D6.3. Assistance à l'outil DSMW
Rapport de mise en œuvre

Marie Lefevre, Anh-Hoang Le, Tristan Dubois, Amélie Cordier

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

Assistance à l'utilisation de wikis sémantiques distribués

Le projet Kolflow (Man-machine collaboration in continuous knowledge-construction flows) [1] vise à concevoir un espace social sémantique. Cet espace doit permettre à des humains et des agents artificiels de manipuler ensemble des connaissances.

En effet, pour pouvoir raisonner, les systèmes informatiques exploitent des connaissances. Ces connaissances sont issues de l'expertise humaine et sont traduites par des ingénieurs de la connaissance en données exploitables par les agents informatiques. Lorsque les connaissances sont mises à jour par les agents artificiels, seuls des personnes connaissant le formalisme de description sont aptes à comprendre les modifications. De la même manière, lorsque l'expertise humaine évolue, il faut de nouveau passer par un expert pour modifier les données du système. L'idée du projet Kolflow est de permettre à des humains et des agents artificiels de travailler ensemble sur les mêmes connaissances. Pour cela, il est nécessaire d'avoir un formalisme commun et de permettre à plusieurs humains et/ou plusieurs agents de travailler ensemble sur les connaissances.

DSMW (Distributed Semantic Media Wiki) [2] est une extension de SMW (Semantic MediaWiki) [3] permettant de déployer un réseau de wikis et de leur faire partager certaines de leurs pages. Comme sur un wiki classique, les utilisateurs peuvent créer ou modifier des pages directement depuis leur navigateur. Cette édition passe par l'utilisation d'une syntaxe simple, pouvant être manipulée graphiquement. Mais comme DSMW est une extension de SMW, les pages peuvent contenir en plus des informations textuelles, des annotations sémantiques sous forme de triplets RDF.

Avec DSMW, les utilisateurs peuvent, contrairement aux wikis classiques, mettre à disposition tout ou partie des pages de leur wiki. Ces pages peuvent ensuite être récupérées par d'autres utilisateurs pour les importer dans leur propre wiki. Lorsqu'un utilisateur veut importer une page d'un autre utilisateur dans son wiki, deux cas se présentent :

- soit la page n'existe pas dans le wiki, et elle est ajoutée ;
- soit la page existe déjà, et les mécanismes associés à DSMW permettent de fusionner les deux pages.

Cet outil permet donc à plusieurs utilisateurs de construire l'espace social sémantique voulu dans le projet Kolflow. Toutefois, DSMW n'est pas simple d'utilisation. Pour des utilisateurs novices, il faut apprendre d'une part à utiliser un wiki classique et à faire des annotations en utilisant des triplets RDF, puis d'autre part, à utiliser les fonctions de partage d'une page. Concrètement, pour partager une page, il faut créer

autant de canaux de communication que l'on a de paires de wikis souhaitant partager des données. Chaque canal se crée par des actions spécifiques sur chacun des deux wikis mettant en place le canal. Ensuite, pour partager des pages, il faut sur le premier wiki les mettre à disposition et sur l'autre les récupérer. Toutes ces actions se font via l'utilisation d'une interface de configuration assez complexe à prendre en main.

La première étape de l'assistance dans le cadre de Kolflow a donc été de fournir une aide aux nouveaux utilisateurs de l'outil à manipuler.

Principe

L'assistance aux utilisateurs peut prendre diverses formes [4]. Dans le cadre du projet Kolflow, nous souhaitons utiliser le raisonnement à partir de traces [5] pour mettre en œuvre une assistance. Dans ce rapport, nous nous intéressons uniquement à l'assistance à l'outil DSMW à partir de traces.

Le principe de cette assistance est d'identifier les différentes tâches pouvant être faites par un utilisateur de DSMW, et de lui fournir pour chaque tâche :

- 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.

Background

Dans cette section, nous présentons les travaux sur lesquels s'appuie notre proposition. Nous présentons ainsi la théorie de la trace, et notamment la notion de trace modélisée et de système de gestion de traces, puis les outils kTBS et DSMW.

Théorie de la trace

La théorie de la trace [6] fournit quelques définitions de base sur les traces, les méthodologies et les outils permettant de manipuler les traces.

Une **trace** est définie comme un ensemble d'éléments observés, appelés ***obsels***. Les obsels sont temporellement situés dans des traces. Un **modèle de trace** définit la structure et les types d'obsels que l'on peut s'attendre à trouver dans la trace, ainsi que les relations entre ces obsels. Une **trace modélisée** (M-Trace) est l'association d'une trace et de son modèle de trace.

Toutes les traces sont enregistrées dans un **Système de Gestion de Bases de Traces** (SGBT). Le processus de collecte produit des traces primaires. Au sein du SGBT, des transformations (filtrage, agrégation, etc.) peuvent être appliquées sur ces traces, produisant ainsi des **traces transformées** qui sont aussi stockées dans le SGBT. Un SGBT possède trois composants principaux :

- Le système de collecte : il est utilisé pour collecter les données observées à partir de différentes sources (logs, enregistrements vidéo, évènements de l'interface, messages du serveur, etc.) et les stocker sous forme d'obsels dans une trace.
- Le système de transformations : il permet d'effectuer plusieurs transformations telles que le filtrage, la réécriture, la fusion d'obsels à partir d'une ou plusieurs M-Traces.
- Le système de requêtes : il permet l'exécution de requêtes sur des traces. Les requêtes permettent de calculer différents résultats sur des traces existantes (le nombre d'obsels, la fréquence d'un obsel, la fréquence d'un motif, etc.).

Chaque ressource dans un SGBT est identifiée par un URI unique (Uniform Resource Identifier [7]). Grâce à ces URI, les utilisateurs et les applications peuvent accéder et manipuler les ressources du SGBT facilement.

Une formalisation complète de la théorie de la trace, présentant la sémantique des modèles de traces, des traces, des requêtes, et des transformations peut être trouvée dans [6].

kTBS

Le kTBS [8] est une implémentation d'un Système de Gestion de Traces proposé par l'équipe Silex du laboratoire LIRIS. Cette implémentation repose sur la représentation des données en RDF et une communication via le protocole HTTP.

Dans le kTBS, une trace contient un ensemble d'obsels. Chaque obsel est muni :

- d'un type ;
- d'une date de début et d'une date de fin ;
- d'un sujet ;
- d'un ensemble d'attributs ;
- de relations avec d'autres obsels.

Pour être exploitable, les traces doivent être explicitement modélisées. Un modèle de trace définit:

- les types d'obsels ;
- les attributs associés à chaque type ;
- les relations possibles entre les types ;
- éventuellement, des contraintes temporelles sur les types d'obsels.

A titre d'exemple, la Figure 1 montre un obsel présent dans le kTBS de l'assistant à l'outil DSMW. Cet obsel est de type « KeyEvent », possède une date de début et de fin stockées dans les attributs « HasBegin » et « HasEnd » ainsi qu'un ensemble d'attributs. Ces différents attributs sont définis dans le modèle associé à la trace dont un extrait est fourni dans la Figure 2.

Chaque kTBS peut ainsi contenir plusieurs traces dites premières, puisqu'elles sont directement issues d'un processus de collecte. Chacune de ces traces est associée à son propre modèle, également contenu dans le kTBS.

Le kTBS peut de plus contenir des traces transformées. Chaque trace transformée est définie par :

- une ou plusieurs trace(s) source(s) ;
- une méthode de transformation ;
- des paramètres éventuels pour cette méthode.

Les traces sources peuvent être des traces premières ou des traces transformées.

```
@prefix : <http://localhost:8001/base1/model1/>.
@prefix _8: <http://localhost:8002/base1/PrivateTrace_0/>.
@prefix _9: <http://localhost:8002/base1/model1/>.
@prefix ktbs: <http://liris.cnrs.fr/silex/2009/ktbs#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfrest: <http://liris.cnrs.fr/silex/2009/rdfrest#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xml: <http://www.w3.org/XML/1998/namespace>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

 _8:keyevent_108167052 a _9:KeyEvent;
     ktbs:hasBegin "1552479982"^^<http://www.w3.org/2001/XMLSchema#integer>;
     ktbs:hasBeginDT "2012-05-14T08:43:53.995Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>;
     ktbs:hasEnd "1552479982"^^<http://www.w3.org/2001/XMLSchema#integer>;
     ktbs:hasEndDT "2012-05-14T08:43:53.995Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>;
     ktbs:hasTrace <http://localhost:8002/base1/PrivateTrace_0/>;
     _9:character "d";
     _9:description """keytype = KeyPress keycode = 100 character = d element_id = wpTextbox1""";
     _9:keycode "100";
     _9:keytype "KeyPress";
     _9:location "http://localhost/wiki2/index.php?title=Main_Page&action=edit";
     _9:site_id "A272A93EE230EA8BC37CC127A9A3BA11";
     _9:title "character = d";
     _9:user_id "0";
     _9:user_name "127.0.0.1".
```

Figure 1 : Exemple d'un obsel de type KeyEvent dans une trace du kTBS.

```
@prefix : <http://localhost:8001/base1/model1/>.
@prefix ktbs: <http://liris.cnrs.fr/silex/2009/ktbs#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfrest: <http://liris.cnrs.fr/silex/2009/rdfrest#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xml: <http://www.w3.org/XML/1998/namespace>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

<http://localhost:8002/base1/> ktbs:contains <>.

<> a ktbs:TraceModel.

 <Obsel> a ktbs:ObselType.

 <KeyEvent> a ktbs:ObselType;
     ktbs:hasSuperObselType <Obsel>.

 <character> a ktbs:AttributeType;
     ktbs:hasAttributeDomain <KeyEvent>;
     ktbs:hasAttributeRange xsd:string.

 <description> a ktbs:AttributeType;
     ktbs:hasAttributeDomain <Obsel>;
     ktbs:hasAttributeRange xsd:string.

 <keycode> a ktbs:AttributeType;
     ktbs:hasAttributeDomain <KeyEvent>;
     ktbs:hasAttributeRange xsd:string.

 <keytype> a ktbs:AttributeType;
     ktbs:hasAttributeDomain <KeyEvent>;
     ktbs:hasAttributeRange xsd:string.
```

Figure 2 : Extrait du modèle de traces du kTBS définissant le type d'obsel KeyEvent.

Le kTBS propose plusieurs méthodes de transformation.

Le **filtre temporel** ne conserve que les obsels situés dans un intervalle temporel donné. Il prend en entrée une unique trace source et fournit une trace transformée dont le modèle est le même que la trace en entrée. Il a besoin de deux paramètres : la date de début et la date de fin. On peut ainsi filtrer une trace pour ne garder par exemple que les obsels des trois derniers jours et dans un intervalle de temps précis comme le montre l'exemple de la Figure 3.

```
@prefix : <http://localhost:8001/base1/model1/>.
@prefix _8: <http://localhost:8002/base1/DateFilterTrace_1491728958/@>.
@prefix ktbs: <http://liris.cnrs.fr/silex/2009/ktbs#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfrest: <http://liris.cnrs.fr/silex/2009/rdfrest#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xml: <http://www.w3.org/XML/1998/namespace>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

 <http://localhost:8002/base1/> ktbs:contains <http://localhost:8002/base1/DateFilterTrace_1491728958/>.

 <> ktbs:descriptionOf <http://localhost:8002/base1/DateFilterTrace_1491728958/>.

 <http://localhost:8002/base1/DateFilterTrace_1491728958/> a ktbs:ComputedTrace;
     rdfs:label "Username: 127.0.0.1 > start: 2012-05-22 14:00:00.000 finish: 2012-07-02 00:00:00.000";
     ktbs:compliesWithModel "?";
     ktbs:hasMethod ktbs:filter;
     ktbs:hasModel <http://localhost:8002/base1/model1/>;
     ktbs:hasObselCollection _8:obsels;
     ktbs:hasOrigin "2012-05-22T13:28:53.695Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>;
     ktbs:hasParameter "finish=3493866305",
         "start=1866305";
     ktbs:hasSource <http://localhost:8002/base1/PrivateTrace_0127.0.0.1/>.
```

Figure 3 : Filtre temporel défini dans un kTBS.

```
@prefix : <http://localhost:8001/base1/model1/>.
@prefix _8: <http://localhost:8002/base1/KeyEventTrace_16386380/@>.
@prefix ktbs: <http://liris.cnrs.fr/silex/2009/ktbs#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfrest: <http://liris.cnrs.fr/silex/2009/rdfrest#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xml: <http://www.w3.org/XML/1998/namespace>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

 <http://localhost:8002/base1/> ktbs:contains <http://localhost:8002/base1/KeyEventTrace_16386380/>.

 <> ktbs:descriptionOf <http://localhost:8002/base1/KeyEventTrace_16386380/>.

 <http://localhost:8002/base1/KeyEventTrace_16386380/> a ktbs:ComputedTrace;
     rdfs:label "Username: 127.0.0.1 > Action: KeyEvent";
     ktbs:compliesWithModel "?";
     ktbs:hasMethod ktbs:obsel_type_filter;
     ktbs:hasModel <http://localhost:8002/base1/model1/>;
     ktbs:hasObselCollection _8:obsels;
     ktbs:hasOrigin "2012-04-26T09:29:14.013Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>;
     ktbs:hasParameter "obsel_types=http://localhost:8002/base1/model1/KeyEvent";
     ktbs:hasSource <http://localhost:8002/base1/PrivateTrace_0/>.
```

Figure 4 : Filtre structurel sur les obsels de type KeyEvent défini dans un kTBS.

Le **filtre structurel** ne conserve que les obsels vérifiant certaines contraintes. Il prend en entrée une unique trace source et fournit en sortie une trace transformée dont le modèle est le même que la trace en entrée. Il a besoin en paramètre du type d'obsels à conserver et éventuellement des valeurs ou des plages

de valeurs pour certains attributs. On peut ainsi ne garder dans une trace de navigation Internet que les obsels relatifs aux clics souris ou à des saisies clavier, comme le montre l'exemple de la Figure 4 où l'on ne garde que les obsels de type « KeyEvent ».

La **fusion** fait l'union des obsels de plusieurs traces. Elle prend en entrée plusieurs sources, et fournit une trace dont le modèle doit englober l'ensemble des modèles des traces sources. Cette transformation n'a pas de paramètre. On peut ainsi fusionner les traces de deux utilisateurs ou les traces collectées à partir de deux outils différents.

La **réécriture de motifs** crée de nouveaux obsels sur la base d'un motif recherché dans la trace source. Elle prend en entrée une seule trace source et fournit une trace dont le modèle dépend de la réécriture. Elle a besoin en paramètre de la définition du motif recherché et de la définition du motif produit. On peut ainsi dans une trace regrouper tous les obsels indiquant des saisies clavier pour ne former qu'un unique obsel contenant le texte saisi.

```
 <http://localhost:8002/base1/SparqlTrace_6b62454595e4d97fd9eba77ad525672f/> a ktbs:ComputedTrace;
     rdfs:label "Sparql request for http://localhost:8002/base1/FusionTrace_fc6ab8e41a3aef48f56d92575e37c2b2/.";
     ktbs:compliesWithModel "yes";
     ktbs:hasMethod ktbs:sparql;
     ktbs:hasModel <http://localhost:8002/base1/model1/>;
     ktbs:hasObselCollection _8:obsels;
     ktbs:hasOrigin "2012-05-22T14:08:14.619Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>;
     ktbs:hasParameter """sparql=PREFIX : <http://localhost:8002/base1/model1/>
PREFIX ktbs: <http://liris.cnrs.fr/silex/2009/ktbs#>

CONSTRUCT {
    [ a :MouseEvent ;
      ktbs:hasTrace <%(__destination__)s> ;
      ktbs:hasBegin ?begin ;
      ktbs:hasBeginDT ?begindt ;
      ktbs:hasEnd ?end ;
      ktbs:hasEndDT ?enddt ;
      :user_id ?user_id ;
      :user_name ?user_name ;
      :site_id ?site_id ;
      :title ?title ;
      :description ?description ;
      :location ?location ;
      :element_type ?element_type ;
      :element_id ?element_id ;
      :element_value ?element_value ;
      :element_text ?element_text ;
      :element_checked ?element_checked ;
      :element_src ?element_src ;
      :element_href ?element_href ;
      ktbs:hasSourceObsel ?obsel ;
    ] .
} WHERE {
    ?obsel a :MouseEvent ;
      ktbs:hasBegin ?begin ;
      ktbs:hasBeginDT ?begindt ;
      ktbs:hasEnd ?end ;
      ktbs:hasEndDT ?enddt ;
      :user_id ?user_id ;
      :user_name ?user_name ;
      :site_id ?site_id ;
      :title ?title ;
      :description ?description ;
      :location ?location ;
      :element_type ?element_type ;
      :element_id ?element_id ;
      :element_value ?element_value ;
      :element_text ?element_text ;
      :element_checked ?element_checked ;
      :element_src ?element_src ;
      :element_href ?element_href.
      FILTER regex(?element_href, "&action=edit&redlink=1", "i") .
}""";
     ktbs:hasSource <http://localhost:8002/base1/FusionTrace_fc6ab8e41a3aef48f56d92575e37c2b2/>.
```

Figure 5 : Méthode de réécriture de motifs définie dans un kTBS.

La méthode **composite** permet de composer plusieurs méthodes (séquentiellement ou parallèlement). Elle prend en entrée une seule trace source et fournit une trace dont le modèle dépend des méthodes composées. Cette méthode de transformation permet par exemple de partir d'une trace source sur laquelle on filtrerait en parallèle les obsels pour ne garder dans une trace que les actions d'édition des pages wiki, comme le montre l'exemple de la transformation de la Figure 5, et dans l'autre que les sauvegardes de pages pour ensuite fusionner les deux traces et obtenir une trace contenant les éditions de pages d'un wiki, sauvegardées ou non.

La méthode **dérivée** permet de spécialiser une méthode existante. Ainsi, on peut fixer des valeurs pour certains paramètres d'une méthode parente, et éventuellement définir des paramètres propres à la méthode dérivée. Cette méthode permet de faciliter l'utilisation de méthodes très génériques en cristallisant une manière particulière de les utiliser. On peut par exemple définir une méthode de filtre structurel permettant de ne garder que les clics souris puis à partir de cette méthode, définir autant de méthodes que l'on veut permettant de ne garder que les clics sur des boutons, sur des liens, etc.

DSMW

DSMW [2] est une extension de Semantic MediaWiki (SMW) [3] développée par des chercheurs du LINA et du LORIA. L'extension DSMW ajoute aux wikis sémantiques classiques les fonctionnalités nécessaires au partage des pages wiki entre les différents wikis du réseau. Cette extension supporte notamment l'édition multi-synchrone [9] qui permet la modification simultanée de pages sur des serveurs différents, la propagation de ces modifications sur le réseau et la gestion des conflits survenant éventuellement suite à la propagation des modifications [10]. Les modifications propagées sont appelées des mises à jour. La propagation des mises à jour est basée sur des opérations de publication-souscription. Ces opérations sont utilisées pour créer des canaux de communication entre les serveurs.

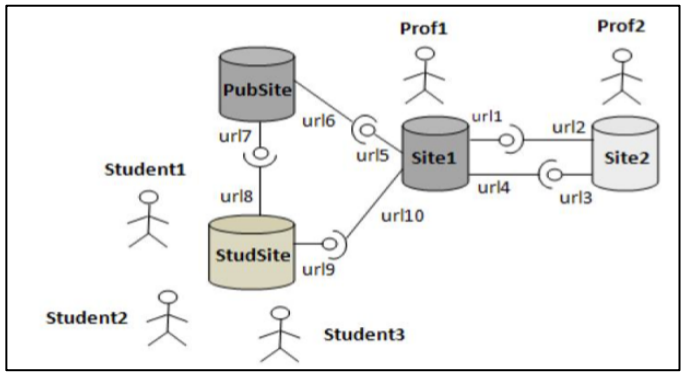

Figure 6 : Un scénario de collaboration supporté par DSMW [10].

La Figure 6 illustre un scénario de collaboration entre deux professeurs et des étudiants utilisant DSMW. Chaque cylindre représente une instance de DSMW. Dans ce scénario, les professeurs préparent leurs différents cours en partageant leurs données, sur leurs instances de wiki (Site1 et Site2). Ces instances contiennent plusieurs cours. Lorsqu'il le souhaite, l'enseignant Prof1 publie ses cours sur l'instance de wiki PubSite. Puis, il avertit ses étudiants qu'il y a de nouveaux cours disponibles sur PubSite. Les étudiants peuvent alors télécharger les cours qui les intéressent depuis PubSite sur leur instance de wiki (StudSite). Ils peuvent annoter ces cours puis publier ces nouvelles modifications et en informer le professeur. Par la suite, le professeur télécharge les modifications des étudiants et les prend en compte si nécessaire.

Pour mettre en œuvre ce scénario, les enseignants et leurs étudiants ont dû créer des canaux de communication en utilisant les opérations de publication-souscription proposées avec l'outil DSMW. Ces opérations sont les suivantes :

- **Create PushFeed** : cette opération est utilisée pour créer un PushFeed. Le contenu d'un PushFeed est une requête sémantique dont le résultat est un ensemble de pages d'un wiki sémantique.
- **Push** : cette opération est utilisée pour calculer l'ensemble des modifications non publiées pour toutes les pages associées à un PushFeed donné. Elle permet donc de déterminer quelles sont les modifications qui devront être propagées.
- **Create PullFeed** : cette opération est utilisée pour créer un PullFeed se référant à un PushFeed donné. La liaison entre un PushFeed et un PullFeed constitue un canal de communication.
- **Pull** : cette opération est utilisée pour télécharger toutes les modifications publiées dans le PushFeed associé au PullFeed. Tous les changements de pages du wiki sémantique téléchargés seront intégrés au wiki effectuant l'action de Pull. C'est donc l'utilisateur, lorsqu'il effectue un Pull, qui décide quelles modifications il souhaite intégrer.

Les enseignants et leurs étudiants doivent, au début du processus, créer des canaux de communication à l'aide des opérations Create PushFeed et Create PullFeed. Ceci leur permet par la suite, à chaque modification, de publier simplement leurs modifications (à l'aide de l'opération Push) et de récupérer les modifications des autres (à l'aide de l'opération Pull). Ces opérations sont déclenchées via l'interface d'administration de DSMW dont un aperçu est donné Figure 7.

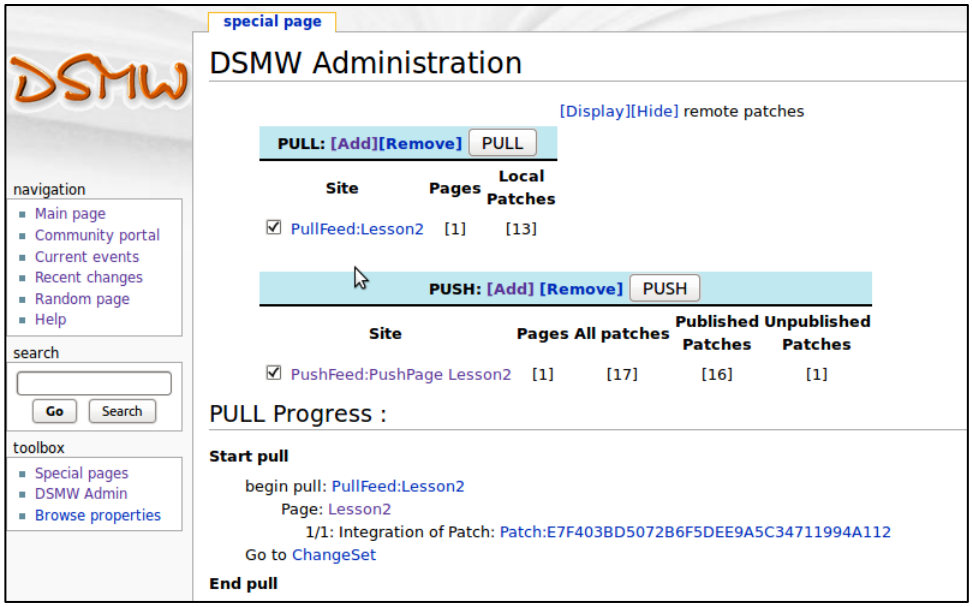

Figure 7 : Interface administrateur de DSMW.

Collectra

Afin de fournir un assistant aux utilisateurs de l'outil DSMW, une première étape a été de doter DSMW d'un collecteur de traces. Le collecteur proposé, Collectra [11-12], permet, pour chaque wiki du réseau, de tracer les actions des utilisateurs. Ces actions peuvent concerner l'édition du wiki mais également le partage de pages entre le wiki observé et les autres wikis du réseau.

Collectra respecte l'architecture présentée dans la Figure 8. Ce collecteur de traces est, comme DSMW, une extension de MediaWiki qui permet d'observer les utilisateurs et de stocker les traces collectées dans un Système de Gestion de Bases de Traces (SGBT). Dans le cas de Collectra, le SGBT utilisé est le kTBS [8] qui permet de stocker des traces modélisées mais également de faire diverses opérations sur ces traces (requêtes, fusion, etc.).

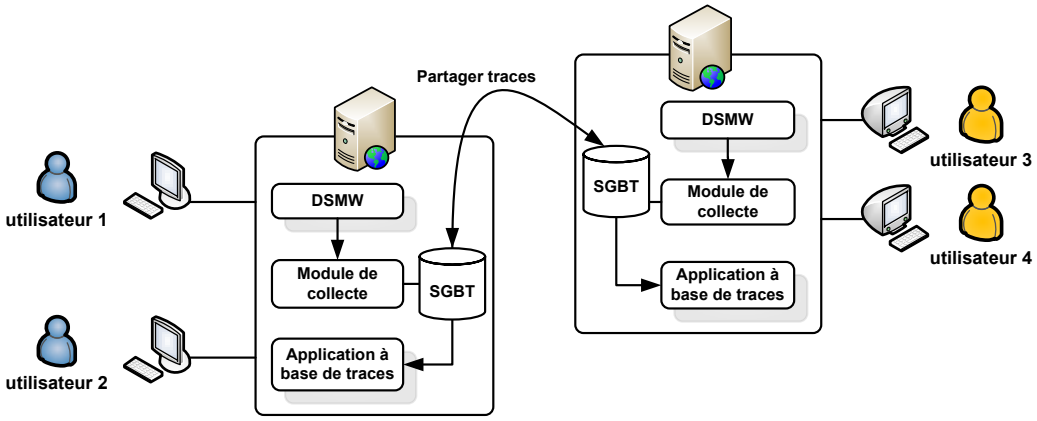

Figure 8 : Architecture de Collectra.

Les traces créées avec Collectra respectent le modèle de traces présenté sur la Figure 9 et détaillé dans [11-12].

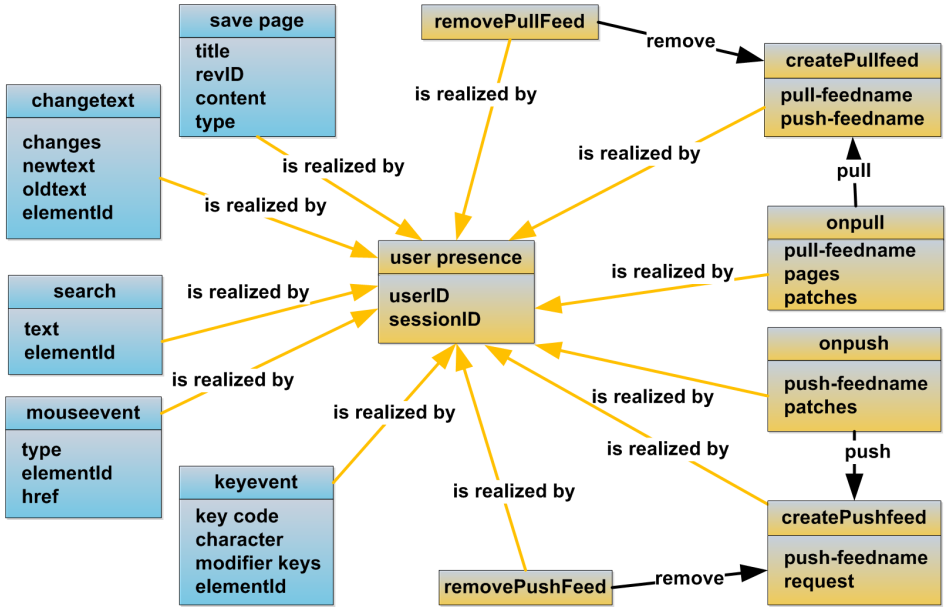

Figure 9 : Modèle de traces de Collectra.

Architecture et fonctionnement de l'assistance à l'outil

L'assistance à l'outil que nous proposons respecte l'architecture présentée sur la Figure 10. Cette assistance est intégrée au wiki MediaWiki sous forme de pages du wiki. Elle s'appuie sur le module de collecte Collectra présenté dans la section précédente.

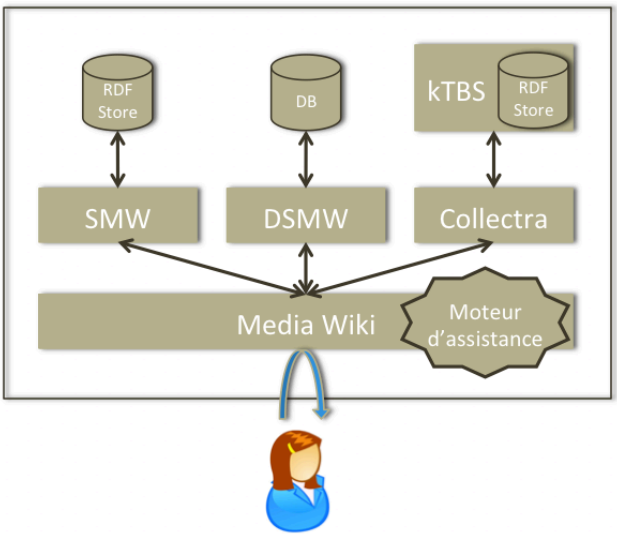

Figure 10 : Architecture de l'assistance à l'outil.

Son fonctionnement est décrit dans le diagramme de la Figure 11. L'utilisateur interagit uniquement avec le wiki DSMW. Lorsqu'il souhaite avoir de l'assistance, il se rend sur la page dédiée et choisit une des questions disponibles (présentées dans la section suivante). Lorsqu'il clique sur une question, le moteur d'assistance calcule la page qui sera proposée à l'utilisateur. Pour cela, il interroge le kTBS avec la requête correspondant à la question. Le kTBS retourne la ou les traces correspondantes à la requête. Le moteur construit alors la page en intégrant les traces récupérées à une page contenant un texte prédéfini répondant à la question. Cette page est alors proposée via le wiki à l'utilisateur qui peut lire le texte et naviguer dans les traces d'exemples.

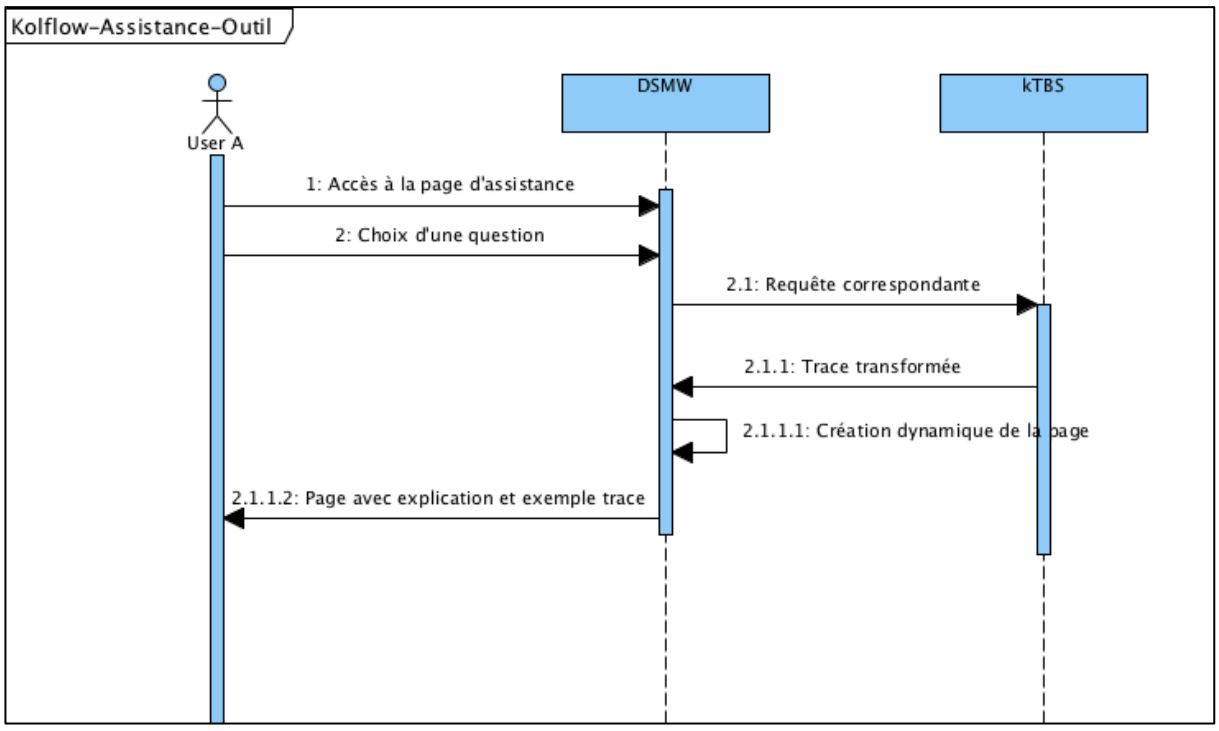

Figure 11 : Diagramme de séquence de l'assistance à l'outil.

Mise en œuvre

Dans cette section, nous présentons la liste des questions disponibles dans notre assistance à l'outil DSMW. Nous indiquons ensuite comment nous avons implémenté l'assistance pour ces questions en présentant la manière de répondre à ces questions, et les mécanismes mis en place pour l'automatisation de tâches. Nous incluons dans ces descriptions les interfaces proposées au sein des wikis.

Listes des questions de l'assistance à l'outil

Les questions de l'assistance portent sur plusieurs points. Elles concernent tout d'abord l'utilisation classique d'un wiki :

1. Comment créer une page ?
2. Comment modifier une page ?
3. Comment trouver une page existante (recherche) ?
4. Comment mettre des commentaires sur une page (onglet discussion) ?

Elles concernent ensuite le partage de pages rendu possible grâce au plugin DSMW :

5. Comment mettre à disposition / partager une page ?
   5a. Comment créer un canal (faire un PushFeed) ?
   5b. comment mettre à jour le partage (faire un Push) ?
6. Comment récupérer le contenu d'une page d'un autre wiki ?
   6a. Comment créer un canal (faire un PullFeed) ?
   6b. Comment mettre à jour (faire un Pull) ?

Elles concernent ensuite la visualisation de différences entre deux versions d'une même page :

7. Comment visualiser les changements que l'on fait sur une page avant de valider ?
8. Comment visualiser l'historique d'une page ?
9. Comment voir ce qui va être fait lors de la récupération des données d'un autre wiki ?

Elles concernent enfin les outils de collecte et d'assistance :

10. Comment fonctionne Collectra ?
11. Comment fonctionne l'assistance ?
12. Comment fonctionne la maintenance ?

Mécanisme de la recherche de traces

Pour trouver dans le kTBS, les traces servant d'exemple, nous suivons la procédure suivante.

Dans un premier temps, il faut définir les signatures de traces relatives à chaque question. Une signature de traces correspond à la description des traces que l'on souhaite récupérer. Pour une question donnée, plusieurs signatures peuvent être possibles, correspondant aux différentes façons de réaliser la tâche. Il faut donc, pour chaque question, lister les différentes manières de faire et pour chaque manière, lister les actions à effectuer sur le wiki. Ensuite il faut faire correspondre chaque action au type d'obsels (et à ses attributs instanciés) connu grâce au modèle de traces de Collectra.

Ensuite, pour chacune des signatures, il faut définir la requête permettant de récupérer tous les obsels dans le kTBS.

Enfin, il faut exécuter cette requête dans le kTBS afin de rechercher dans ses propres traces, ou dans celles des autres utilisateurs, les traces qui seront fournies comme exemple à l'utilisateur (cf. Figure 13).

Pour illustrer ce principe, prenons l'exemple de la question 1 : « Comment créer une page ? ». Pour créer une page dans MediaWiki, il existe au moins quatre façons de faire, dont les descriptions sont les suivantes :

Liste des actions pour la possibilité A :

- A1. Saisir le nom de la page dans la zone de recherche
- A2. Valider avec le bouton « Go »
- A3. Si la page existe, recommencer avec un autre nom
- A4. Si la page n'existe pas, cliquer sur le lien du nom de la page
- A5. Saisir du texte dans la zone de saisie
- A6. Cliquer sur le bouton « Save page »

Liste des actions pour la possibilité B :

- B1. Saisir le nom de la page dans la zone de recherche
- B2. Valider en tapant « Entrée » sur le clavier
- B3. Si la page existe, recommencer avec un autre nom
- B4. Si la page n'existe pas, cliquer sur le lien du nom de la page
- B5. Saisir du texte dans la zone de saisie
- B6. Cliquer sur le bouton « Save page »

Liste des actions pour la possibilité C :

- C1. Dans une page en édition, ajouter un lien avec le bouton dédié de l'interface
- C2. Eventuellement saisir du texte dans la zone de saisie
- C3. Cliquer sur le bouton « Save page » de la page en cours
- C4. Cliquer sur le lien de la page à créer
- C5. Saisir du texte
- C6. Cliquer sur le bouton « Save page » de la page à créer

Liste des actions pour la possibilité D :

- D1. Dans une page en édition, saisir avec le clavier un lien
- D2. Eventuellement saisir du texte dans la zone de saisie
- D3. Cliquer sur le bouton « Save page » de la page en cours
- D4. Cliquer sur le lien de la page à créer
- D5. Saisir du texte
- D6. Cliquer sur le bouton « Save page » de la page à créer

Une fois la tâche décrite, il faut pour chaque possibilité faire le lien entre les actions sur le wiki et les obsels que l'on observe dans la trace. Ainsi, pour la première façon de faire, on obtient la correspondance présentée dans le Tableau 1.

Tableau 1 : Correspondante entre actions faites sur MediaWiki et les types d'obsels de Collectra.

Liste des actions pour la possibilité A :	Listes des obsels correspondants :
A1. Saisir le nom de la page dans la zone de recherche	A1. Obsels de type *ChangeText*, attribut *title* = "changed searchInput"
A2. Valider avec le bouton « Go »	A2. Obsel *MouseEvent*, attribut *element_id* = "SearchGo Button"
A3. Si la page existe, recommencer avec un autre nom	A3. …
A4. Si la page n'existe pas, cliquer sur le lien du nom de la page	A4. Obsel *MouseEvent*, attribut *element_href* = "http://localhost/wiki2/index.php?title=NomPage&action=edit&redlink=1" en remplaçant *l'adresse du wiki2* par la bonne adresse et en remplaçant *NomPage* par le nom saisie par l'utilisateur. Pour trouver le nom de la page, il faut récupérer dans la trace A1, les noms de pages. Le nom de la page est dans l'attribut *Changes* du dernier *ChangeText* avant chaque *MouseEvent* est un *SearchGo*
A5. Saisir du texte dans la zone de saisie	A5. Si on décide de montrer un exemple de saisie, il faut récupérer toutes les actions faites dans la zone de saisie (*KeyEvent*, *ChangeText*, clic sur les boutons de mise en page...) donc toutes les actions entre la date de fin de la trace A4 et la date de début de la trace A6.
A6. Cliquer sur le bouton « Save page »	A6. Obsel *MouseEvent*, attribut *location* = NomPage avec le chemin complet, attribut *element_id* = wpSave

Ensuite, il faut pour chaque type d'obsels, créer une requête afin de récupérer les traces relatives à ces obsels. Un exemple de requête pour les obsels de type A4 est fourni sur la Figure 12. Pour la manière de faire A de la question 1, nous devons définir 6 requêtes permettant de récupérer 6 traces. Une dernière requête permet de fusionner ces différentes traces pour obtenir une unique trace contenant toutes les actions listées dans le Tableau 1.

Notons qu'avec ce principe, certaines traces transformées peuvent être communes à différentes possibilités ou différentes questions. Comme le kTBS mémorise les traces transformées, il peut les réutiliser pour répondre à plusieurs questions.

Notons également que la trace finale peut contenir plusieurs exemples de création de page que l'on découpe lorsqu'on les fournit à l'utilisateur.

```
@prefix ktbs: <http://liris.cnrs.fr/silex/2009/ktbs#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfrest: <http://liris.cnrs.fr/silex/2009/rdfrest#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xml: <http://www.w3.org/XML/1998/namespace>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
 <> ktbs:contains <Trace_CreatePageLink9/>.

 <Trace_CreatePageLink9/> a ktbs:ComputedTrace;
 ktbs:hasParameter """sparql=
PREFIX : <http://localhost:8001/base1/model1/>
PREFIX ktbs: <http://liris.cnrs.fr/silex/2009/ktbs#>
CONSTRUCT {
 [ a :MouseEvent ;
 ktbs:hasTrace <%(__destination__)s> ;
 ktbs:hasBegin ?begin ;
 ktbs:hasBeginDT ?begindt ;
 ktbs:hasEnd ?end ;
 ktbs:hasEndDT ?enddt ;
 :user_id ?user_id ;
 :user_name ?user_name ;
 :site_id ?site_id ;
 :title ?title ;
 :description ?description ;
 :location ?location ;
 :element_type ?element_type ;
 :element_id ?element_id ;
 :element_value ?element_value ;
 :element_text ?element_text ;
 :element_checked ?element_checked ;
 :element_src ?element_src ;
 :element_href ?element_href ;
 ktbs:hasSourceObsel ?obsel ;
 ] .
} WHERE {
 ?obsel a :MouseEvent ;
 ktbs:hasBegin ?begin ;
 ktbs:hasBeginDT ?begindt ;
 ktbs:hasEnd ?end ;
 ktbs:hasEndDT ?enddt ;
 :user_id ?user_id ;
 :user_name ?user_name ;
 :site_id ?site_id ;
 :title ?title ;
 :description ?description ;
 :location ?location ;
 :element_type ?element_type ;
 :element_id ?element_id ;
 :element_value ?element_value ;
 :element_text ?element_text ;
 :element_checked ?element_checked ;
 :element_src ?element_src ;
 :element_href ?element_href.
 FILTER regex(?element_href, "&action=edit&redlink=1", "i") .
}""";
 ktbs:hasMethod ktbs:sparql;
 ktbs:hasSource <http://localhost:8001/base1/PrivateTrace_0/>.
```

Figure 12 : Requête kTBS permettant d'obtenir tous les MouseEvent sur des liens d'édition.

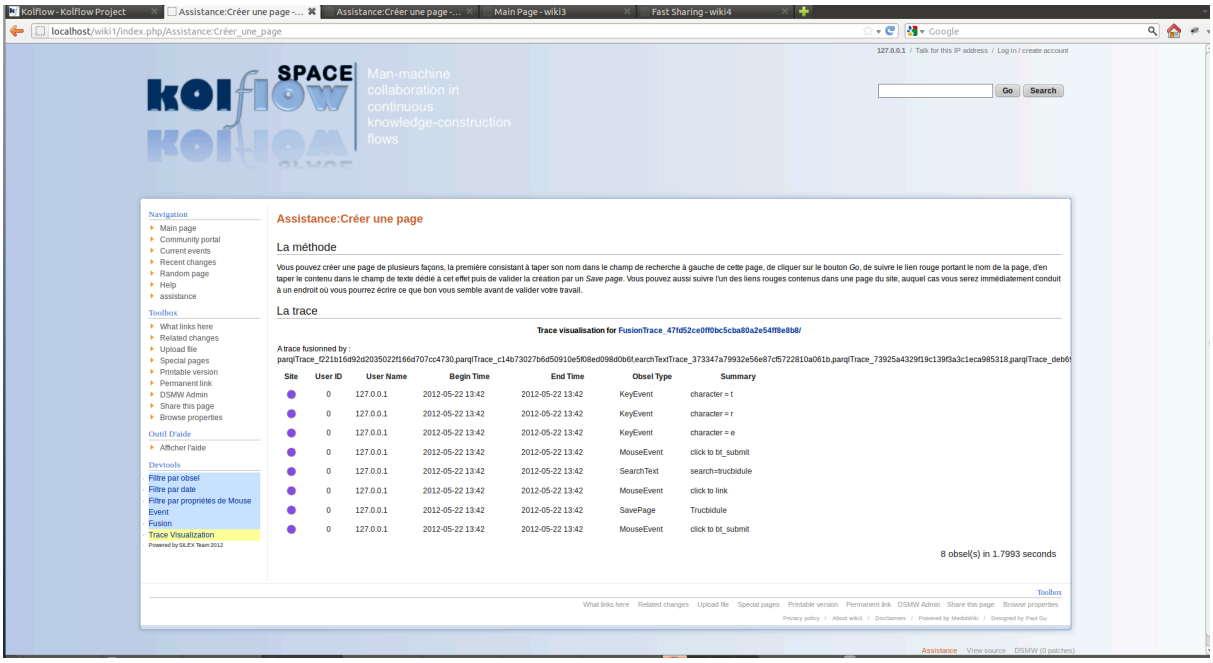

Figure 13 : Page d'assistance à l'outil DSMW montrant les traces correspondant à la création d'une page.

Automatisation de tâches

L'assistance à l'outil repose donc sur une présentation d'exemples de traces montrant les actions faites précédemment par l'utilisateur lui-même, ou d'autres utilisateurs pour réaliser une tâche donnée.

Pour certaines actions, nous avons souhaité mettre en place une automatisation de la tâche. C'est le cas pour le partage de page. En effet, cette tâche est compliquée d'une part puisqu'il faut maitriser les principes de partage (création de canaux de communication, mise à disposition, récupération, etc.) et d'autre part puisque l'interface d'administration permettant de faire ces actions n'est pas adaptée à des utilisateurs novices (cf. Figure 7).

Pour cette tâche de partage, nous avons donc ajouté un lien sur les pages du wiki permettant d'indiquer à l'assistance que l'on souhaite partager le contenu d'un page. Lorsque l'utilisateur clique sur ce lien, l'assistance exploite les fonctions de DSMW pour savoir si cette page est concernée par un canal de communication (PushFeed). Dans l'affirmative, l'assistance fournit l'adresse du PushFeed à l'utilisateur. Sinon, elle crée un nouveau PushFeed et fournit l'adresse de celui-ci (cf. Figure 14).

L'utilisateur peut alors communiquer le lien fourni aux utilisateurs avec lesquels il souhaite partager du contenu. Ceux-ci pourront créer un flux d'entrée correspondant au flux de sortie et, ainsi, obtenir l'ensemble des modifications effectuées sur la page partagée.

L'utilisateur peut aussi mettre directement son contenu dans un autre wiki. Pour cela, il fournit à l'assistance l'adresse du wiki concerné. L'assistance crée alors sur cet autre wiki, la seconde moitié du canal de communication (PullFeed), toujours en exploitant les fonctionnalités de DSMW, et intègre le contenu à l'aide d'un Pull.

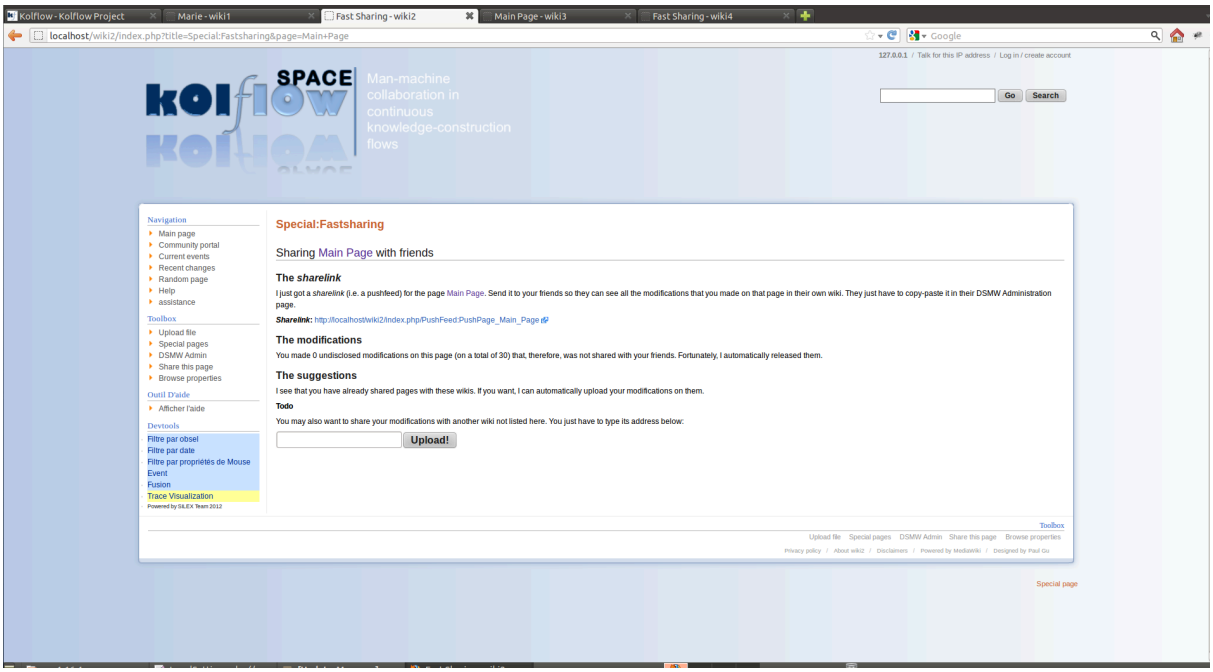

Figure 14 : Interface d'automatisation du partage de pages dans DSMW.

Bilan

Nous avons présenté dans ce rapport les principes et la mise en œuvre de l'assistance à l'outil DSMW que nous proposons dans le cadre du projet Kolflow.

Pour mettre en place cette assistance, nous avons développé Collectra, un collecteur de traces pour l'outil DSMW permettant d'obtenir, dans un kTBS, des traces d'actions d'édition de pages de wikis mais aussi de partages de ces traces au sein d'un réseau de wikis. Nous avons donc des traces d'activités individuelles et collectives.

Nous avons ensuite défini la majorité des requêtes SPARQL au sein du kTBS permettant d'obtenir les traces relatives aux différentes questions d'interface. Seules les requêtes relatives aux traces de partages de pages restent à définir.

Nous avons ensuite mis en œuvre l'assistance en l'intégrant au wiki. Pour chaque question, nous fournissons un texte d'explication, et, excepté pour les questions relatives au partage, nous interrogeons le kTBS afin d'obtenir des exemples de traces.

Nous avons enfin mis en place l'automatisation de tâche pour le partage de pages entre wikis.

Ces différents points montrent la faisabilité de notre démarche. Toutefois, nous n'avons pas intégré un véritable outil de visualisation de traces. Pour l'instant nous affichons les traces dans un tableau. De la même manière, nous ne fournissons pas de vidéos des actions à faire.

Ces deux derniers points, ainsi que les requêtes manquantes ne seront toutefois pas proposés. En effet, une réorientation stratégique du projet Kolflow nous a amené à abandonner l'outil DSMW pour mettre en place l'espace social sémantique. En effet, pour mettre en œuvre un assistant à la négociation de sens [13], nous avons besoin de faire passer des tests sur plusieurs évolutions possibles d'une même ontologie. En gardant DSWM comme support pour nos bases de connaissances, nous aurions dû dupliquer les wikis

DSMW pour pouvoir effectuer ces séries de tests. Le passage à l'échelle de cette solution n'étant pas réaliste, nous avons opté pour l'abandon de DSMW au profit de wikis sémantiques classiques, sauvegardant leurs bases de connaissances dans des RDFStores. Avec cette solution, seul la duplication des RDFStores sera nécessaire pour effectuer les séries de tests.

Notons toutefois que Collectra a été développé comme un plugin pour collecter les traces de DSMW, et le modèle de trace qu'il collecte a été défini au regard des activités réalisées par des utilisateurs de DSMW. Cependant, ce travail est réutilisable et applicable à d'autres contextes, moyennant deux modifications principales : la définition d'un nouveau modèle de trace dépendant du modèle d'application visé (dans notre cas, un Semantic Media Wiki couplé à des pages web classiques) et la modification d'une des interfaces de l'API Collectra afin de la connecter à l'application correspondante.

Références bibliographiques et webographiques

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

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

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

[4] P-A. Champin, A. Cordier, E. Lavoué, M. Lefevre, H. Skaf-Molli. *User Assistance for Collaborative Knowledge Construction*. Workshop Semantic Web Collaborative Spaces, WWW2012, Lyon, France, 17 avril 2012.

[5] A. Cordier, M. Lefevre, P-A. Champin, O. Georgeon, A. Mille. *Trace-Based Reasoning - Modeling interaction traces for reasoning on experiences*. 26th International FLAIRS Conference, St. Pete Beach, Florida, USA, mai 2013.

[6] 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.

[7] URI. http://en.wikipedia.org/wiki/Uniform_resource_identifier (consulté en mars 2013).

[8] 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.

[9] P. Dourish. *The parting of the ways : Divergence, data management and collaborative work*. P. 215–230. Kluwer Academic Publishers, 1995.

[10] C. Rahhal, H. Skaf-Molli, P. Molli, S. Weiss. *Multi-synchronous collaborative se- mantic wikis*. Wise'09 : International Conference on Web Information Systems, 2009.

[11] A-H. Le, M. Lefevre, A. Cordier. *Collecter les traces d'interaction de wikis sémantiques distribués pour assister leurs utilisateurs*. 20ème atelier français de raisonnement à partir de cas (RàPC 2012), Paris, France, 25 juin 2012.

[12] 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.

[13] M. Lefevre, A. Cordier, P-A. Champin. *D5.3. KOLFLOW - Description et fonctionnement de l'assistant distribué pour la co-construction de connaissances*. Rapport de recherche RR-LIRIS-2013-003, 2013.