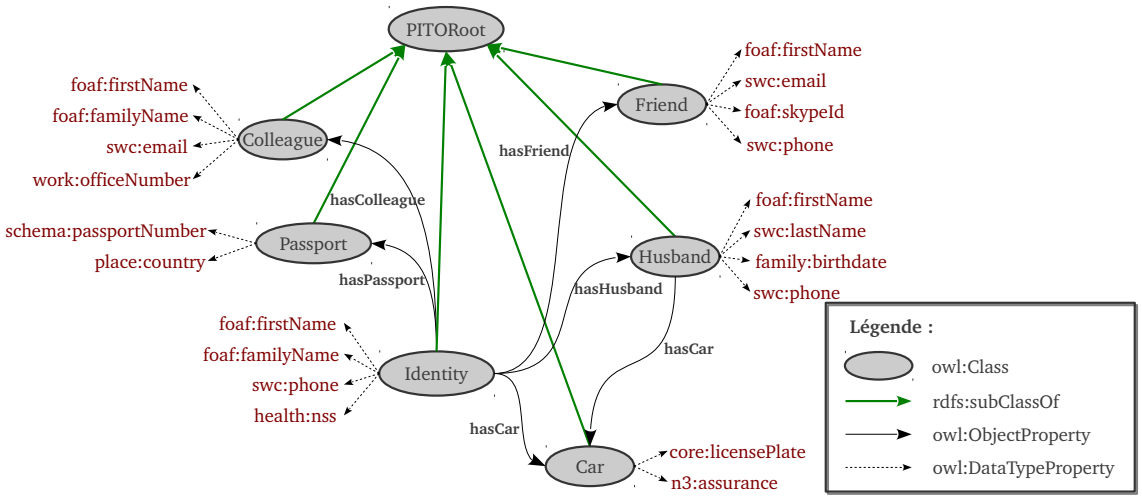


FIGURE 4.3: Extrait d'une ontologie PITO

— $\mathfrak{M}_p$ *est l'ensemble de tous les mappings des propriétés des PIT et les propriétés représentées dans les différents points de vue. Il est défini comme suit :*

$$\mathfrak{M}_p = \bigcup_{c \in \mathfrak{C}} c.M_p$$

La figure 4.3 présente l'ontologie PITO d'Alice où elle sépare la description de ces contacts entre amis et collègues et chacune des classe est décrite par un ensemble différent de propriétés. La classe *pito*:*Friend* est décrite par *foaf*:*firstName*, *swc*:*email*, *swc*:*skypeId* et *swc*:*phone* ; et la classe *pito*:*Colleague* est décrite par *foaf*:*firstName*, *foaf*:*familyName*, *swc*:*email* et *work*:*officeNumber*. La description OWL de l'ontologie PITO représentée en figure 4.3 est donnée en annexe B.

Exemple 4.2. *L'ontologie PITO d'Alice donnée en figure 4.3 est définie comme suit :*

— id = *Identity*
— $\mathfrak{C}$ = {*Car, Colleague, Husband, Friend, Passport*}
— $\mathfrak{R}$ = {*hasCar, hasColleague, hasHusband, hasFriend, hasPassport*}
— $\mathfrak{M}_p$ = {
(pito:Identity.firstName ≡ foaf:firstName),
(pito:Identity.familyName ≡ foaf:familyName),
(pito:Identity.phone ≡ swc:phone),
(pito:Identity.nss ≡ health:nss),
(pito:Passport.passportNumber ≡ schema:passportId),
(pito:Passport.country ≡ place:country),
(pito:Car.licensePlate ≡ core:licencePlate),

(pito:Car.assurance ≡ n3:assurance),
(pito:Colleague.firstName ≡ foaf:firstName),
(pito:Colleague.familyName ≡ foaf:familyName),
(pito:Colleague.email ≡ swc:email),
(pito:Colleague.officeNumber ≡ work:officeNumber),
(pito:Husband.firstName ≡ foaf:firstName),
(pito:Husband.lastName ≡ swc:lastName),
(pito:Husband.birthDate ≡ family:birthdate),
(pito:Husband.phone ≡ swc:phone),
(pito:Friend.firstName ≡ foaf:firstName),
(pito:Friend.email ≡ swc:email),
(pito:Friend.skypeId ≡ foaf:accountId),
(pito:Friend.phone ≡ swc:phone) }

4.1.3 Cohérence de l'ontologie PITO

La cohérence d'une ontologie consiste en la validation de son contenu. Plus précisément, elle réfère à la sémantique formelle des différents constructeurs de l'ontologie et à sa satisfiabilité dans le sens où elle doit être sémantiquement correcte et ne présente pas de contradictions logiques. Dans une ontologie OWL, différents axiomes sont associés à la déclaration de classes et de propriétés comme mentionné dans le chapitre 3 en section 3.2.1. Une interprétation de l'ontologie est dite satisfiable si et seulement si tous les axiomes de l'ontologie sont satisfiables *i.e.* leur application ne génère pas de contradictions.

Pour vérifier la cohérence de notre ontologie PITO, nous considérons :
— les axiomes de subsomption entre classes (*rdfs:subClassOf*)
— les axiomes de subsomption entre propriétés (*rdfs:subPropertyOf*)
— les axiomes de disjonction entre classes (*owl:disjointWith*)

Étant donné que l'ontologie PITO est construite en réutilisant d'autres ontologies (niveau points de vue), sa cohérence doit considérer le contenu et les axiomes de l'ontologie PITO mais aussi ceux des ontologies points de vue. Pour ce faire, les règles d'inférence présentées dans le tableau 3.1 sont appliquées afin d'inférer toutes les connaissances relatives à ces règles. À cela nous ajoutons deux règles supplémentaires permettant de saturer l'ensemble des disjonctions entre classes d'une même ontologie et de celles impliquant des classes d'ontologies différentes (ontologie PITO et ontologies points de vue).

Le calcul de disjonctions dans une même ontologie O est effectué en exploitant les déclarations de disjonctions entre classes.

La règle 4.1 ci-dessous permet d'inférer les disjonctions entre une classe c_i et les sous-classes de c_j si c_i et c_j sont déclarées disjointes dans l'ontologie.

$$\frac{c_i, c_j, c_k \in O, c_i \bot c_j, c_k \preceq c_j}{c_i \bot c_k} \tag{4.1}$$

La règle 4.2 permet d'inférer des disjonctions entre les sous-classes de c_i et les sous-classes de c_j si c_i et c_j sont déclarées disjointes dans l'ontologie.

$$\frac{c_i, c_j, c_l, c_k \in O, c_i \bot c_j, c_l \preceq c_i, c_k \preceq c_j}{c_l \bot c_k} \tag{4.2}$$

Le calcul de disjonctions entre les classes des différentes ontologies exploite également l'ensemble de mappings $\mathcal{M}_{pv}$ spécifiant l'ensemble de classes équivalentes à partir des différents points de vue. Soient O_1:c_i et O_2:c_p deux classes équivalentes appartenant à deux ontologies O_1 et O_2 respectivement. La règle d'inférence 4.3 appliquée aux deux classes c_i et c_p permet d'inférer que toutes les classes déclarées disjointes de c_i dans O_1 sont disjointes de c_p et vice-versa.

$$\frac{c_i, c_j \in O_1, c_p \in O_2, (c_i \equiv c_p) \in \mathcal{M}_{pv}, c_i \bot c_j}{c_p \bot c_j} \tag{4.3}$$

La règle d'inférence 4.4 appliquée aux deux classes c_i et c_p permet d'inférer que toutes les sous-classes c_k des classes c_j déclarées disjointes de c_i sont disjointes de c_p et vice-versa.

$$\frac{c_i, c_j, c_k \in O_1, c_p \in O_2, (c_i \equiv c_p) \in \mathcal{M}_{pv}, c_i \bot c_j, c_k \preceq c_j}{c_p \bot c_k} \tag{4.4}$$

La règle d'inférence 4.5 appliquée aux deux classes c_i et c_p permet d'inférer que si c_i est disjointe d'une classe c_j appartenant à O_1 alors toutes les sous-classes c_k de c_j sont disjointes des sous-classes c_l de c_p.

$$\frac{c_i, c_j, c_k \in O_1, c_p, c_l \in O_2, (c_i \equiv c_p) \in \mathcal{M}_{pv}, c_i \bot c_j, c_k \preceq c_j, c_l \preceq c_p}{c_l \bot c_k} \tag{4.5}$$

En plus de l'ensemble d'axiomes OWL, nous considérons un ensemble de contraintes spécifiques décrites ci-dessous :

1. toute classe PIT doit être décrite en utilisant uniquement des classes points de vue non disjointes
2. toute classe PIT est décrite par un ensemble de propriétés non équivalentes afin d'éviter les redondances.
3. si deux classes PIT sont décrites en utilisant des classes disjointes alors elles sont disjointes

Sur cet ensemble de règles et de contraintes, nous vérifions que nous ne générons pas d'incohérences.

4.1.4 Instanciation de l'ontologie PITO

Dans cette section, nous présentons l'instanciation de l'ontologie PITO sur des informations personnelles. Nous définissons tout d'abord ce que l'on appelle information personnelle puis nous présentons la procédure d'instanciation de l'ontologie PITO. Comme l'ontologie est représentée dans le langage OWL, l'ensemble des informations personnelles sont représentées en RDF.

4.1.4.1 Information personnelle

Une information personnelle est une instance d'un type d'informations personnelles (PIT) spécifié dans l'ontologie de l'utilisateur PITO. Pour instancier un PIT, il faut instancier tout d'abord toutes ses propriétés.

De plus, pour pouvoir réaliser la contextualisation des informations personnelles (voir section 4.2), nous considérons que toutes les propriétés décrivant les PIT sont multi-valuées.

Une information personnelle est identifiée par un identifiant unique *uri* et elle est décrite par l'ensemble de valeurs instanciant les propriétés décrivant le PIT.

Définition 4.5 (Information personnelle)**.** *Une information personnelle est un tuple de la forme* $\langle uri, T, P_{inst} \rangle$ *où*

- *uri est un identifiant unique utilisé pour représenter l'instance de l'information personnelle dans le système,*
- T *représente la classe PIT à instancier,*
- P_{inst} *est un ensemble de couples (p, V) instanciant le PIT, où* $p \in \mathcal{P}$ *et V est un ensemble de valeurs pouvant être vide.*

Exemple 4.3. *Dans la figure 4.3, le PIT Friend est décrit par les propriétés pito:firstName, pito:email, pito:phone, pito:skypeId. Une instance de PIT qui représente une amie appelée Carole possédant deux adresses mail, deux numéros de téléphone et deux identifiants Skype est décrite comme suit :*

- uri = *friend001*
- T = *Friend*
- P_{inst} =
  - *(firstName, {Carole})*
  - *(email, {carole.bernard@u-psud.fr, carole21@gmail.com})*
  - *(phone, {0633333333, 0121212121})*
  - *(skypeId, {caroleB, carole.bernard})*

Les informations personnelles sont stockées dans le format RDF. Ci-dessous la représentation RDF de l'exemple 4.3.

```
<friend001, firstName, Carole>
<friend001, email, carole.bernard@u-psud.fr>
<friend001, email, carole21@gmail.com>
<friend001, phone, 0633333333>
<friend001, phone, 0121212121>
<friend001, skypeId, caroleB>
<friend001, skypeId, carole.bernard>
```

4.2 Contextualisation d'informations personnelles

Pour représenter la façon dont les individus utilisent leurs informations personnelles, nous avons étendu la modélisation des informations personnelles de façon à prendre en compte les contextes d'utilisation. Ces derniers peuvent être de différents types : sociaux, géographiques ou encore temporels. En effet, l'utilisation de certaines informations personnelles peut se révéler non-pertinente dans certains contextes et même parfois inapproprié.

L'utilisation des informations personnelles par l'utilisateur est en étroite relation avec le contexte et la situation de celui-ci. Une même information peut avoir des valeurs utilisables dans plusieurs contextes. Par exemple, *Carole* est une étudiante, elle utilise son adresse mail *u-psud* pour échanger avec ses enseignants et son adresse *gmail* pour communiquer avec ses amis. *Bob* est un homme d'affaire habitant 6 mois à Paris et 6 mois à Londres. Pour se soigner, il utilise son numéro de sécurité social français quand il est en France et un autre numéro quand il est à Londres.

Dans notre travail, un contexte correspond à toute information représentant une situation d'utilisation des informations personnelles. Nous représentons chaque type de contexte dans une ontologie décrivant l'ensemble de valeurs de contextes possibles par un ensemble de classes. Ces classes sont organisées en hiérarchie et liées par des relations : de subsomption, d'appartenance, de disjonction ou d'équivalence. Nous considérons également qu'à chaque classe est associé un label.

Définition 4.6 (Ontologie de contextes)**.** *Une ontologie de contextes est définie par un tuple* $\langle \mathscr{C}, \mathcal{A}_{\text{cont}} \rangle$ *où*

- $\mathscr{C}$ *est l'ensemble de classes représentant les différents contextes,*
- $\mathcal{A}_{\text{cont}}$ *est l'ensemble d'axiomes déclarés dans l'ontologie de contextes pouvant être des axiomes de subsomption, d'appartenance, de disjonction ou d'équivalence.*

La figure 4.4 (a) représente une ontologie de contextes géographiques, dont laquelle les contextes sont reliés par des relations de subsomption, d'appartenance, de disjonctions ou d'équivalence. La figure 4.4 (b) présente une ontologie de contextes

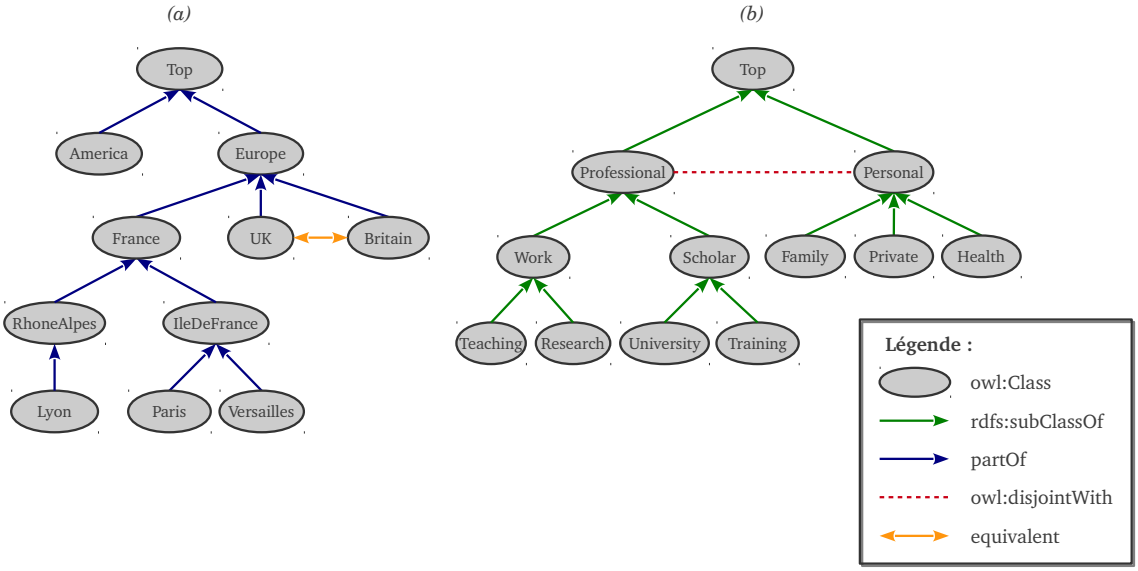


FIGURE 4.4: Ontologies de contextes : *(a)* contextes géographiques, *(b)* contextes sociaux

sociaux par exemple le contexte *Work* est subsumé par le contexte *Professional* et ce dernier est disjoint du contexte *Personal*.

Description contextuelle des informations personnelles

Pour associer des valeurs de contextes aux informations personnelles, nous représentons les valeurs des propriétés par, en plus du nom de la propriété et de l'ensemble de ses valeurs (voir définition 4.5), un troisième élément représentant les différents contextes associés aux différentes valeurs. Ainsi, une valeur d'une propriété est représentée par le triplet (p, v, c). Ce triplet pourrait être interprété par le fait que la valeur v de la propriété p est utilisable dans le contexte c. On note que la pertinence d'utilisation des valeurs dans un contexte donné pourrait être pondérée par un degré. En d'autres termes, au lieu de considérer une interprétation booléenne (vrai/faux) de l'association des contextes aux valeurs, nous considérons que cette association est pondérée par un *degré d'utilisabilité*.

Le degré d'utilisabilité est une valeur entre 0 et 1 attribuée à chaque triplet instanciant une propriété. Une valeur ayant un degré d'utilisabilité de 1 signifie que cette valeur est utilisée avec certitude dans le contexte considéré. Un degré d'utilisabilité de 0 signifie que cette valeur ne peut pas être utilisée dans ce contexte. Une valeur v déclarée avec un contexte c avec un degré dans un intervalle $]0..1[$ représente le cas où cette valeur peut être utilisée dans ce contexte lorsque aucune

autre valeur n'est déclarée avec un degré de 1 ou lorsque l'on s'intéresse à toutes les valeurs de la propriété. Par exemple un même numéro de téléphone portable pourrait être utilisé dans le contexte personnel avec un degré de 1 et dans le contexte professionnel avec un degré de 0.5.

Définition 4.7 (Information personnelle contextuelle)**.** *Une information personnelle contextuelle est un tuple de la forme* $\langle uri, T, P^{\mathscr{C}}_{inst} \rangle$ *où*

- *uri est un identifiant unique utilisé pour représenter l'instance de l'information personnelle dans le système,*
- *T représente le PIT à instancier,*
- $P^{\mathscr{C}}_{inst}$ *est un ensemble de quadruplets* (p, v, c, δ) *où p est la propriété à instancier, v est la valeur de p,* $c \in \mathscr{C}$ *est le contexte d'utilisation de v et* $\delta \in [0..1]$ *est le degré d'utilisabilité de v dans c.*

Puisque les contextes sont représentés dans des ontologies, nous pouvons exploiter toute la puissance de raisonnement sous-jacente à la sémantique logique associée aux connaissances déclarées dans l'ontologie. Nous pouvons ainsi exploiter notamment la sémantique de la subsomption entre les classes, de la disjonction et de l'équivalence. Par exemple, si l'utilisateur spécifie qu'il utilise son numéro de téléphone portable dans le contexte professionnel cela permet d'inférer qu'il peut être aussi utilisé pour tous les contextes subsumés par le contexte professionnel (*i.e.* enseignement, recherche) déclarés dans l'ontologie (figure 4.4 (b)).

Exemple 4.4. *En figure 4.3, le PIT* Friend *est décrit par les propriétés* firstName, email, phone, skypeId*. En utilisant l'ontologie de contextes sociaux (figure 4.4 (b)), une instance de ce PIT est représentée comme suit :*

- uri = *friend001*
- T = *Friend*
- P_{inst} =
  - *{(firstName, Carole, Top, 1)}*
  - *{(email, carole.bernard@u-psud.fr, Scholar, 1), (email, carole21@gmail.com, Personal, 1)}*
  - *{(phone, 0633333333, Personal, 1), (phone, 0121212121, University, 0.5)}*
  - *{(skypeId, caroleB, Personal, 0.5), (skypeId, carole.bernard, Professional, 1)}*

Les contextes et les degrés sont représentés en RDF en utilisant le mécanisme de la réification (voir section 3.2.2). Chaque information personnelle est ainsi représentée en RDF comme suit :

```
<uriSta rdf:type rdf:Statement>
<uriSta rdf:subject s>
<uriSta rdf:predicate p>
<uriSta rdf:object v>
<uriSta pimi:context c>
<uriSta pimi:degree d>
```

avec : *uriSta* représente l'URI du triplet RDF, s représente le sujet du triplet, p représente le prédicat du triplet et v représente l'objet du triplet. Les propriétés *pimi*:*context* et *pimi*:*degree* représentent les propriétés réifiées, exprimant respectivement le contexte associé au triplet et le degré d'utilisabilité de la valeur v dans le contexte c pour la propriété p.

Exemple 4.5. *La représentation de l'exemple 4.4 réifiée en RDF est donnée comme suit :*

```
<t1 rdf:type rdf:Statement>
<t1 rdf:subject friend001>
<t1 rdf:predicate firstName>
<t1 rdf:object Carole>
<t1 pimi:context Top>
<t1 pimi:degree 1>

<t2 rdf:type rdf:Statement>
<t2 rdf:subject friend001>
<t2 rdf:predicate email>
<t2 rdf:object carole.bernard@u-psud.fr>
<t2 pimi:context Scholar>
<t2 pimi:degree 1>

<t3 rdf:type rdf:Statement>
<t3 rdf:subject friend001>
<t3 rdf:predicate email>
<t3 rdf:object carole21@gmail.com>
<t3 pimi:context Personal>
<t3 pimi:degree 1>

<t4 rdf:type rdf:Statement>
<t4 rdf:subject friend001>
<t4 rdf:predicate phone>
<t4 rdf:object 0633333333>
<t4 pimi:context Personal>
<t4 pimi:degree 1>

<t5 rdf:type rdf:Statement>
<t5 rdf:subject friend001>
<t5 rdf:predicate phone>
<t5 rdf:object 01212121121>
<t5 pimi:context University>
<t5 pimi:degree 0.5>

<t6 rdf:type rdf:Statement>
<t6 rdf:subject friend001>
<t6 rdf:predicate skypeId>
<t6 rdf:object caroleB>
<t6 pimi:context Personal>
```

```
<t6 pimi:degree 0.5>

<t7 rdf:type rdf:Statement>
<t7 rdf:subject friend001>
<t7 rdf:predicate skypeId>
<t7 rdf:object carole.bernard>
<t7 pimi:context Professional>
<t7 pimi:degree 1>
```

4.3 Classification des informations personnelles

La façon dont les individus gèrent leurs informations personnelles numérisées n'est pas loin de celle qu'ils utilisent pour organiser leurs documents personnels classiques. Il est très naturel d'organiser ses documents par catégorie (***e.g.,*** fiscal, santé, banque). Les individus font de même pour les informations personnelles numérisées. Nous proposons donc dans ce travail de permettre à l'utilisateur d'organiser ses informations par catégorie. Cette solution a été adoptée dans le cadre du projet ANR PIMI dans lequel un ensemble de catégories d'informations personnelles ont été identifiées [SMDWD11]. Parmi ces catégories nous citions famille, professionnel, finance, fiscal, banque et contact.

Pour associer les catégories aux informations personnelles, chaque classe PIT est décrite par une catégorie. Par conséquent, toutes les instances du PIT vont appartenir aussi à cette catégorie. L'utilisation de ces catégories est nécessaire pour limiter et contrôler l'utilisation des informations personnelles par les applications en ligne (pour plus de détails sur la gestion des politiques de sécurité, voir la section 6.1.4).

Pour représenter ces catégories (définition 4.8), nous avons défini une ontologie OWL dans laquelle les classes représentent les différentes catégories et sont organisées en utilisant des relations de subsomption, de disjonction et d'équivalence.

Définition 4.8 (Ontologie de catégories d'informations personnelles)**.** *L'ontologie de catégories des informations personnelles est définie par un tuple* $\mathcal{O}_{\text{Info}} = \langle \mathcal{C}_{\text{Info}}, \mathcal{A}_{\text{Info}} \rangle$*, où*

- $\mathcal{C}_{\text{Info}}$ *est l'ensemble des catégories des informations personnelles,*
- $\mathcal{A}_{\text{Info}}$ *est l'ensemble des axiomes déclarés dans l'ontologie de catégories d'informations personnelles pouvant être des axiomes de subsomption, d'appartenance, de disjonction ou d'équivalence.*

La figure 4.5 montre un extrait de l'ontologie de catégories utilisée dans le projet PIMI. L'utilisation de cette ontologie permet d'exprimer par exemple que toutes les informations appartenant à la catégorie *Enseignement* appartiennent aussi à la

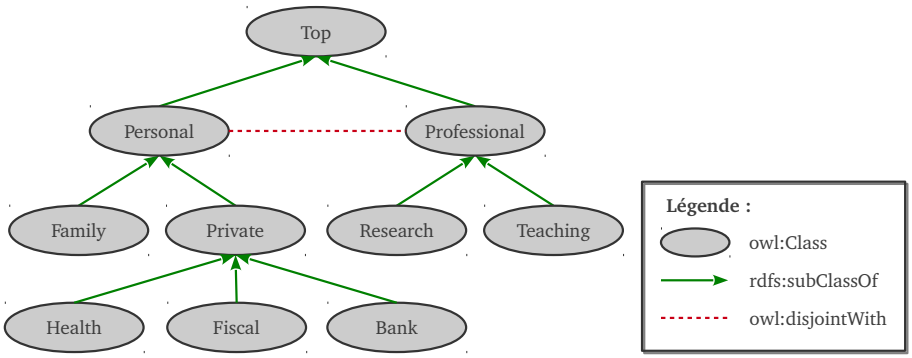


FIGURE 4.5: Extrait de l'ontologie de catégories du projet PIMI

catégorie *Professionnel*.

4.4 CoPIMS : prototype pour la gestion d'informations personnelles sensibles aux contextes

Pour montrer la faisabilité de notre approche, nous avons développé un prototype permettant la création de l'ontologie de types d'informations personnelles (PITO) en utilisant différents points de vue (ontologies). Il est également possible à un utilisateur d'instancier cette ontologie sur ses informations personnelles. De plus, en utilisant des ontologies de contextes existantes, l'utilisateur pourra s'il le souhaite associer des contextes et des degrés d'utilisabilité à ses informations personnelles. Par défaut, le contexte générique Top avec un degré de 1 est associé à toute valeur de propriété. Enfin, l'utilisateur via le prototype CoPIM a la possibilité de classer ses informations par catégorie en utilisant l'ontologie de catégories comme celle montrée en figure 4.5. Ainsi il sera plus facile d'appliquer des politiques de sécurité sur les informations personnelles de l'utilisateur.

Pour créer une ontologie PITO, le prototype CoPIM considère une ontologie de départ avec seulement la classe PITORoot. Ensuite, l'utilisateur devra créer la classe *Identity* qui représente les informations personnelles l'identifiant. Puis, en utilisant les ontologies points de vue il pourra créer les autres PITs de son ontologie. Ces derniers seront liés à la classe *Identity* via des *owl:ObjectProperty* de la forme hasNomPIT automatiquement générés. La classe *Identity* ainsi que toutes les autres classes représentant les autres PITs sont liées à la classe PITRoot via un lien de

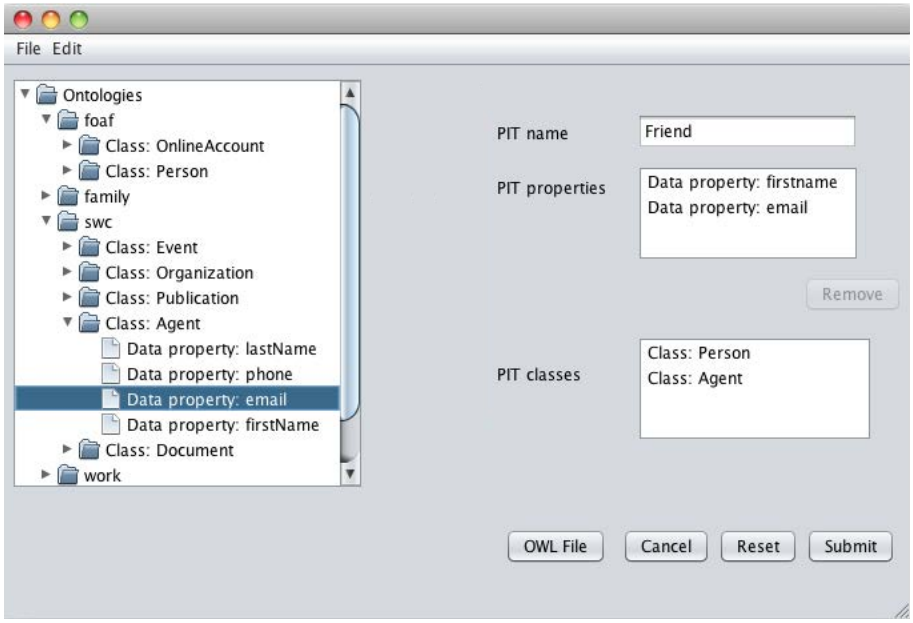


FIGURE 4.6: Interface de création de nouveau PIT

subsomption (*rdfs:subClassOf*).

La figure 4.6 présente l'interface de création d'un nouveau PIT. Nous trouvons à gauche de la figure l'ensemble des ontologies points de vue disponibles dans le PIMS servant à la description des PITs. Un nouveau PIT est créé en spécifiant un label et en sélectionnant des propriétés à partir de l'ensemble des ontologies. Ces propriétés sont ajoutées à la liste *PIT properties* et leurs classes apparaissant en domaine sont insérées dans la liste *PIT classes* permettant ainsi de construire la liste de *mappings* pour chaque PIT. Le PIT créé est ajouté à l'ontologie PITO en tant que sous-classe de la classe PITORoot suite à la pression du bouton *Submit*. Une fois que tous les PITs sont créés, l'ontologie PITO est ainsi stockée dans un fichier OWL en cliquant sur le bouton OWL File (un exemple d'une ontologie PITO en OWL est donné en annexe B).

Dans le but de stocker les informations personnelles de l'utilisateur, l'instanciation des PITs est indispensable. La figure 4.7 présente l'interface d'instanciation des PITs. À gauche de la fenêtre nous trouvons tous les PIT créés précédemment par l'utilisateur. Ce dernier sélectionne tout d'abord le PIT qu'il souhaite instancier. Puis, il sélectionne les propriétés à instancier et il introduit ensuite dans les zones de texte dédiées la valeur, le contexte et le degré de chacune. Le résultat d'instanciation est enregistré sous la forme de *rdf:Statement* en RDF (un exemple est donné en annexe C).

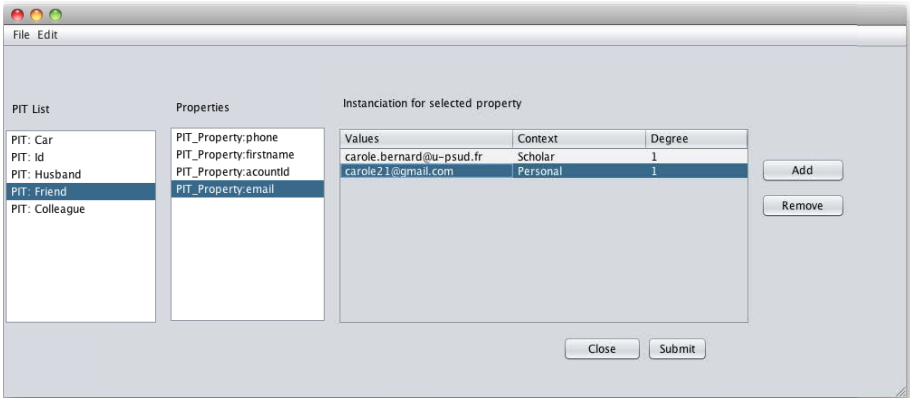


FIGURE 4.7: Interface d'instanciation des PIT

Il est également possible d'importer un fichier RDF et d'en extraire les informations personnelles que l'utilisateur souhaiterait stocker dans son PIS. Il lui suffit d'indiquer les liens de mises en correspondance entre l'ontologie PITO et le schéma du fichier RDF importé.

Conclusion

Nous avons présenté dans ce chapitre une approche de représentation contextuelle des informations personnelles de l'utilisateur dont les principaux résultats ont été publiés dans [KPS12a, KPS12b]. Cette approche permet aux utilisateurs de décrire leurs informations personnelles en utilisant le vocabulaire de diverses ontologies de domaine existantes. En effet, l'utilisateur peut créer sa propre ontologie PITO qu'il pourra ensuite instancier sur ses informations personnelles. Le modèle que nous avons proposé est composé de deux niveaux. Le premier est le niveau points de vue et il est composé d'ontologies de domaines. Le second concerne le niveau de types d'informations personnelles.

L'une des originalités de notre approche réside dans l'utilisation de contextes d'utilisabilité des informations personnelles. Selon le besoin des applications, d'autres critères peuvent être associés à une instance de propriété tels que des critères de qualité d'information comme par exemple la fraîcheur des données ou la fiabilité des sources.

Chapitre 5

Interrogation contextuelle des informations personnelles

Sommaire

L'intérêt d'avoir modélisé sémantiquement les informations personnelles est de pouvoir automatiser le plus possible leur exploitation. Les informations personnelles peuvent être exploitées par l'utilisateur propriétaire des données mais aussi par des applications nécessitant la saisie d'informations personnelles notamment via des formulaires. Dans le cadre du projet PIMI, les informations personnelles sont utilisées pour la réalisation de procédures administratives. Chaque procédure est constituée d'un ensemble de formulaires. Une étude réalisée par nos partenaires du projet a permis de mettre en évidence le fait qu'au sein d'une même procédure, une

même information pouvait être demandée plusieurs fois [WGV+11].

L'utilisateur est alors conduit à renseigner les mêmes informations maintes fois. Afin d'aider l'utilisateur à éviter la tâche répétitive et fastidieuse du remplissage des formulaires, nous avons proposé une approche qui utilise les informations personnelles stockées dans le PIS pour remplir automatiquement ces formulaires. Dans le cadre applicatif du projet PIMI, nous considérons que chaque procédure est constituée d'un seul formulaire. De plus, nous supposons que toutes les informations demandées dans le formulaire sont stockées dans le PIS. En outre, puisque nous représentons les informations personnelles en leur associant des contextes, nous considérons qu'à chaque formulaire est associé un contexte d'utilisation (***e.g.,*** *Professional*, France).

Nous présentons dans ce chapitre une approche qui permet d'exploiter les informations personnelles en vue du remplissage automatique de formulaires. Cette approche, dans un premier temps, génère automatiquement une *requête sémantique* à partir d'une description annotée des formulaires (section 5.1). Dans un second temps, ces requêtes sont contextualisées puis évaluées suivant deux algorithmes (section 5.2) : (*i*) une évaluation stricte des contextes (algorithme *SQE*) et (*ii*) une évaluation flexible des contextes (algorithme *FQE*).

5.1 Génération automatique de requêtes sémantiques à partir de formulaires

Un formulaire est un espace de saisie dans une interface utilisateur. Il peut comporter plusieurs zones où un utilisateur peut saisir du texte, cocher des cases, sélectionner des valeurs dans une liste ou cliquer sur des boutons. Les formulaires sont décrits en utilisant des conventions et des vocabulaires différents. En effet, bien que les champs des formulaires représentent souvent les mêmes informations, il n'en demeure pas moins qu'ils peuvent être syntaxiquement différents. Pour avoir une représentation uniforme des formulaires, nous considérons qu'il est possible de les annoter sémantiquement en utilisant l'ontologie PITO. Ainsi, la génération de la requête à partir du formulaire annoté devient plus aisée.

5.1.1 Annotation d'un formulaire par une ontologie

Nous considérons qu'un formulaire est composé d'un ensemble de zones. Chaque zone est constituée d'un ensemble de champs à remplir. Une zone peut également

contenir des sous-zones. Par exemple, un formulaire de déclaration d'impôts peut nécessiter des informations concernant le demandeur et d'autres concernant son conjoint. Dans ce cas les informations du conjoint apparaissent dans une sous-zone.

En général, chaque zone du formulaire décrit une entité complexe d'informations, par exemple, les informations identifiant la personne, des informations décrivant son véhicule personnel. Ces entités complexes peuvent être représentées par des classes de l'ontologie PITO (***e.g.,*** *pito*:*Car*, *pito*:*Identity*). Chaque zone d'un formulaire contient un ensemble de champs élémentaires, comme par exemple nom, prénom ou numéro d'immatriculation. Ces champs peuvent être représentés par des propriétés de l'ontologie PITO (*pito*:*Identity.firstName*, *pito*:*Car.licensePlate*).

Nous considérons que l'annotation des formulaires est a priori effectuée manuellement ou (semi-)automatiquement en utilisant des outils d'annotation sémantique guidée par une ontologie [HSM01, MEG+03]. Le résultat attendu de l'annotation d'un formulaire en utilisant l'ontologie PITO est le suivant :

- chaque zone est annotée par une classe de l'ontologie PITO,
- chaque champ est annoté par une propriété de l'ontologie PITO,
- pour représenter les liens (zone - sous-zone) des propriétés de type *owl*:*ObjectProperty* sont déclarées entre les classes annotant les zones et celles annotant les sous-zones.

Définition 5.1. ***(Formulaire annoté.)*** *Étant donné une ontologie de l'utilisateur* $\text{PITO} = (id, \mathfrak{C}, \mathfrak{R}, \mathfrak{M}_p)$*, un formulaire annoté* $\mathcal{F}$ *est un tuple* $\langle Z, Ch, An \rangle$ *où :*

- *Z est un ensemble de zones composant le formulaire,*
- *Ch est un ensemble de champs décrivant les zones du formulaire,*
- *An est l'ensemble des annotations exprimé sous la forme d'un ensemble de tuples représentant les différentes mises en correspondance entre le formulaire et l'ontologie PITO. Nous définissons trois types de mise en correspondance :*

  1. *annotation zone-classe* (z, c) *où* z *est une zone du formulaire et* c *est une classe de l'ontologie PITO,*
  2. *annotation champs-propriété* (ch, p) *où* ch *est un champs du formulaire et* p *est une propriété dans l'ontologie PITO de la classe* c *associée à la zone incluant ce champs,*
  3. *annotation relation-zone-sous-zone* (r, z_1, z_2) *où* z_1 *est une zone du formulaire incluant* z_2*,* r *est une owl*:*ObjectProperty de l'ontologie PITO ayant comme domaine* c_1 *et comme co-domaine* c_2 *qui sont les deux classes associées respectivement à* z_1 *et* z_2*.*

Prenons l'exemple du formulaire donné en figure 5.1. Notons que *pito*:*Identity*, *pito*:*Husband* et *pito*:*Car* sont des classes de l'ontologie de l'utilisateur PITO, *hasHusband* et *hasCar* sont des propriétés de type *owl*:*ObjectProperty* de l'ontologie PITO, et *pito*:*Identity.familyName*, *pito*:*Identity.firstName*, *pito*:*Husband.lastName*,

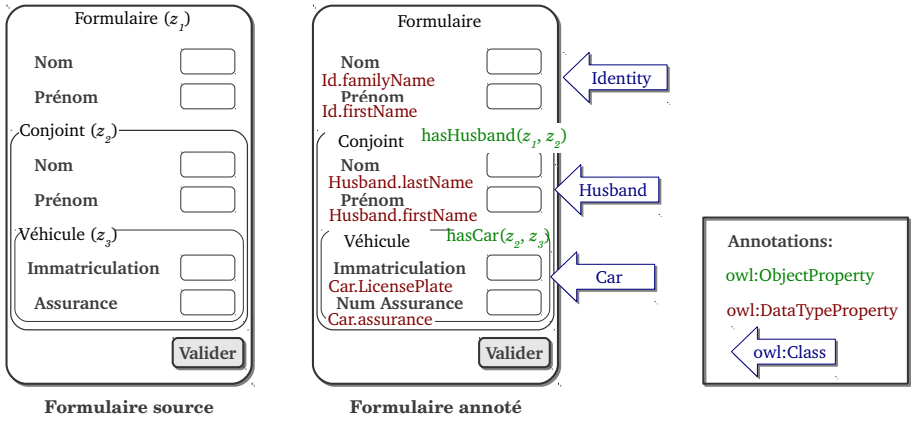


FIGURE 5.1: Formulaire annoté par l'ontologie PITO

pito:*Husband.firstName*, *pito*:*Car.licensePlate* et *pito*:*Car.assurance* sont des propriétés de type *owl*:*DataTypeProperty* de l'ontologie PITO.

La zone z_1 du formulaire annotée par **Identity** est composée d'une sous-zone z_2 annotée par **Husband**, elle-même composée d'une autre sous-zone z_3 annotée par **Car**. Pour représenter les relations entre zones et sous-zones une *owl*:*ObjectProperty* *hasHusband* est déclarée entre z_1 et z_2 et une autre *owl*:*ObjectProperty* *hasCar* est déclarée entre z_2 et z_3.

5.1.2 Représentation arborescente d'un formulaire annoté

Chaque formulaire est composé d'un ensemble de zones et chaque zone est composée d'un ensemble de champs et/ou d'un ensemble de sous-zones. Chaque zone correspond à une classe dans l'ontologie PITO, et chaque champs correspond à une propriété de type *owl*:*DataTypeProperty* de l'ontologie PITO. Les zones d'un même formulaire sont reliées par des propriétés de type *owl*:*ObjectProperty*, qui représentent les relations sémantiques entre les différentes zones du formulaire.

Pour générer la requête sémantique qui permettrait de remplir automatiquement le formulaire, nous représentons le formulaire annoté sous une forme arborescente.

Dans cet arbre, nous distinguons deux types de nœuds : les *nœuds terminaux* et les *nœuds non-terminaux*.

- *un nœud terminal* est un nœud représentant une propriété de type *owl*:*DataTypeProperty* correspondant à un champs du formulaire,
- *un nœud non-terminal* est un nœud représentant une classe de l'ontologie PITO annotant une zone du formulaire.

Ces nœuds sont reliés par deux types d'arcs :

— un *arc terminal* est un arc qui lie un nœud non-terminal à un nœud terminal,
— un *arc non-terminal* est un arc qui lie deux nœuds non-terminaux. Cet arc est étiqueté par le nom de la propriété (*owl:ObjectProperty*) exprimant la relation entre les zones du formulaire représentées par ces deux nœuds.

Définition 5.2. ***(Arbre de formulaire).*** *Un arbre de formulaire est défini par un tuple* $\mathcal{T} = \langle \mathcal{N}, \mathcal{E} \rangle$*, où*

— $\mathcal{N}$ *est l'ensemble de nœuds de l'arbre* ($\mathcal{N} = \mathcal{N}_{\text{NT}} \cup \mathcal{N}_{\text{T}}$)*, où* $\mathcal{N}_{\text{NT}}$ *(resp.* $\mathcal{N}_{\text{T}}$*) représente les nœuds non terminaux (resp. les nœuds terminaux),*
— $\mathcal{E} = \mathcal{E}_{\text{NT}} \cup \mathcal{E}_{\text{T}} :\rightarrow \mathcal{N} \times \mathcal{N}$ *où* $\mathcal{E}_{\text{NT}}$ *représente l'ensemble d'arcs reliant les nœuds non-terminaux entre eux et* $\mathcal{E}_{\text{T}}$ *représente l'ensemble d'arcs reliant les nœuds non-terminaux aux nœuds terminaux.*

La transformation d'un formulaire en arbre est réalisée comme le montre l'algorithme *F2T* détaillé en algorithme 1.

Étant donné un formulaire $\mathcal{F} = \langle Z, Ch, An \rangle$, pour :

— chaque zone, un nœud non-terminal est créé.
— chaque champ, un nœud terminal est créé. De plus, un arc terminal est créé entre ce nœud terminal et le nœud non-terminal représentant la zone contenant ce champ.
— chaque relation, un arc non-terminal étiqueté par le nom de la relation est créé.

Exemple 5.1. *L'arbre correspondant au formulaire annoté de la figure 5.1 est montré en figure 5.2.*

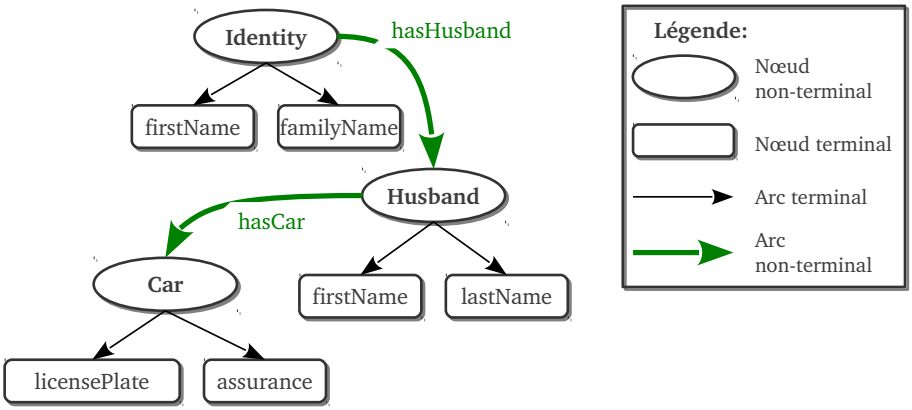


FIGURE 5.2: Traduction arborescente du formulaire annoté

Algorithme 1 : Traduction arborescente d'un formulaire annoté ($F2T$)

Entrées : $\mathcal{F}$: un formulaire annoté $\langle Z, Ch, An\rangle$
Sorties : $\mathcal{T}$: arbre de formulaire $\langle \mathcal{N}, \mathcal{E}\rangle$

début
  $\mathcal{N} \leftarrow \emptyset$
  $\mathcal{E} \leftarrow \emptyset$
  ; /* Création des nœuds non-terminaux */
  pour chaque $(z, c) \in An$ **faire**
    $terminalNode \leftarrow createTerminalNode(label(c))$
    $\mathcal{N} \leftarrow \mathcal{N} \cup terminalNode$
  ; /* Création des nœuds terminaux et des arcs terminaux */
  pour chaque $(ch, p) \in An$ **faire**
    $nonTerminalNode \leftarrow createNonTerminalNode(label(p))$
    $\mathcal{N} \leftarrow \mathcal{N} \cup nonTerminalNode$
    $nonTerminalArc \leftarrow createNonTerminalArc(label(p), domain(p))$
    $\mathcal{E} \leftarrow \mathcal{E} \cup nonTerminalArc$
  ; /* Création des arcs non-terminaux */
  pour chaque $(r, z_1, z_2) \in An$ **faire**
    $terminalArc \leftarrow createTerminalArc(label(r), domain(r), range(r))$
    $\mathcal{E} \leftarrow \mathcal{E} \cup terminalArc$
fin

5.1.2.1 Traduction d'un formulaire annoté en requête sémantique

Avant de présenter notre méthode de traduction d'un formulaire annoté en requête sémantique, nous définissons d'abord formellement la notion de requête sémantique.

Requête sémantique. Nous considérons qu'une requête sémantique est une requête conjonctive exprimée en utilisant le vocabulaire de l'ontologie PITO. Nous rappelons qu'une requête conjonctive dans le domaine des bases de données relationnelles est une requête dont l'expressivité est restreinte à la logique du premier ordre [CR97].

Définition 5.3 (Requête sémantique)**.** *Soit $\mathcal{O}$ l'ontologie PITO $\langle \mathcal{C}, \mathcal{P}, \mathcal{A}\rangle$, une requête sémantique Q sur $\mathcal{O}$ est une expression de la forme :*

$$Q(\overrightarrow{x}) \leftarrow S_1(\overrightarrow{y_1}), S_2(\overrightarrow{y_2}), \ldots, S_n(\overrightarrow{y_n})$$

où $Q(\overrightarrow{x})$ est la tête de la requête et la conjonction des $S_i(\overrightarrow{y_i})$ est son corps. Le symbole « , » exprime la conjonction, et $n > 1$ et $S_1(\overrightarrow{y_1}), S_2(\overrightarrow{y_2}), \ldots, S_n(\overrightarrow{y_n})$ sont

des classes ou des propriétés de $\mathcal{O}$, $\overrightarrow{x}$ est le vecteur des variables apparaissant dans la tête de la requête qui doivent apparaître au moins une fois dans le corps de la requête, et $\overrightarrow{y_1}, \ldots, \overrightarrow{y_n}$ sont des vecteurs de variables.

La traduction du formulaire en requête sémantique est effectuée en deux phases : (*i*) la traduction du formulaire annoté en arbre par l'algorithme *F2T* (algorithme 1) (*ii*) la génération automatique de la requête sémantique à partir de l'arbre de formulaire en utilisant l'algorithme *T2Q* (algorithme 2).

L'algorithme *F2T* prend en entrée un formulaire annoté et produit un arbre représentant le contenu du formulaire ainsi que sa structure.

Une fois l'arbre du formulaire construit, l'algorithme *T2Q* prend cet arbre en entrée et génère automatiquement la requête sémantique. Cet algorithme prend en entrée un formulaire annoté et renvoie en résultat une requête sémantique.

Pour chaque nœud de l'arbre passé en paramètre, l'algorithme *T2Q* génère une variable dans l'ensemble {X,Y,Z...} associée. La variable associée à un nœud n est notée $n.var$. En suite, L'algorithme *T2Q* génère les atomes de la requête comme suit :

1. pour tout nœud n non-terminal, un atome de prédicat unaire est généré de la forme : $n.name(n.var)$ qui représente le nom de la classe à laquelle fait référence cette zone du formulaire.
2. Pour tout arc terminal liant un nœud non terminal $n1$ à un nœud terminal $n2$, un atome de prédicat binaire de la forme : $n2.name(n1.var, n2.var)$
3. Pour tout arc e non terminal liant un nœud $n1$ à un nœud $n2$, un atome de prédicat binaire de la forme : $e.etiquette(n1.var, n2.var)$

L'ensemble des atomes générés sont unis sous une forme conjonctive. L'ensemble des variables correspondant aux nœuds terminaux sont ajoutées à la tête de la requête formant ainsi l'ensemble de variables de projection (*i.e.* celles pour lesquelles des valeurs doivent être renvoyées dans les réponses).

Exemple 5.2. *Prenons l'exemple du formulaire annoté en figure 5.1. La requête sémantique correspondant à son arborescence donnée en figure 5.2 est la suivante :*
$Q(A, B, C, D, E, F) \leftarrow Identity(X) \wedge Husband(Y) \wedge Car(Z) \wedge hasHusband(X, Y) \wedge hasCar(Y, Z) \wedge Identity.firstName(X, A) \wedge Identity.familyName(X, B) \wedge Husband.firstName(Y, C) \wedge Husband.lastName(Y, D) \wedge Car.assurance(Z, E) \wedge Car.licensePlate(Z, F)$

5.2 Évaluation de requêtes contextuelles

Les informations qu'un utilisateur renseigne lors du remplissage d'un formulaire peuvent avoir différentes valeurs en fonction du contexte. En effet, il peut utiliser

Algorithme 2 : Traduction Arbre-Requête sémantique ($T2Q$)

Entrées : $\mathcal{T}$: arbre de formulaire $\langle \mathcal{N}, \mathcal{E} \rangle$
Sorties : Q : requête sémantique $\langle head, body \rangle$

début
  $vars \leftarrow \emptyset$
  $Atomes \leftarrow \emptyset$
  $headVars \leftarrow \emptyset$
  pour chaque $n \in \mathcal{N}$ **faire**
    $vars \leftarrow vars \cup newVariables()$
  ; /* Traduction des nœuds non-terminaux en prédicats unaires */
  pour chaque $n \in \mathcal{N}$ **faire**
    si $nonTerminal(n)$ **alors**
      $atome \leftarrow createUnaryPredicate(n.name, n.var)$
      $Atomes \leftarrow Atomes \cup atome$
  ; /* Traduction des arcs en prédicat binaire */
  pour chaque $e(n_1, n_2) \in \mathcal{E}$ **faire**
    si $nonTerminal(n_1)$ *and* $terminal(n_2)$ **alors**
      $atome \leftarrow createBinaryPredicate(n_2.name, n_1.var, n_2.var)$
      $headVars \leftarrow headVars \cup n_2.var$
    si $nonTerminal(n_1)$ *and* $nonTerminal(n_2)$ **alors**
      $atome \leftarrow createBinaryPredicate(e.etiquette, n_1.var, n_2.var)$
    $Atomes \leftarrow Atomes \cup atome$
  ; /* Écriture de la requête sémantique */
  $body \leftarrow set2String(Atomes, `\wedge')$
  $head \leftarrow set2String(headVars, `,')$
  $Q \leftarrow concat(head, \leftarrow, body)$
fin

son adresse et son téléphone professionnel dans le cadre du travail et son adresse de domicile et son téléphone portable dans le cadre personnel. Dans le but de remplir automatiquement un formulaire, l'utilisateur doit spécifier dans quel contexte ce formulaire va être utilisé. Une fois le contexte spécifié, une requête contextuelle est créée en utilisant la requête sémantique générée à partir du formulaire annoté (section 5.1). Cette requête retourne les informations personnelles (réponses) qui sont déclarées comme utilisables dans ce contexte. Les réponses seront ordonnées par degré de pertinence. Ce degré est dit *local*, lorsqu'il est calculé pour une instance de propriété et il est dit *global* lorsqu'il est calculé pour l'ensemble d'instances de propriétés d'une réponse.

Dans la suite nous présentons deux algorithmes de sélection d'informations personnelles pour le remplissage automatique du formulaire. Ces deux algorithmes sont donnés en termes d'une requête sémantique, d'un contexte de référence et d'une fonction de comparaison sur les contextes. Ils sont détaillés respectivement dans les sections 5.2.3 et 5.2.4.

5.2.1 Contextualisation des requêtes sémantiques

Dans cette section nous montrons comment une requête contextuelle est générée à partir d'une requête sémantique.

Une requête contextuelle est une requête sémantique à laquelle sont ajoutés un *contexte de référence* et une *fonction de comparaison entre contextes*. Le contexte de référence représente le contexte dans lequel le formulaire est utilisé (***e.g.,*** *Personnal*, *Professional*, France) que l'on note par $\underline{c}$. Ce contexte doit être présent dans l'ontologie de contextes $\mathcal{O}^{\mathscr{C}}$.

En ce qui concerne la fonction de comparaison f, il s'agit d'une fonction qui permet d'exprimer la pertinence des contextes des informations personnelles stockées dans le PIS par rapport au contexte de référence. Nous avons défini trois types de fonctions possibles : stricte ($=$), similaire ($\sim$) et disjointe ($\perp$).

— *Stricte :* exprime que les réponses retournées par la requête sont celles utilisables dans exactement le contexte de référence.
— *Similaire :* exprime que les réponses retournées par la requête sont celles utilisables dans des contextes similaires au contexte de référence.
— *Disjoint :* exprime que les réponses retournées par la requête sont celles utilisables dans des contextes déclarés disjoints du contexte de référence.

Définition 5.4 (Requête contextuelle)**.** *Une requête contextuelle est définie par un tuple* $\mathcal{Q}^{f\underline{c}} = \langle Q, f, \underline{c} \rangle$ *où*

— Q *est une requête sémantique (définition 5.3),*
— $f \in \{=, \sim, \perp\}$ *est une fonction de comparaison,*
— $\underline{c} \in \mathscr{C}$ *est le contexte de référence.*

Une requête contextuelle retourne un ensemble de réponses ($\mathscr{R}$) satisfaisant la requête et la fonction de comparaison sur les contextes.

Définition 5.5 (Une réponse)**.** *Étant donnée une requête contextuelle* $\mathcal{Q}^{f\underline{c}}(\overrightarrow{x})$ *où* n *est la cardinalité de* $\overrightarrow{x}$*, une réponse* r_j *de la requête* $\mathcal{Q}^{f\underline{c}}(\overrightarrow{x})$ *est une expression de la forme* $r_j = \langle P_j^{inst}, \Delta_j \rangle$ *où*

— P_j^{inst} *est l'ensemble d'instances correspondant à* $\overrightarrow{x}$*,* $P_j^{inst} = \langle p_1^{inst}, \ldots, p_n^{inst} \rangle$ *avec :* p_i^{inst} *est l'instance de la* $i^{\text{ème}}$ *propriété de* $\overrightarrow{x}$*,* $p_i^{inst} = (v_i, c_i, \delta_i^r)$ *où* δ_i^r *est*

le degré de pertinence local du contexte c_i par rapport au contexte de référence $\underline{c}$,

— *Δ_j est le degré de pertinence global de la $j^{ème}$ réponse r_j de l'ensemble de réponses $\mathscr{R}$.*

Les calculs du degré de pertinence local δ^r et de pertinence global Δ sont détaillés dans la section 5.2.2.

5.2.2 Calcul du degré de pertinence

Nous distinguons deux types de degrés de pertinence : un degré local et un degré global définis ci-dessous.

Degré de pertinence local. Le degré de pertinence local est un score de similarité calculé entre le contexte de référence et le contexte de l'instance d'information personnelle stockée dans le PIS. Ces deux contextes appartiennent à une même ontologie de contextes. Le calcul de ce degré exploite la hiérarchie de l'ontologie de contextes.

Étant donnés une ontologie de contextes $\mathcal{O}^{\mathscr{C}}$, une instance d'information personnelles (p,v,c,δ) et un contexte de référence $\underline{c}$, le degré de pertinence local $\delta^r(\underline{c}, c)$ est calculé comme suit :

— $\delta^r(\underline{c}, c) = 0$ si $\underline{c} \perp c$,
— $\delta^r(\underline{c}, c) = \delta$ si $\underline{c} \equiv c$, si $\underline{c} \sqsubset c$ ou si $c \preceq \underline{c}$,
— $\delta^r(\underline{c}, c) = sim_{WP}(\underline{c}, c)$ si $c \sqsubset \underline{c}$ ou si $\underline{c} \preceq c$

où sim_{WP} la une mesure de similarité structurelle développée par Wu et Palmer [WP94], et δ est le degré d'utilisabilité associé à la valeur v de la propriété p pour le contexte c dans le PIS. Comme nous l'avons vu en section 3.3, la mesure sim_{WP} exploite les relations de subsomption de l'ontologie pour calculer la distance entre deux classes.

Degré de pertinence global. Le degré de pertinence global est un score calculé comme étant la moyenne de tous les degrés de pertinence locaux associés aux instances de propriétés composant une réponse r à la requête contextuelle.

Étant donné un ensemble d'instances de propriétés d'une réponse r, $P^{inst} = \langle (v_1, c_1, \delta_1^r), (v_2, c_2, \delta_2^r), \ldots, (v_n, c_n, \delta_n^r) \rangle$, le degré de pertinence global Δ est calculé comme suit :

$$\Delta = \frac{\sum_{i=1}^{n} \delta_i^r}{n}$$

Le degré global est utilisé pour ordonner les réponses à une requête contextuelle. Il exprime en effet le degré de pertinence de l'ensemble des valeurs de propriétés

par rapport au contexte de référence.

Dans la suite nous présentons les deux algorithmes que nous avons développé pour l'évaluation des requêtes contextuelles : évaluation exacte de requêtes contextuelles et évaluation flexible de requêtes contextuelles.

5.2.3 Évaluation exacte de requêtes contextuelles *SQE*

Nous présentons dans cette section l'algorithme d'évaluation exacte de requête contextuelle *SQE* (algorithme 3). Le terme *exacte* exprime le fait que toutes les instances de propriétés d'une même réponse doivent être déclarées comme utilisables dans le même contexte.

L'évaluation de la requête en utilisant l'algorithme *SQE* (algorithme 3) peut être effectuée pour les trois valeurs possibles de la fonction f.

Le cas où « f *est* $=$ », la requête doit retourner l'ensemble de réponses pour lesquelles les instances de propriétés sont déclarées comme utilisables dans le contexte de référence.

Le cas où « f *est* $\sim$ », la requête doit retourner les réponses ayant un contexte équivalent ou similaire au contexte de référence. Bien que nous restreignions les réponses de l'algorithme *SQE* à celles dont les instances de propriétés sont utilisables dans un même contexte pour une réponse donnée, entre les différentes réponses, ces contextes peuvent être différents. Le calcul de l'ensemble de contextes similaires au contexte de référence est donné par la fonction $similarContexts(\mathcal{O}^{\mathscr{C}}, \underline{c})$. Cette fonction prend en paramètre une ontologie de contextes et un contexte. Elle retourne l'ensemble de contextes similaires à ce dernier.

Le cas où « f *est* $\perp$ », l'évaluation de la requête est analogue à celle où « f *est* $\sim$ ». En revanche les contextes qui nous intéressent ne sont pas les contextes similaires mais ceux qui sont disjoints du contexte de référence. Le calcul de l'ensemble de contextes disjoints est effectué en utilisant la fonction $disjointContexts(\mathcal{O}^{\mathscr{C}}, \underline{c})$. Cette fonction prend en paramètres une ontologie de contexte et un contexte de référence et calcule l'ensemble de contextes déclarés ou inférés par saturation comme disjoint avec ce dernier.

Une fois l'ensemble de réponses calculé, nous procédons au calcul du degré global de pertinence correspondant à chaque réponse (voir section 5.2.2).

Exemple 5.3. *Soit l'ontologie PITO donnée en figure 4.3, le PIT Friend concernant Alice est instancié comme le montre le tableau 5.3. Chaque instance de propriété est de la forme* (p,v,c,δ).

Algorithme 3 : Évaluation exacte d'une requête contextuelle (SQE)

Entrées :
Q : requête sémantique
$\underline{c}$: contexte de référence
f : fonction de comparaison des contextes
$\mathcal{O}^{\mathscr{C}}$: ontologie de contextes
PIS : ensemble des informations personnelles de l'utilisateur
Sorties : $\mathscr{R}$: ensemble de réponses de la requête

```
1  début
2      rsList ⟵ ∅
3      ℛ ⟵ ∅
4      suivant f faire
           ; /* Cas où f est =                                   */
5          cas où (=)
6              rsList ⟵ queryEval(Q, c, PIS)
7              pour chaque rep ∈ rsList faire
8                  ℛ ⟵ ℛ ∪ ⟨rep, globalDegree(rep)⟩
           ; /* Cas où f est ∼                                   */
9          cas où (∼)
10             contexts ⟵ similarContexts(O^𝒞, c)
11             pour chaque c ∈ contexts faire
12                 rsList ⟵ queryEval(Q, c, PIS)
13                 pour chaque rep ∈ rsList faire
14                     localDegrees(rep)
15                     ℛ ⟵ ℛ ∪ ⟨rep, globalDegree(rep)⟩
           ; /* Cas où f est ⊥                                   */
16         cas où (⊥)
17             contexts ⟵ disjointContexts(O^𝒞, c)
18             pour chaque c ∈ contextList faire
19                 rsList ⟵ queryEval(Q, c, PIS)
20                 pour chaque rep ∈ rsList faire
21                     localDegrees(rep)
22                     ℛ ⟵ ℛ ∪ ⟨rep, globalDegree(rep)⟩
23     sort(ℛ)
24 fin
```

TABLEAU 5.1: Instanciation du PIT *Friend*

Friend001	***Friend002***
(firstName,Carole,Top,1)	(firstName,Bob,Top,1)
(email,carole.bernard@u-psud.fr,Scholar,1)	(email,bob.lebeau@gmail.com,Personal,1)
(email,carole21,Personal,1)	(email,lebeau@univ-paris1.fr,Work,1)
(phone,0633333333,Personal,1)	(phone,0625252525,Personal,1)
(phone,0121212121,University,0.5)	(phone,0150505050,Work,0.5)
(skypeId,caroleB,Personal,0.5)	(phone,0230303030,Professional,1)
(skypeId,carole.bernard,Professional,1)	(skypeId,leBeau,personal,1)

La requête contextuelle à évaluer est la suivante : $\mathcal{Q}^{-Professional}(Friend.email, Friend.Phone)$ *demandant les adresses électroniques et les numéros de téléphone des amis d'Alice utilisables dans le contexte professionnel.*

En évaluant la requête ci-dessus sur les informations personnelles d'Alice en utilisant SQE *nous obtenons les réponses suivantes :*

TABLEAU 5.2: Évaluation de la requête contextuelle avec *SQE*

f	***Email***	***Phone***	Δ
$=$	$\langle bob.lebeau@gmail.com, Professional, 1\rangle$	$\langle 0230303030, Professional, 1\rangle$	1
$\sim$	$\langle bob.lebeau@gmail.com, Professional, 1\rangle$	$\langle 0230303030, Professional, 1\rangle$	1
	$\langle lebeau@univ-paris1.fr, Work, 1\rangle$	$\langle 0150505050, Work, 0.5\rangle$	0.75
$\perp$	$\langle carol21@gmail.com, Personal, 1\rangle$	$\langle 0633333333, Personal, 1\rangle$	1

En s'appuyant sur les réponses à la requête obtenues dans le cas où « f *est* $\sim$ », nous constatons que Carole dispose d'une adresse mail utilisable dans le contexte *Scholar* et un numéro de téléphone utilisable dans le contexte *University*. Cette réponse pourrait être pertinente pour le contexte *Professional* car *Scholar* et *University* sont des sous-classes de *Professional*. Dans le cas où l'utilisateur souhaite aussi retrouver les informations de Carole, le résultat de cet algorithme n'est pas totalement satisfaisant. Afin de remédier à ce problème nous avons proposé un nouvel algorithme dont le but est d'étendre l'évaluation de la requête et d'autoriser l'obtention de réponses dont les valeurs de propriétés sont déclarées comme utilisables dans des contextes différents.

5.2.4 Évaluation flexible de requêtes contextuelles *FQE*

Nous avons voulu autoriser le fait qu'une réponse puisse être retournée même si l'ensemble des valeurs de ses propriétés ne sont pas déclarées comme utilisables dans le même contexte. De plus, le fait d'avoir différents contextes dans une même

réponse pose le problème de la pertinence de la réponse.

Algorithme 4 : Évaluation flexible d'une requête contextuelle (*FQE*)

Entrées :
Q : requête sémantique
$\underline{c}$: contexte de référence
f : fonction de comparaison des contextes
$\mathcal{O}^{\mathscr{C}}$: ontologie de contextes
PIS : ensemble des informations personnelles de l'utilisateur
Sorties : $\mathscr{R}$: ensemble de réponses de la requête

début
  $rsList \longleftarrow \emptyset$, $\mathscr{R} \longleftarrow \emptyset$
  suivant f **faire**
    ; /* Cas où f est $=$ */
    cas où $(=)$
      $rsList \longleftarrow queryEval(Q, \underline{c}, \text{PIS})$
      pour chaque $rep \in rsList$ **faire**
        $\mathscr{R} \longleftarrow \mathscr{R} \cup \langle rep, globalDegree(rep)\rangle$
    ; /* Cas où f est $\sim$ */
    cas où $(\sim)$
      $contexts \longleftarrow similarContexts(\mathcal{O}^{\mathscr{C}}, \underline{c})$
      $rsList \longleftarrow queryEval(Q, contexts, \text{PIS})$
      pour chaque $rep \in rsList$ **faire**
        $localDegrees(rep)$
        $\mathscr{R} \longleftarrow \mathscr{R} \cup \langle rep, globalDegree(rep)\rangle$
    ; /* Cas où f est $\perp$ */
    cas où $(\perp)$
      $contexts \longleftarrow disjointContexts(\mathcal{O}^{\mathscr{C}}, \underline{c})$
      $rsList \longleftarrow queryEval(Q, contexts, \text{PIS})$
      pour chaque $rep \in rsList$ **faire**
        $localDegrees(rep)$
        $\mathscr{R} \longleftarrow \mathscr{R} \cup \langle rep, globalDegree(rep)\rangle$
  $sort(\mathscr{R})$
fin

Les détails de l'algorithme *FQE* sont donnés par l'algorithme 4. Il considère trois cas possibles pour la fonction de comparaison sur les contextes :

Le cas où « f *est* $=$ », est identique à celui de l'algorithme *SQE*.

Dans le cas où « f *est* $\sim$ », le résultat de la requête est un ensemble de réponses ordonné par les degrés de pertinence globaux, où chaque réponse peut être composée d'instances de propriétés avec différents contextes. Pour construire les réponses, chacune doit contenir des instances de propriétés utilisables dans un contexte similaire à celui demandé par la requête.

Dans le cas où « f *est* $\perp$ », après le calcul de l'ensemble de contextes disjoints de celui de la requête, la même procédure que celle utilisée pour « f *est* $\sim$ » est utilisée pour sélectionner les réponses.

Exemple 5.4. *En évaluant la requête donnée dans l'exemple 5.3 sur les informations personnelles d'Alice nous obtenons les réponses suivantes :*

TABLEAU 5.3: Évaluation de la requête contextuelle avec FQE

f	***Email***	***Phone***	Δ
$=$	$\langle bob.lebeau@gmail.com, Professional, 1\rangle$	$\langle 0230303030, Professional, 1\rangle$	1
$\sim$	$\langle bob.lebeau@gmail.com, Professional, 1\rangle$	$\langle 0230303030, Professional, 1\rangle$	1
	$\langle lebeau@univ-paris1.fr, Work, 1\rangle$	$\langle 0150505050, Work, 0.5\rangle$	0.75
	$\langle carole.bernard@u-psud.fr, Scholar, 1\rangle$	$\langle 0121212121, University, 0.5\rangle$	0.75
$\perp$	$\langle carol21@gmail.com, Personal, 1\rangle$	$\langle 0633333333, Personal, 1\rangle$	1

Nous remarquons que la réponse concernant Carole a bien été renvoyée par l'algorithme FQE et non pas par SQE.

Conclusion

Dans ce chapitre, nous avons présenté nos contributions concernant l'exploitation d'informations personnelles. La première concerne le remplissage automatique de formulaires dans laquelle nous avons développé un algorithme qui génère une requête sémantique à partir de la représentation annotée d'un formulaire. La seconde concerne le développement de deux algorithmes d'évaluation de requêtes contextuelles (SQE, FQE). Les deux algorithmes calculent un degré de pertinence local et global servant à ordonner les réponses. Le calcul du degré de pertinence pourrait être étendu pour prendre en compte d'autres informations en lien avec la qualité des données comme la fraîcheur ou la fiabilité des sources.

Chapitre 6

Approche de composition de services Web sensible aux contextes

Sommaire

Dans ce chapitre nous présentons notre contribution sur la génération automatique d'un modèle de composition de services correspondant à un besoin utilisateur défini en terme de tâches abstraites. Ce besoin est, dans la plupart des cas, complexe, et nécessite l'enchaînement et la composition de plusieurs services. Les données utilisées lors de la composition de services sont les données personnelles de l'utilisateur contenues dans son PIS. Ce type de données à caractère sensible engendre un certain nombre de précautions à prendre lors de leur exploitation en ligne et donc lors de la génération du modèle de composition.

Dans ce qui suit, nous présentons notre approche de composition de services, et l'outil correspondant, **CoCliCo**. Notre approche est sensible aux contextes et elle prend en compte un ensemble de politiques de sécurité définies par l'utilisateur lors de la composition, offrant ainsi la possibilité de limiter la divulgation des données personnelles.

Ce chapitre est organisé comme suit : en section 6.1, nous présentons l'ensemble des éléments considérés dans le problème de composition auquel nous faisons face. Ensuite, en section 6.2 nous détaillons l'encodage de ce problème de composition en un problème de planification afin de permettre son traitement automatique. Ce dernier est présenté en section 6.3. Nous présentons en section 6.4 un jeu de test et une évaluation de notre approche de composition. Nous terminons ce chapitre par une conclusion.

6.1 Modélisation du problème de composition

Un problème de composition, suivant la présentation faite en section 2.1.1, est la donnée d'un ensemble de services disponibles pour la composition, de données que l'utilisateur est capable de fournir (provenant de son PIS) et de l'objectif qu'il souhaite accomplir.

Dans cette section, nous présentons successivement ces différents éléments. Nous présentons le besoin que l'utilisateur cherche à réaliser en section 6.1.1, les services disponibles en section 6.1.2, les données contextualisées à prendre en compte en section 6.1.3 et les politiques de sécurités à respecter en section 6.1.4.

6.1.1 Procédures

Le premier élément que nous considérons est le besoin de l'utilisateur. Ce besoin est spécifié par une procédure c'est-à-dire un ensemble de tâches abstraites à effectuer appelées *capacités* ou *fonctionnalités*. Il est possible d'associer aux procédures une conversation définie sous forme de *workflow* [HH93] ou d'automate dont l'alphabet correspond aux capacités à réaliser. Dans le cadre de ce travail, les

conversations seront modélisées par un *workflow* de capacités. Ce choix est motivé par l'expressivité, le caractère graphique et l'outillage disponible pour cette famille de formalismes. Nous nous basons ici sur une forme simple des *workflows* avec des opérateurs de séquences, choix exclusif, et parallélisme. Nos *workflows* sont bien équilibrés [KtHB00] et sans boucles.

Définition 6.1 (Procédure [Kie03]). *Étant donné un ensemble d'activités A, une procédure est définie par un tuple $Proc = \langle N, \rightarrow, \lambda \rangle$ où*

- *N est l'ensemble des nœuds composant la procédure. Ces nœuds peuvent être divisés en un ensemble disjoint de nœuds $N = N_A \cup N_{JA} \cup N_{JO} \cup N_{SA} \cup N_{SO}$ où N_A est l'ensemble d'activités basiques $N_A \in A$, N_{JA} est l'ensemble des nœuds de type AND-joins, N_{JO} est l'ensemble des nœuds de type XOR-joins, N_{SA} est l'ensemble des nœuds de type AND-splits et N_{SO} est l'ensemble des nœuds de type XOR-splits. Les nœuds XOR-splits et XOR-joins modélisent le choix exclusif et les nœuds AND-splits et AND-joins modélisent le parallélisme (figure 6.1),*
- *$\rightarrow \subseteq N \times N$ dénote le flux de contrôle (séquence),*
- *$\lambda : N_A \longrightarrow A$ est une fonction permettant d'assigner une activité à un nœud d'activité.*

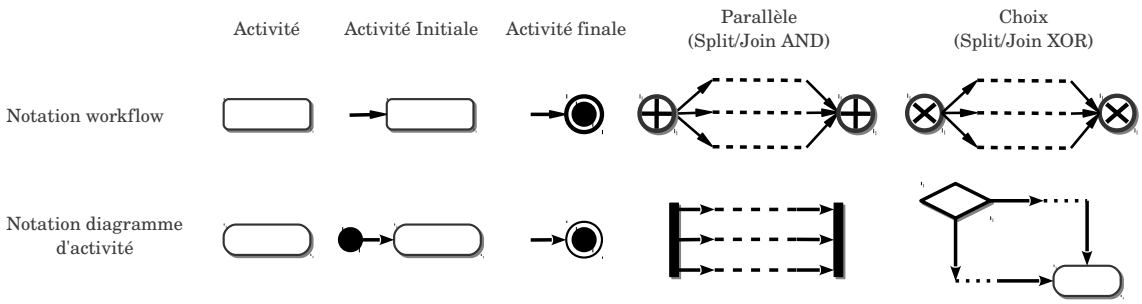


FIGURE 6.1: Notations graphiques d'un *workflow* (Notations BPMN et UML)

On note $\bullet x = \{y \in N | y \rightarrow x\}$ l'ensemble des nœuds qui précèdent immédiatement x et $x\bullet = \{y \in N | x \rightarrow y\}$ l'ensemble des nœuds qui succèdent immédiatement à x. Un nœud initial n'est précédé par aucun autre nœud ($\bullet x = \emptyset$) et un nœud final n'a pas de successeur ($x\bullet = \emptyset$).

La notion de bonne structuration d'un *workflow* signifie que les XOR-splits et les XOR-joins (respectivement les AND-splits et les AND-joins) apparaissent par paire (figure 6.1). De plus, nous exigeons que seuls les nœuds joints puissent avoir plus d'un prédécesseur ($|\bullet x| \leq 1$ pour chaque $x \in N_A \cup N_{SA} \cup N_{SO}$) et que seuls les nœuds splits puissent avoir plus d'un successeur ($|x\bullet| \leq 1$ pour chaque $x \in N_A \cup N_{JA} \cup N_{JO}$).

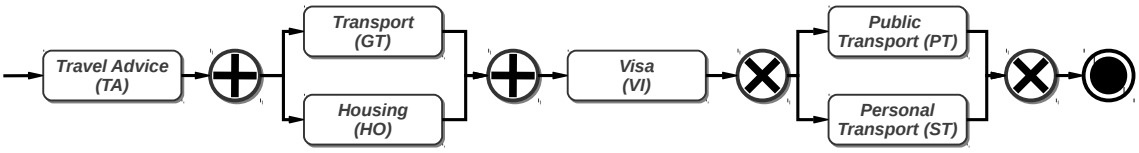


FIGURE 6.2: Procédure de planification d'un voyage

Exemple 6.1. *Un exemple de procédure est donné en figure 6.2. Il s'agit de la planification d'un voyage*

Après avoir obtenu des conseils de voyage, il faut obtenir un moyen de transport et un logement en préalable à l'obtention d'un visa. On termine ensuite en demandant soit un transport public soit un transport personnel. Les capacités ici sont abstraites.

6.1.2 Services

Le deuxième composant nécessaire pour notre problème de composition concerne les services disponibles pour réaliser un besoin abstrait par composition de services. D'un point de vue pratique, nous considérons qu'un service consiste en un ensemble de formulaires qui sont décrits en terme de capacités, d'attributs d'entrée et d'attributs de sortie. De plus, un service peut appartenir à une catégorie (***e.g.,*** gouvernemental, commercial), ce qui facilite la définition de politiques d'accès aux données. Pour cela nous introduisons le concept d'ontologie de catégories de services $\mathcal{O}_{\text{Serv}}$.

Définition 6.2 (Ontologie de catégories de services)**.** *Une ontologie de catégories de services est définie par un tuple* $\mathcal{O}_{\text{Serv}} = \langle \mathcal{C}_{\text{Serv}}, \mathcal{A}_{\text{Serv}} \rangle$ *où*

- $\mathcal{C}_{\text{Serv}}$ *est l'ensemble des catégories des services,*
- $\mathcal{A}_{\text{Serv}}$ *est l'ensemble des axiomes déclarés dans l'ontologie de catégories de services pouvant être des axiomes de subsomption, d'appartenance, de disjonction ou d'équivalence.*

Étant données deux catégories cat_{S1} et cat_{S2}, $cat_{S2} \preceq_{\text{Serv}} cat_{S1}$ signifie que tout service appartenant à la catégorie cat_{S2} appartient aussi à la catégorie cat_{S1}

Exemple 6.2. *Un exemple d'ontologie de catégories de services est donnée en figure 6.3. On peut y voir que les services de voyage (catégorie Travel) sont aussi des services de e-commerce. On peut aussi y voir que les services de la catégorie Manipulation et de la catégorie Government sont disjoints.*

Nous pouvons maintenant définir les services.

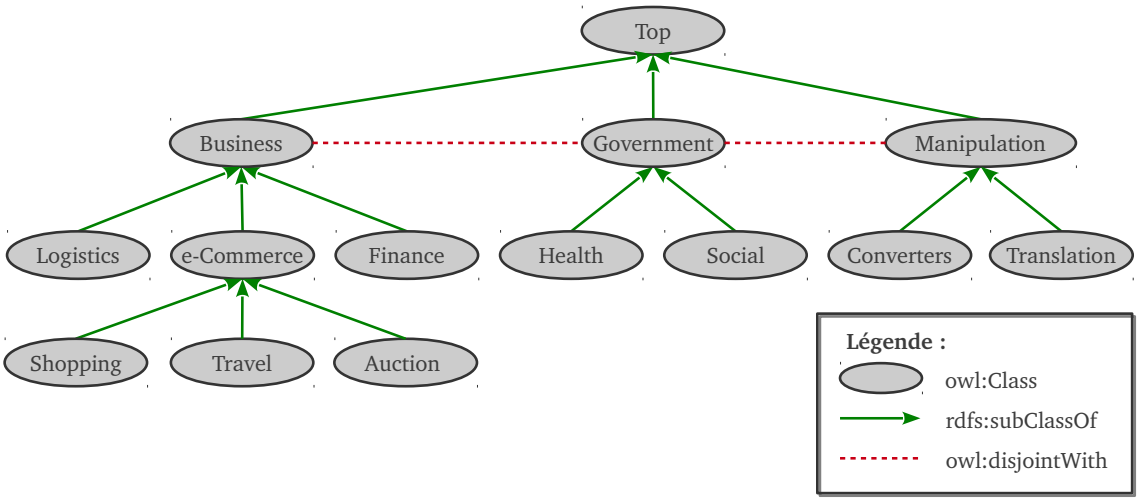


FIGURE 6.3: Ontologie de catégories de services

Définition 6.3 (Service)**.** *Étant donnés un ensemble de propriétés* $\mathfrak{P}$*, une ontologie de capacités* $\mathcal{O}_{\text{Cap}} = (\mathcal{C}_{\text{Cap}}, \mathcal{A}_{\text{Cap}})$ *et une ontologie de catégories* $\mathcal{O}_{\text{Serv}} = (\mathcal{C}_{\text{Serv}}, \mathcal{A}_{\text{Serv}})$*, définition 6.2, un service est défini par un tuple* $w = \langle \text{u}, I, O, K, C \rangle$ *où*

- u *est l'identifiant du service (nom ou adresse unique),*
- $I \subseteq \mathfrak{P}$ *est l'ensemble d'attributs en entrée du service,*
- $O \subseteq \mathfrak{P}$ *est l'ensemble d'attributs en sortie du service,*
- $K \subseteq \mathcal{C}_{\text{Cap}}$ *est l'ensemble des capacités réalisées par le service,*
- $C \in \mathcal{C}_{\text{Serv}}$ *est la catégorie à laquelle appartient le service* w*.*

Exemple 6.3. *Dans le but d'accomplir le besoin de l'utilisateur donné en figure 6.2, un ensemble de services est nécessaire. Nous supposerons disponibles les services décrits dans le tableau 6.1. La première colonne donne le nom du service, la deuxième décrit son interface et la dernière donne sa catégorie. L'interface d'un service est de la forme*

{attributs en entrée} $\rightarrow$ *{attributs en sortie} [capacités]*

6.1.3 Données contextuelles

Un autre élément important dans notre problème de composition est l'ensemble des données personnelles de l'utilisateur. Dans le chapitre 4, nous avons proposé un modèle de représentation des données personnelles sensible aux contextes. Ces données sont enregistrées dans le PIS sous la forme d'un ensemble de tuples $\langle p, v, c, \delta \rangle$ où $p \in \mathfrak{P}$ est la donnée, v est sa valeur, $c \in \mathscr{C}$ est le contexte d'utilisation de la valeur v et δ est le degré d'utilisabilité de v dans le contexte c. Par ailleurs, chaque type de données PIT appartient à une catégorie de données $cat^{Info} \in \mathcal{C}_{\text{Info}}$.

TABLEAU 6.1: Ensemble de services pour l'exemple Planification de voyage

Service	*Interface*	*Catégorie*
w_1	nationality, food-prefs → travel-country,travel-city [**Travel-Advice (TA)**]	Travel
w_2	nationality, hobbies-prefs → travel-country, travel-city [**Travel-Advice (TA)**]	Travel
w_3	identity, passport, dep-date, return-date, dep-airport-code, arrival-airport-code → e-ticket [**Transport (GT)**]	Travel
w_4	identity, travel-city, dep-date, return-date → hotel-reservation, hotel-address [**Housing (HO)**]	Travel
w_5	identity, passport, nationality, dep-date, return-date, hotel-reservation, e-ticket, travel-country → visa [**Visa (VI)**]	Government
w_6	arrival-airport-code, hotel-address → bus-return-ticket [**Public-Transport (PT)**]	e-Commerce
w_7	arrival-airport-code, hotel-address → [**Personal-Transport (ST)**]	
w_8	conf, dep-date, return-date, research-id → hotel-reservation, hotel-address [**Housing (HO)**]	e-Commerce
w_9	travel-country, freedate → dep-date, return-date []	Travel
w_{10}	travel-country, freedate → conf, dep-date, return-date []	Travel
w_{11}	home-city → dep-airport-code []	Manipulation
w_{12}	identity, passport, nationality, travel-country → visa [**Visa (VI)**]	Government
w_{13}	travel-city → arrival-airport-code []	

Dans la suite nous dénotons par PIS l'ensemble des données personnelles du PIS en associant à chaque propriété la catégorie de son type d'informations PIT. Chaque donnée contextuelle est ainsi représentée par un tuple de la forme $\langle p, v, c, \delta, cat^{Info} \rangle$.

Exemple 6.4. *Dans le but d'accomplir le besoin de l'utilisateur, nous disposons d'un certain nombre de données personnelles. Ces données sont présentées dans le tableau 6.2. On voit par exemple que notre utilisateur aime manger exotique dans le cadre de ses loisirs mais continental dans le cadre de son travail.*

6.1.4 Politiques de sécurité

Utiliser les données personnelles d'un utilisateur de façon automatique en les transférant à des services en ligne peut créer des problèmes de sécurité liés à la divulgation d'informations à des tiers qui ne seraient pas de confiance. Afin de limiter ce risque, nous proposons que l'utilisateur puisse contrôler ses données en spécifiant

TABLEAU 6.2: Ensemble de données contextuelles

Propriété	*Valeur*	*Contexte*	*Degré*	*Catégorie*
nationality	{French}	Top	1.0	Top
food-prefs	{french, exotic}	Leasure	1.0	Personal
food-prefs	{french,continental}	Work	1.0	Personal
hobbies-prefs	{history,science}	Leasure	0.5	Personal
identity	{John Doe}	Top	1.0	Top
passport	{T758960,2013}	Top	1.0	Private
passport	{R234589,2018}	Top	1.0	Private
freedate	{01/05/2014–15/07/2014}	Work	0.5	Personal
freedate	{01/08/2014–30/08/2014}	Leasure	1.0	Personal
research-id	{RK2861127}	Work	1.0	Research
home-city	{Paris}	Top	1.0	Personal

la manière dont elles peuvent être utilisées lors de la réalisation de la procédure par composition automatique de services. L'utilisateur peut ainsi préciser des politiques telles que, « utiliser mes informations privées uniquement pour les services gouvernementaux ». Ce type de politiques s'appuie sur l'utilisation de catégories d'informations et de catégories de services. Ces catégories sont représentées par des ontologies (voir la section 4.3 pour les catégories de données et la section 6.1.2 pour les catégories de services). Le choix des ontologies nous permet d'inférer de nouvelles règles à partir de celles définies par l'utilisateur et de garder la définition des politiques assez concises. En effet, le nombre important de catégories de services et de celles des données existantes imposerait à l'utilisateur une tâche difficile et fastidieuse s'il devait définir et déclarer toutes les règles possibles.

Définition 6.4 (Politique de sécurité)**.** *Étant données une ontologie de catégories d'information* $\mathcal{O}_{\text{Info}} = (\mathcal{C}_{\text{Info}}, \mathcal{A}_{\text{Info}})$*, définition 4.8, et une ontologie de catégories de services* $\mathcal{O}_{\text{Serv}} = (\mathcal{C}_{\text{Serv}}, \mathcal{A}_{\text{Serv}})$*, définition 6.2, une règle de sécurité est un tuple* $pol = (x, y)$ *aussi noté* $x \rhd y$*, avec :*

- $x \in \mathcal{C}_{\text{Info}}$
- $y \in \mathcal{C}_{\text{Serv}}$

Une politique $x \rhd y$ signifie que l'utilisateur autorise les services de catégorie y à utiliser les données de catégorie x. En appliquant la fermeture transitive de chaque politique par rapport aux axiomes de subsomptions déclarés dans les ontologies $\mathcal{O}_{\text{Info}}$ et $\mathcal{O}_{\text{Serv}}$, nous obtenons que y et toutes ses sous-catégories sont aussi autorisés à utiliser les données de catégorie x ou d'une de ses sous-catégories. Nous dénotons par Pol, l'ensemble des politiques de sécurités définies par l'utilisateur ou obtenues à partir de celles-ci par transitivité :
si $x \rhd y \in Pol$ alors $\forall x',\ x' \preceq_{\text{Info}} x, \forall y', y' \preceq_{\text{Serv}} y,\ x' \rhd y'$

L'ensemble des politiques de sécurité traduit uniquement ce que l'utilisateur autorise, ainsi toutes les combinaisons qui n'appartiennent pas à cet ensemble sont exclues et ne sont pas autorisées.

6.1.5 Problèmes de composition

Le but de la composition est de trouver un enchaînement de services qui permette d'accomplir une procédure abstraite en utilisant uniquement les données personnelles de l'utilisateur (enregistrées dans son PIS) ou les données produites par les services utilisés. Cet usage doit respecter les politiques d'usage de l'utilisateur. La procédure peut de plus être réalisée pour un contexte particulier. Ce contexte est sélectionné par l'utilisateur parmi ceux contenus dans l'ontologie de contexte utilisée lors de l'instanciation des informations personnelles (section 4.1.4). Enfin, en raison de la pertinence (l'utilisabilité) des données par rapport aux contextes (voir section 4.2),

il est ainsi important de préciser la valeur minimale à utiliser pour les degrés d'utilisabilité des différentes instances de propriétés. Sur la base de ces contraintes, un problème de composition est défini comme suit.

Définition 6.5 (Problème de composition)**.** *Étant donnée une ontologie de contextes* $\mathcal{O}^{\mathscr{C}} = (\mathscr{C}, \mathcal{A}_{\text{cont}})$*,définition 4.6, une ontologie de catégories de données* $\mathcal{O}_{\text{Info}} = (cat^{Info}, \mathcal{A}_{\text{Info}})$*, définition 4.8, une ontologie de catégories de services* $\mathcal{O}_{\text{Serv}} = (\mathcal{C}_{\text{Serv}}, \mathcal{A}_{\text{Serv}})$*, définition 6.2, et une ontologie de capacités* $\mathcal{O}_{\text{Cap}} = (\mathcal{C}_{\text{Cap}}, \mathcal{A}_{\text{Cap}})$*, un problème de composition est défini par un tuple* $\mathscr{P} = \langle Proc, \underline{c}, \epsilon, W, \text{PIS}, Pol \rangle$ *où*

- $Proc$ *est la procédure (définition 6.1) que l'utilisateur souhaite accomplir,*
- $\underline{c} \in \mathscr{C}$ *est le contexte dans lequel la procédure doit être accomplie (Top pour indiquer une composition sans contexte précis),*
- $\epsilon \in [0..1]$ *est le degré d'utilisabilité minimal acceptable,*
- W *est l'ensemble des services (définition 6.3) disponibles,*
- PIS *est l'ensemble des informations personnelles de l'utilisateur (i.e. l'ensemble de valeurs de propriétés contextuelles (définition 4.7) de l'utilisateur),*
- Pol *est l'ensemble des politiques de sécurité (définition 6.4) que l'utilisateur a défini en fonction de* $\mathcal{O}_{\text{Serv}}$ *et de* $\mathcal{O}_{\text{Info}}$*.*

Exemple 6.5. *Pour l'illustration de notre exemple, la procédure est celle de la planification d'un voyage présentée en figure 6.2, l'ensemble des services est donné dans le tableau 6.1, les données de l'utilisateur sont présentées dans le tableau 6.2, nous choisissons le contexte Leasure pour la réalisation de la procédure et un seuil de 0.3 pour le degré d'utilisation minimal.*

6.2 Encodage automatique du problème de composition

Lors de l'étude des travaux de composition de services réalisée en chapitre 2, nous avons noté que plusieurs travaux ont utilisé la technique de *GraphPlan* comme base à leurs approches. Ces approches traitent notamment la qualité de services et exploitent les conversations. De plus, *GraphPlan* a l'avantage de générer des solutions courtes en maximisant le parallélisme. Nous avons donc choisi d'adopter cette technique pour résoudre notre problème de composition (définition 6.5).

Dans ce but, nous traduisons tout d'abord les problèmes de composition en problèmes de planification (section 2.3.1).

Définition 6.6 (Problème de planification)**.** *Étant donné un ensemble de propositions* P*, un problème de planification est un tuple* $\langle \mathbf{I}, \mathbf{G}, \mathbf{A} \rangle$ *où :*

- $\mathbf{I}$ *est un état initial composé d'un ensemble de propositions initiales* $\mathbf{I} \subseteq P$*,*

— $\mathbf{G}$ *est un état final composé d'un ensemble de propositions objectif* $\mathbf{G} \subseteq P$,
— $\mathbf{A}$ *est un ensemble d'actions où chaque action* $\mathbf{a}$ *est définie par (*$\mathbf{Pre}(\mathbf{a})$, $\mathbf{Eff}^-(\mathbf{a})$, $\mathbf{Eff}^+(\mathbf{a})$*) où* $\mathbf{Pre}(\mathbf{a})$ *est l'ensemble des préconditions de* $\mathbf{a}$, $\mathbf{Eff}^-(\mathbf{a})$ *est l'ensemble de ses effets négatifs et* $\mathbf{Eff}^+(\mathbf{a})$ *est l'ensemble de ses effets positifs.*

Dans la suite nous présentons comment les différents éléments du problème de composition peuvent être encodés en tant que problème de planification.

6.2.1 Encodage de la procédure

L'utilisation d'une conversation pour la représentation d'une procédure augmente l'expressivité des problèmes de composition par la prise en compte de contraintes d'ordre sur la composition de services. En effet, dans le cadre par exemple d'une procédure séquençant les capacités $k_0 \dots k_j \dots k_n$, un service de capacité k_j ne peut être utilisé que lorsque toutes les capacités $k_0..k_{j-1}$ ont été réalisées par appel de services correspondants. Dans un processus de composition uniquement basé données, un service est utilisable si tous ses attributs en entrée sont présents. Toutefois, il pourrait arriver que cette condition soit vérifiée mais que la capacité du service ne soit pas encore autorisée. Par exemple, pour la réalisation de la procédure *planification de voyage* (figure 6.2), le service w_{12} (tableau 6.1) effectuant la capacité *Visa* pourrait être appliqué avant le service w_3 (tableau 6.1) qui lui, effectue la capacité *Transport* si ses données d'entrée sont disponibles. Ceci contreviendrait cependant à l'ordonnancement décrit par la procédure. Dans le but d'avoir un plan résultant de la planification qui soit valide, il faut prendre en compte la conversation de la procédure lors de l'encodage du problème en actions de planification. Pour ce faire, nous avons utilisé la transformation de *workflow* en réseau de Petri définie par Kiepusewski [Kie03], qui a été à son tour transformée, par Poizat et Yan [PY10], en un ensemble d'actions de planification.

Les contraintes du comportement (*e.g.* une action précède/succède une autre) sont assurées par l'utilisation de deux propositions : $r_{x,y}$ et $c_{x,y}$. Nous avons également deux propositions indiquant le début et la fin de la procédure, $\sharp$ encode l'état initial et $\surd$ encode la terminaison correcte de la procédure. Nous traduisons ainsi la procédure selon les règles suivantes :

— pour chaque $x \in N_{SA}$, nous avons une action $\mathbf{a} = \oplus x$, pour chaque $x \in N_{JA}$, nous avons $\mathbf{a} = \overline{\oplus} x$, et pour chaque $x \in N_A$, nous avons une action $\mathbf{a} = [\lambda(x)]x$. Dans les trois cas nous avons de plus :
$\mathbf{Pre}(\mathbf{a}) = \mathbf{Eff}^-(\mathbf{a}) = \bigcup_{y \in \bullet x} \{\mathbf{r_{x,y}}\}$, et $\mathbf{Eff}^+(\mathbf{a}) = \bigcup_{y \in x\bullet} \{\mathbf{c_{x,y}}\}$.
— pour chaque $x \in N_{SO}$, pour chaque $y \in x\bullet$, nous avons une action $\mathbf{a} = \otimes x, y$ avec :
$\mathbf{Pre}(\mathbf{a}) = \mathbf{Eff}^-(\mathbf{a}) = \bigcup_{z \in \bullet x} \{\mathbf{r_{x,z}}\}$, et $\mathbf{Eff}^+(\mathbf{a}) = \{\mathbf{c_{x,y}}\}$.

— pour chaque $x \in N_{JO}$, pour chaque $y \in \bullet x$, nous avons une action $\mathbf{a} = \overline{\otimes} x, y$ avec :
$\mathbf{Pre}(\mathbf{a}) = \mathbf{Eff}^-(\mathbf{a}) = \{\mathbf{r_{x,y}}\}$ et $\mathbf{Eff}^+(\mathbf{a}) = \bigcup_{z \in \bullet x} \{\mathbf{c_{x,z}}\}$
— pour chaque $x \to y$, nous avons une action $\mathbf{a} = \rightsquigarrow x, y$ avec :
$\mathbf{Pre}(\mathbf{a}) = \mathbf{Eff}^-(\mathbf{a}) = \{\mathbf{c_{x,y}}\}$ et $\mathbf{Eff}^+(\mathbf{a}) = \{\mathbf{r_{y,x}}\}$.
— pour toute action initiale $\mathbf{a}$ nous ajoutons $\sharp$ à $\mathbf{Pre}(\mathbf{a})$ et à $\mathbf{Eff}^-(\mathbf{a})$.
— pour toute action finale $\mathbf{a}$ nous ajoutons $\surd$ à $\mathbf{Eff}^+(\mathbf{a})$.

Ceci permet d'encoder l'ordonnancement des actions de la procédure prescrites dans sa conversation. Il reste à s'assurer que les services soient bien planifiés en correspondance. Il faut donc *lier* l'encodage de la procédure et celui des services.

Pour ceci, nous proposons de lier chaque capacité k à trois propositions $\mathbf{enabled_k}$, $\mathbf{done_k}$ et $\mathbf{link}_k$ (voir la figure 6.4). Dans l'encodage de la procédure, nous remplaçons toute action $\mathbf{a} = [k]x$ par deux actions $\mathbf{a}'$ et $\bar{\mathbf{a}}'$, dont l'ordre d'exécution est assuré par la proposition $\mathbf{link}_k$, avec :

$\mathbf{Pre}(\mathbf{a}') = \mathbf{Pre}(\mathbf{a})$,
$\mathbf{Eff}^-(\mathbf{a}') = \mathbf{Eff}^-(\mathbf{a})$,
$\mathbf{Eff}^+(\mathbf{a}') = \{\mathbf{enabled_k}, \mathbf{link}_x\}$
$\mathbf{Pre}(\bar{\mathbf{a}}') = \{\mathbf{done_k}, \mathbf{link}_x\}$,
$\mathbf{Eff}^-(\bar{\mathbf{a}}') = \{\mathbf{done_k}, \mathbf{link}_x\}$, et
$\mathbf{Eff}^+(\bar{\mathbf{a}}') = \mathbf{Eff}^+(\mathbf{a})$.

Supposons que dans la procédure, la capacité k_1 doive être réalisée avant la capacité k_2, et que l'on ait un service w_1 de capacité k_1. Alors les dépendances entre encodages sont :

— quand k_1 devient possible ($\mathbf{enabled_{k_1}}$), alors w_1 le devient, donc $\mathbf{enabled_{k_1}}$ est une précondition de w_1 mais aussi un effet négatif pour empêcher qu'un autre service de capacité k_1 soit exécuté en plus de w_1, réalisant la capacité

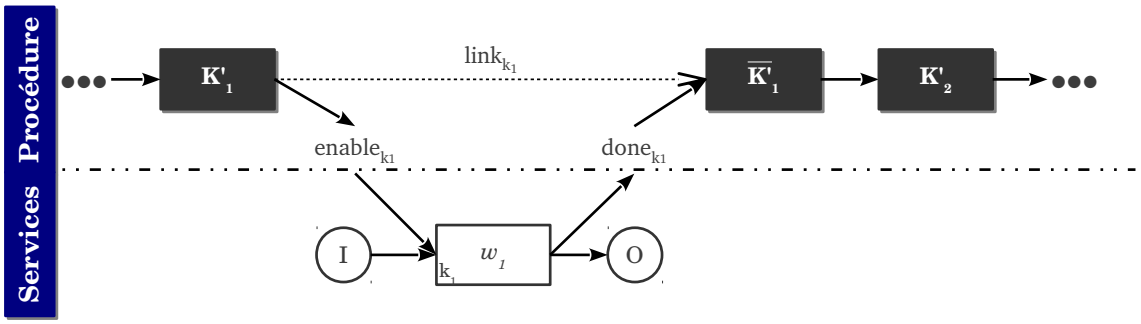


FIGURE 6.4: Intéraction entre procédures et services dans l'encodage de planification

TABLEAU 6.3: Encodage de la procédure de la figure 6.2

Action	*Préconditions*	*Effects négatifs*	*Effects positifs*
JO_end-xor	{r_end-xor_PT, r_end-xor_ST}	{r_end-xor_PT, r_end-xor_ST}	{√ }
SO_xor	{r_xor_VI}	{r_xor_VI}	{c_xor_PT, c_xor_ST}
JA_end-and	{r_end-and_HO, r_end-and_GT}	{r_end-and_HO, r_end-and_GT}	{c_end-and_VI}
A_PT_1	{done_PT, link_PT}	{done_PT, link_PT}	{c_PT_end-xor}
A_PT	{r_PT_xor}	{r_PT_xor}	{enabled_PT, link_PT}
A_VI_1	{done_VI, link_VI}	{done_VI, link_VI}	{c_VI_xor}
A_VI	{r_VI_end-and}	{r_VI_end-and}	{enabled_VI, link_VI}
A_HO_1	{done_HO, link_HO}	{done_HO, link_HO}	{c_HO_end-and}
A_HO	{r_HO_and}	{r_HO_and}	{enabled_HO, link_HO}
A_GT_1	{done_GT, link_GT}	{done_GT, link_GT}	{c_GT_end-and}
A_GT	{r_GT_and}	{r_GT_and}	{enabled_GT, link_GT}
A_ST_1	{done_ST, link_ST}	{done_ST, link_ST}	{c_ST_end-xor}
A_ST	{r_ST_xor}	{r_ST_xor}	{enabled_ST, link_ST}
A_TA_1	{done_TA, link_TA}	{done_TA, link_TA}	{c_TA_and}
A_TA	{♯ }	{♯ }	{enabled_TA, link_TA}
SA_and	{r_and_TA}	{r_and_TA}	{c_and_GT, c_and_HO}
TR_PT_end-xor	{c_PT_end-xor}	{c_PT_end-xor}	{r_end-xor_PT}
TR_xor_PT	{c_xor_PT}	{c_xor_PT}	{r_PT_xor}
TR_xor_ST	{c_xor_ST}	{c_xor_ST}	{r_ST_xor}
TR_end-and_VI	{c_end-and_VI}	{c_end-and_VI}	{r_VI_end-and}
TR_VI_xor	{c_VI_xor}	{c_VI_xor}	{r_xor_VI}
TR_HO_end-and	{c_HO_end-and}	{c_HO_end-and}	{r_end-and_HO}
TR_GT_end-and	{c_GT_end-and}	{c_GT_end-and}	{r_end-and_GT}
TR_ST_end-xor	{c_ST_end-xor}	{c_ST_end-xor}	{r_end-xor_ST}
TR_TA_and	{c_TA_and}	{c_TA_and}	{r_and_TA}
TR_and_GT	{c_and_GT}	{c_and_GT}	{r_GT_and}
TR_and_HO	{c_and_HO}	{c_and_HO}	{r_HO_and}

en double.

— quand w_1 est terminé, k_1 est réalisée ($\mathbf{done_{k_1}}$).

— quand k_1 est terminé, k_2 devient réalisée.

Exemple 6.6. *La procédure présentée en figure 6.2, est ainsi encodée en un ensemble d'actions comme le montre le tableau 6.3 :*

6.2.2 Encodage des services

Un service peut être exécuté uniquement si tous ses attributs d'entrée sont disponibles et si sa (ou ses) capacité(s) est (sont) activée(s) par l'état courant de l'exécution de la procédure. Le service génère alors ses attributs de sortie et indique que sa (ou ses) capacité (s) a (ont) été accomplie (s). Chaque service $w = \langle \mathrm{u}, I, O, K, C \rangle$ est donc encodé par une action
$\mathbf{a} = (\mathbf{Pre}(\mathbf{a}), \mathbf{Eff}^-(\mathbf{a}), \mathbf{Eff}^+(\mathbf{a}))$ avec :

— $\mathbf{Eff}^-(\mathbf{a}) = \bigcup_{k \in K} \{\mathbf{enabled_k}\}$,

— $\mathbf{Eff}^+(\mathbf{a}) = O \cup \bigcup_{k \in K} \{\mathbf{done_k}\}$, et

— $\mathbf{Pre}(\mathbf{a}) = I \cup \mathbf{Eff}^-(\mathbf{a})$.

Exemple 6.7. *L'encodage des services décrits dans le tableau 6.1 est effectué comme le montre le tableau 6.4 :*

TABLEAU 6.4: Encodage des services

Action	***Préconditions***	***Effects négatif***	***Effets positifs***
A_w1	{nationality, food-prefs, enabled_TA}	{enabled_TA}	{travel-country, travel-city, done_TA}
A_w2	{nationality, hobbies-prefs, enabled_TA}	{enabled_TA}	{travel-country, travel-city, done_TA}
A_w3	{identity, passport, dep-date, return-date, dep-airport-code, arrival-airport-code, enabled_GT}	{enabled_GT}	{e-ticket, done_GT}
A_w4	{identity, travel-city, dep-date, return-date, enabled_HO}	{enabled_HO}	{hotel-reservation, hotel-address, done_HO}
A_w5	{identity, passport, nationality, dep-date, return-date, e-ticket, hotel-reservation, travel-country, enabled_VI}	{enabled_VI}	{visa, done_VI}
A_w6	{arrival-airport-code, hotel-address, enabled_PT}	{enabled_PT}	{bus-return-ticket, done_PT}
A_w7	{arrival-airport-code, hotel-address, enabled_ST}	{enabled_ST}	{done_ST}
A_w8	{conf, dep-date, return-date, research-id, enabled_ST}	{enabled_ST}	{tel-soctravel, done_ST}
A_w9	{travel-country, freedate}	{}	{dep-date, return-date}
A_w10	{travel-country, freedate}	{}	{conf, dep-date, return-date}
A_w11	{home-city}	{}	{dep-airport-code}
A_w12	{identity, passport, nationality, travel-country, enabled_VI}	{enabled_VI}	{visa, done_VI}
A_w13	{travel-city}	{}	{arrival-airport-code}

6.2.3 Pré-sélection des données

Comme nous l'avons présenté dans la section 4.1.4, une propriété d'une donnée personnelle peut avoir plusieurs valeurs utilisables dans différents contextes. Exécuter une procédure dans un contexte donné consiste à utiliser uniquement un sous-ensemble de données personnelles enregistrées dans le PIS lors de la composition de services, celles susceptibles d'être utilisées dans le contexte visé. Pour cela, nous sélectionnons uniquement les instances de propriétés ayant un contexte similaire à celui demandé par l'utilisateur que nous notons $\underline{c}$. Cette similarité entre le contexte d'une instance de propriété et le contexte cible définit le degré d'utilisabilité de l'instance. De plus, nous gardons uniquement une seule instance

par propriété. Pour un contexte donné, il est possible d'avoir plusieurs instances, avec différents degrés d'utilisation. Nous sélectionnons la meilleure. Cette sélection permet de réduire le nombre de données candidates à l'utilisation et donc la taille du problème de composition. Cela permet aussi d'utiliser les données les plus pertinentes afin de produire des données résultantes de la meilleur qualité possible, sachant que les données produites lors de la composition sont éventuellement ajoutées au PIS de l'utilisateur. La sélection des meilleures informations lors de la composition favorise aussi la construction de solutions adaptées au contexte utilisé.

Les données de PIS sont pré-sélectionnées selon le contexte de référence du problème et en fonction d'un seuil minimal de pertinence prédéfini, les détails de cette pré-sélection sont donnés dans l'algorithme 5.

Algorithme 5 : Pré-sélection des données

Entrées :
PIS : l'ensemble des données personnelles de l'utilisateur
$\underline{c}$: le contexte de réalisation de la procédure
ϵ : le seuil minimal acceptable des degrés de pertinence des instances de propriétés
$\mathcal{O}^{\mathscr{C}}$: ontologie de contexte à considérer
Sorties : $\mathscr{D}$: l'ensemble des propositions contextuelles

```
1  début
2      D = ∅
3      pour chaque (p, v, c, δ, cat^Info) ∈ PIS faire
4          Δ^{O_C}_{c,c} ⟵ contextRel(O_C, c, c)
5          δ' ⟵ δ × Δ^{O_C}_{c,c}
6          si (δ' > ε) alors
7              δ'' ⟵ exists(D, p)
8              si (δ'' = −1) alors
9                  D ⟵ D ∪ (p, v, c, δ', cat^Info)
10             sinon si (δ' > δ'') alors
11                 remove(D, p)
12                 D ⟵ D ∪ (p, v, c, δ', cat^Info)
13 fin
```

L'algorithme 5 détaille les étapes de sélection des données personnelles à utiliser par les services, permettant ainsi de limiter le temps de la génération de la composition de service. Tout d'abord, pour chaque instance de propriétés $(p, v, c, \delta, cat^{Info})$ du PIS nous calculons un degré de pertinence $\Delta^{\mathcal{O}^{\mathscr{C}}}_{\underline{c},c}$ entre le contexte d'utilisation c et le contexte de référence de la procédure $\underline{c}$, en fonction de l'ontologie de contextes $\mathcal{O}^{\mathscr{C}}$. Le nouveau degré d'utilisation δ', relatif à l'utilisation de l'information dans le

TABLEAU 6.5: Pré-sélection des informations personnelles (tableau 6.2) selon les contextes *Leasure* et *Work*

(a) $\underline{c}$ = *Leasure*

Propriété	c	δ	$\Delta_{\underline{c},c}^{\mathscr{O}}$	δ' dans $\mathbf{PL}_0$
nationality	Top	1.0	1.0	1.0
food-prefs	Leasure	1.0	1.0	1.0
food-prefs	Work	1.0	0.0	0.0
hobbies-prefs	Leasure	0.5	1.0	0.5
identity	Top	1.0	1.0	1.0
passport	Top	1.0	1.0	1.0
passport	Top	1.0	1.0	1.0
freedate	Work	0.5	0.0	0.0
freedate	Leasure	1.0	1.0	1.0
research-id	Work	1.0	0.0	0.0
home-city	Top	1.0	1.0	1.0

(b) $\underline{c}$ = *Work*

Propriété	c	δ	$\Delta_{\underline{c},c}^{\mathscr{O}}$	δ' dans $\mathbf{PL}_0$
nationality	Top	1.0	1.0	1.0
food-prefs	Leasure	1.0	0.0	0.0
food-prefs	Work	1.0	1.0	1.0
hobbies-prefs	Leasure	1.0	0.0	0.0
identity	Top	1.0	1.0	1.0
passport	Top	1.0	1.0	1.0
passport	Top	1.0	1.0	1.0
freedate	Work	0.5	1.0	0.5
freedate	Leasure	1.0	0.0	0.0
research-id	Work	1.0	1.0	1.0
home-city	Top	1.0	1.0	1.0

contexte $\underline{c}$, est ainsi calculé en multipliant le degré d'utilisation de base (dans c) de l'instance, δ, par le degré de pertinence $\Delta_{\underline{c},c}^{\mathscr{O}}$ calculé entre les deux contextes, $\underline{c}$ et c. Nous comparons ensuite ce degré avec le seuil minimal acceptable, ϵ. Si ce degré est supérieur au seuil et qu'il n'existe aucune instance de cette propriété (*i.e.* $\delta'' = -1$) dans l'ensemble de propositions contextuelles $\mathscr{D}$, un tuple composé de la propriété p, du contexte c et du nouveau degré d'utilisation δ' est ajouté à $\mathscr{D}$. C'est ce dernier qui sera utilisé lors de la composition. En revanche, s'il existe déjà une instance correspondante dans $\mathscr{D}$ (*i.e.* $\delta'' \in [0..1]$), nous comparons alors le degré de pertinence de l'ancienne instance δ'' avec celui récemment calculé δ'. Si la nouvelle instance a un meilleur degré de pertinence que l'ancienne alors celle-ci la remplacera dans l'ensemble de données contextuelles candidates à la composition.

Exemple 6.8. *Nous appliquons l'algorithme de sélection des données personnelles dans le contexte "Leasure" et "Work". Les résultats sont donnés dans le tableau 6.5.*

Sachant que le seuil à considérer est de 0, nous gardons uniquement les propriétés ayant un degré δ' non nul pour participer à la constitution des données initiales du problème de planification.

6.2.4 Pré-sélection des services

Dans la section 6.1.4, nous avons présenté les politiques de sécurité que l'utilisateur spécifie en fonction des deux ontologies de catégories, les catégories de données $\mathcal{O}_{\text{Info}}$ et les catégories de services $\mathcal{O}_{\text{Serv}}$. Nous dénotons par Pol, l'ensemble des règles de sécurité directement spécifiées par l'utilisateur ainsi que celles inférées à partir de ces dernières en exploitant les axiomes de subsomption entre les deux ontologies.

Les politiques de sécurité sont utilisées pour restreindre les services susceptibles d'être utilisés par le moteur de composition. Un tel service est celui dont la catégorie est autorisée à accéder à l'ensemble des catégories de ses données d'entrée. Par exemple, un service $s1$ de catégorie *cat-S1* recevant en entrée les données a et b n'est sélectionné que s'il existe des règles de sécurité indiquant que la catégorie de a et la catégorie de b sont autorisées à être utilisées par *cat-S1*.

Le déroulement de la sélection des services en fonction des politiques de sécurité est détaillé dans l'algorithme 6.

Algorithme 6 : Pré-sélection des services

Entrées :
Pol : l'ensemble des politiques de sécurité
W : l'ensemble des services disponibles
$\mathscr{D}$: les données contextuelles à utiliser

Sorties : W' : l'ensemble de services autorisés à être utilisés

```
1 début
2     W' = ∅
3     pour chaque w = (u, I, O, K, C) ∈ W faire
4         si ∀i ∈ I, ∃ catégorie(i) ▷ C ∈ Pol alors
5             W' ⟵ W' ∪ {w}
6 fin
```

Cette étape de pré-sélection de services favorise la réduction de l'espace mémoire et du temps de calcul des compositions en limitant le nombre de combinaisons possibles lors de la création du graphe de planification (section 6.3). Elle assure ainsi que seuls les services respectant les contraintes de l'utilisateur seront utilisables.

6.2.5 Encodage global

Nous avons fait le choix de résoudre nos problèmes de composition par encodage en problème de planification (section 2.3.1).

Notre problème de composition $\mathscr{P} = \langle Proc, \underline{c}, \epsilon, W, \text{PIS}, Pol \rangle$, définition 6.5, est encodé en un problème de planification $\langle \mathbf{I}, \mathbf{G}, \mathbf{A} \rangle$ où :

— $\mathbf{I}$ est l'état initial composé de l'ensemble de données contextuelles présélectionnées et de la proposition initiale $\sharp$ résultant de l'encodage de la procédure : $\mathbf{I} = \mathscr{D} \cup \{\langle \sharp, \underline{c}, 1.0 \rangle\}$

— $\mathbf{G}$ est l'état final (ou le but) qui correspond à la proposition $\surd$ résultant de l'encodage de la procédure : $\mathbf{G} = \{\langle \surd, \_, \_ \rangle\}$

— $\mathbf{A}$ est l'ensemble des actions qui correspondent à l'union de l'ensemble des actions résultant de l'encodage de la procédure (section 6.2.1) et de l'ensemble

de celles résultant de l'encodage des services pré-sélectionnés (section 6.2.2 et algorithme 6)

6.3 Résolution du problème de composition

Pour la résolution du problème de composition nous avons choisi de nous baser sur *GraphPlan* pour les raisons présentées précédemment (section 6.2). Nous avons dans cette même section, présenté l'encodage de nos problèmes de planification.

Ce type de problème est résolu en appliquant un algorithme de planification. L'approche que nous proposons étend l'algorithme *GraphPlan* pour la prise en compte de données (propositions) contextualisées.

6.3.1 Algorithme

Une fois l'encodage de la procédure et des services en actions de planification effectué, nous pouvons appliquer les deux étapes du déroulement de l'algorithme de *GraphPlan* : la construction du graphe de planification et la recherche par *backtracking*. La seconde étape est identique à celle de l'algorithme classique [BF97]. En revanche, en raison de la contextualisation des données, la première étape doit être modifiée pour gérer la propagation des contextes et des degrés de pertinence tout au long de la construction du graphe.

Cette nouvelle version de construction du graphe de planification est réalisée par l'algorithme 7. Il peut être détaillé comme suit.

Initialisation du premier niveau – $\mathbf{PL}_1$. Les niveaux de propositions vont contenir des propositions contextualisées, c'est-à-dire des tuples (p, c, δ). Le premier niveau de propositions doit contenir la proposition initiale issue de l'encodage de la procédure, $\sharp$ (pour présenter la possibilité de commencer dans l'état initial de cette dernière), ainsi que les propositions correspondant à la pré-sélection des données du PIS. Ces propositions sont déjà contextualisées (section 6.2.3). Nous rappelons que nous conservons une unique instance de ces données (la meilleure pour le contexte de référence, $\underline{c}$). Dans le cas où deux instances auraient le même degré de pertinence, nous faisons un choix aléatoire.

Exemple 6.9. *En utilisant le contexte Leasure, le premier niveau de propositions* $\mathbf{PL}_1$ *contient les propositions contextuelles suivantes :*
{(identity, Top, 1.0), (nationality, Top, 1.0), (home-city, Top, 1.0), (passport, Top, 1.0), (freedate, Leasure, 1.0), (hobbies-pres, Leasure, 0.5), (food-prefs, Leasure, 1.0), (♯, Leasure, 1.0) }

Algorithme 7 : Construction du graphe de planification étendu

Entrées :
A : ensemble d'actions,
I : propositions initiales,
G : proposition objectif,
$\mathcal{O}^{\mathscr{C}}$: l'ontologie de contexte à considérer,

début
  $currentLayer \longleftarrow 1$
  ; /* Initialisation $\mathbf{PL}_1$ */
  $\mathbf{PL}_{currentLayer} \longleftarrow contextualisation(\mathbf{I})$
  répéter
    $\mathbf{AL}_{currentLayer} \longleftarrow \emptyset$
    ; /* Construction $\mathbf{AL}_{currentLayer}$ */
    pour chaque $\mathbf{a} \in \mathbf{A}$ **faire**
      si $isEnabled(\mathbf{a})$ **alors**
        si $\mathbf{Pre}(\mathbf{a}) \underset{\sim}{\subseteq} \mathbf{PL}_{CurrentLayer}$ **alors**
          $\mathbf{AL}_{currentLayer} \longleftarrow \mathbf{AL}_{currentLayer} \cup \{\mathbf{a}\}$
    ; /* Construction $\mathbf{PL}_{currentLayer+1}$ */
    $currentLayer \longleftarrow currentLayer + 1$
    $\mathbf{PL}_{currentLayer} \longleftarrow \mathbf{PL}_{currentLayer-1}$
    pour chaque $\mathbf{a} \in \mathbf{AL}_{currentLayer-1}$ **faire**
      $\mathbf{PL}_{currentLayer} \longleftarrow \mathbf{PL}_{currentLayer} \cup convert2CP(\mathbf{Eff}^{+}(\mathbf{a}), \mathcal{O}^{\mathscr{C}})$
      $\mathbf{PL}_{currentLayer} \longleftarrow \mathbf{PL}_{currentLayer} - \mathbf{Eff}^{-}(\mathbf{a})$
    $redundancyClear(\mathbf{PL}_{currentLayer})$
    ; /* Terminaison de la construction */
  jusqu'à $(\mathbf{PL}_{currentLayer-1} \cong \mathbf{PL}_{currentLayer})$ *ou* $(\mathbf{PL}_{currentLayer} \underset{\sim}{\subseteq} \mathbf{G})$;
fin

Si le contexte de référence était le contexte Professional, le niveau de propositions initiales $\mathbf{PL}_1$ *serait constitué comme ceci :*
{(identity, Top, 1.0), (nationality, Top, 1.0), (home-city, Top, 1.0), (passport, Top, 1.0), (freedate, Work, 0.33), (hobbies-prefs, Work, 0.66), (food-prefs, Work, 0.66), (♯, Professional, 1.0) }

Le fait d'avoir deux instances d'une même propriété ayant le même score de pertinence peut correspondre à un défaut de mise à jour des données personnelles dans le PIS de l'utilisateur, par exemple deux passeports, le premier expire en 2018 et le second a expiré en 2011. Ces deux passeports sont utilisables dans les mêmes contextes, et de ce fait ils ont le même score d'utilisabilité. Pour remédier à cela, un ensemble de critères de qualité pourrait être utilisées ([HKPS13], section 3). Le score

de chaque propriété serait ainsi calculé non seulement en fonction du contexte de référence mais aussi en fonction des critères de qualité. Ceci pourrait alors produire deux instances avec deux degrés différents favorisant ainsi la sélection de la meilleure instance pour le niveau de propositions initial $\mathbf{PL}_1$. Dans notre exemple, l'instance de passeport valide jusqu'en 2018 sera sélectionnée. Cette approche ne remet pas en cause la suite de notre approche car nous aurons aussi à gérer des tuples $\langle p, c, \delta \rangle$.

Construction d'un niveau d'actions – $\mathbf{AL}_i$. Chaque niveau d'actions contient l'ensemble des actions dont les préconditions sont satisfaites, c'est-à-dire disponibles dans le niveau de propositions courant. Il faut cependant prendre en compte le fait que les propositions dans les niveaux d'actions sont contextualisées ($\langle p, c, \delta \rangle$), ce qui n'est pas le cas des préconditions (P). Nous définissons donc l'appartenance modulo contextualisation par $P \underset{\sim}{\in} \mathbf{PL}_i$ *ssi* $\exists \langle p, c, \delta \rangle \in \mathbf{PL}_i$. De même, pour l'inclusion $P \underset{\sim}{\subset} \mathbf{PL}_i$ ssi $\forall p \in P,\ p \underset{\sim}{\in} \mathbf{PL}_i$

La condition pour qu'une action $\mathbf{a}$ puisse être mise dans le niveau d'actions $\mathbf{AL}_i$ est alors : $\mathbf{a} \in \mathbf{AL}_i$ ssi $\mathbf{Pre}(\mathbf{a}) \underset{\sim}{\subset} \mathbf{PL}_i$

Une action $\mathbf{a}$ correspondant à un service w est ajoutée à un niveau d'actions si et seulement si w est autorisé à utiliser ses attributs en entrée. Ceci est assuré par l'étape de pré-sélection en exploitant les politiques de sécurité (voir la section 6.2.4).

Exemple 6.10. *Dans le cadre de notre exemple les actions applicables dans le niveau* $\mathbf{AL}_1$ *sont A_TA et A_w11.*

- *A_TA (tableau 6.3) est la capacité initiale de la procédure, elle est donc possible initialement et elle est liée au niveau* $\mathbf{PL}_1$ *à la proposition contextuelle (♯, Leasure, 1.0).*
- *A_w11 (tableau 6.4) est un service sans capacité. Il n'est donc pas contraint par la procédure et est possible si ses entrées sont disponibles, ce qui est le cas. A_w11 est donc liée à la proposition contextuelle (home-city, Top, 1.0) du niveau* $\mathbf{PL}_1$.

Construction d'un niveau de propositions – $\mathbf{PL}_{i+1}$. Le niveau de propositions $\mathbf{PL}_{i+1}$ est calculé en fonction du niveau des propositions et du niveau d'actions précédent, $\mathbf{PL}_i$ et $\mathbf{AL}_i$. Toutefois, comme nous l'avons vu plus haut pour les préconditions, les actions sont décrites par des propositions simples. Tandis que, les niveaux de propositions sont constitués de propositions contextuelles. Il convient donc de contextualiser automatiquement les proposition générées par les actions.

Pour transformer une proposition simple en une proposition contextuelle, nous associons à chaque proposition simple un contexte et un degré (fonction *converst2CP*). Le contexte est calculé en fonction des contextes de propositions

associés aux préconditions de l'action et le degré représente la moyenne des degrés de pertinence des propositions associées aux préconditions de l'action.

Étant donnée une ontologie de contextes $\mathcal{O}^{\mathscr{C}}$ et une action $\mathbf{a}$ au niveau actions $\mathbf{AL}_i$ dont les préconditions correspondent à l'ensemble des propositions contextuelles suivantes $\{(p_1, c_1, \delta_1), \ldots, (p_n, c_n, \delta_n)\}$, chaque effet positif de $\mathbf{a}$ est associé à un contexte $c(\mathbf{a})$ et à un degré d'utilisabilité $\delta(\mathbf{a})$ calculé comme suit :

— $c(\mathbf{a}) = LCS(\{c_1, \ldots, c_n\}, \mathcal{O}^{\mathscr{C}})$ avec LCS est la fonction calculant le subsumant commun le plus spécifique entre $\{c_1, \ldots, c_n\}$,

— $\delta(\mathbf{a}) = \frac{\sum_{j=1}^{n} \delta_j}{n}$.

La construction du niveau de propositions $\mathbf{PL}_{i+1}$ est effectuée en trois étapes :

— la première étape consiste à calculer l'union des propositions contextuelles du niveau de propositions précédent, $\mathbf{PL}_i$, et des propositions contextuelles correspondant aux effets positifs de toutes les actions au niveau d'actions $\mathbf{AL}_i$,

— la deuxième étape consiste à retirer du niveau $\mathbf{PL}_{i+1}$ les propositions contextuelles correspondant aux effets négatifs des actions du niveau $\mathbf{AL}_i$,

— la dernière étape consiste à garder une unique instance de chaque propriété présente dans le niveau de propositions $\mathbf{PL}_{i+1}$. Ceci est effectué en utilisant la fonction de filtrage *redundancyClear* qui permet de garder l'instance de propriété ayant le meilleur degré de pertinence.

Exemple 6.11. *Le niveau de proposition* $\mathbf{PL}_1$*, dans le cas où le contexte utilisé est Leasure est ainsi composé des propositions suivantes :*
{ (identity, Top, 1.0), (nationality, Top, 1.0), (home-city, Top, 1.0), (passport, Top, 1.0), (freedate, Leasure, 1.0), (hobbies-pres, Leasure, 1.0), (food-prefs, Leasure, 1.0), (dep-airport-code, Top, 1.0), (enabled_ TA, Leasure, 1.0), (link_ TA, Leasure, 1.0) }

Terminaison de la construction. La construction du graphe de planification s'arrête soit par un succès (l'objectif est atteint) soit par un échec (l'objectif n'est pas atteignable).

— Dans le cas du succès, il est possible d'utiliser les critères de *GraphPlan*. Le but est inclus dans le niveau de propositions courant. Il convient juste de faire abstraction de la contextualisation : $\mathbf{G} \subset \mathbf{PL}_i$. Ceci inclut la proposition $\mathbf{G}$.

— Dans le cas de l'échec (point-fixe, $\mathbf{PL}_i \cong \mathbf{PL}_{i-1}$), il convient aussi de supporter la contextualisation. Pour cela nous considérons qu'un tuple $\langle p, \_, \delta_i \rangle$, $\delta_i \geq \epsilon$, est une nouvelle proposition s'il n'existe aucun tuple $\langle p, \_, \_ \rangle \in \mathbf{PL}_{i-1}$ ou s'il n'existe pas de tuple $\langle p, \_, \delta_{i-1} \rangle$ tel que $\delta_{i-1} \geq \delta_i$

Recherche de la solution. La recherche de solution est faite en appliquant l'algorithme de recherche *backtrack* habituel du *GraphPlan*. Le principe de cet algorithme

de recherche permet de construire un chemin d'exécution commençant par le but et permettant de parcourir les états qui peuvent conduire à ce but, jusqu'à retrouver l'état initial. Dans notre cas, l'algorithme de recherche commence par l'objectif **G** et s'arrête lorsque toutes les propositions à atteindre appartiennent à l'ensemble de propositions initiales **I**. Les détails de cet algorithme sont donnés en algorithme 8. Nous obtenons la solution en appelant *backtrack(***G***, i-1, GP, <>)*.

Fonction `backtrack`(**G**, *i*, *GP*, π)

```
1 début
2   si i = 0 alors
3     retourner π
4   sinon
5     A ⟵ ensembleMinimalAction(AL_i) tel que Eff⁺(A) ≅ G
6     π' ⟵ A;π
7     G' ⟵ Pre(A)
8     backtrack(G', i − 1, GP, π')
9 fin
```

6.3.2 CoCliCo

Dans le but de tester et de vérifier notre approche de composition, nous avons développé un outil de composition, CoCliCo, dont l'architecture est donnée en figure 6.5. CoCliCo supporte l'utilisation de données contextuelles, de politiques d'accès à ces données et de procédures définies sous forme de *workflows*.

CoCliCo est implanté en Java. Il prend en entrée plusieurs fichiers. Le premier correspond à la description de la procédure, le deuxième contient la description des services, le troisième contient les politiques de sécurité définies par l'utilisateur et le dernier correspond aux données personnelles. L'outil prend également en compte le contexte de référence dans lequel la procédure sera appliquée et le seuil minimal d'utilisation. CoCliCo est paramétré par plusieurs ontologies : l'ontologie de contextes, l'ontologie de catégories d'informations et l'ontologie de catégories de services.

Le fonctionnement de CoCliCo est le suivant. Tout d'abord, CoCliCo parse les différents fichiers (resp. procédure, e-services, politiques de sécurité et PIS) et les stocke sous la forme de modèles correspondant à chaque type de fichier (resp. modèle de procédure, modèle de service, modèle de sécurité et modèle de données). Par la suite, il exécute les deux fonctions de sélection decrites en section 6.2.2 et en section 6.2.3. La première (section 6.2.2) est appliquée sur l'ensemble des services en exploitant l'ensemble *Pol* qui a été calculé en utilisant les politiques

de sécurité de l'utilisateur, l'ontologie de catégories d'informations et l'ontologie de catégories de services. Cette étape de sélection permet d'obtenir le *modèle de services optimisé* qui représente l'ensemble des services autorisés à être utilisé dans la composition de services. La seconde fonction de sélection (section 6.2.3) est appliquée sur les données personnelles. Elle utilise le contexte de référence, le seuil minimal et l'ontologie de contextes. Cette étape de sélection permet d'obtenir un *modèle de données optimisé* contenant une seule instance pour chaque propriété. Le modèle de procédure, le modèle de services optimisé et le modèle de données optimisé sont ensuite encodés afin d'obtenir des parties du problème de planification que le système fusionne par la suite pour construire le *modèle du problème global*.

Le modèle du problème global correspond à un problème de planification qui est par la suite traité par notre moteur de composition. La première phase consiste en la construction du graphe de planification, et la seconde consiste en l'extraction de la solution. Cette solution, si elle existe, sera présentée à l'utilisateur sous la forme d'un workflow d'actions.

La version actuelle de CoCliCo utilise des fichiers texte et qui peuvent être facilement remplacés par d'autres types de fichiers afin de décrire la procédure, les

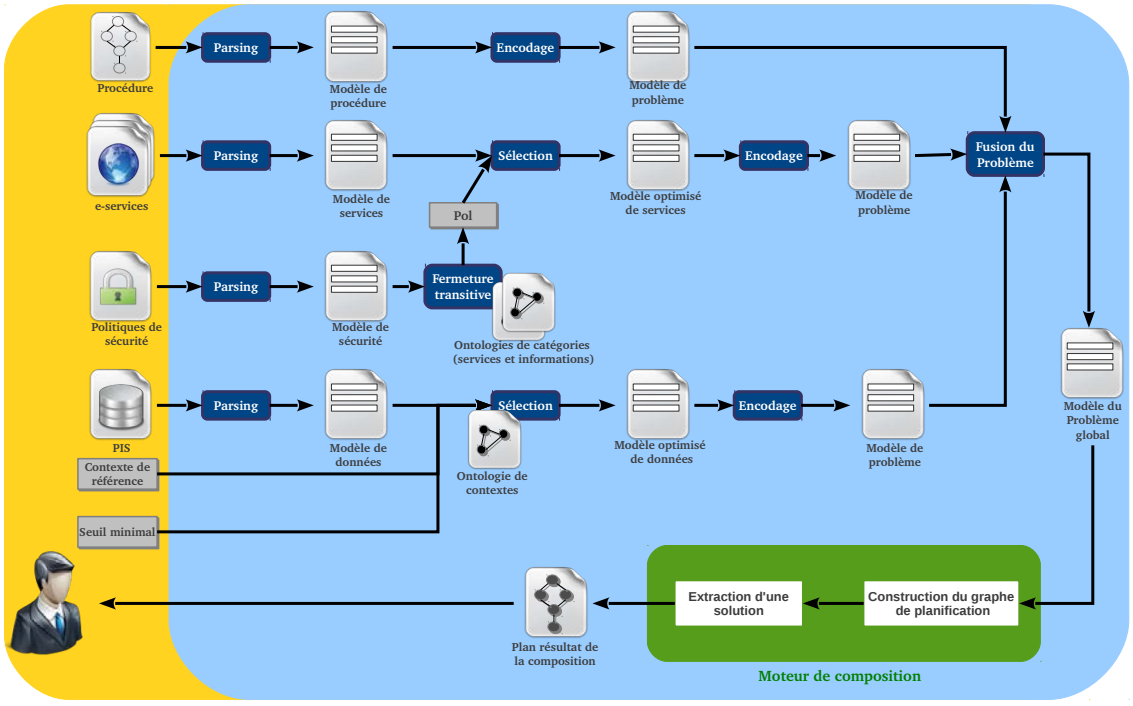


FIGURE 6.5: Architecture du système CoCliCo

politiques de sécurité et les services. Il suffit d'implanter l'analyse syntaxique (*parsing*) le *parser* vers le modèle correspondant. Les ontologies peuvent être remplacées par d'autres qui sont utilisées lors de la description des données contextuelles et le contexte de référence.

6.4 Expérimentations

Dans les précédentes sections de ce chapitre, nous avons présenté notre approche de composition et l'outil associé, CoCliCo. Nous présentons dans cette section l'évaluation qualitative et quantitative de notre approche. L'objectif de cette évaluation est de montrer d'une part la faisabilité de notre approche et d'autre part que l'exploitation des contextes et de leurs degrés n'altère pas l'efficacité de l'approche. De plus, nous avons souhaité montrer que l'utilisation des contextes permet d'augmenter la pertinence des solutions (*i.e.* compositions) trouvées. Pour ce faire deux évaluations ont été réalisées. La première consiste à évaluer notre approche qualitativement à travers une étude de cas. La seconde consiste à l'évaluer quantitativement sur un ensemble de problèmes générés aléatoirement.

6.4.1 Étude qualitative

Cette étude qualitative a pour but de montrer le rôle important des différents éléments du problème de composition. Nous nous intéressons au contexte et à l'exploitation des degrés d'utilisabilité.

Contexte. Nous avons vu dans les chapitres précédents que les contextes jouent un rôle important dans la représentation et l'interrogation des informations personnelles. Ils sont également importants lors de la composition de services en permettant de fournir des solutions plus pertinentes. Pour cela, nous utilisons notre étude de

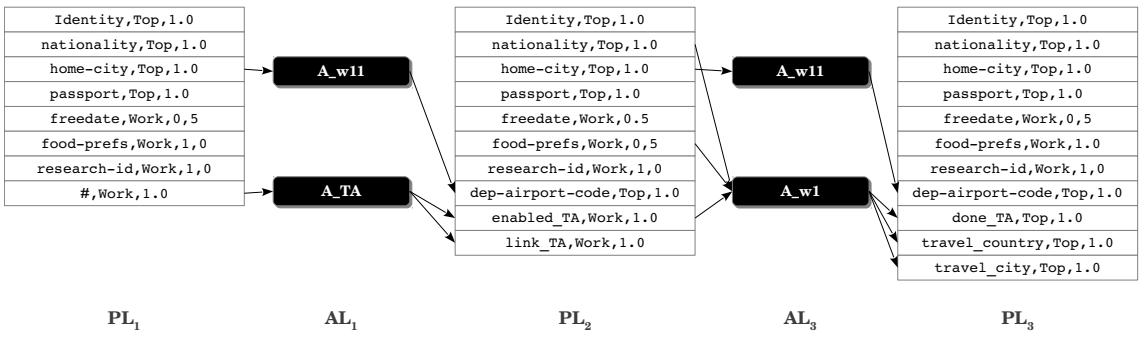


FIGURE 6.6: Extrait du graphe de planification de l'exemple 6.5 (contexte *Work*)

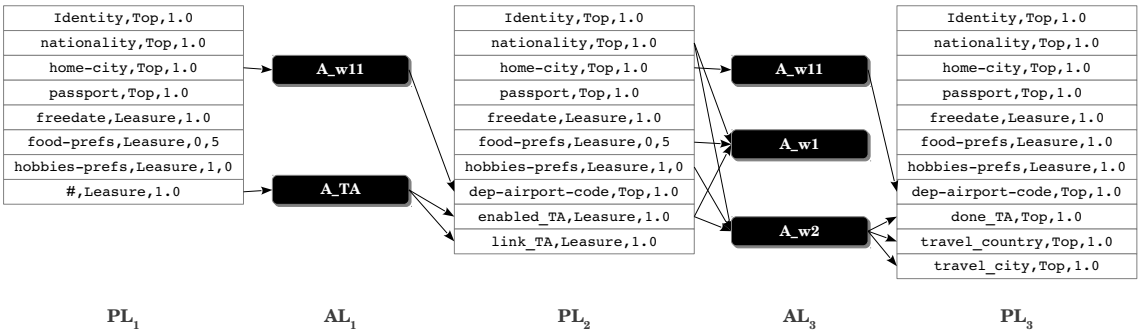


FIGURE 6.7: Extrait du graphe de planification de l'exemple 6.5 (contexte *Leasure*)

cas dans les contextes *Leasure* et *Work*. Un extrait des graphes de planification correspondants est donné respectivement dans les figures 6.7 et 6.6. En utilisant le contexte *Leasure*, les actions sélectionnées pour le deuxième niveau d'actions sont $\mathbf{A\_w_1}$ et $\mathbf{A\_w_2}$. Ces deux actions permettent de réaliser la capacité *Travel_Advice (TA)*. Toutefois, dans le cas où le contexte de la procédure est *Work*, seule l'action $\mathbf{A\_w_1}$ sera sélectionnée. Ceci est dû, d'une part, à l'absence de valeur pour la propriété *hobbies_prefs* dans le contexte *Work* et, d'autre part, à la disjonction des deux contextes *Work* et *Leasure* dans l'ontologie de contextes considérée.

Degré d'utilisabilité. L'impact de l'exploitation des degrés d'utilisabilité des informations personnelles peut être étudié en analysant les graphes de planification générés par notre approche. Nous constatons que dans le cas où le contexte de la procédure est *Leasure*, le niveau d'actions $\mathbf{AL}_2$ contient deux actions effectuant la capacité *Travel_Advice*. Ces deux actions donnent comme effets positifs les mêmes propositions, à savoir *travel_country* et *travel_city*. Dans le graphe de planification ces propositions sont données en fonction d'un contexte et d'un degré d'utilisabilité. Ces derniers sont calculés en fonction des degrés d'utilisabilité des préconditions de chaque action. L'action $\mathbf{A\_w_1}$ a comme précondition les propositions contextuelles (*nationality,Top,1*) et (*foof_prefs, Leasure, 0.5*). Ses effets positifs auront donc le contexte *Top* (*i.e.* LCS(*Leasure,Top*)) et le degré 0.75. Cependant l'action $\mathbf{A\_w_2}$ qui a comme préconditions les propositions contextuelles (*nationality, Top, 1*) et (*hobbies_prefs, Leasure, 1*) donne le contexte *Top* et le degré 1 pour les propositions correspondant à ses effets positifs. Vu que la construction de notre graphe de planification s'appuie sur l'utilisation des meilleurs scores d'utilisabilité, les plans finaux se différencient ainsi selon le contexte utilisé, notamment la procédure réalisée dans le contexte *Leasure* va contenir l'action $\mathbf{A\_w_2}$ et celle réalisée dans le contexte *Work* va contenir l'action $\mathbf{A\_w_1}$.

6.4.2 Étude quantitative

Pour réaliser l'étude quantitative, nous avons eu besoin de générer des jeux de données représentant notre problème. En dépit de l'existence d'outils de génération de jeux de données dans les compétitions des services Web [SWS], il n'existe pas d'outils ou de *benchmarks* correspondant exactement à notre problème, à savoir l'utilisation d'un workflow et l'utilisation de données contextuelles. Dans le but d'évaluer notre approche, nous avons donc développé un *plug-in* Eclipse spécifique permettant la génération de problèmes de composition, CoCliCo-gen.

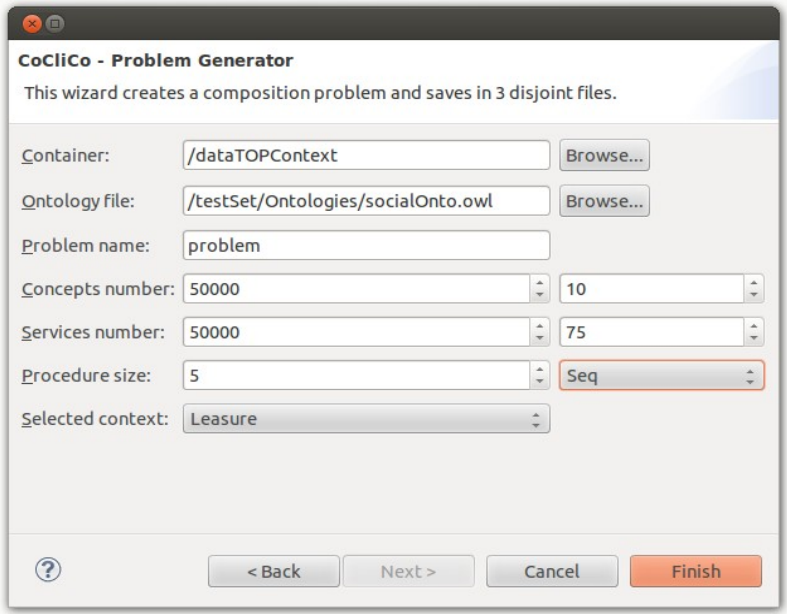


FIGURE 6.8: Interface de l'outil de génération de *benchmarks*, CoCliCo-gen

Notre outil, dont l'interface est donnée en figure 6.8, prend en paramètres un répertoire où les fichiers générés vont être enregistrés (*Container*), une ontologie de contextes (*Ontology file*), un nom de problème qui va servir à différencier les problèmes générés, un nombre de concepts à utiliser et le pourcentage de ceux à contextualiser, un nombre de services et le pourcentage de ceux avec capacité et, un nombre de capacités ainsi que la nature de *Workflow* à utiliser (séquentiel, parallèle, choix). Cet outil permet de générer trois fichiers préfixés par le nom du problème : le premier concerne les données contextuelles, le deuxième concerne l'ensemble de services disponibles et le dernier correspond à la procédure à considérer. Tous les problèmes générés par notre outil possèdent au moins une solution.

La taille de la procédure spécifiée lors de la génération d'un problème (*Procedure size*) désigne le nombre de capacités à utiliser. Les *workflow* que nous utilisons dans

notre approche sont bien équilibrés et sans boucles (section 6.1.1). Par exemple, étant donnée une procédure séquentielle de taille 4, elle est composée de $K0$ suivie de $K1$ suivie de $K2$ et enfin, suivie de $K3$. Le résultat de la génération est un fichier texte décrivant la procédure. La génération de la procédure pourrait être effectuée avec un autre type de fichier par exemple du BPEL en utilisant une transformation de modèles et une traduction modèle-texte. L'exemple d'une procédure séquentielle de quatre capacités est donné comme suit :

```
#A, K0/K0, I
#A, K1/K1
#A, K2/K2
#A, K3/K3, F
@
#K0,K1
#K1,K2
#K2,K3
```

Le nombre de propriétés spécifiées dans l'interface représente les propriétés à utiliser dans la description du problème à savoir les données contextuelles et les attributs utilisés pour la description des services. Un sous-ensemble de ces propriétés va être contextualisé en fonction de l'ensemble de contextes définis dans l'ontologie de contextes et du pourcentage de contextualisation sépécifié. Dans l'ensemble des problèmes générés, la moitié des concepts sont contextualisés. Comme nous l'avons présenté en chapitre 5, chaque donnée peut être instanciée avec plusieurs valeurs. Pour ce faire, notre outil permet de choisir aléatoirement pour chaque propriété un nombre d'instances. Chaque instance est ainsi associée à un contexte aléatoirement sélectionné parmi les contextes de l'ontologie et un degré d'utilisation. Ce dernier est calculé par rapport au nombre d'instances de la propriété et à son rang : le degré de la $i^{ème}$ instance est égal à $\frac{1}{i}$. Ci-dessous nous donnons un extrait d'un fichier de données contextuelles où la propriété data0 est instanciée deux fois et où la propriété data1 est instanciée quatre fois :

```
#data0,val0,Finance,1.0
#data0,val1,Research,0.5
#data1,val2,Family,1.0
#data1,val3,Professional,0.5
#data1,val4,Leasure,0.3333333333333333
#data1,val5,Work,0.25
#data2,val6,Private,1.0
```

Notre outil permet aussi de générer des données pour un unique contexte. Pour ce faire, il faut sélectionner un contexte dans la liste de *Selected context*. Par exemple, si nous choisissons de sélectionner le contexte « *Work* », l'ensemble de données contextuelles vont avoir ce contexte et le degré d'utilisation 1.0. Un extrait d'un tel fichier de données contextuelles est donné comme suit :

```
#data0,val0,Work,1.0
#data1,val1,Work,1.0
#data2,val2,Work,1.0
```

```
#data3,val3,Work,1.0
#data4,val4,Work,1.0
```

Concernant la génération des services, l'outil permet de créer deux types de services : avec ou sans capacités. Un nombre d'attributs en entrée et un nombre d'attributs en sortie sont calculés aléatoirement pour chaque service. Ces attributs vont être sélectionnés aléatoirement parmi les concepts du problème. De plus, notre outil attribue pour certains services les capacités utilisées dans la procédure. Un extrait du fichier contenant les services est donné ci-dessous :

nomService:entrées:sorties:capacités:catégorie

```
#serv0:data8,data13::data72:K0:catS1
#serv1:data82::data29,data6:K1:catS1
#serv2:data43,data39::data35,data6::catS1
#serv3:data29::data61::catS1
#serv4:data21::data88:K2:catS1
#serv5:data82,data17::data23:K3:catS1
```

Les différents fichiers générés par notre outil sont ensuite utilisés comme entrée de CoCliCo afin d'être évalués.

Dans la suite, nous présentons les résultats des évaluations que nous avons effectuées. La première évaluation concerne la capacité de passage à l'échelle et la seconde concerne la comparaison du temps de composition lors de l'utilisation des contextes et sans utilisation des contextes. Pour ce faire nous avons fait varier le nombre de services entre {10, 100, 1000, 10000, 100000}, le nombre de données entre {10, 100, 1000, 10000, 100000} et le nombre de capacités entre {5, 10, 15, 20, 25}.

6.4.2.1 Passage à l'échelle

Pour vérifier le passage à l'échelle de notre approche, nous avons générer trois ensembles de problèmes et nous avons opté pour l'utilisation de *workflow* séquentiel pour chaque problème.

Services. Le premier ensemble de test a permis d'étudier le comportement de notre approche en faisant varier le nombre de services disponibles pour chaque problème.

Dans la figure 6.9 (a), nous avons tracé une courbe pour chacune des valeurs de l'ensemble des données. Nous avons fixé le nombre de capacités (taille de la procédure) à 5 pour chacune de ces courbes. Nous remarquons que pour chacune de ces courbes, le temps de composition est quasi-linéaire en fonction du nombre de données et du nombre de services. De plus, le temps de composition d'un problème décrit par 100000 services, 100000 concepts et d'une procédure de taille 5 est de 87s.

Dans la figure 6.9 (b), nous avons tracé une courbe pour chacune des valeurs de l'ensemble des capacités. Nous avons fixé le nombre des données à 1000 pour

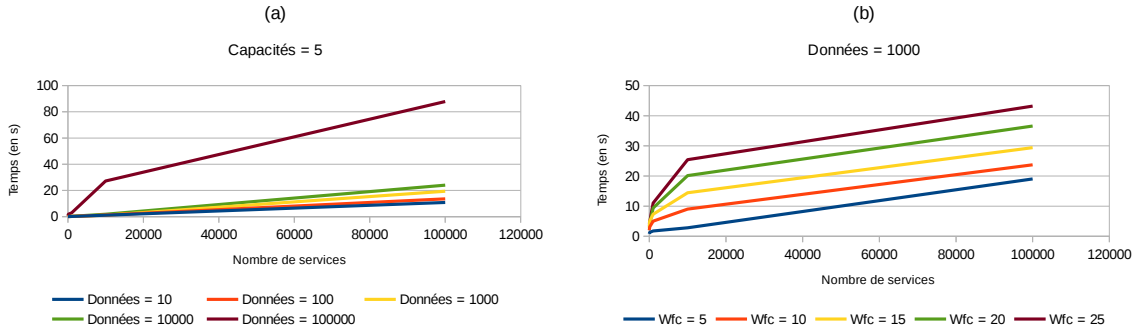


FIGURE 6.9: Temps de composition par rapport au nombre de services

chacune de ces courbes. Nous remarquons que le temps de composition est quasi-linéaire et il augmente proportionnellement au nombre de capacités. En outre, le temps de composition d'un problème décrit par 100000 services, 1000 concepts et d'une procédure de taille 25 est de 38s.

Données. Le deuxième ensemble de tests permet d'étudier le comportement de notre approche lorsque nous faisons varier le nombre de données de chaque problème. Les résultats sont donnés en figure 6.10.

Dans la figure 6.10 (a), nous avons tracé une courbe pour chacune des valeurs de l'ensemble de services. Nous avons fixé le nombre des capacités (taille de la procédure) à 5 pour chacune de ces courbes. Nous remarquons que le temps de composition augmente en fonction du nombre de services et de la taille de procédure considérée dans le problème.

Dans la figure 6.10 (b), nous avons tracé une courbe pour chacune des valeurs de l'ensemble des capacités. Nous avons fixé le nombre des services à 1000 pour chacune de ces courbes. Nous avons le même comportement que pour la précédente

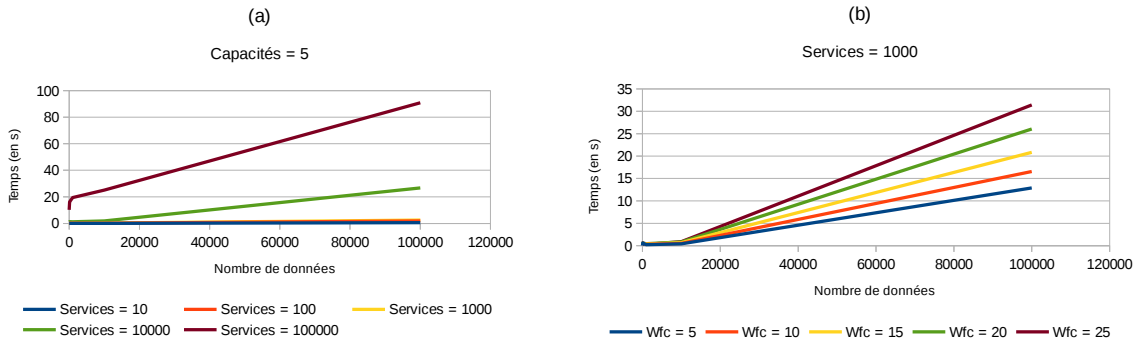


FIGURE 6.10: Temps de composition par rapport au nombre de données

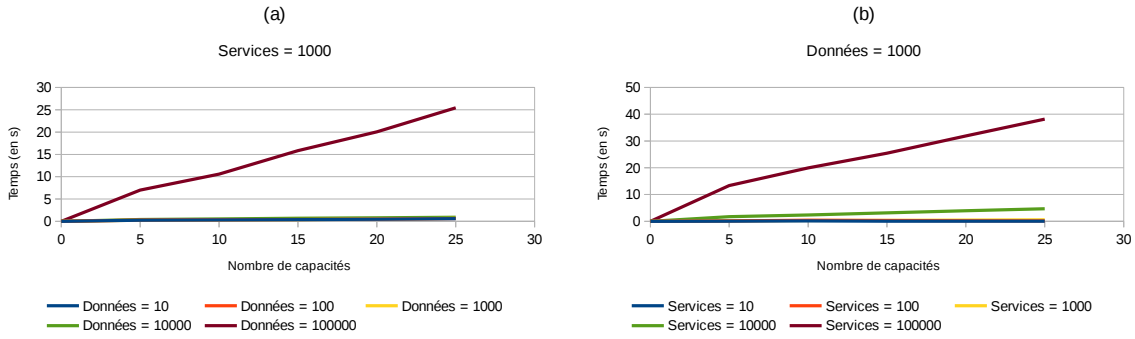


FIGURE 6.11: Temps de composition par rapport au nombre de capacités

évaluation, c'est-à-dire que le temps de composition est quasi-lineaire et il le temps augmente de façon proportionnelle au nombre de capacités.

Capacités. Le dernier ensemble de problèmes permet d'étudier le comportement de notre approche lorsque nous faisons varier le nombre de capacités de chaque problème. Les résultats sont donnés en figure 6.11.

Dans la figure 6.11 (a), nous avons tracé une courbe pour chacune des valeurs de l'ensemble de données. Nous avons fixé le nombre de services à 1000 pour chacune de ces courbes. Nous remarquons que le temps de composition augmente en fonction du nombre de données et du nombre de services considérés dans le problème.

Dans la figure 6.11 (b), nous avons tracé une courbe pour chacune des valeurs de l'ensemble de services. Nous avons fixé le nombre de données à 1000 pour chacune de ces courbes. Nous remarquons que pour une capacité donnée le temps de composition augmente avec le nombre de services. Cette augmentation reste proportionnelle au nombre de capacités.

Nous remarquons pour un nombre de données fixe (resp. un nombre de services fixe) le temps de la composition évolue linéairement. De plus, en faisant varier le nombre de données (resp. le nombre de services) considéré l'augmentation reste proportionnelle au nombre de capacités. Ceci prouve bien que notre approche passe à l'échelle.

6.4.2.2 Utilisation de contextes

Notre seconde expérimentation concerne la comparaison du temps de composition des problèmes avec et sans l'utilisation de contextes. Pour ce faire, nous avons générer deux ensembles de problèmes correspondant à ces deux cas. Ces problèmes ont été générés pour chaque ensemble de services (resp. ensemble de données et ensemble de capacités) et en fixant le nombre de données et le nombre de capacités (resp. en fixant le nombre de services et le nombre de capacités, en fixant le nombre

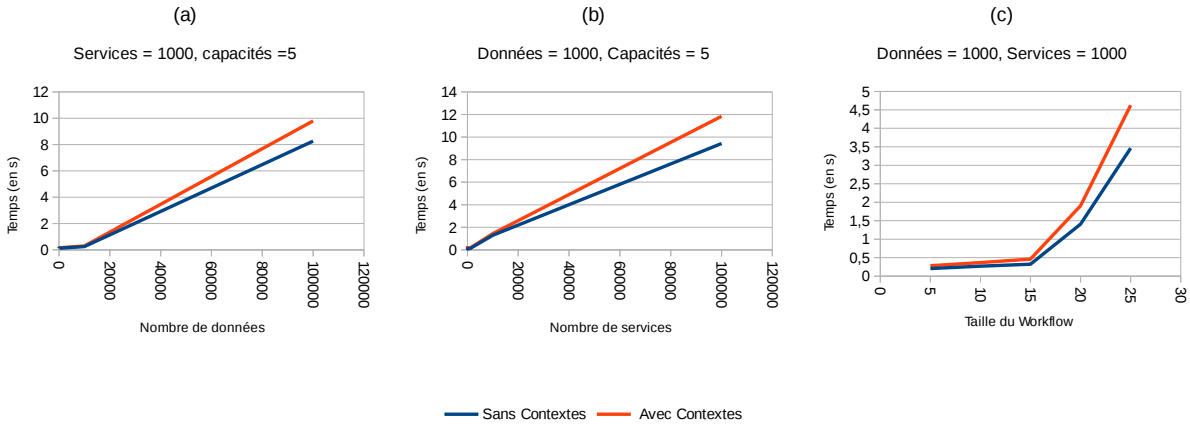


FIGURE 6.12: Évolution du temps de composition avec et sans utilisation de contextes

de services et le nombre de données). Dans ces trois cas, nous avons tracé une courbe pour chacun des deux problèmes (avec et sans contextes).

En utilisant ces ensembles comme entrées pour notre outil CoCliCo, nous avons obtenu les résultats présentés dans la figure 6.12.

La figure 6.12 (a) montre le temps de composition en fonction du nombre de services. Nous avons fixé le nombre de données à 1000 et le nombre de capacité à 5.

La figure 6.12 (b) montre le temps de composition en fonction du nombre de données. Nous avons fixé le nombre de services à 1000 et le nombre de capacité à 5.

La figure 6.12 (c) montre le temps de composition en fonction du nombre de capacités. Nous avons fixé le nombre de données et le nombre de services à 1000.

Ces trois figures montrent que le temps de composition pour les problèmes sans contextes est plus rapide que pour les problèmes avec contextes. Cependant la différence de temps n'est pas très importante et s'explique par le temps de calcul des similarités dans l'ontologie de contextes. Ces résultats permettent aussi de confirmer les expérimentations qualitatives sur l'utilisation des contextes.

Conclusion

Dans ce chapitre, nous avons présenté notre approche de composition, les éléments considérés dans notre problème, l'encodage de nos éléments en problème de planification et la résolution de ce problème en étendant l'algorithme de *GraphPlan* afin de prendre en compte la contextualisation des données et donc des propositions utilisées par la planification.

Notre approche est implantée grâce à CoCliCo. Elle permet de traiter des problèmes de composition expressifs tout en garantissant la terminaison et l'obten-

tion d'une solution en un temps raisonnable, si elle existe. Une autre contribution est le *plug-in* de génération de problèmes de composition avec des données contextuelles, CoCliCo-gen. Cet outil pourrait être utilisé par d'autres concepteurs d'algorithmes de composition.

Conclusion et Perspectives

Cette thèse porte sur la modélisation et l'utilisation d'informations personnelles à validité contextuelle. Plus particulièrement, elle a pour objectif le support de l'utilisateur quant à la réalisation de procédures en ligne, administratives ou personnelles. Dans ce contexte, les problématiques abordées sont celles de la représentation d'informations hétérogènes, l'interrogation d'espaces d'informations personnelles (PIS) contextualisés, le remplissage automatique de formulaires et la réalisation automatique de procédures définies à un haut niveau d'abstraction par composition de services disponibles en ligne.

Dans ce chapitre nous présentons tout d'abord une synthèse de nos contributions puis nous détaillerons les perspectives que nous avons dégagées pour nos travaux.

1 Synthèse des contributions

Dans cette section, nous résumons chacune des contributions que nous avons apporté pour la résolution des problématiques abordées dans cette thèse.

Modélisation contextuelle et multi-points de vue des informations personnelles. L'approche que nous avons proposée se caractérise par les originalités suivantes : (*i*) l'utilisation de plusieurs points de vue, (*ii*) la création personnalisée de modèles de représentation des informations personnelles et (*iii*) la contextualisation de ces dernières. Notre approche permet en effet à l'utilisateur de créer une ontologie personnelle (*PITO*) où il peut décrire ses informations personnelles sans être contraint par une représentation existante. Pour ce faire, nous avons utilisé de nombreuses ontologies de domaine que nous avons appelées *ontologies de points de vue*. Cette utilisation permet d'avoir un vocabulaire riche à la disposition de l'utilisateur. Une fois l'ontologie PITO créée, elle peut être instanciée sur les informations personnelles de l'utilisateur en leur associant un ensemble de contextes et de degrés d'utilisabilité. Pour tester la faisabilité de cette approche, nous avons développé un prototype CoPIMS permettant à la fois l'utilisation de points de vue existants pour la création de l'ontologie PITO pour la représentation des informations personnelles et l'instanciation de ces dernières selon une ontologie de contextes.

Interrogation contextuelle des informations personnelles. Pour exploiter les informations personnelles stockées dans le PIS, nous avons présenté deux approches. La première concerne le remplissage automatique de formulaires, en développant un algorithme qui génère une requête sémantique à partir d'un formulaire annoté par l'ontologie de l'utilisateur PITO. La seconde concerne le développement de deux algorithmes d'évaluation de requêtes contextuelles (*SQE*, *FQE*). Ces requêtes exploitent différentes contraintes sur les contextes d'utilisation des données : égalité, similarité, ou différence par rapport à un contexte donné. Les deux algorithmes exploitent une ontologie de contextes permettant ainsi d'ordonner les réponses à la requête en terme de degré de pertinence. Un contexte et une fonction (l'égalité, la similarité ou la disjonction) sont associés aux requêtes.

Composition de services sensible aux contextes. Notre dernière contribution concerne la réalisation d'un objectif utilisateur de haut niveau par composition de services. L'objectif utilisateur que nous avons considéré, est une procédure représentée sous la forme de *workflow* abstrait. Cette procédure doit pouvoir être réalisée dans un contexte particulier, précisé par l'utilisateur. Pour résoudre cette problématique, nous avons proposé une approche automatique de composition de services en exploitant les informations personnelles de l'utilisateur. Notre approche est une extension de l'algorithme *GraphPlan* pour gérer la contextualisation des informations personnelles. De plus, notre approche permet de prendre en compte des politiques de sécurité pour limiter le risque de divulgation des informations personnelles aux services en ligne et ainsi offrir à l'utilisateur les moyens de contrôler ses informations. Pour évaluer et valider nos contributions, nous avons implémenté un outil CoCliCo supportant notre approche. Cet outil a été évalué sur un ensemble de problèmes générés par un outil de génération de jeux de données. L'outil que nous avons implémenté a été inspiré de l'outil de génération de problèmes utilisé dans les compétitions sur la composition de services Web WSC (Web Services Challenge). Notre approche a prouvé son efficacité pour un nombre réaliste de données, de services et de capacités dans la procédure à réaliser.

2 Perspectives du travail

Dans cette section nous présentons les perspectives que nous envisageons de réaliser.

Extension du processus de composition de services. Nous envisageons de prendre en compte des critères de qualité de service (QoS) dans les descriptions de services et les besoins de l'utilisateur. Pour cela il est possible de se baser sur [YCY12]. Nous envisageons aussi de pouvoir prendre en compte des services

à états (state-full), c'est-à-dire des conversations pour ces services. Ici il est possible de se baser sur [PY10]. Nous souhaitons étendre l'expressivité de notre approche par la prise en compte des contextes dans la description des services et des procédures. Dans le premier cas, il s'agit de pouvoir spécifier des contraintes sur le contexte d'utilisation des services ou sur la validité des entrées et des sorties par rapport à un contexte. On peut ainsi envisager un service de demande de visa qui ne soit valable que dans un contexte professionnel. On peut aussi envisager qu'un tel service ne soit valide que si une entrée « *adresse personnelle* » est elle-même valide sur le territoire de la communauté européenne. Dans le second cas (procédures), il s'agirait de pouvoir équiper les procédures de contraintes contextuelles, en précisant ainsi que si telle capacité abstraite est réalisée en utilisant un service pour un contexte c, ou une donnée personnelle valide pour le contexte c, alors une autre capacité abstraite (plus en avant dans la procédure) doit aussi être réalisée par rapport à ce contexte.

Annotation des services. La composition de services permet de développer rapidement des applications en réutilisant des services existants. Ces services appartenant à de nombreux domaines d'application, sont souvent présentés avec différents niveaux d'hétérogénéité : physique et sémantique. L'hétérogénéité physique concerne les protocoles utilisés (***e.g.,*** REST, SOAP) et le type de fichiers (***e.g.,*** WSDL, OWL-S, WSMO). L'hétérogénéité sémantique concerne l'utilisation de différents vocabulaires pour la description des services. La problématique de l'hétérogénéité, freinant la composition automatique, doit être aussi considérée et traitée. Nous envisageons de nous focaliser sur l'hétérogénéité sémantique. Afin d'assembler les services, les techniques de composition supposent en général qu'ils sont sémantiquement décrits à l'aide d'un vocabulaire commun. Ce n'est toutefois pas le cas en pratique. Pour résoudre ce problème, une solution possible pourrait être l'annotation sémantique des services guidée par une ontologie.

Politiques de sécurité. Notre approche de composition se base sur une forme simple de politiques de sécurité. Ces dernières permettent de préciser quelles catégories de services sont autorisés à utiliser quelles catégories d'informations. Il s'agit en quelque sorte d'invariants (l'autorisation est toujours donnée, ou pas). Des politiques plus expressives pourraient être prises en compte par l'utilisation de logiques modales (ce qui inclut les logiques temporelles). Ainsi, l'utilisateur pourrait par exemple spécifier que si telle catégorie d'information est utilisée par un service pour réaliser une procédure alors telle autre ne le peut pas. Pour cela il est possible de se baser sur [MABS+12].

Traçabilité de la divulgation d'informations. L'utilisation des approches de composition de services dans le cadre de la gestion des informations personnelles pose le problème de la traçabilité des données. Les données utilisées par les services

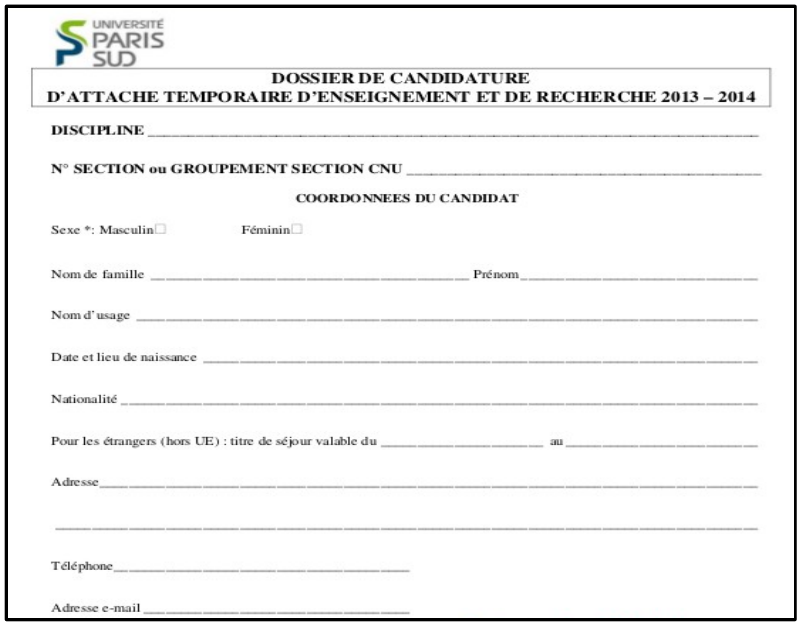
en ligne sont des informations personnelles. Il est important de permettre à l'utilisateur de suivre ses informations et de savoir quels sont les services qui les utilisent, quels sont les services qui les modifient. Nous envisageons de concevoir et de développer une approche de traçabilité des informations personnelles. Cette approche devra permettre d'identifier les services utilisant ces informations personnelles et la nature de cette utilisation. La méthode de traçabilité doit pouvoir fournir un suivi détaillé de toutes les informations (***e.g.,*** le nombre de fois qu'une information a été utilisée durant un laps de temps, les services l'ayant utilisés). En analysant les résultats de la méthode, l'utilisateur pourrait ainsi prendre des décisions par rapport à l'enrichissement, l'affinement de ses politiques de sécurité et à l'utilisation de ses informations. Par exemple il pourrait limiter l'utilisation de certaines informations afin de prévenir leur divulgation, ou interdire leur utilisation par certains services.

Modélisation et gestion de PIMS collaboratifs. Une autre perspective en relation avec le projet PIMI est de pouvoir proposer un modèle collaboratif des informations personnelles. Il s'agit d'un réseau social des PIMS de plusieurs personnes, offrant la possibilité de partager et d'utiliser les informations de ces dernières. Bien évidemment cette possibilité requiert non seulement de pouvoir modéliser le réseau social de l'utilisateur (graphe avec plusieurs types d'arcs selon les relations possibles : collègue, famille, etc) mais aussi d'adapter les politiques de sécurité en conséquence (x autorisant l'utilisateur par telle type de relation de telle type d'information).

Liage de services. L'idée est de construire un graphe en ligne des services Web existants et d'insérer tout nouveau service dans ce graphe. Pour ce faire, il faut créer des liens de *mappings* entre les attributs en entrée et les attributs en sortie de tout service. Ces *mappings* s'appuieraient sur le calcul de similarité entre les attributs. De plus, il serait intéressant de créer des relations sémantiques entre les différents services par exemple des liens d'équivalences entre deux services effectuant une même fonctionnalité, ou des liens de composition entre un service composite et les services atomiques qui le compose. Ce graphe permettrait ainsi de faciliter la sélection et la composition des services.

Exemples de Formulaires

Nous présentons dans cet annexe deux exemples de formulaires de candidature à un poste d'attaché temporaire d'enseignement (ATER) dans deux universités françaises : à université Paris Sud A.1 et à l'université Paris Dauphine A.2.



UNIVERSITÉ PARIS SUD

DOSSIER DE CANDIDATURE
D'ATTACHE TEMPORAIRE D'ENSEIGNEMENT ET DE RECHERCHE 2013 – 2014

DISCIPLINE _____

N° SECTION ou GROUPEMENT SECTION CNU _____

COORDONNEES DU CANDIDAT

Sexe *: Masculin☐ Féminin☐

Nom de famille _____ Prénom_____

Nom d'usage _____

Date et lieu de naissance _____

Nationalité _____

Pour les étrangers (hors UE) : titre de séjour valable du _____ au _____

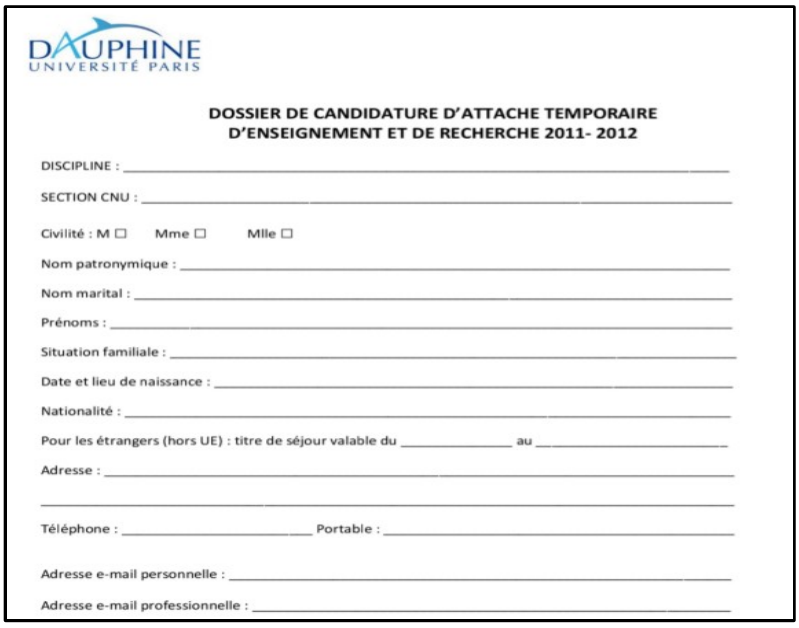
Adresse_____

_____

Téléphone_____

Adresse e-mail _____

FIGURE A.1: Formulaire de candidature ATER à l'université Paris Sud



DAUPHINE UNIVERSITÉ PARIS

DOSSIER DE CANDIDATURE D'ATTACHE TEMPORAIRE
D'ENSEIGNEMENT ET DE RECHERCHE 2011- 2012

DISCIPLINE : _____

SECTION CNU : _____

Civilité : M ☐ Mme ☐ Mlle ☐

Nom patronymique : _____

Nom marital : _____

Prénoms : _____

Situation familiale : _____

Date et lieu de naissance : _____

Nationalité : _____

Pour les étrangers (hors UE) : titre de séjour valable du _____ au _____

Adresse : _____

_____

Téléphone : _____ Portable : _____

Adresse e-mail personnelle : _____

Adresse e-mail professionnelle : _____

FIGURE A.2: Formulaire de candidature ATER à l'université Paris Dauphine

ANNEXE B

Ontologie PITO

Dans cette annexe nous donnons la représentation en OWL/XML de l'ontologie PITO de l'utilisateur présentée dans la section 4.1.2.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:swc="http://www.semanticweb.org/khefifi/ontologies/2014/5/swc#"
    xmlns:geo="http://semanticweb.org/khefifi/ontologies/geoOnto.owl#"
    xmlns:pimi="http://www.semanticweb.org/khefifi/ontologies/usersPims#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:foaf="http://www.semanticweb.org/khefifi/ontologies/2014/5/foaf#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:work="http://www.semanticweb.org/khefifi/ontologies/work.owl#"
    xmlns:family="http://www.semanticweb.org/khefifi/ontologies/family.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <owl:Ontology rdf:about="http://www.semanticweb.org/khefifi/ontologies/
      usersPims#"/>
  <owl:Class rdf:about="http://www.semanticweb.org/khefifi/ontologies/
      usersPims#PITORoot"/>

  <owl:ObjectProperty rdf:about="http://www.semanticweb.org/khefifi/
      ontologies/usersPims#hasHusband">
    <rdfs:range>
      <pimi:PITORoot rdf:about="http://www.semanticweb.org/khefifi/
          ontologies/usersPims#Husband">
        <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/swc#Agent"/>
        <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/family.owl#Person"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/family.owl#birthDate"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/swc#lastName"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
```

```
            ontologies/2014/5/swc#firstName"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/swc#phone"/>
      </pimi:PITORoot>
    </rdfs:range>
    <rdfs:domain>
      <pimi:PITORoot
      rdf:about="http://www.semanticweb.org/khefifi/ontologies/usersPims#
          Identity">
      <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
          ontologies/2014/5/foaf#Person"/>
      <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
          ontologies/2014/5/foaf#OnlineAccount"/>
        <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/swc#Agent"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/foaf#firstname"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/foaf#familyName"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/foaf#acountId"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/swc#phone"/>
      </pimi:PITORoot>
    </rdfs:domain>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="http://www.semanticweb.org/khefifi/
      ontologies/usersPims#hasColleague"> <rdfs:range>
      <pimi:PITORoot rdf:about="http://www.semanticweb.org/khefifi/
          ontologies/usersPims#Colleague">
      <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
          ontologies/work.owl#Employee"/>
      <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
          ontologies/2014/5/foaf#Person"/>
      <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
          ontologies/2014/5/swc#Agent"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/foaf#familyName"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/work.owl#officeNumber"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/foaf#firstname"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/swc#email"/>
      </pimi:PITORoot>
    </rdfs:range>
    <rdfs:domain
    rdf:resource="http://www.semanticweb.org/khefifi/ontologies/usersPims#
        Identity"/>
    </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="http://www.semanticweb.org/khefifi/
      ontologies/usersPims#hasFriend">
    <rdfs:range>
      <pimi:PITORoot rdf:about="http://www.semanticweb.org/khefifi/
          ontologies/usersPims#Friend">
        <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/foaf#Person"/>
```

```
        <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/foaf#OnlineAccount"/>
        <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/swc#Agent"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/swc#email"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/foaf#firstname"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/foaf#acountId"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/2014/5/swc#phone"/>
      </pimi:PITORoot>
    </rdfs:range>
    <rdfs:domain rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Identity"/>
  </owl:ObjectProperty>

<owl:ObjectProperty rdf:about="http://www.semanticweb.org/khefifi/ontologies
    /usersPims#hasCar">
    <rdfs:range>
      <pimi:PITORoot rdf:about="http://www.semanticweb.org/khefifi/
          ontologies/usersPims#Car">
        <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/core.owl#Car"/>
        <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/n3.owl#Auto"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/core.owl#licensePlate"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/n3.owl#assurance"/>
      </pimi:PITORoot>
    </rdfs:range>
    <rdfs:domain rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Identity"/>
 </owl:ObjectProperty>

 <owl:ObjectProperty rdf:about="http://www.semanticweb.org/khefifi/
     ontologies/usersPims#hasPassport">
    <rdfs:range>
      <pimi:PITORoot rdf:about="http://www.semanticweb.org/khefifi/
          ontologies/usersPims#Passport">
        <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/Work.owl#Place"/>
        <pimi:Classes rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/schema#Document"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/schema#passportId"/>
        <pimi:Properties rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/Work.owl#country"/>
      </pimi:PITORoot>
    </rdfs:range>
    <rdfs:domain rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Identity"/>
  </owl:ObjectProperty>

</rdf:RDF>
```

ANNEXE C

Instanciation de l'ontologie PITO

Dans cette annexe nous présentons un extrait de l'instanciation de l'ontologie de l'utilisateur PITO donnée en annexe B.

```
<?xml version="1.0" ?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:context="http://semanticweb.org/khefifi/ontologies/socialOnto.owl#"
   xmlns:swc="http://www.semanticweb.org/khefifi/ontologies/2014/5/swc#"
   xmlns:foaf="http://www.semanticweb.org/khefifi/ontologies/2014/5/foaf#"
   xmlns:work="http://www.semanticweb.org/khefifi/ontologies/work.owl#"
   xmlns:family="http://www.semanticweb.org/khefifi/ontologies/family.owl#"
   xmlns:pimi="http://www.semanticweb.org/khefifi/ontologies/usersPims#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
    usersPims#tuple1">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Friend001"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/foaf#firstname"/>
    <rdf:object>Carole</rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Top"/>
    <pimi:degree> 1 </pimi:degree>
</rdf:Statement>

<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
    usersPims#tuple2">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
```

```
        /usersPims#Friend001"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/swc#email"/>
    <rdf:object>carole.bernard@u-psud.fr</rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Scholar"/>
    <pimi:degree> 1 </pimi:degree>
</rdf:Statement>

<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
    usersPims#tuple3">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Friend001"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/swc#email"/>
    <rdf:object>carole21@gmail.com</rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Personal"/>
    <pimi:degree> 1 </pimi:degree>
</rdf:Statement>

<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
    usersPims#tuple4">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Friend001"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/swc#phone"/>
    <rdf:object>0633333333 </rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Personal"/>
    <pimi:degree> 1 </pimi:degree>
</rdf:Statement>

<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
    usersPims#tuple5">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Friend001"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/swc#phone"/>
    <rdf:object>0121212121 </rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#University"/>
    <pimi:degree> 0.5 </pimi:degree>
</rdf:Statement>


 <rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
     usersPims#tuple6">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Friend001"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/foaf#acountId"/>
    <rdf:object> caroleB </rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Personal"/>
    <pimi:degree> 0.5 </pimi:degree>
</rdf:Statement>

<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
```

```
    usersPims#tuple7">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Friend001"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/foaf#acountId"/>
    <rdf:object> carole.bernard </rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Professional"/>
    <pimi:degree> 1 </pimi:degree>
</rdf:Statement>

<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
    usersPims#tuple8">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Friend002"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/foaf#firstname"/>
    <rdf:object>Bob</rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Top"/>
    <pimi:degree> 1 </pimi:degree>
</rdf:Statement>

<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
    usersPims#tuple9">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Friend002"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/swc#email"/>
    <rdf:object>lebeau@univ-paris1.fr</rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Work"/>
    <pimi:degree> 1 </pimi:degree>
</rdf:Statement>

<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
    usersPims#tuple10">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Friend002"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/swc#email"/>
    <rdf:object>bob.lebeau@gmail.com</rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Personal"/>
    <pimi:degree> 1 </pimi:degree>
</rdf:Statement>

<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
    usersPims#tuple11">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Friend002"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/swc#phone"/>
    <rdf:object>0625252525 </rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Personal"/>
    <pimi:degree> 1 </pimi:degree>
</rdf:Statement>
```

```
<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
    usersPims#tuple12">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Friend002"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/swc#phone"/>
    <rdf:object>0150505050 </rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Work"/>
    <pimi:degree> 0.5 </pimi:degree>
</rdf:Statement>

<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
    usersPims#tuple13">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Friend002"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/swc#phone"/>
    <rdf:object>0230303030 </rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Professional"/>
    <pimi:degree> 1 </pimi:degree>
</rdf:Statement>

<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
    usersPims#tuple14">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Friend002"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/foaf#acountId"/>
    <rdf:object> leBeau </rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Personal"/>
    <pimi:degree> 1 </pimi:degree>
</rdf:Statement>

<rdf:Statement rdf:about="http://www.semanticweb.org/ontologies/pimsUsers#
    tuple100">
    <rdf:subject>
      <rdf:Description rdf:about="http://www.semanticweb.org/khefifi/
          ontologies/pimsUsers#Identity001">
        <pimi:hasFriend rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/pimsUsers#Friend001"/>
        <pimi:hasFriend rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/pimsUsers#Friend002"/>
        <pimi:hasCar rdf:resource="http://www.semanticweb.org/khefifi/
            ontologies/pimsUsers#Car001"/>
      </rdf:Description>
    </rdf:subject>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/foaf#firstname"/>
    <rdf:object>Alice</rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Top"/>
    <pimi:degree> 1 </pimi:degree>
 </rdf:Statement>

<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
    usersPims#tuple101">
```

```
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Identity001"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/foaf#familyname"/>
    <rdf:object> bernard </rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Top"/>
    <pimi:degree> 1 </pimi:degree>
</rdf:Statement>

<rdf:Statement rdf:about="http://www.semanticweb.org/khefifi/ontologies/
    usersPims#tuple102">
    <rdf:subject rdf:resource="http://www.semanticweb.org/khefifi/ontologies
        /usersPims#Identity001"/>
    <rdf:predicate rdf:resource="http://www.semanticweb.org/khefifi/
        ontologies/2014/5/swc#phone"/>
    <rdf:object> 0633343536 </rdf:object>
    <pimi:context rdf:resource="http://semanticweb.org/khefifi/ontologies/
        socialOnto.owl#Personal"/>
    <pimi:degree> 1 </pimi:degree>
</rdf:Statement>
</rdf:RDF>
```

ANNEXE D

Notations utilisées

Notation	***Signification***
$\mathcal{O}$	une ontologie
$\mathcal{C}$	ensemble de concepts de l'ontologie $\mathcal{O}$
$\mathcal{P}$	ensemble de propriétés de l'ontologie $\mathcal{O}$
$\mathcal{A}$	ensemble d'axiomes de l'ontologie $\mathcal{O}$
$\mathcal{O}^{\mathscr{C}}$	une ontologie de contextes
$\mathcal{O}_{\text{Info}}$	ontologie de catégories d'informations personnelles
$\mathcal{C}_{\text{Info}}$	ensemble de classes de l'ontologie de catégories d'informations personnelles
cat^{Info}	une catégorie d'informations personnelles
$\mathcal{O}_{pv}$	ensemble d'ontologies points de vue
$\mathcal{M}_{pv}$	ensemble de *mappings* entre les ontologies points de vue
$\mathcal{PV}$	niveau point de vue
PITO	ontologie d'informations personnelles
$\mathfrak{C}$	ensemble de classes de l'ontologie PITO (ensemble de PIT)
$\mathfrak{R}$	ensemble d'axiomes déclaré dans PITO
$\mathfrak{M}_p$	ensemble de *mappings* de l'ontologie PITO
$label$	label du PIT
M_p	ensemble de mappings entre un PIT et des points de vue
uri	identifiant unique d'une instance d'information personnelle
T	type d'informations personnelles
P_{inst}	ensemble d'instances de propriétés d'un PI

$P^{\mathscr{C}}_{inst}$	ensemble d'instances contextuelles de propriétés du PI
p	propriété d'une PI
v	valeur de la propriété p
c	contexte de la valeur v
δ	le degré d'utilisabilité de la valeur v dans le contexte c
$\mathcal{F}$	formulaire annoté
Z	ensemble de zones d'un formulaire
Ch	ensemble de champs d'un formulaire
An	ensemble d'annotation d'un formulaire
$\mathcal{T}$	arbre de formulaire
$\mathcal{E}$	ensemble d'arcs de $\mathcal{T}$
$\mathcal{N}$	ensemble de nœuds de $\mathcal{T}$
$\mathcal{E}_{\text{T}}$	ensemble d'arcs reliant des nœuds non-terminaux aux nœuds terminaux
$\mathcal{E}_{\text{NT}}$	ensemble d'arcs reliant deux nœuds non-terminaux
$\mathcal{N}_{\text{T}}$	ensemble de nœuds terminaux
$\mathcal{N}_{\text{NT}}$	ensemble de nœuds non-terminaux
Q	requête sémantique
$\mathcal{Q}^{f\underline{c}}$	requête contextuelle
$\underline{c}$	contexte de référence (du formulaire/procédure)
f	fonction de comparaison
δ^r	degré de pertinence local
Δ	degré de pertinence global (d'une réponse)
sim_{WP}	mesure de similarité de Wu et Palmer
r	réponse à une requête
$\mathscr{R}$	ensemble de réponses à une requête
$Proc$	procédure abstraite à accomplir
A	ensemble d'activités d'une procédure
N	ensemble de nœuds d'une procédures
λ	fonction permettant d'associer une activité à un nœud
N_A	ensemble de nœuds de type activité
N_{JA}	ensemble de nœuds de type AND-Joint
N_{JO}	ensemble de nœuds de type XOR-Joint
N_{SA}	ensemble de nœuds de type AND-Split
N_{SO}	ensemble de nœuds de type XOR-Split
$\rightarrow$	séquencement entre deux nœuds appartenant à N
w	un service
u	adresse ou nom unique d'un service
I	ensemble d'attributs en entrée d'un service

O	ensemble d'attributs en sortie d'un service
K	ensemble de capacités d'un service
$\mathcal{O}_{\text{Serv}}$	ontologie de catégories d'un service
C	une catégorie du service
W	ensemble de services disponibles
W'	ensemble de services disponibles et autorisés à être utilisés
$\mathscr{D}$	ensemble de propositions contextuelles
δ'	degré de pertinence utilisé par les propositions contextuelles
pol	politique de sécurité
Pol	ensemble de politiques de sécurité
$\mathscr{P}$	problème de composition
ϵ	seuil minimal d'acceptation
$\mathbf{A}$	ensemble d'actions
$\mathbf{I}$	ensemble de propositions contextuelles initiales
$\mathbf{G}$	ensemble de propositions finales
$\mathbf{a}$	action nommée a
$\mathbf{Pre}(\mathbf{a})$	préconditions de l'action $\mathbf{a}$
$\mathbf{Eff}^{-}(\mathbf{a})$	effets négatifs de l'action $\mathbf{a}$
$\mathbf{Eff}^{+}(\mathbf{a})$	effets positifs de l'action $\mathbf{a}$
$\mathbf{AL}_i$	$i^{ème}$ niveau d'actions
$\mathbf{PL}_i$	$i^{ème}$ niveau de propositions
$\mathbf{\Pi}$	problème de planification
LCS	concept le plus spécifique commun entre deux concepts dans une ontologie

Bibliographie

[AB04] Alain Abran and Pierre Bourque. *SWEBOK : Guide to the software engineering Body of Knowledge*. 2004. ix

[Afe09] Jmal-Maâlej Afef. Évaluation de politiques d'auto-adaptabilité basées sur la mobilité des services web orchestrés. Master's thesis, École Nationale d'Ingénieurs de Sfax, 2009. x

[Arn94] André Arnold. *Finite transition systems - semantics of communicating systems*. 1994. 33

[Ars04] Ali Arsanjani. Service-oriented modeling and architecture. *IBM developer works*, pages 1–15, 2004. 18

[AS04] Knarig Arabshian and Henning Schulzrinne. Gloserv : Global service discovery architecture. In *The First Annual International Conference on Mobile and Ubiquitous Systems : Networking and Services*, pages 319–325, 2004. 38

[Baa03] Franz Baader. *The description logic handbook : theory, implementation, and applications*. 2003. 47

[BAHP82] Mordechai Ben-Ari, Joseph Y Halpern, and Amir Pnueli. Deterministic propositional dynamic logic : Finite models, complexity, and completeness. *Journal of computer and system sciences*, pages 402–417, 1982. 34

[BAS12] Victoria Beltran, Knarig Arabshian, and Henning Schulzrinne. Ontology-based user-defined rules and context-aware service composition system. In *The Semantic Web*, pages 139–155, 2012. 38, 41

[BBD03] Jeremy Bryans, Howard Bowman, and John Derrick. Model checking stochastic automata. *ACM Transactions on Computational Logic (TOCL)*, pages 452–492, 2003. 33

[BBHB11] Karim Benouaret, Djamal Benslimane, Allel Hadjali, and Mahmoud Barhamgi. Top-k web service compositions using fuzzy dominance relationship. In *IEEE International Conference on Services Computing (SCC)*, pages 144–151, 2011. 20

[BBMN+08] Ofer Bergman, Ruth Beyth-Marom, Rafi Nachmias, Noa Gradovitch, and Steve Whittaker. Improved search engines and navigation preference in personal information management. *ACM Transactions on Information Systems (TOIS)*, page 20, 2008. 10

[BBS10] Omer Boyaci, Victoria Beltran, and Henning Schulzrinne. Bridging communications and the physical world : Sense everything, control everything. In *GLOBECOM Workshops (GC Wkshps), IEEE*, pages 1735–1740, 2010. 38, 41

[BC07] Antonio Brogi and Sara Corfini. Behaviour-aware discovery of web service compositions. *International Journal of Web Services Research*, pages 1–25, 2007. 31, 32, 41

[BCDG+03] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Massimo Mecella. Automatic composition of e-services that export their behavior. In *Service-Oriented Computing-ICSOC 2003*, pages 43–58. 2003. 33

[BCDG+05] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Richard Hull, and Massimo Mecella. Automatic composition of transition-based semantic web services with messaging. In *International conference on Very large data bases*, pages 613–624, 2005. 33

[BCG+05] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Massimo Mecella. Automatic service composition based on behavioral descriptions. *International Journal of Cooperative Information Systems*, pages 333–376, 2005. 33, 34, 35, 41

[BCGP08] Daniela Berardi, Fahima Cheikh, Giuseppe De Giacomo, and Fabio Patrizi. Automatic service composition via simulation. *International Journal of Foundations of Computer Science*, pages 429–451, 2008. 33, 34, 35, 41

[BEK+00] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple object access protocol (soap) 1.1, 2000. 17

[BF97] Avrim Blum and Merrick L. Furst. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence*, pages 281–300, 1997. 24, 25, 104

[BGX+13] Xiaoan Bao, Jun Gao, Xiaoming Xie, Hui Lin, and Na Zhang. Research of the web service automatic composition based on generalized stochastic petri nets. *International Journal of Multimedia & Ubiquitous Engineering*, 8(4), 2013. 32, 41

[BMGI06] Sonia Ben Mokhtar, Nikolaos Georgantas, and Valérie Issarny. Cocoa : conversation based service composition for pervasive computing environments. In *ACS/IEEE International Conference on Pervasive Services*, pages 29–38, 2006. 34, 35, 41

[BMGI07] Sonia Ben Mokhtar, Nikolaos Georgantas, and Valérie Issarny. Cocoa : Conversation-based service composition in pervasive computing environments with qos support. *Journal of Systems and Software*, pages 1941–1955, 2007. 34, 35, 41

[BN95] Deborah Barreau and Bonnie A Nardi. Finding and reminding : file organization from the desktop. *ACM SigChi Bulletin*, pages 39–43, 1995. xii, 4

[BP10] George Baryannis and Dimitris Plexousakis. Automated Web Service Composition : State of the Art and Research Challenges. Technical report, 2010. 23

[BS04] Richard Boardman and M Angela Sasse. Stuff goes into the computer and doesn't come out : a cross-tool study of personal information management. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 583–590, 2004. xi, 5, 10

[BSD14] Athman Bouguettaya, Quan Z Sheng, and Florian Daniel. *Web Services Foundations*. 2014. 16, 21

[Bus45] Vannevar Bush. As we may think. 1945. xi, 4

[CDGL+08] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Massimo Mecella, and Fabio Patrizi. Automatic service composition and synthesis : the roman model. *IEEE Data Eng. Bull.*, pages 18–22, 2008. 33

[CDH+05] Yuhan Cai, Xin Luna Dong, Alon Halevy, Jing Michelle Liu, and Jayant Madhavan. Personal information management with semex. In *ACM SIGMOD international conference on Management of data*, pages 921–923, 2005. 7, 11, 13

[CEHMR11] Yudith Cardinale, Joyce El Haddad, Maude Manouvrier, and Marta Rukoz. Cpn-tws : a coloured petri-net approach for transactional-qos driven web service composition. *International Journal of Web and Grid Services*, pages 91–115, 2011. 32, 41

[CGG+05] Paul Alexandru Chirita, Rita Gavriloaie, Stefania Ghita, Wolfgang Nejdl, and Raluca Paiu. Activity based metadata for semantic desktop search. In *The Semantic Web : Research and Applications*, pages 439–454. 2005. 9, 11, 13

[CGP99] Edmund M Clarke, Orna Grumberg, and Doron Peled. *Model checking*. 1999. 30

[CGWW09] Jidong Chen, Hang Guo, Wentao Wu, and Wei Wang. imecho : an associative memory based desktop search system. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 731–740, 2009. 8, 12, 13

[CMP+06] Carlos Canal, Juan Manuel Murillo, Pascal Poizat, et al. Software adaptation. *L'objet*, pages 9–31, 2006. 18

[CMRW07] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman, and Sanjiva Weerawarana. Web services description language (wsdl) version 2.0 part 1 : Core language. *W3C Recommendation*, 2007. 17

[CPRT03] Alessandro Cimatti, Marco Pistore, Marco Roveri, and Paolo Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, pages 35–84, 2003. 34

[CPY14] Min Chen, Pascal Poizat, and Yuhong Yan. Adaptive composition and qos optimization of conversational services through graph planning encoding. In *Web Services Foundations*, pages 423–449. 2014. 27, 28, 41

[CR97] Chandra Chekuri and Anand Rajaraman. Conjunctive query containment revisited. In *Database Theory*, pages 56–70. 1997. 78

[DCC+03] Susan Dumais, Edward Cutrell, Jonathan J Cadiz, Gavin Jancke, Raman Sarin, and Daniel C Robbins. Stuff i've seen : a system for personal information retrieval and re-use. In *International ACM SIGIR conference on Research and development in informaion retrieval*, pages 72–79, 2003. 6, 12, 13

[DEL+00] Paul Dourish, W Keith Edwards, Anthony LaMarca, John Lamping, Karin Petersen, Michael Salisbury, Douglas B Terry, and James Thornton. Extending document management systems with user-specific active properties. *ACM Transactions on Information Systems (TOIS)*, pages 140–170, 2000. 7, 13

[Dey01] Anind K Dey. Understanding and using context. *Personal and ubiquitous computing*, pages 4–7, 2001. 6

[Dij59] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, pages 269–271, 1959. 27

[DLPT02] Ugo Dal Lago, Marco Pistore, and Paolo Traverso. Planning with a language for extended goals. In *AAAI/IAAI*, pages 447–454, 2002. 34

[DS05] Schahram Dustdar and Wolfgang Schreiner. A survey on web services composition. *International Journal of Web and Grid Services*, pages 1–30, 2005. 16, 21, 23

[DS06] Jens-Peter Dittrich and Marcos Antonio Vaz Salles. idm : A unified and versatile data model for personal dataspace management. In *International conference on Very large data bases*, pages 367–378, 2006. 9

[DSK06] Jens-Peter Dittrich, M Salles, and S Karaksashian. imemex : A platform for personal dataspace management. In *SIGIR PIM Workshop*, pages 22–29, 2006. 13

[EHN94] Kutluhan Erol, James Hendler, and Dana S Nau. Htn planning : Complexity and expressivity. In *AAAI*, pages 1123–1128, 1994. 24

[Erl05] Thomas Erl. Service-oriented architecture (soa) concepts, technology and design. 2005. ix

[Erl08] Thomas Erl. *Soa : principles of service design*, volume 1. 2008. ix

[ES13] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching, Second Edition*. 2013. 59

[Evé10] Florian Evéquoz. *Supporting personal information management with visual facets*. PhD thesis, University of Fribourg, 2010. 11

[FG96] Eric Freeman and David Gelernter. Lifestreams : A storage model for personal data. *ACM SIGMOD Record*, pages 80–86, 1996. 6, 10, 13

[FOA00] FOAF Vocabulary Specification. http://xmlns.com/foaf/spec/, 2000. [en ligne ; accès 22-05-2014]. 46

[GBL+02] Jim Gemmell, Gordon Bell, Roger Lueder, Steven Drucker, and Curtis Wong. Mylifebits : fulfilling the memex vision. In *ACM international conference on Multimedia*, pages 235–238, 2002. 7, 11, 13

[GNT04] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated planning : theory & practice*. 2004. 24

[Gru95] Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing ? *International journal of human-computer studies*, pages 907–928, 1995. 46

[GT00] Fausto Giunchiglia and Paolo Traverso. Planning as model checking. In *Recent Advances in AI Planning*, pages 1–20. 2000. 34

[GV03] Claude Girault and Rüdiger Valk. *Petri nets for systems engineering : a guide to modeling, verification, and applications*. 2003. 19

[HB03] Rachid Hamadi and Boualem Benatallah. A petri net-based model for web service composition. In *Australasian Database Conference*, pages 191–200, 2003. x, 17, 18, 31, 32, 41

[HH93] David Hollingsworth and UK Hampshire. Workflow management coalition the workflow reference model. *Workflow Management Coalition*, 1993. 19, 90

[HKPS13] Joyce El Haddad, Rania Khéfifi, Pascal Poizat, and Fatiha Saïs. Model-Based techniques for functional PIMS composition, 2013. 105

[Hor02] Ian Horrocks. Daml+oil : a description logic for the semantic web. *IEEE Data Engineering Bulletin*, pages 4–9, 2002. 47

[HSM01] Siegfried Handschuh, Steffen Staab, and Alexander Maedche. Cream : creating relational metadata with a component-based, ontology-driven annotation framework. In *Proceedings of the 1st international conference on Knowledge capture*, pages 76–83, 2001. 75

[Jon04] William Jones. Finders, keepers ? the present and future perfect in support of personal information management. 2004. 3

[Kap03] Victor Kaptelinin. Umea : translating interaction histories into project contexts. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 353–360, 2003. 6, 10, 13

[KBS05] Dirk Krafzig, Karl Banke, and Dirk Slama. *Enterprise SOA : service-oriented architecture best practices*. 2005. ix

[KGS05] Matthias Klusch, Andreas Gerber, and Marcus Schmidt. Semantic web service composition planning with owls-xplan. In *Proceedings of the AAAI Fall Symposium on Semantic Web and Agents*, 2005. 27, 28, 41

[Kie03] Bartosz Kiepusewski. *Expressiveness and suitability of languages for control flow modelling in workflows*. 2003. 91, 97

[Kim12] Jinyoung Kim. *Retrieval and evaluation techniques for personal information*. PhD thesis, University of Massachusetts Amherst, 2012. 9

[KJ05] Aparna Krishnan and Steve Jones. Timespace : activity-based temporal visualisation of personal information spaces. *Personal and Ubiquitous Computing*, pages 46–65, 2005. 6, 10, 13

[KPS+05] Vivi Katifori, Antonella Poggi, Monica Scannapieco, Tiziana Catarci, and Yannis E Ioannidis. Ontopim : how to rely on a personal ontology for personal information management. In *Semantic Desktop Workshop*, 2005. 8, 11, 13

[KPS12a] Rania Khéfifi, Pascal Poizat, and Fatiha Saïs. Modeling and querying context-aware personal information spaces. In *DEXA (2)*, pages 103–110, 2012. xii, 72

[KPS12b] Rania Khéfifi, Pascal Poizat, and Fatiha Saïs. Modélisation et interrogation d'espaces d'informations personnelles sensibles au contexte. In *EGC*, pages 573–574, 2012. xii, 72

[KS92] Henry A. Kautz and Bart Selman. Planning as satisfiability. In *ECAI*, pages 359–363, 1992. 39

[KSP+05] Ugur Kuter, Evren Sirin, Bijan Parsia, Dana Nau, and James Hendler. Information gathering during planning for web service composition. *Web semantics : science, services and agents on the World Wide Web*, pages 183–205, 2005. 26, 28, 41

[KtHB00] Bartek Kiepuszewski, Arthur Harry Maria ter Hofstede, and Christoph J Bussler. On structured workflow modelling. In *Advanced Information Systems Engineering*, pages 431–445, 2000. 91

[Lak76] Imre Lakatos. *Proofs and refutations : The logic of mathematical discovery*. 1976. 36

[Lan88a] Mark W Lansdale. The psychology of personal information management. *Applied ergonomics*, pages 55–66, 1988. xi

[Lan88b] Mark W Lansdale. The psychology of personal information management. *Applied ergonomics*, pages 55–66, 1988. 4

[Lar] Dictionnaires de français. http://www.larousse.fr/dictionnaires/francais/réification/67739?q=réification. [en ligne ; accès 22-05-2014]. 50

[LEHP14] Amine Louati, Joyce El Haddad, and Suzanne Pinson. A distributed decision making and propagation approach for trust-based service discovery in social networks. In *Group Decision and Negotiation. A Process-Oriented View*, pages 262–269. 2014. xii

[LF94] Mik Lamming and Mike Flynn. Forget-me-not : Intimate computing in support of human memory. In *Next Generation Human Interface*, page 4, 1994. 6, 10, 13

[Lin98] Dekang Lin. An information-theoretic definition of similarity. In *ICML*, pages 296–304, 1998. 53

[LKS08] Naiwen Lin, Ugur Kuter, and Evren Sirin. Web service composition with user preferences. In *The Semantic Web : Research and Applications*, pages 629–643. 2008. 24, 25, 27, 28, 41

[LRE+94] Hector J Levesque, Raymond Reiter, YVES LESP ERANCE, FANGZHEN LIN, and Richard B Scherl. Golog : A logic programming language for dynamic domains. *J. LOGIC PROGRAMMING*, pages 1–679, 1994. 38

[MABS+12] Karima Mokhtari-Aslaoui, Salima Benbernou, Soror Sahri, Vasilios Andrikopoulos, Frank Leymann, and Mohand-Said Hacid. Timed privacy-aware business protocols. *International Journal of Cooperative Information Systems*, 2012. 121

[McG82] James J McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Software : Practice and Experience*, pages 23–34, 1982. 24

[MEG+03] Luke McDowell, Oren Etzioni, Steven D Gribble, Alon Halevy, Henry Levy, William Pentney, Deepak Verma, and Stani Vlasseva. Mangrove : Enticing ordinary people onto the semantic web via instant gratification. In *The Semantic Web-ISWC 2003*, pages 754–770. 2003. 75

[Mey88] Bertrand Meyer. *Object-oriented software construction*, volume 2. 1988. ix

[MFGI05] Sonia Ben Mokhtar, Damien Fournier, Nikolaos Georgantas, and Valérie Issarny. Context-aware service composition in pervasive computing environments. In *Rapid Integration of Software Engineering Techniques (RISE)*, pages 129–144, 2005. 34, 35, 41

[MP09] Annapaola Marconi and Marco Pistore. Synthesis and composition of web services. In *Formal Methods for Web Services*, pages 89–157. 2009. 16, 21

[MPM08] Tarek Melliti, Pascal Poizat, and Sonia Ben Mokhtar. Distributed behavioural adaptation for the automatic composition of semantic services. In *FASE*, pages 146–162, 2008. 33, 34, 35, 41

[MRS08] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. 2008. 9

[MS02] Sheila McIlraith and Tran Cao Son. Adapting golog for composition of semantic web services. *International Conference on Principles and Knowledge Representation and Reasoning*, pages 482–493, 2002. 38, 41

[MVH+04] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, page 10, 2004. 47

[MZ95] Jock D Mackinlay and Polle T Zellweger. Browsing vs. search : can we find a synergy ?(panel session). In *Conference companion on Human factors in computing systems*, pages 179–180, 1995. 9

[NM02] Srini Narayanan and Sheila A McIlraith. Simulation, verification and automated composition of web services. In *World Wide Web*, pages 77–88, 2002. x, 32, 41

[OMG13] Omg uml specification. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel, 2013. [en ligne ; accès 23-05-2014]. 19

[Orn04] Elizabeth Orna. *Information strategy in practice*. Gower, 2004. 4

[OWL12] OWL 2 Web Ontology Language Profiles. http://www.w3.org/TR/owl2-profiles/, 2012. [en ligne ; accès 23-02-2014]. 49

[Par72] David Lorge Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, pages 1053–1058, 1972. ix

[Pee05] Joachim Peer. Web service composition as ai planning-a survey. *Second revised version*, 2005. 23

[Pet62] Carl Adam Petri. Fundamentals of a theory of asynchronous information flow. In *IFIP Congress*, pages 386–390, 1962. 29

[PF02] Shankar R Ponnekanti and Armando Fox. Sword : A developer toolkit for web service composition. In *International World Wide Web Conference*, 2002. 37, 38, 41

[PPS06] Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive (1) designs. In *Verification, Model Checking, and Abstract Interpretation*, pages 364–380, 2006. 33

[PTB05] Marco Pistore, Paolo Traverso, and Piergiorgio Bertoli. Automated composition of web services by planning in asynchronous domains. In *ICAPS*, pages 2–11, 2005. 33, 34, 35, 41

[PY10] Pascal Poizat and Yuhong Yan. Adaptive composition of conversational services through graph planning encoding. In *Leveraging Applications of Formal Methods, Verification, and Validation*, pages 35–50. 2010. 18, 24, 25, 27, 28, 41, 97, 121, 153

[RDF04a] RDF 1.1. http://www.w3.org/standards/techs/rdf, 2004. [en ligne ; accès 22-05-2014]. 49

[RDF04b] RDF Schema 1.1. http://www.w3.org/TR/rdf-schema/, 2004. [en ligne ; accès 22-05-2014]. 47

[RDQ04] RDQL - A Query Language for RDF. http://www.w3.org/Submission/RDQL/, 2004. [en ligne ; accès 07-07-2014]. 12

[Rek99] Jun Rekimoto. Time-machine computing : a time-centric approach for the information environment. In *ACM symposium on User interface software and technology*, pages 45–54, 1999. 6, 10, 13

[Res95] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*, 1995. 53

[RF11] Philippe Ramadour and Myriam Fakhri. Modèle et langage de composition de services. In *INFORSID*, pages 59–76, 2011. 23

[RKM04] Jinghai Rao, Peep Kungas, and Mihhail Matskin. Logic-based web services composition : from service description to process model. In *IEEE International Conference on Web Services*, pages 446–453, 2004. 37, 38, 41

[RLC+08] Kaijun Ren, Xiao Liu, Jinjun Chen, Nong Xiao, Junqiang Song, and Weimin Zhang. A qsql-based efficient planning algorithm for fully-automated service composition in dynamic service environments. In *IEEE International Conference on Services Computing*, pages 301–308, 2008. 32, 41

[RMBB89] Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, pages 17–30, 1989. 53

[Rob65] John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM)*, pages 23–41, 1965. 36

[RS05] Jinghai Rao and Xiaomeng Su. A survey of automated web service composition methods. In *Semantic Web Services and Web Process Composition*, pages 43–54. 2005. ix, 16, 21, 23

[SAW94] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Mobile Computing Systems and Applications*, pages 85–90, 1994. 6

[SBD05] Leo Sauermann, Ansgar Bernardi, and Andreas Dengel. Overview and outlook on the semantic desktop. In *Semantic Desktop Workshop*, 2005. 8, 13

[SDA99] Daniel Salber, Anind K Dey, and Gregory D Abowd. The context toolkit : aiding the development of context-enabled applications. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 434–441, 1999. 6

[SM10] Shirin Sohrabi and Sheila A McIlraith. Preference-based web service composition : A middle ground between execution and search. In *The Semantic Web–ISWC*, pages 713–729. 2010. 27, 28, 41

[SMDWD11] Dominique Scapin, Pascal Marie-Dessoude, Marco Winckler, and Claudia Detraux. Personal information systems : User views and information categorization. In *International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services*, pages 40–47, 2011. 69

[SPA06] SPARQL Query Language For RDF. http://fr.wikipedia.org/wiki/SPARQL, 2006. [en ligne ; accès 23-04-2014]. 51

[SPDG08] Sebastian Sardina, Fabio Patrizi, and Giuseppe De Giacomo. Behavior composition in the presence of failure. In *KR*, pages 640–650, 2008. 33

[SPM09] Shirin Sohrabi, Nataliya Prokoshyna, and Sheila A McIlraith. Web service composition via the customization of golog programs with user preferences. In *Conceptual Modeling : Foundations and Applications*, pages 319–334. 2009. x, 17, 20, 38, 41

[Spo09] Apple MAC OS X spotlight. http://support.apple.com/kb/ht3636, 2009. [en ligne ; accès 02-02-2014]. 11

[ST94] Bill N Schilit and Marvin M Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, pages 22–32, 1994. 6

[SVED07] Leo Sauermann, Ludger Van Elst, and Andreas Dengel. Pimo - a framework for representing personal information models. *Proceedings of I-Semantics*, pages 270–277, 2007. 8, 12, 13

[SWC07] FOAF Vocabulary Specification. http://data.semanticweb.org/ns/swc/swc_2009-05-09.html, 2007. [en ligne ; accès 22-05-2014]. 58

[SWS] SWS-Challenge. sws-challenge.org/wiki/index.php/Main_PageSWS. [en ligne ; accès 06-06-2014]. 112

[Szy02] Clemens Szyperski. *Component software : beyond object-oriented programming*. 2002. ix

[TAAK04] Jaime Teevan, Christine Alvarado, Mark S Ackerman, and David R Karger. The perfect search engine is not enough : a study of orienteering behavior in directed search. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 415–422, 2004. 9

[Tay97] David A Taylor. *Object technology : a manager's guide*. 1997. ix

[TBS+06] Benjamin Turnbull, Barry Blundell, Jill Slay, et al. *Google desktop as a source of digital evidence*. 2006. 11

[TFZT10] Wei Tan, Yushun Fan, MengChu Zhou, and Zhong Tian. Data-driven service composition in enterprise soa solutions : a petri net approach. *Automation Science and Engineering*, pages 686–694, 2010. 31, 32, 41

[TJB06] Jaime Teevan, William Jones, and Benjamin B Bederson. Personal information management. *Communications of the ACM*, pages 40–43, 2006. 4

[UKM03] Sebastian Uchitel, Jeff Kramer, and Jeff Magee. Behaviour model elaboration using partial labelled transition systems. In *ACM SIGSOFT Software Engineering Notes*, pages 19–27, 2003. 19

[VR04] Maja Vukovic and Peter Robinson. Adaptive, planning based, web service composition for context awareness. In *Second International Conference on Pervasive Computing, Vienna*, 2004. 27, 28, 41

[W3C14] W3C. Web services glossary. *W3C Working Group Note, http://www.w3.org/TR/ws-gloss*, 2014. 17

[WGV+11] Marco Winckler, Vicent Gaits, Dong-Bach Vo, Firmenich Sergio, and Gustavo Rossi. An approach and tool support for assisting users to fill-in web forms with personal information. In *ACM international conference on Design of communication*, pages 195–202, 2011. 74

[WP94] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Association for Computational Linguistics*, pages 133–138, 1994. 53, 82

[WPS+03] Dan Wu, Bijan Parsia, Evren Sirin, James Hendler, and Dana Nau. *Automating DAML-S web services composition using SHOP2*. 2003. 26, 28, 41

[WS-14] Oasis web services business process execution language. http://www.omg.org/spec/UML/2.5/Beta2/, 2014. [en ligne ; accès 23-05-2014]. 21

[WSH+06] Dan Wu, Evren Sirin, James Hendler, Dana Nau, and Bijan Parsia. Automatic web services composition using shop2. Technical report, DTIC Document, 2006. x, 17, 24, 25

[XC05] Huiyong Xiao and Isabel F Cruz. A multi-ontology approach for personal information management. In *Semantic Desktop Workshop*, 2005. 8, 10, 12, 13

[XZLZ13] Yunni Xia, Xiang Zhang, Xin Luo, and Qingsheng Zhu. Modelling of ontology-based service compositions using petri net. *Electronics and Electrical Engineering*, pages 75–78, 2013. 32

[YCY12] Yuhong Yan, Min Chen, and Yubin Yang. Anytime qos optimization over the plangraph for web service composition. In *ACM Symposium on Applied Computing*, pages 1968–1975, 2012. 27, 28, 41, 120

Liste des figures

Liste des tableaux

Liste des Algorithmes