D5.3. KOLFLOW
Description et fonctionnement de l'assistant distribué pour la co-construction de connaissances

Marie Lefevre, Amélie Cordier, Pierre-Antoine Champin

Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France
{prenom.nom}@liris.cnrs.fr

Ce rapport présente les différents mécanismes proposés dans le cadre de la tâche 5 du projet Kolflow [1] dans le but de fournir un assistant alter-ego permettant une co-construction de connaissances entre humain et agents artificiels intelligents.

Assistance à l'édition de connaissances

Lorsque des utilisateurs éditent des wikis sémantiques afin de construire une base de connaissances, ceux-ci doivent à la fois maitriser le fonctionnement et la syntaxe propres à un wiki, mais également le formalisme utilisé afin de définir des relations sémantiques.

À titre d'exemple, le site WikiTaaable [2-3] est un wiki utilisé pour définir des ontologies relatives à la cuisine. Chaque ingrédient est ainsi décrit en utilisant lasyntaxe du wiki et annoté avec des triplets RDF (cf. partie droite de la Figure 1) afin d'être affiché dans le wiki pour les utilisateurs (cf. partie gauche de la Figure 1).

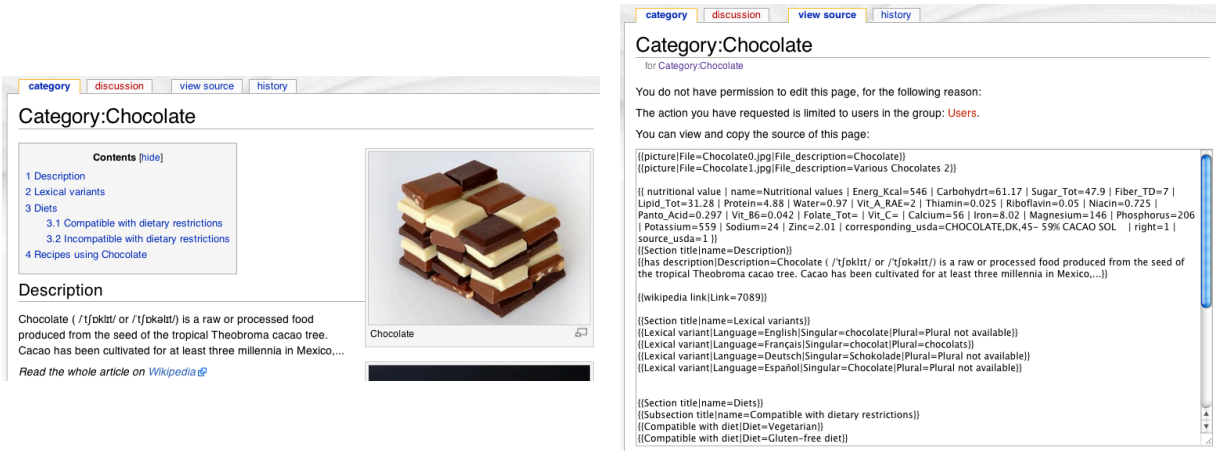

Figure 1 : Définition d'un ingrédient dans WikiTaaable : vue utilisateur à gauche, vue système à droite.

Dans le cadre du projet Kolflow, nous utilisons Semantic MediaWiki [4] pour la représentation des connaissances et le stockage de données est effectué dans des RDFStores. Les annotations doivent donc être des triplets RDF. Cette situation est illustrée par la Figure 2.

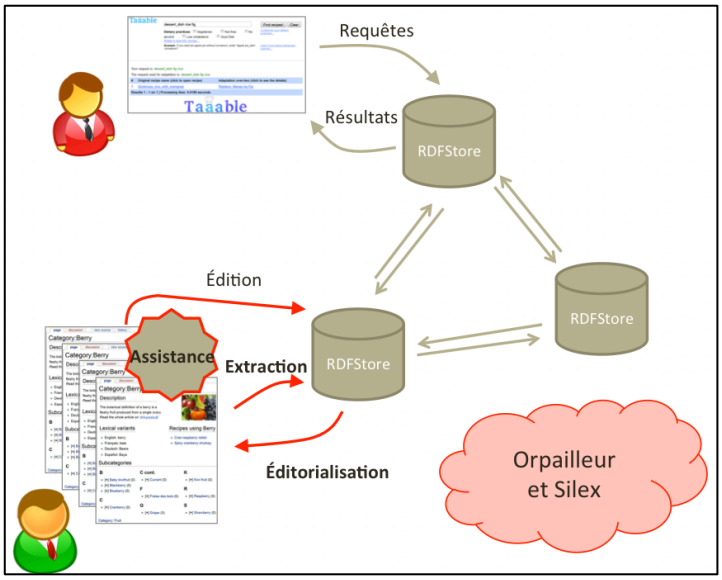

Figure 2 : Assistance à l'édition, l'extraction et l'éditorialisation de connaissances.

Dans ce contexte, un premier type d'assistance concerne l'aide à l'édition de connaissances. Cette aide peut porter sur l'outil lui-même, sur la syntaxe de ce wiki mais également sur la création d'annotations.

Toujours dans le cadre du projet Kolflow, un assistant a été développé pour aider les utilisateurs de l'outil DSMW (Distributed Semantic MediaWiki) [5]. Cet assistant repose sur l'exploitation des traces d'interaction d'une communauté d'utilisateurs de DSMW [6]. Pour chacune des différentes tâches pouvant être faites par un utilisateur de DSMW, l'assistant fournit à l'utilisateur :

- une explication textuelle avec éventuellement des schémas illustrant les différents composants de l'explication et leurs interactions ;
- une vidéo des actions à faire et/ou une capture d'écran ;
- des exemples de traces d'utilisateurs ayant réalisé cette tâche ;
- éventuellement une automatisation de la tâche.

Cet assistant pourrait être réutilisé pour permettre une assistance aux wikis classiques et pourrait être complété pour guider les utilisateurs lors de la création d'annotations sémantiques.

Cette réutilisation de l'assistant à l'outil DSMW pour fournir de l'assistance à l'édition d'annotation sémantique est actuellement à l'état de projet. Sa mise en œuvre fera l'objet d'une collaboration entre l'équipe SILEX du LIRIS et l'équipe Orpailleur du LORIA sur la prochaine période du projet.

Assistance à l'éditorialisation de connaissances

Dans la situation illustrée dans la Figure 2, les différentes bases RDF (RDFStores) échangent des modifications. Dans le cas où la base RDF est la représentation d'un wiki sémantique, ces changements sont répercutés dans la page Wiki correspondante. Cette répercussion ne peut cependant pas tenir

compte du sens du texte dans la page Wiki. Il est donc nécessaire qu'un utilisateur humain reprenne ces modifications afin de réinstaurer une cohérence entre les connaissances textuelles et les connaissances formelles (triplets RDF). Nous appelons cette étape l'éditorialisation des changements sémantiques importés depuis les autres bases.

Reprenons l'exemple de WikiTaaable. La page du *Melon* contient le texte « Le melon est un fruit originaire d'Afrique » ou les mots *fruit* et *Afrique* portent des liens sémantiques (de sous-type et de provenance, respectivement). Supposons qu'on importe les changements d'une autre base, entrainant l'ajout du triplet <Melon, sous-type, Faux-Fruit> et la suppression du triplet <Melon, sous-type, Fruit>. Ce changement sera répercuté de manière automatique dans le wiki, en ajoutant en dessous du texte un lien (sans phrase) vers la rubrique *Faux-Fruit*, et supprimant le lien du mot *fruit* (mais sans retirer le mot lui même, afin de garder l'intégrité du texte).

L'utilisateur, informé de ces changements, pourra en prendre connaissance en consultant l'historique de la page, et décider des changements éditoriaux à apporter au texte pour en tenir compte. Dans ce cas, il est probable qu'il réintègre le lien vers *Faux-Fruit* dans la phrase originale, en la ré-écrivant : « Le melon est un faux-fruit originaire d'Afrique. »

À titre de preuve de concept, nous avons implémenté un wiki sémantique minimal permettant la répercussion des changements RDF dans le texte des pages, et l'éditorialisation a posteriori par l'utilisateur [7].

Assistance à l'acquisition de tests

Au sein du projet Kolflow, nous avons proposé un processus appelé K-CIP pour l'intégration de connaissances. K-CIP signifie « Knowledge Continuous Integration Process » et est, comme son nom l'indique, inspiré des approches d'intégration continues bien connues dans le monde du génie logiciel. L'objectif de K-CIP est de donner les lignes directrices permettant l'implémentation d'un processus de gestion et d'évolution des connaissances dans un environnement distribué. Les détails du processus, qui sont exposés dans [10], peuvent être résumé très simplement : nous considérons que chaque modification d'ontologie, aussi minime soit-elle, doit faire l'objet de tests. Le résultat des tests conditionne l'acceptation ou non de la modification. Ces tests sont appliqués à la fois pour les modifications locales, mais aussi lors de la fusion de deux ontologies entre utilisateurs, et même entre un utilisateur et une « ressource de référence ».

Le processus K-CIP repose sur une pierre angulaire : le moteur de tests. Ce moteur, développé par des chercheurs du LINA et du LORIA, permet de tester les résultats produits par un système et de les comparer à des résultats de référence. Le moteur de test interroge l'application à tester (dans notre cas d'utilisation, il s'agit de Taaable) et compare les résultats obtenus avec les résultats de référence. Les résultats des tests sont retournés sous forme de rapports qui spécifient : les résultats positifs (i.e. les résultats inchangés), les résultats négatifs (i.e. des éléments qui n'apparaissent plus dans les résultats alors qu'ils le devraient) et les résultats pour lesquels on ne peut se prononcer, faute d'information préalables). K-CIP est en mesure d'exploiter les résultats de cette forme afin de conditionner le processus d'évolution des bases de connaissances.

Toute la difficulté du processus est de construire et d'identifier les tests qui doivent être exécutés pour tester les modifications d'ontologies. Plus précisément, nous avons deux problèmes à résoudre :

- Construire un jeu de tests suffisamment riche au regard de la taille de l'ontologie

- Choisir dynamiquement les tests à effectuer lors de la modification d'une ontologie, afin d'éviter d'avoir à exécuter l'ensemble des tests (afin de résoudre des problèmes évidents de performances).

Pour traiter du premier point, nous adoptons une approche systématique qui consiste à enregistrer, sur un système initial, toutes les requêtes des utilisateurs, et à collecter, pour chacune de ces requêtes, du feedback simple sous la forme « oui, cette recette correspond à ma requête », « non, cette recette ne correspond pas à ma requête » ou rien (i.e. l'utilisateur ne se prononce pas). Ce travail (en cours au moment de la rédaction de ce livrable) est essentiellement mené par des membres de l'équipe Orpailleur du LORIA.

Pour traiter du second point, suite à un travail préliminaire avec les concepteurs du système de tests, nous avons pensé qu'il serait opportun d'utiliser également une approche à base de traces pour fournir une assistance à l'acquisition et à l'utilisation des tests, comme indiqué sur la **Erreur ! Source du renvoi introuvable.**.

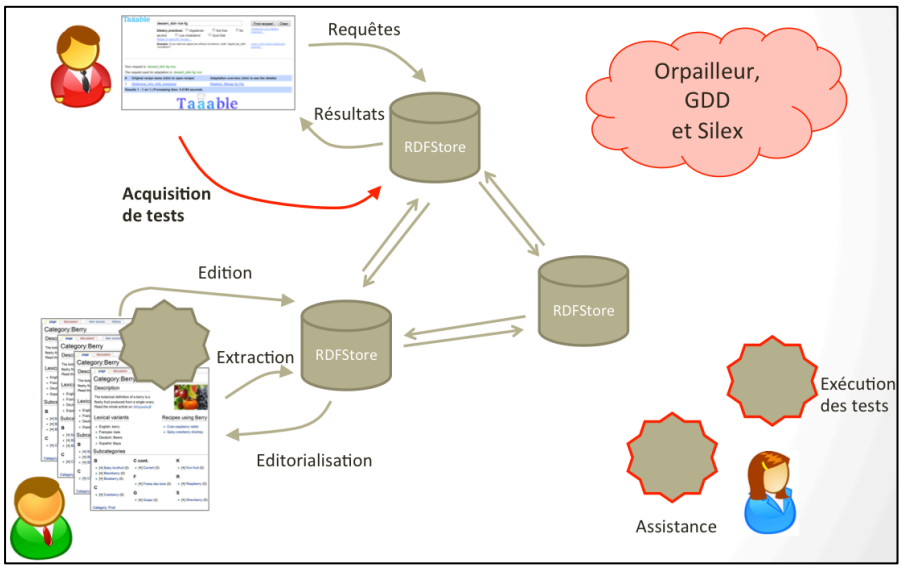

Figure 3 : Assistance à l'acquisition de tests

Nous avons envisagé plusieurs niveaux d'exploitation des traces, qui sont détaillés ci-dessous.

Afin d'amorcer le système, nous proposons d'observer un expert effectuant des modifications sur l'ontologie. Dans une telle phase, deux solutions sont possibles : soit l'expert exécute tous les tests à chaque modification de l'ontologie (terriblement coûteux à la fois en temps de calcul et en temps d'analyse des résultats par l'expert), ou bien il choisit lui-même les tests à effectuer, en s'appuyant sur ses connaissances d'expert. Etant donné la complexité des ontologies sur lesquelles nous travaillons, il est fortement probable que ce soit le second scénario qui soit retenu. Dans cette hypothèse, nous proposons, à partir des traces, d'appliquer des processus d'extraction de connaissances afin d'identifier des liens contextuels entre les modifications apportées à l'ontologie et les tests retenus. Notre hypothèse est que ces connaissances permettront de faire émerger des règles de sélection des tests qui, une fois généralisées, pourront être facilement appliquées dans d'autres contextes. L'approche d'extraction de connaissances à partir des traces telle que nous l'envisageons ici est similaire à des travaux que nous avons menés précédemment (voir [13]) et dont nous pourrions réutiliser et adapter les principes.

Un niveau d'exploitation plus complexe à mettre en œuvre mais qui semble aussi plus prometteur pour acquérir des connaissances fines et pertinentes est d'instrumenter plus spécifiquement l'interface utilisateur de l'application finale afin d'identifier « à la source » les problèmes ontologiques afin de conduire à l'élaboration de nouveaux tests. Pour illustrer cette forme d'exploitation, considérons le scénario suivant : un utilisateur interroge table afin de trouver des recettes de tarte au citron meringuée. Taaable retourne des recettes de tarte à la rhubarbe meringuée, de tarte aux pommes, et de meringue citronnée. L'utilisateur, partiellement satisfait, mais surtout soucieux d'améliorer la qualité du système, peut donner du feedback sur les différents résultats. Taaable possède déjà une interface d'acquisition des connaissances pour les connaissances d'adaptation [2]. A moindre coût, il serait possible de modifier cette interface afin de la coupler au système de tests, permettant ainsi d'identifier, grâce au feedback de l'utilisateur, quels tests auraient dû être exécutés afin d'éviter que les mauvais résultats soient retournés par Taaable.

Un troisième scénario, dont la mise en œuvre n'a pas encore été étudiée à ce jour, consiste à exploiter les traces d'utilisateurs (probablement experts) pour assister le processus de construction d'un jeu de tests pour un cas particulier. L'objectif ici est de guider le processus de construction de tests en s'appuyant sur un ensemble de bonnes pratiques qui émergeraient des traces des utilisateurs précédents.

L'ensemble de ces scénarios est actuellement à l'état de projet. Leur mise en œuvre fera l'objet d'une collaboration entre l'équipe SILEX du LIRIS et l'équipe Orpailleur du LORIA sur la prochaine période du projet.

Assistance à la fusion de connaissances

Prenons une communauté d'utilisateurs possédant tous leur propre instance d'une base de connaissances. Sur la Figure 4 par exemple, nous avons trois utilisateurs ayant chacun une instance de WikiTaaable [3] dans laquelle ils ont un ensemble de recettes de cuisine, ainsi qu'un ensemble d'ontologies représentant l'organisation des ingrédients, des types de plats, etc. Ces ontologies sont exploitées par le moteur de raisonnement du système Taaable qui permet d'adapter des recettes de cuisine aux contraintes des utilisateurs [8].

Chacun des utilisateurs peut modifier son wiki afin d'ajouter, de supprimer ou de modifier les recettes de cuisine et/ou les ontologies.

Lorsqu'un utilisateur veut partager le contenu de son wiki avec un autre utilisateur, le problème de la fusion de deux bases de connaissances se pose [9]. Cette fusion de connaissances peut aboutir à des conflits et l'utilisateur voulant intégrer des connaissances d'un autre wiki dans son propre wiki devra gérer ces conflits. Pour ce faire, nous proposons un assistant permettant de guider l'utilisateur à partir de traces d'interactions d'autres utilisateurs se trouvant dans la même situation ainsi qu'à partir des résultats d'un ensemble de tests validés par une communauté d'utilisateurs et/ou par des experts.

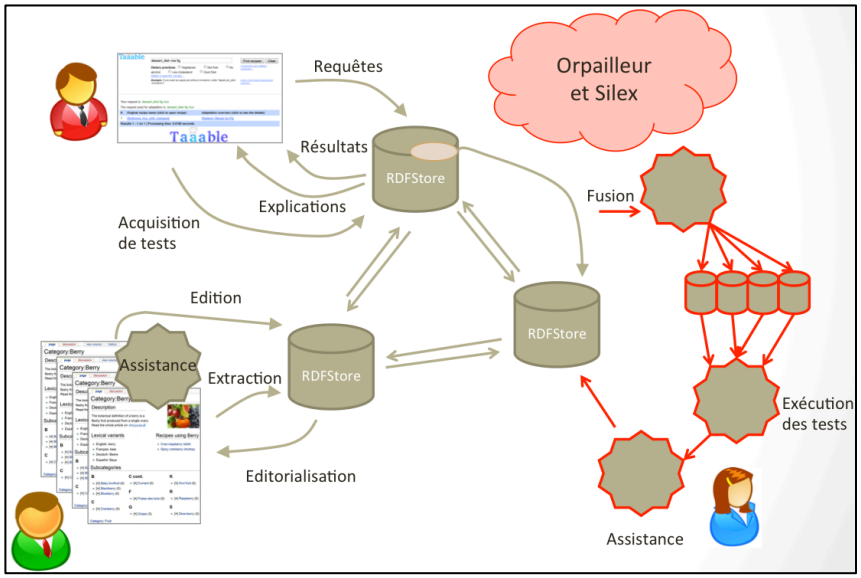

Figure 4 : Assistance à la fusion de connaissances.

Les différentes étapes suivies par les utilisateurs voulant fusionner deux bases de connaissances et celles de notre assistant sont représentées dans le diagramme de séquence de la Figure 5. L'outil de fusion présent sur cette figure correspond au moteur de raisonnement sur lequel travaille de l'équipe Orpailleur du LORIA. L'outil de tests sera le résultat d'un travail collaboratif entre les équipes Orpailleur du LORIA, GDD du LINA et Silex du LIRIS. L'assistant, son interface et le serveur d'échange seront proposés par l'équipe Silex du LIRIS. Pour permettre la mise en œuvre de ce scénario, nous nous sommes accordés sur le modèle de connaissances et le format d'échange.

Sur la Figure 5, nous pouvons ainsi voir qu'un premier utilisateur, noté A, possède son propre wiki et décide de partager une partie de celui-ci avec un autre utilisateur, noté B. L'utilisateur A décide des données à partager et l'assistant selectionne la partie du RDFStore correspondant. Cette sous-partie du RDFStore de l'utilisateur A est noté RDFStore A'. L'utilisateur B peut alors choisir d'intégrer, dans son wiki, tout ou partie des données partagées par A. Cette selection de l'utilisateur B correspond au RDFStore A''.

Notre assistant dispose donc d'un premier RDFStore, A'', correspondant à une sous-partie des données du wiki de l'utilisateur A, données mises à disposition par A et voulant être intégrées par B. Il dispose également d'un second RDFStore, B, contenant les données du wiki de l'utilisateur B.

Ces deux RDFStores sont alors fournis à l'outil permettant la fusion d'ontologie. Cet outil ne fournit pas un unique RDFStore résultant de la fusion des deux bases de connaissances. Pour chaque conflit détecté, il fournit quatre solutions possibles.

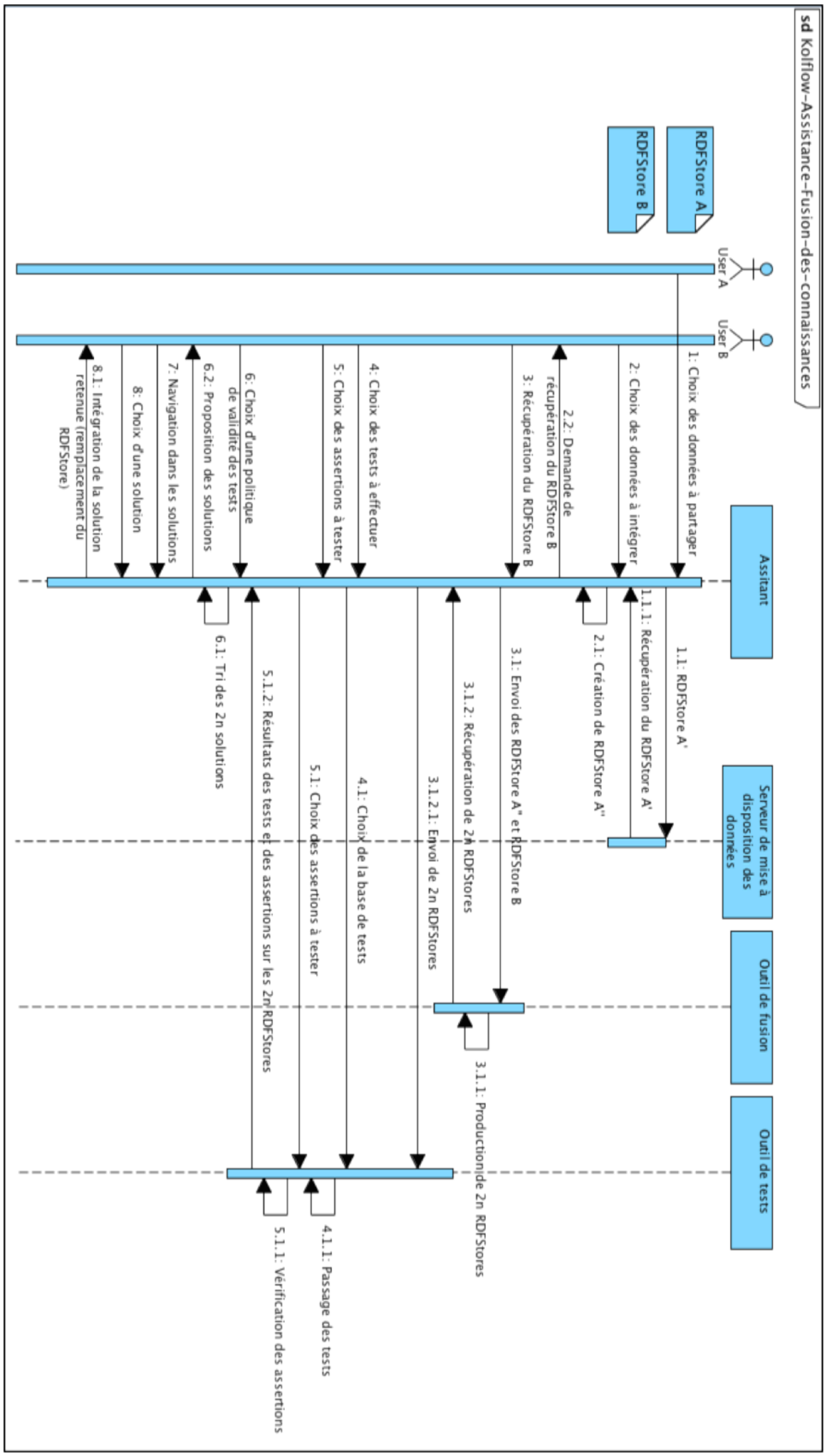

Figure 5 : Diagramme de séquence de l'assistance à la fusion des connaissances.

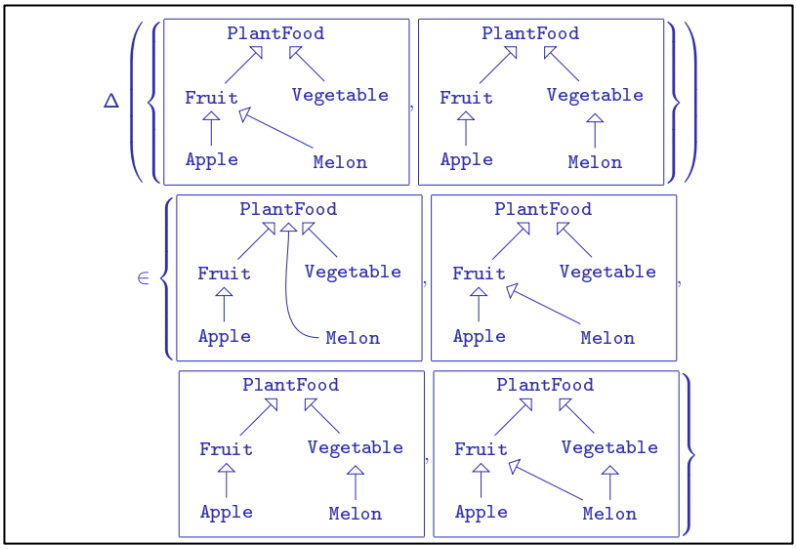

Figure 6 : Résultat de l'outil de fusion de deux ontologies.

Prenons l'exemple de l'ontologie des ingrédients dans WikiTaaable. Dans l'ontologie de A, le concept *Melon* se trouve sous le concept *Fruit* , qui se trouve lui même sous le concept de *PlantFood*. Alors que dans l'ontologie de B, le concept *Melon* se trouve sous le concept *Vegetable* , qui se trouve lui même sous le concept de *PlantFood.* Dans ce cas, l'outil de fusion propose quatre solutions (cf. Figure 6) :

- si l'on décide de supprimer simplement les conflits : le concept *Melon* ne se trouve ni sous le concept *Fruit* , ni sous le concept *Vegetable* mais directement sous le concept de *PlantFood* ;
- si l'on considère que les connaissances de l'utilisateur A doivent prévaloir : le concept *Melon* reste sous le concept *Fruit* , et le lien entre le concept *Melon* et le concept *Vegetable* est supprimé ;
- si l'on considère que les connaissances de l'utilisateur B doivent prévaloir : le concept *Melon* reste sous le concept *Vegetable* , et le lien entre le concept *Melon* et le concept *Fuit* n'est pas gardé ;
- si l'on souhaite conserver tous les liens définis par les deux utilisateurs : le concept *Melon* se trouve à la fois sous le concept *Fruit* , et sous le concept *Vegetable*.

Cet outil de fusion retourne donc à l'assistant l'ensemble des RDFStores pouvant être construits à partir des deux RDFStores en entrée. Dans le cas d'un seul conflit, il retourne ainsi quatre RDFStores.

Afin de guider l'utilisateur B dans le choix de la base de connaissances à conserver (et du RDFStore associé), nous exploitons, dans un premier temps, un outil de tests proposé dans le cadre du projet Kolflow [10].

L'outil de tests repose sur une base de tests acquis précédemment (voir section « Assistance à l'acquisition de tests »). Dans cette base de tests, un test est un triplet contenant (cf. Figure 7) :

- une requête sémantique,
- une ontologie,
- un ensemble de réponses à cette requête où chaque réponse est notée comme acceptée, refusée ou inconnue par un ensemble d'experts.

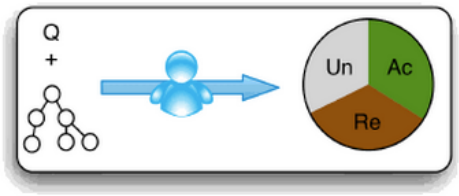

Figure 7 : Un test dans le cadre de l'approche K-CIP.

Dans notre exemple d'utilisation de Taaable et de sa base de connaissances WikiTaaable, une requête sémantique est une demande de recettes faite par un utilisateur, contenant un certain nombre de contraintes. Par exemple, « je désire une recette de tarte au melon, sans sucre, sans œuf ». Cette requête est soumise à un moteur de raisonnement permettant soit de retrouver une recette existante dans WikiTaaable, soit d'adapter une recette, à partir d'une recette existante et des connaissances connues dans les différentes ontologies contenues dans WikiTaaable. Les recettes proposées par le système de raisonnement sont alors annotées par l'utilisateur comme bonnes ou mauvaises recettes. Si l'utilisateur n'annote pas certaines recettes, elles sont annotées comme inconnues.

L'outil de tests utilise donc la base de tests en exécutant les requêtes sémantiques contenues dans les tests mais sur une nouvelle ontologie. Les nouveaux résultats peuvent alors être comparés à ceux correspondant à l'ontologie de référence dans les tests pour savoir si la nouvelle ontologie permet de retrouver plus de bonnes réponses, plus de mauvaises, autant de bonnes mais plus de mauvaises, etc. Ces différentes comparaisons sont appellées des assertions.

L'outil d'assistance utilise donc l'outil de tests pour exécuter les tests sur chacun des RDFStores récupérés en sortie de l'outil de fusion et pour comparer une série d'assertions relatives aux résultats de ces tests.

Ces assertions, validées ou non, sont utilisées pour trier les RDFStores candidats en fonction d'une des politiques prédéfinies ou d'une politique définie par l'utilisateur. Ces politiques contiennent des règles de priorité entre les assertions testées. Par exemple, il est préférable d'avoir « autant de bonnes recettes et moins de mauvaises » que « plus de bonnes recettes et plus de mauvaises ».

L'assistant propose donc les bases de connaissances (RDFStores) triées à l'utilisateur selon la politique de validité des tests retenue. Puis il complète son assistance en exploitant les traces d'interactions de la communauté d'utilisateurs avec d'une part, les wikis permettant d'éditer les connaissances, et d'autre part, l'outil d'assistance à la fusion des connaissances (cf. Figure 8). Ces traces sont sockées dans un SGBT (Système de Gestion de Bases de Traces), qui fournit, en plus de la fonctionnalité de stockage des traces, des fonctionnalités permettant de faire des requêtes et des transformations sur les traces [11]. Dans le cadre de notre assistant, nous exploitons le kTBS [12] pour stocker et interroger les traces. À partir de l'exploitation des traces, l'assistant va pouvoir indiquer les politiques de tests que l'utilisateur utilise le plus souvent, celles utilisées par des utilisateurs se trouvant dans le même contexte que lui (par exemple, même concept mis en cause, ou mêmes caractéristiques utilisateur). Il peut aussi reconstruire l'historique des concepts concernés : par qui ils ont été modifiés, à quel moment et dans quel contexte ?

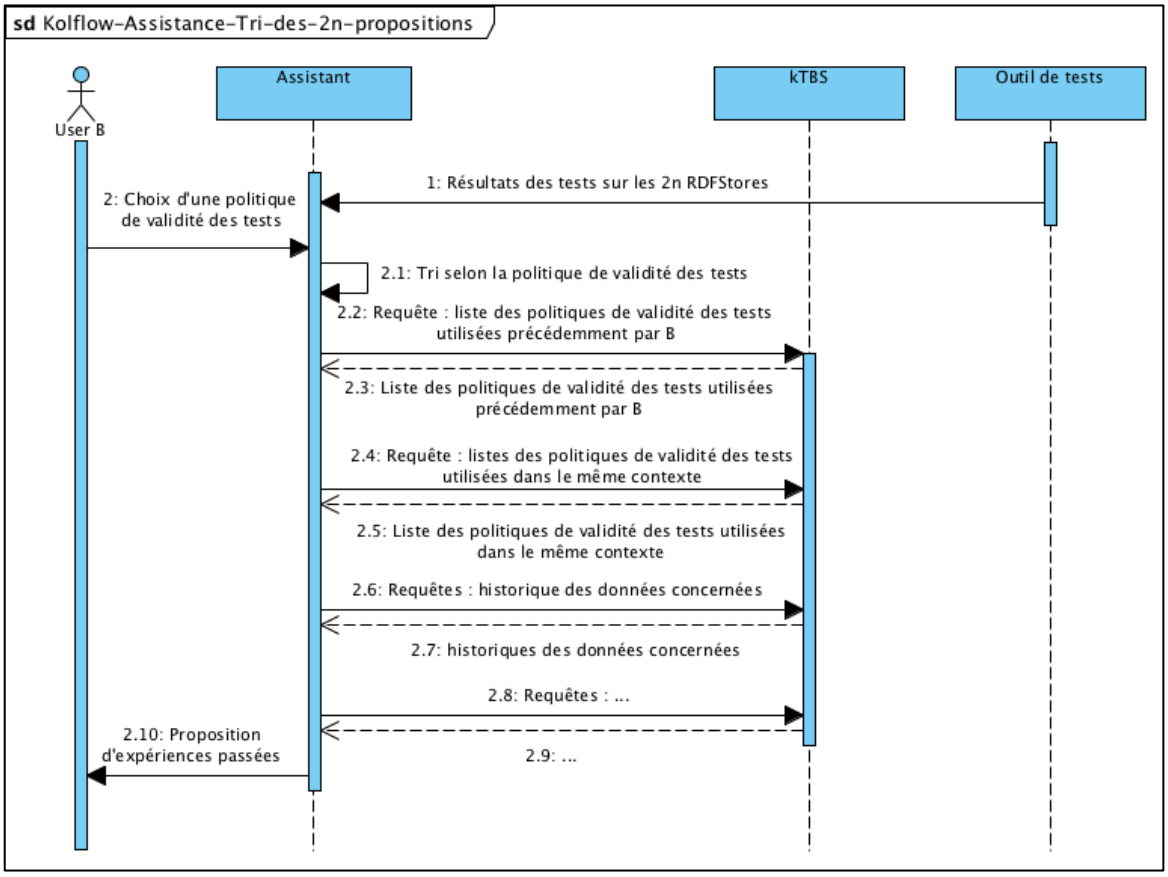

Figure 8 : Diagramme de séquence de l'étape de tri des propositions dans l'assistance à la fusion des connaissances.

Assistance à l'évolution des connaissances

Lorsque qu'un utilisateur modifie les connaissances contenues dans son wiki, les résultats des raisonnements pouvant être faits sur cette base de connaissances s'en trouvent modifiés. L'objectif de l'utilisateur est que les modifications apportées améliorent les résultats. Pour vérifier ce fait, il peut utiliser l'outil de tests décrit dans la section précédente. Cet outil va comparer le résultat d'un certain nombre d'assertions sur les résultats obtenus en faisant passer les tests sur l'ancienne version de la base de connaissances et ceux obtenus à partir de la nouvelle base de connaissances.

L'assistance fournie à l'utilisateur dans ce contexte (cf. Figure 9) porte sur le choix des tests à effectuer, les assertions à utiliser et l'explication du delta entre les deux ensembles résultats. Toutes ces informations peuvent être déduites à partir de l'exploitation des traces d'interaction d'une communauté d'utilisateurs avec l'application permettant de vérifier que l'évolution des connaissances est pertinente.

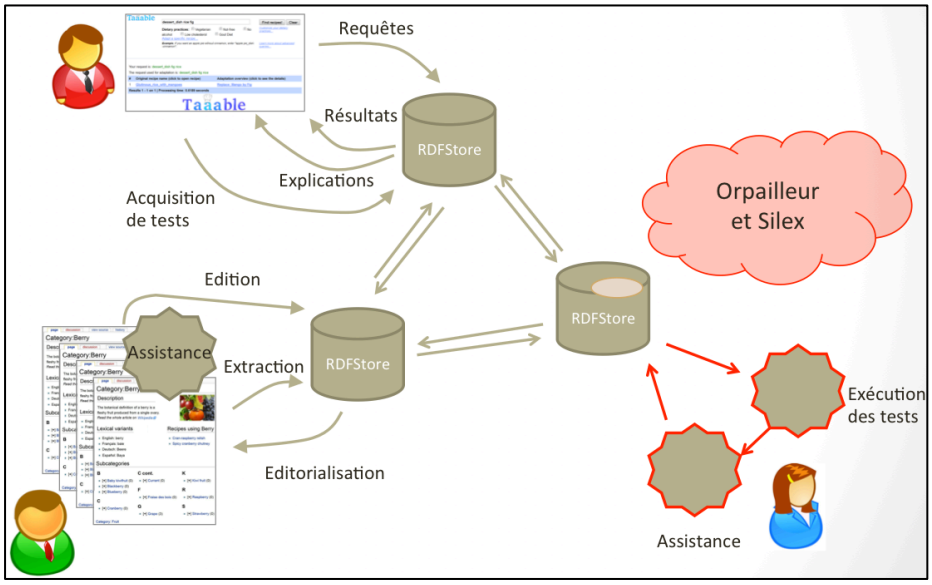

Figure 9 : Assistance à l'évolution des connaissances.

Le mécanisme mis en œuvre pour cet assistant est décrit sur la Figure 10. Lorsqu'un utilisateur modifie ses connaissances, une sauvegarde de l'état initial est effectuée sous la forme d'un RDFStore, nommé A_0. Lorsque l'utilisateur demande à passer les tests unitaires pour vérifier la validité de ses modifications, un second RDFStore est récupéré par l'assistance : le RDFStore courant, noté A. Ces deux RDFStores sont alors envoyés à l'outil de tests.

Afin de déterminer les tests à effectuer, l'assistance recherche dans les traces de la communauté, les tests effectués dans un contexte similaire. L'utilisateur peut alors choisir les tests à effectuer. La même opération a lieu pour le choix des assertions à vérifier.

Une fois les tests passés et les résultats des assertions connus, l'assistant cherche, en interrogeant les traces, des explications sur le delta obtenu entre les tests passés sur chacune des deux versions du RDFStore. Ces explications sont alors proposées à l'utilisateur. Celui-ci peut ainsi choisir de garder ou non les modifications apportées.

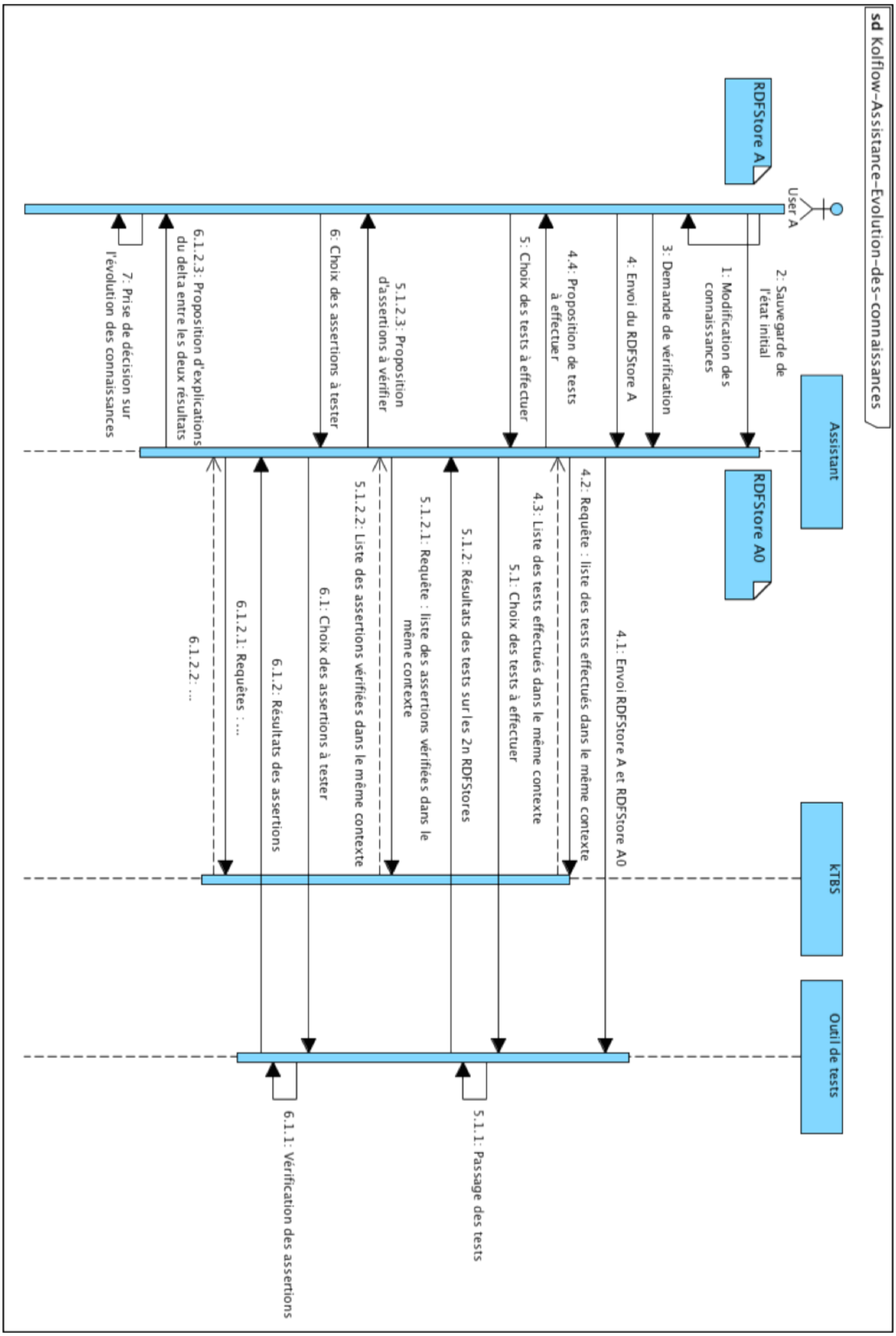

Figure 10 : Diagramme de séquence de l'assistance à l'évolution des connaissances.

Conclusion

Les différents scénarios proposés dans ce rapport couvrent les différentes assistances à la co-construction de connaissances envisagées dans le cadre du projet Kolflow.

Ces scénarios reposent tous sur l'utilisation du raisonnement à partir de traces, et s'appuient donc sur un assistant respectant l'architecture présentée sur la Figure 11. Chaque assistant s'appuie ainsi sur un module de collecte de traces, nommé Collectra [14], qui capture les interactions des utilisateurs avec les différents sites web et qui stocke cette expérience observée dans un système de gestion de traces. Le système de gestion de traces que nous exploitons est le kTBS. En s'appuyant sur son module de collecte et sur les fonctionnalités offertes par le kTBS, le moteur d'assistant peut ainsi fournir aux utilisateurs des indicateurs leur permettant de construire, faire évoluer ou fusionner leurs bases de connaissances. Ces utilisateurs pourront interagir avec l'assistant soit par l'intermédiaire des sites web, comme lors de l'édition et de l'éditorialisation des connaissances, soit directement, comme lors de la fusion de connaissances ou lors d'évolution de celles-ci.

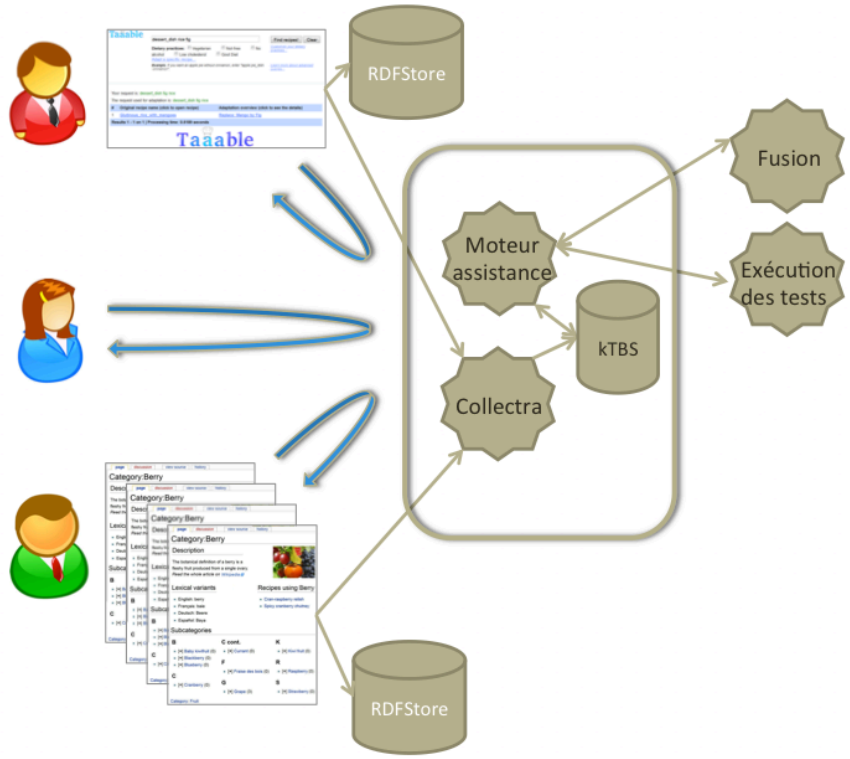

Figure 11 : Architecture de l'assistant alter-égo de Kolflow.

Références bibliographiques et webographiques

[1] Projet Kolflow. http://kolflow.univ-nantes.fr/ (consulté en février 2013).

[2] F. Badra, J. Cojan, A. Cordier, J. Lieber, T. Meilender, A. Mille, P. Molli, E. Nauer, A. Napoli, H. Skaf-Molli, and Y. Toussaint. *Knowledge acquisition and discovery for the textual case-based cooking system WIKITAAABLE*. In S. J. Delany, editor, 8th International Conference on Case-Based Reasoning - ICCBR 2009, Workshop Proceedings, pages 249–258, Seattle, United States, Juillet 2009.

[3] WikiTaaable. http://wikitaaable.loria.fr/index.php/Main_Page (consulté en février 2013).

[4] Semantic MediaWiki. http://semantic-mediawiki.org/ (consulté en février 2013).

[5] H. Skaf-Molli, G. Canals, P. Molli. *DSMW: a distributed infras- tructure for the cooperative edition of semantic wiki documents*. In Proceedings of the 10th ACM symposium on Document engineering, DocEng'10, pages 185–186, New York, NY, USA, 2010.

[6] M. Lefevre, A-H. Le, T. Dubois, A. Cordier. *D6.3.KOLFOW - Assistance à l'outil DSMW. Rapport de mise en œuvre*. Rapport de recherche RR-LIRIS-2013-004, 2013.

[7] SemWiki. https://github.com/pchampin/semwiki (consulté en février 2013).

[8] J. Lieber. *Le moteur de raisonnement à partir de cas de WikiTaaable*. In 17ème atelier sur le raisonnement à partir de cas – RàPC, 2009.

[9] A. Cordier, J. Lieber, J. Stevenot. *Towards an operator for merging taxonomies*. In Belief change, nonmonotonic reasoning, conflict resolution: BNC, Workshop at ECAI 2012., Montpellier, France, 2012..

[10] H. Skaf-Molli, E. Desmontils, E. Nauer, G. Canals, A. Cordier, M. Lefevre, P. Molli, Y. Toussaint. *Knowledge Continuous Integration Process (K-CIP)*. In Workshop Semantic Web Collaborative Spaces, WWW2012, Lyon, France, 17 avril 2012.

[11] L. S. Settouti. *Systèmes à base de traces modélisées - Modèles et langages pour l'exploitation des traces d'interactions*. Thèse de doctorat en informatique, Université Claude Bernard Lyon 1, janvier 2011.

[12] P-A. Champin, Y. Prié, O. Aubert, F. Conil, D. Cram. *kTBS: Kernel for Trace-Based Systems*, http://liris.cnrs.fr/sbt-dev/ktbs/, 2011.

[13] R. Zarka, A. Cordier, E. Egyed-Zsigmond, A. Mille. *Contextual Trace-Based Video Recommendations.* Dans 21st international conference companion on World Wide Web (WWW-XperienceWeb'12), Lyon, France. pp. 751-754. WWW '12 Companion . ACM New York, NY, USA. ISBN 978-1-4503-1230-1. 2012.

[14] A-H. Le, M. Lefevre, A. Cordier, H. Skaf-Molli. *Collecting Interaction Traces in Distributed Semantic Wikis*. Conférence WIMS'13 (Web Intelligence, Mining and Semantics), Madrid, Espagne, 12-14 juin 2013.