Collecter les traces d'interaction de wikis sémantiques distribués pour assister leurs utilisateurs

Anh-Hoang Le, Marie Lefevre, Amélie Cordier
Université de Lyon
Université de Lyon 1, LIRIS
UMR5205, F-69622, France
{prenom.nom@liris.cnrs.fr}

Résumé

Cet article s'intéresse à la construction d'un moteur d'assistance à base de traces pour les utilisateurs de wikis sémantiques distribués. Nous nous concentrons sur la première brique de ce moteur : le module de collecte de traces. Nous identifions les éléments qui doivent être collectés de sorte à alimenter convenablement le moteur d'assistance. Nous identifions également les spécificités de notre environnement, et en particulier son caractère distribué et propice aux activités collaboratives. En nous appuyant sur ces observations, nous définissons un modèle et des principes pour la collecte des traces d'interaction des utilisateurs interagissant avec des outils distribués. Ce modèle et ces principes sont implémentés dans une extension de MediaWiki appelée Collectra. Nous montrons, au travers d'un scénario, comment les traces collectées permettent de fournir une assistance pertinente et contextualisée aux utilisateurs. Nous discutons également des bénéfices de notre proposition dans d'autres contextes applicatifs.

Mots clés : Raisonnement à partir de l'expérience tracées, Assistance aux utilisateurs, Collecte de traces, Wiki Sémantiques Distribuées, Activités collaboratives.

1 Introduction

L'édition collaborative de sites Web est désormais une réalité, comme le montre la réussite de projets tels que Wikipédia. Les wikis sont particulièrement adaptés aux activités d'édition collaborative, mais ils rencontrent encore des problèmes en cas d'éditions simultanées : des conflits apparaissent fréquemment et leur résolution est difficile [3]. Par ailleurs, les wikis fonctionnent selon une architecture centralisée : tous les utilisateurs contribuent à la même instance du wiki. Une telle architecture ne permet pas aux utilisateurs de maintenir plusieurs versions concurrentes d'une même page, par exemple pour exprimer des points de vue différents. Pour pallier ces problèmes, Rahhal et al. [21] ont proposé de mettre en œuvre une architecture de wikis selon un modèle distribué. Les wikis constituant cette architecture sont appelés DSMW pour *Distributed Semantic Media Wiki*. Dans une architecture de DSMW, chaque utilisateur peut travailler sur sa propre version du wiki et décider de ce qu'il souhaite partager avec les autres utilisateurs. L'architecture contient des modèles qui assurent le bon fonctionnement du réseau et qui régissent les modalités de partage des objets au sein de ce réseau. Les DSMW sont des wikis sémantiques, c'est-à-dire qu'ils permettent d'enrichir les pages d'annotations sémantiques. Ces annotations peuvent ensuite être utilisées par des agents intelligents chargés d'effectuer des tâches de raisonnement.

Dans le projet Kolflow [1], nous cherchons à développer des environnements informatiques permettant aux humains et aux agents intelligents de collaborer afin de produire des connaissances partagées. Pour les raisons évoquées ci-dessus, les wikis sémantiques distribués sont des outils tout à fait appropriés pour développer de tels environnements. Cependant, une des premières constatations du projet Kolflow est que les outils tels que DSMW sont difficiles à utiliser, même pour les utilisateurs expérimentés [6]. Un des aspects les plus difficiles à appréhender est le partage d'éléments entre les utilisateurs car il nécessite non seulement une bonne

1. `http://kolflow.univ-nantes.fr`

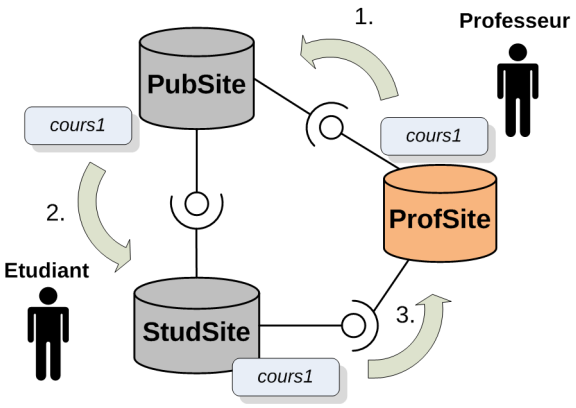

FIGURE 1 – Un scénario de collaboration supportée par DSMW.

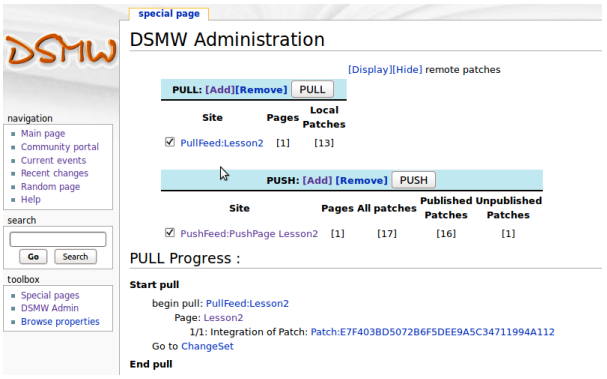

FIGURE 2 – Interface d'administration de DSMW.

maîtrise technique de l'outil, mais aussi une connaissance de l'environnement dans lequel se situe l'outil. Un des objectifs du projet Kolflow est donc de fournir aux utilisateurs l'assistance nécessaire à la fois dans l'utilisation de l'outil et dans la phase de partage des connaissances.

Dans cet article, nous discutons de la pertinence d'utiliser le raisonnement à partir de traces [12], fortement inspiré du raisonnement à partir de cas, au sein du moteur d'assistance que nous développons. Nous montrons en quoi ce mode de raisonnement facilite la collecte des expériences des utilisateurs et la prise en compte du caractère distribué de l'environnement.

Nous présentons, dans la section 2, des scénarios illustrant le fonctionnement d'un réseau de wikis sémantiques distribués. Nous mettons en évidence les besoins en matière d'assistance et nous montrons comment le raisonnement à partir de l'expérience tracée pourrait y contribuer. Dans la section 3, nous justifions le choix de l'utilisation des traces comme support à la fois pour collecter les expériences des utilisateurs et pour construire les propositions d'assistance qui leurs seront fournies. Dans la section 4, nous présentons le modèle que nous avons développé pour le contexte des wikis sémantiques distribués, puis, dans la section 5, nous montrons comment ce modèle a été implémenté dans l'outil Collectra. La section 6 illustre le fonctionnement de l'ensemble du système. Nous concluons cet article en montrant en quoi ces propositions peuvent être appliquées dans des contextes différents avant de présenter nos perspectives de recherche.

2 Assister les utilisateurs de wikis sémantiques distribués

DSMW est une extension de Semantic MediaWiki (SMW) développée par des chercheurs du LINA et du LORIA [25]. 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 [13] qui permet la modification simultanée de pages sur des serveurs différents, la propagation de ces modifications sur le réseau et la gestion des conflits résultant éventuellement suite à la propagation des modifications [21]. 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.

La figure 1 (adaptée de l'article [21]) illustre un scénario très simple de collaboration entre un professeur et un étudiant utilisant DSMW. Chaque cylindre représente une instance de DSMW. Dans ce scénario, le professeur prépare ses différents cours sur son instance du wiki (ProfSite). Cette instance contient plusieurs cours. Lorsqu'il le souhaite, le professeur publie ses cours sur l'instance de wiki PubSite. Puis, il avertit ses étudiants qu'il y a de nouveaux cours disponibles sur PubSite. Chaque étudiant peut télécharger les cours qui l'interressent depuis PubSite sur son instance personnelle de wiki, ici StudSite, et faire quelques changements (par exemple, répondre à un questionnaire en ligne). Puis, chaque étudiant publie ses nouvelles modifications et en informe 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, l'enseignant et ses 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 tous 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.

L'enseignant et ses 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 2.

Cette interface, bien que parfaitement fonctionnelle, est difficile à prendre en main. Par exemple, la création d'un PushFeed ou d'un PullFeed requiert plusieurs étapes et implique des actions à effectuer sur plusieurs pages. Par conséquent, une première forme d'assistance consisterait à automatiser certaines actions complexes (comme la création de canaux de communication) ou à guider les utilisateurs dans la réalisation de ces tâches.

Comme on le voit dans l'exemple, il est également difficile pour les utilisateurs de percevoir l'ensemble des actions possibles dans le réseau de wikis. Les utilisateurs doivent avoir recours à des moyens de communication externes pour s'informer mutuellement de la mise à disposition de ressources. Ils sont également confrontés à d'importantes difficultés lors de l'application de mises à jour impliquant la fusion de ressources existantes. Une seconde forme d'assistance consiste donc à accompagner les utilisateurs dans le processus de partage des ressources contenues dans leurs wikis respectifs. Les différents types d'assistance pouvant être proposées dans ce cadre sont détaillés dans l'article [6].

Pour mieux illustrer ce second besoin d'assistance, considérons l'exemple de la communauté des utilisateurs de Taaable [11]. Taaable est un wiki contenant un ensemble de recettes de cuisine et une ontologie des connaissances du domaine culinaire permettant à un moteur de raisonnement de proposer des adaptations de recettes en fonction des requêtes exprimées par les utilisateurs. Bob, membre de la communauté, possède sa propre instance de DSMW qu'il a créée en récupérant les données d'un DSMW existant. Dans sa version, il a donc un ensemble de recettes de cuisine qu'il a augmenté en ajoutant sa recette de *Tarte au Melon*. Il a par ailleurs modifié d'autres recettes et a enrichi l'ontologie des ingrédients en ajoutant des fruits (kiwis, ananas, melon, *etc.*). Alice possède également une instance de DSMW créée à partir de la même source que celle de Bob. Elle a, de son coté, ajouté sa propre recette de *Tarte au Melon*. Elle a également modifié l'ontologie des ingrédients. Bob a mis les kiwis dans la catégorie *Fruits Exotiques* alors qu'Alice les a placés dans la catégorie générale des *Fruits*. Lorsque Bob et Alice vont vouloir partager le contenu de leur wikis, des conflits vont donc survenir : recettes différentes avec le même nom, ingrédients placés différemment, etc. Des mécanismes de raisonnement peuvent permettre de détecter ces conflits, mais leur résolution reste à la charge des utilisateurs. Un système d'assistance pourrait les aider à négocier les connaissances contenues dans leurs wikis et leur monter les étapes à suivre pour résoudre les conflits ou leur donner des informations complémentaires, telles que l'historique des modifications des ressources impliquées dans le conflit, fain de les aider dans leurs prises de décision.

3 Pourquoi utiliser des traces d'interaction pour l'assistance ?

Dans la section précédente, nous avons vu qu'il était nécessaire de fournir une assistance aux utilisateurs de DSMW à au moins deux niveaux : utilisation de l'outil et accompagnement dans les activités de partage de connaissances. Dans cette section, nous discutons tout d'abord des différentes façons de fournir de l'assistance et nous montrons en quoi l'exploitation des traces d'interaction peut permettre de fournir de tels types d'assistance. À la lumière de ces observations, nous nous intéressons au problème de la collecte de traces dans les applications Web. La section se termine par une présentation rapide du vocabulaire spécifique de la théorie de la trace sur laquelle nous nous appuyons.

3.1 Comment fournir de l'assistance ?

La majorité des travaux conçoit l'assistance comme la capacité du système à fournir une réponse à un problème posé par l'utilisateur. Cette assistance consiste à implémenter des mécanismes de raisonnement permettant de faciliter ou d'automatiser certaines tâches en s'appuyant sur l'utilisation de connaissances et de stratégies prédéfinies, le rôle de l'utilisateur se limitant à fournir les informations nécessaires à la recherche d'une solution [27]. Cette conception de l'assistance est critiquée puisque [4] :

- elle ne permet pas l'acquisition d'un savoir supplémentaire pour l'utilisateur ;
- elle est contraire aux pratiques d'assistance en situation réelle consistant à guider l'utilisateur pour arriver à la solution plutôt que de lui fournir directement cette solution [9] ;
- elle ne permet pas de dialogue entre humain et machine, dialogue qui permettrait de guider et d'améliorer la recherche de solution [22].

Pour dépasser ces limites, d'autres types d'assistance ont été proposés, visant une résolution des problèmes plus interactive. Ainsi, les outils adaptatifs sont conçus pour s'adapter aux utilisateurs [26]. Ils sont capables de modifier automatiquement leurs propres caractéristiques en fonction des besoins des utilisateurs [19]. Ces outils sont particulièrement pertinents dans les contextes où les utilisateurs doivent s'approprier rapidement des environnements alors qu'ils n'ont parfois même pas conscience de leurs propres besoins. Les outils adaptatifs exploitent les connaissances dont ils disposent sur les utilisateurs pour adapter leur comportement. Ils font également usage de connaissances relatives au domaine et à l'application elle-même afin de pouvoir faire des inférences et d'identifier les éléments de l'application qui peuvent être adaptés à l'utilisateur. Ainsi, les outils adaptatifs sont centrés sur la façon dont l'utilisateur interagit avec le système et sur la façon dont les interfaces peuvent s'adapter pour faciliter ces interactions.

Ces outils adaptatifs, s'ils s'adaptent à l'utilisateur, ne permettent pas d'apporter de l'aide sur des aspects non anticipés par leurs concepteurs. Pour cela, les systèmes d'assistance doivent donc être capable d'enrichir leurs connaissances au cours du temps [10]. Or, la combinatoire des situations imprévisibles rend toute tentative d'aide prescrite très difficile [17]. En utilisant le raisonnement à partir de l'expérience tracée [16], il est possible de dépasser cette limite et ainsi de proposer des assistants capables de s'adapter aux besoins des utilisateurs ainsi qu'au changement de contexte [5, 20, 12]. Pour cela, le raisonnement à partir de traces (RàPT) propose de réutiliser les principes du raisonnement à partir de cas et d'exploiter des expériences non structurées stockées dans des traces [16]. Les traces peuvent être vues comme des conteneurs de connaissances qui permettent de conserver les expériences « en contexte ». Les mécanismes de raisonnement du RàPT permettent ensuite d'extraire des traces les connaissances nécessaires aux différentes tâches, et de les utiliser à bon escient. Le RàPT s'appuie donc naturellement sur les interactions entre l'utilisateur et l'application pour apporter un élément de réponse à la problématique de l'évolution des connaissances nécessaires aux assistants [12].

3.2 Comment tracer les interactions des utilisateurs ?

Dans nos travaux, nous nous appuyons sur le paradigme du RàPT pour construire le moteur d'assistance de DSMW. Il est donc nécessaire de collecter les traces des utilisateurs de DSMW. Deux problématiques apparaissent alors : que collecter et comment collecter ? Dans cette partie, nous discutons de ces deux problématiques au regard de l'assistance que nous souhaitons fournir aux utilisateurs.

Dans le contexte du projet Kolflow, l'outil à tracer est une application Web qui a la spécificité d'être une partie intégrante d'une architecture distribuée. Il est donc important d'acquérir des informations sur l'activité de l'utilisateur, donc côté client, mais également sur le partage des données entre les utilisateurs, donc côté serveur. Nous étudions donc les différentes propositions qui ont été faites pour l'observation des utilisateurs de sites Web puis nous faisons une synthèse illustrant les problèmes relatifs à notre contexte.

La collecte des interactions des utilisateurs avec un site Web peut s'effectuer (1) manuellement par des observateurs humains ou automatiquement, soit par (2) des équipements externes (*i.e.* des capteurs vidéo et audio), soit par (3) des systèmes de traçage intégrés aux applications ou aux outils eux-mêmes. La **collecte manuelle** est réalisée par des observateurs humains qui observent les utilisateurs en présentiel ou à l'aide d'outils informatiques. Les résultats de cette collecte sont des notes, le plus souvent en langage naturel. La **collecte avec équipements externes** exige l'utilisation des caméras, d'appareils photos ou d'outils d'enregistrements audios. Cette approche de collecte permet d'obtenir des traces audiovisuelles. Ces deux approches permettent d'obtenir des informations très détaillées sur les utilisateurs, mais souvent peu formalisées. Par conséquent, l'exploitation des résultats doit être faite par des opérateurs humains. De plus, l'exploitation automatique de sources audiovisuelles est très complexe. Par conséquent, ces deux approches ne conviennent pas pour construire un assistant informatique interactif.

Les **systèmes de traçage** permettent d'obtenir des traces formalisées et donc exploitables par des moteurs de raisonnement. Dans le contexte des sites Web, les systèmes de traçage peuvent se situer soit côté client (plugin du navigateur, ou intégration dans l'application tracée), soit côté serveur. Le traçage côté client permet une collecte très fine des observations mais présente des contraintes (portabilité, rapidité, compatibilité). Le traçage côté serveur ne permet pas d'observer toutes les interactions du client, mais est souvent plus générique et plus simple à mettre en œuvre. Nous expliquons à présent les caractéristiques de chacun de ces systèmes de traçage.

Navigateur Web : le navigateur lui-même est un système de traçage. Les informations sur l'activité de l'utilisateur sont enregistrées les logs du navigateur. Ces informations portent généralement sur les liens que l'utilisateur a visité [18].

Extension de navigateur ou plugin : ces systèmes de traçage ont pour but d'obtenir des informations plus détaillées que celles, assez pauvres, contenues dans les fichiers log des navigateurs. Ces extensions sont souvent connues sous le nom *key-logger*. Elles sont capables de capter les événements clavier et souris et ainsi de récupérer des informations relatives aux actions de l'utilisateur. Ces extensions ont plusieurs buts, dont celui de faciliter des tâches répétitives (*i.e.* aide à la saisie dans un moteur de recherche). Ces extensions doivent être installées par les utilisateurs. Ceci peut être vu comme un inconvénient, mais permet néanmoins de sensibiliser l'utilisateur au fait que ses actions seront observées et enregistrées.

Serveur Web : comme les navigateurs, les serveurs Web comme Apache ont aussi des systèmes de traçage intégrés qui sont capables de générer des fichiers log contenant des informations sur les actions des utilisateurs. Les logs respectent un format prédéfini appelé Common Log Format [2] et contiennent l'adresse des clients, une indication temporelle, la requête exécutée par le serveur, le code correspondant à l'exécution de la requête et la taille de l'objet retourné au client. Ces traces peuvent être utilisées comme entrées pour les assistants à la navigation sur le Web [28]. Pourtant, comme pour le fichier de log du navigateur, celui du serveur Web ne contient pas d'informations assez détaillées sur les actions de l'utilisateur. Par exemple, les logs des serveurs ne contiennent aucune information sur les actions faites côté client.

Extension de serveur Web : collecter et synchroniser des traces à la fois côté client et côté serveur pose souvent des problèmes. Cependant, ces problèmes peuvent être résolus à l'aide d'extensions pour le serveur. Par exemple, [15] a proposé une extension permettant de tracer des forums de discussions. Cette extension se compose d'un collecteur Javascript côté client associé à un code côté serveur. Toutefois, les extensions actuellement proposées se limitent à l'observation d'un seul serveur.

Le tableau 1 synthétise les différentes approches permettant de tracer les applications Web au regard des caractéristiques qui nous intéressent dans la collecte de traces à des fins d'exploitation par un système d'assistance à base de traces. L'analyse de ce tableau révèle que les systèmes de traçage de site Web peuvent prendre plusieurs formes, et qu'ils permettent d'obtenir des traces ayant des contenus et formes de représentation variés.

2. `http://en.wikipedia.org/w/index.php?title=Common_Log_Format&oldid=474668876`

Sources de traçage	Actions clavier	Actions souris	Requêtes sur le serveur	Partage entre serveurs	Multi-utilisateurs	Exploitation par un système informatique	Exemple
Observateur humain	√	√	√	∅	∅	∅	Heraud et al., 2005 [14]
Equipement externe	√	√	√	∅	∅	∅	Avouris et al., 2005 [1]
Navigateur Web	∅	∅	√	∅	∅	√	Oh et al., 2011 [18]
Extension de navigateur	√	√	√	∅	∅	√	Bell et al., 2012 [2]
Serveur Web	∅	∅	√	∅	√	√	Schechter et al., 1998 [23]
Extension de serveur Web	√	√	√	∅	√	√	May et al., 2007 [15]

TABLE 1 – Sources de traçage et contenu des traces obtenues.

Cette analyse montre également que pour avoir des informations détaillées sur les actions d'un ou plusieurs utilisateurs sur un site Web, il faut développer des extensions côté serveur. Toutefois, dans le contexte des serveurs distribués (*i.e.* des wikis sémantiques distribués), les collaborations entre les serveurs et le partage de traces entre ces serveurs doivent pouvoir être tracés. Les systèmes de traçage actuels doivent donc être étendus pour capturer les traces qui reflètent l'activité entre les utilisateurs. Nous proposons pour cela un système de traçage qui permet la collecte de traces dans un environnement distribué. Ce modèle, présenté dans la section 4, s'appuie sur la théorie de la trace que nous présentons dans le paragraphe suivant.

3.3 Théorie de la trace

Dans cette section, nous présentons rapidement la théorie de la trace sur laquelle nous nous appuyons [24]. Nous fournissons quelques définitions de base sur les traces, les méthodologies et les outils pour manipuler les traces. Ces éléments sont nécessaires à la compréhension de notre proposition.

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, événements 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 [3]). 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 traces, présentant la sémantique des modèles de traces, des traces, des requêtes, et des transformations peut être trouvée dans [24].

3. `http://en.wikipedia.org/w/index.php?title=URI&oldid=462250668`

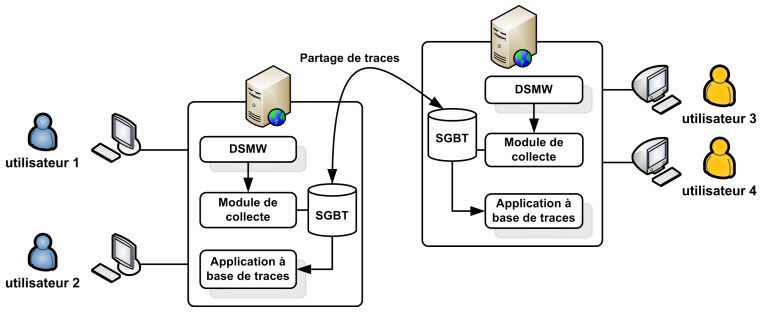

FIGURE 3 – Collecte et partage de traces dans DSMW.

4 Collecter les traces dans un environnement distribué

Dans cette section, nous décrivons notre approche pour la conception d'un module pour la collecte des traces d'interactions dans DSMW. Nous présentons l'architecture du module ainsi que les modèles et les processus qu'il met en œuvre. Notre approche étend un cadre général pour la gestion des traces dans les systèmes collaboratifs synchrones décrit dans [8]. Dans cette approche, l'architecture proposée est plutôt adaptée aux applications classiques qu'aux applications Web, et le modèle de traces est trop général pour être opérationnalisé. Nos expériences préliminaires ont montré que cette architecture n'est pas directement applicable dans notre contexte. Par conséquent, en adaptant l'idée originale, nous proposons une architecture dédiée aux applications Web multi-synchrones telles que DSMW. Notre idée est de mettre en œuvre un module de collecte de traces sur chaque serveur DSMW afin d'observer et d'enregistrer un maximum d'actions des utilisateurs. Dans notre approche, les traces collectées sont stockées dans un SGBT associé à chaque DSMW. Nous étendons les fonctionnalités usuelles des SGBT pour les doter de la possibilité de partager des traces entre serveurs.

4.1 Une architecture pour la collecte et le partage des traces

La figure 3 illustre notre architecture pour la collecte et le partage des traces dans DSMW. Dans cette architecture, chaque serveur est utilisé par un groupe d'utilisateurs et est relié à un SGBT qui stocke les traces collectées et les partage avec d'autres serveurs. Le module de collecte est une interface entre DSMW et son SGBT associé. Le processus de collecte mis en œuvre dans DSMW envoie des messages à ce module. Le module analyse les messages afin de construire les obsels qui sont écrit dans la trace stockée. Pour obtenir des traces partagées à partir de plusieurs serveurs, nous utilisons une communication inter-SGBT qui peut être mise en place grâce à une méthode particulière qui permet la collecte de traces provenant de sources externes. Ces sources externes peuvent être d'autres SGBT.

Grâce à cette architecture, nous pouvons recueillir à la fois des traces individuelles et collectives. Le SGBT que nous avons associé à notre plugin de collecte est le kTBS (*Kernel for Trace-based Systems*) [7]. Le kTBS est une implémentation d'un SGBT en Python. Il permet de définir explicitement les ressources utilisées, telles que les traces primaires, les traces transformées, les modèles de traces (avec des types d'obsels, les types d'attributs et de types de relations entre les types d'obsels). La section 5 revient sur l'implémentation de notre module de collecte et sur son articulation avec le kTBS.

4.2 Tracer la collaboration

DSMW est un système d'édition collaborative multi-synchrone permettant à un groupe d'utilisateurs d'effectuer des activités d'édition sur les pages d'un wiki sémantique et de synchroniser ces pages. Par conséquent, nous nous sommes intéressés à la collecte des traces d'activités de collaboration. Afin de manipuler les traces de collaboration, nous avons dans un premier temps séparé les activités des utilisateurs de DSMW en deux types d'activités principales : activité individuelle et activité collaborative.

– Une **activité individuelle** définit une activité qui est faite par l'utilisateur sur son serveur et qui ne nécessite pas d'avoir conscience des activités faites sur d'autres serveurs.

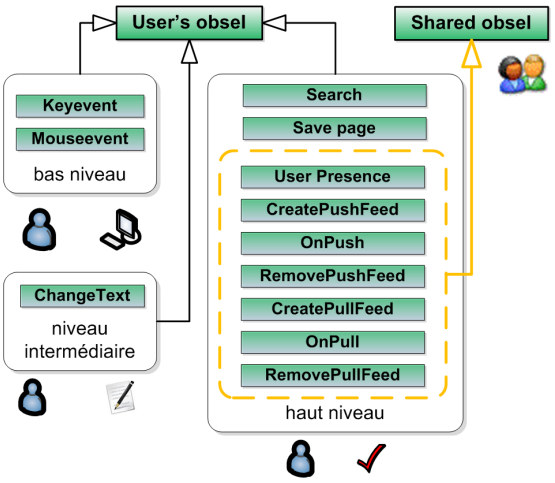

FIGURE 4 – Classification des obsels.

– Une **activité collaborative** définit une activité faite par un utilisateur sur le réseau et qui nécessite d'avoir conscience des autres utilisateurs et des autres serveurs.

Cette distinction nous permet d'utiliser deux types de traces : la trace individuelle et la trace de collaboration. Au niveau sémantique, une **trace individuelle** est un terme utilisé pour parler des traces qui reflètent les activités individuelles d'un utilisateur. Le terme **trace de collaboration** est utilisé pour les traces reflétant des activités de collaboration. Du point de vue implémentation, une trace individuelle est identifiée par un utilisateur, et est appelée **trace privée (*private trace*)**. Une trace qui reflète une activité de collaboration est appelé **trace partagée (*shared trace*)**. Une fusion de traces partagées est appelée **trace de groupe partégée (*group-shared trace*)**. L'objectif de cette organisation est de faciliter le processus de construction des traces partagées à partir des traces individuelles.

4.2.1 Classification des obsels

La notion d'activité est un concept abstrait. Nous proposons une catégorisation simple des obsels en trois groupes : obsels de bas niveau, de niveau intermédiaire et de haut niveau.

Au niveau le plus bas, les obsels reflètent une action unique effectuée par un utilisateur sur l'interface du wiki, tels que les événements clavier et les événements de souris. Au niveau intermédiaire, les obsels caractérisent un ensemble d'actions que nous pouvons associer à une tâche. Ces tâches sont déterminées par le système. Par exemple, la tâche *ChangeText* est une suite d'événements clavier et souris permettant de modifier un texte donné. Au niveau le plus haut, les obsels caractérisent un ensemble de tâches qui sont nécessaires pour la réalisation d'une activité. Par exemple : la recherche, l'édition, l'accès à une page, etc.

La figure 4 récapitule notre proposition pour la classification des obsels dans DSMW. Cette classification nous permet de modéliser les traces des activités des utilisateurs. La signification et la structure des obsels sont les suivants :

– **Key event** : cet obsel peut prendre trois formes (*key down, key up, key press*). Il contient des informations sur le code de la touche clavier et le caractère que l'utilisateur a pressé, y compris les touches de modification telles que Ctrl, Alt et Maj. Notez que le code touche et code caractère sont des concepts différents.
– **Mouse event** : il contient des informations sur l'élément de l'interface sur lequel un utilisateur a cliqué, par exemple les boutons, les liens, les images.
– **Change text** : il contient des informations sur le texte initial et les modifications que l'utilisateur a effectuées sur ce texte initial.
– **Search** : il contient des informations sur une recherche effectuée par l'utilisateur.
– **Save page** : il contient des informations sur la modification de la page du wiki après sa sauvegarde. Un attribut *type* est utilisé pour distinguer les actions : *new* pour une nouvelle page, *edit* pour l'édition d'une

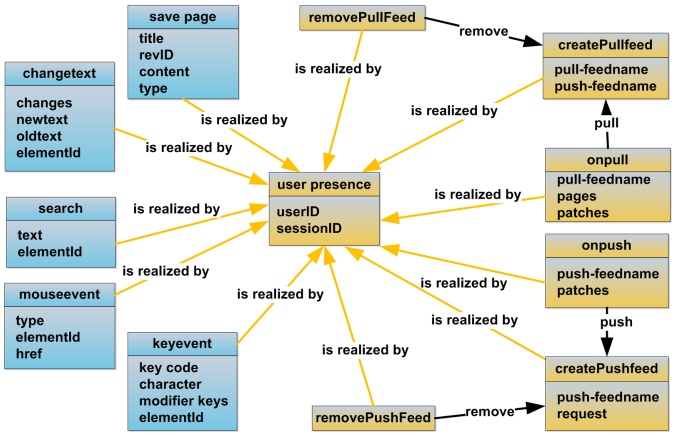

FIGURE 5 – Modèle de trace primaire dans DSMW.

page existante, *cancel* pour l'annulation d'une action, *rollback* pour l'annulation des modifications et le retour à l'état initial.

- **User presence** : il est créé lorsque le système crée une nouvelle session d'un utilisateur. Il contient des informations sur l'utilisateur et sa session.
- **Create PushFeed** : il contient des informations sur la création d'un *PushFeed* telles que le nom et l'URL du *PushFeed* et la requête sémantique conçue lorsqu'un utilisateur crée le *PushFeed*.
- **On Push** : il contient des informations sur l'opération de publication faite par l'utilisateur, y compris les modifications publiées.
- **Remove PushFeed** : il contient les informations sur la suppression d'un *PushFeed*.
- **Create PullFeed** : il contient des informations sur la création d'un *PullFeed* telles que l'URL du *PushFeed* associé.
- **On Pull** : il contient des informations sur l'opération de récupération telles que les correctifs téléchargés et les pages relatives à ces correctifs.
- **Remove PullFeed** : il contient les informations sur la suppression d'un *PullFeed*.

Notez que chaque obsel possède un attribut temporel indiquant la date précise (date et heure) de l'observation. Ces attributs temporels sont fixés par le module de collecte.

4.2.2 Modèle de traces

La figure 5 présente notre modèle de traces. Ce modèle décrit la structure et les types d'obsels qui sont contenus dans une trace primaire. Il illustre également les relations entre les obsels. Il y a deux groupes d'obsels dans le modèle de traces :

- sur le côté gauche de la figure, nous présentons les obsels utilisés uniquement pour construire les traces individuelles : enregistrement d'une page, changement de texte, recherche, événements souris, évènements clavier.
- sur le côté droit, nous présentons les obsels utilisés pour construire les traces de collaboration : présence de l'utilisateur, création de *PushFeed*, création de *PullFeed*, *Push*, *Pull*, suppression de *PushFeed*, suppression de *PullFeed*.

Notre modèle de traces est extensible. Il peut être étendu pour intégrer de nouveaux obsels pour d'autres entités (pages, correctifs, etc.) et d'autres activités (créer un nouveau compte, consulter l'historique des modifications, *etc.*). Les types d'attributs et les types de relations entre les obsels peuvent également être modifiés ou ajoutés, en fonction des besoins spécifiques des applications exploitant les traces.

4.2.3 Modèle de transformations

Les transformations offertes par les SGBT nous permettent de manipuler des traces (filtrage, fusion, *etc.*). Dans notre proposition, nous utilisons les transformations pour construire des traces de collaboration à partir

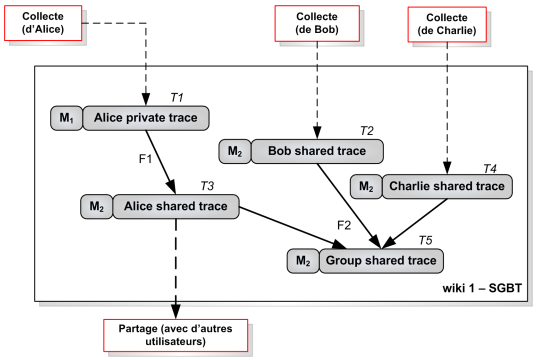

FIGURE 6 – Modèle de transformation des traces de collaboration.

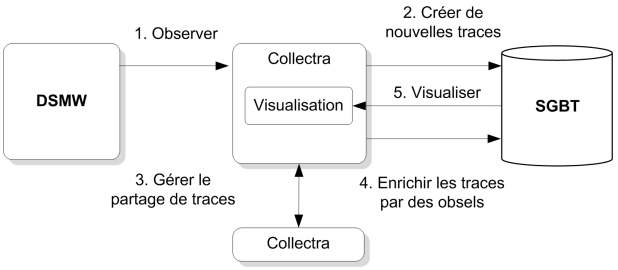

FIGURE 7 – Fonctionnalités implémentées dans Collectra.

des traces individuelles. Nous rappelons que nous distinguons trois types de traces :

- **Private trace** : la trace individuelle d'un utilisateur sur un wiki unique.
- **Shared trace** : pour chaque trace individuelle, une trace partagée est construite. Les traces partagées sont partagées avec d'autres wikis à travers un protocole de partage entre SGBT.
- **Group shared trace** : cette trace est le résultat de la fusion de plusieurs traces partagées. Elle reflète l'activité de collaboration entre les différents membres d'un groupe.

Afin de construire les *shared traces* et les *group shared traces*, nous utilisons des transformations. Pour passer des *private traces* aux *shared traces*, la transformation utilisée est un filtre appliqué à la *private trace*. Le résultat de cette transformation est une nouvelle trace qui contient uniquement les obsels partagés. Pour passer des *shared traces* aux *group shared traces*, la transformation utilisée est une fusion de toutes les traces partagées relatives aux utilisateurs d'un même groupe. Le résultat de cette transformation est une nouvelle trace qui contient tous les obsels partagés de tous les utilisateurs d'un groupe. Les deux transformations sont effectuées automatiquement lors de la collecte.

La figure 6 montre un exemple de notre modèle de transformations appliqué pour trois utilisateurs (Alice, Bob et Charlie). Alice travaille sur wiki_1. Bob et Charlie travaillent sur un autre serveur, mais ils partagent leurs traces avec wiki_1. Cette figure représente le SGBT associé au wiki_1. Nous pouvons voir que le SGBT stocke la trace privée de Alice (T1) et deux traces partagées recueillies auprès de Bob (T2) et Charlie (T4). Le modèle de la trace T1 est M1 qui contient une description des types d'obsels de la trace T1. Il est le même pour toutes les traces individuelles enregistrées sur les différents wikis. Afin de créer une trace partagée pour Alice, nous utilisons une méthode de filtrage F1. Cette méthode est utilisée pour trouver les obsels partagés dans la trace T1 et les copier dans une nouvelle trace T3. La nouvelle trace est associé au modèle M2 qui contient uniquement les obsels partagés. Afin de créer une trace de groupe pour les trois utilisateurs, nous utilisons une méthode de fusion F2. Cette méthode est utilisée pour copier tous les obsels contenus dans les traces partagées des trois utilisateurs. Le résultat de cette fusion est une nouvelle trace T5. Cette nouvelle trace est également associée au modèle M2, car elle ne contient que des obsels partagés.

5 Collectra : un module de collecte de traces pour DSMW

Nous avons implémenté notre proposition dans un module de collecte appelé Collectra. Les données recueillies (obsels) sont stockées dans un SGBT spécifique (un kTBS [7]), qui stocke et transforme des traces collectées. Collectra possède une fonctionnalité lui permettant partager des traces avec d'autres serveurs DSMW. Nous avons doté Collectra de cette fonctionnalité supplémentaire car le kTBS ne la propose pas par défaut.

La figure 7 montre les principales fonctionnalités de Collectra :

1. L'observation des actions des utilisateurs du côté client et leurs conséquences côté serveur ;
2. La détection de nouveaux utilisateurs côté serveur et la création leurs traces dans le SGBT ;
3. La mise à jour des traces partagées à partir d'autres serveurs (synchronisation de traces partagées) et l'intégration de ces traces dans les SGBT ;

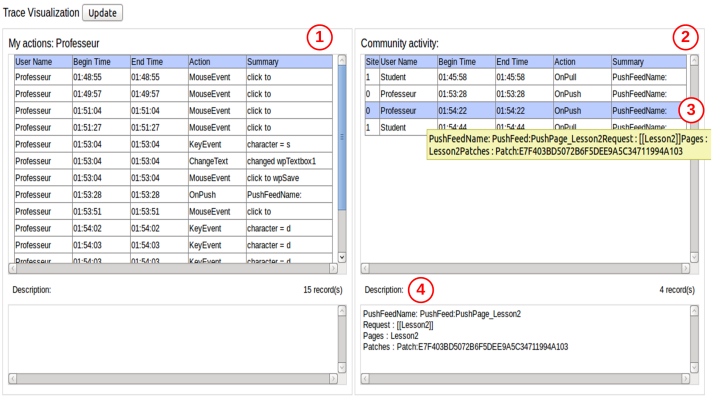

FIGURE 8 – Interface de visualisation de Collectra.

4. L'insertion des données recueillies (obsels) dans les traces du SGBT.

Par ailleurs, nous avons associé à Collectra une interface de visualisation simple des traces pour les utilisateurs finaux afin d'observer les effets de la visualisation de la trace partagée sur la prise de conscience des actions des autres utilisateurs de DSMW. Cet outil de visualisation, présenté figure 8, n'est pas l'élément principal dans Collectra. Cependant, il peut fournir des informations aux utilisateurs sur leurs actions.

6 Comment utiliser les traces pour l'assistance ?

Dans cette section, nous montrons comment les traces peuvent être exploitées au sein d'un moteur d'assistance. Nous montrons d'abord comment les traces collectées permettent d'apporter très simplement, au travers de la visualisation, une première forme d'assistance aux utilisateurs. Dans la deuxième partie de la section, nous discutons des différentes formes d'assistance que nous envisageons d'implémenter en utilisant le raisonnement à partir de traces.

6.1 Visualisation de traces

Afin d'illustrer l'effet des traces recueillies par Collectra, nous considérons un scénario de collaboration entre un professeur et un étudiant dans lequel la visualisation de traces est disponible. Le professeur travaille sur le wiki_1. L'étudiant travaille sur le wiki_2. Ils travaillent ensemble sur le même cours. Les modules de collecte des deux serveurs ont été configurés pour se connecter l'un à l'autre. Dans un premier temps, le professeur modifie la page du cours1 sur son serveur et partage ses modifications. L'étudiant peut consulter l'action de publication du professeur à travers son interface de visualisation de traces. Il récupère les modifications du professeur. Ensuite, l'étudiant édite la page du cours1 sur son serveur et partage également ses modifications. Le professeur peut également consulter l'action de publication de l'étudiant à travers son interface de visualisation de traces. Il peut alors décider de récupérer les modifications de l'étudiant. Grâce à l'outil de visualisation de traces, chacun peut se tenir au courant des actions partagées par les autres. En comparaison avec le scénario original (voir section 2), dans lequel la visualisation de traces n'est pas possible, les utilisateurs finaux peuvent choisir de prendre en compte ou non les différentes modifications disponibles. Cette interface de visualisation est une preuve de concept et n'est pas encore facile à utiliser. Néanmoins, elle montre que la visualisation de traces constitue une amélioration pour les utilisateurs finaux.

6.2 Raisonner à partir de traces

Pour mettre en œuvre des mécanismes d'assistance plus évolués, nous nous appuyons sur le raisonnement à partir de traces. Nous allons donc implémenter un assistant à base de traces tel que défini dans [10]. Un assistant à base de traces exploite plusieurs bases de connaissances (connaissances sur l'application, sur le type d'assistance à fournir, sur le domaine, *etc.*) pour construire des propositions d'assistances contextualisées.

Dans notre cas, les traces collectées peuvent servir différents buts : guider l'utilisateur, proposer des optimisations lors de la réalisation de tâches ou encore, automatiser une tâche. Pour guider l'utilisateur lors de la réalisation d'une tâche, il faut lui présenter les étapes à suivre. Nous proposons de montrer des exemples à l'utilisateur pour illustrer chacune des étapes. Ainsi, lorsqu'un utilisateur demande au système les étapes à effectuer pour faire une action, par exemple créer une nouvelle page dans son wiki, l'assistant recherchera dans la base de traces, ses traces d'utilisations précédentes ou celles d'autres utilisateurs montrant l'action demandée. L'assistant pourra ensuite montrer les traces retrouvées à l'utilisateur afin de le guider dans la résolution de sa tâche. Cette assistance repose donc sur la recherche de traces à partir d'une **signature** connue à l'avance. Cette signature, définie par un expert, indique les différents obsels que l'on doit rechercher, ainsi que l'ordre dans lequel ils doivent apparaître dans la trace.

Reprenons l'exemple de la création d'une page. Cette tâche peut être faite de différentes manières. La recherche de traces correspondantes se fera donc à partir de plusieurs signatures. Une des signatures définies par l'expert indique qu'il faut rechercher les traces correspondantes aux actions suivantes :

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*.

Avec le modèle de traces que nous proposons, ces actions correspondent aux obsels suivants :

O1. Obsels de type *Change text*, attribut *title* = "changed searchInput".
O2. Obsel *Mouse event*, attribut *element_id* = "SearchGo Button".
O3. Obsel *Mouse event*, attribut *element_href* = "`http://localhost/wiki2/index.php?title=NomPage\&action=edit\&redlink=1`". Pour trouver le nom de la page, il faut récupérer dans l'obsel O1, les noms de pages.
O4. Obsels d'édition compris entre O3 et O5. En effet, si l'on décide de montrer un exemple de saisie, il faut récupérer toutes les actions faites dans la zone de saisie (*Key event, Change text*, appui sur les boutons de mise en page, etc.) donc toutes les actions entre la date de fin de l'obsel O3 et la date de début de l'obsel O5.
O5. Obsel *Mouse event*, attribut *location* = NomPage avec le chemin complet, attribut *element_id* = "wpSave".

Le moteur d'assistance créera des traces transformées contenant uniquement les obsels retenus ci-dessus afin de les présenter à l'utilisateur.

Une autre assistance consiste à présenter à l'utilisateur des manières de faire équivalentes pour une tâche donnée. Pour cela, la signature des traces qu'il faut rechercher est identifiée en observant l'utilisateur. Il faut réussir à déterminer la situation dans laquelle se trouve l'utilisateur ainsi que la tâche en cours d'exécution afin de construire la signature. L'assistant recherche ensuite des traces ayant une situation de départ identique ou équivalente et traitant la même tâche. Il peut ensuite adapter les traces retrouvées afin de les faire correspondre au contexte de l'utilisateur.

Le troisième type d'assistance est l'automatisation d'une tâche. L'objectif est de permettre aux utilisateurs de faire faire des tâches spécifiques à l'assistant. L'utilisateur pourra choisir la tâche à effectuer dans une liste qui lui sera proposée ou bien définir lui-même les spécifications de la tâche à réaliser. Dans le premier cas, la liste des tâches disponibles sera créée à partir des traces de la base de traces. L'assistant proposera les tâches pour lesquelles il possède de traces de réalisation. Il devra donc parcourir la base de traces afin de découvrir les signatures des traces disponibles. Dans le second cas, l'utilisateur pourra préciser la tâche à faire en indiquant les étapes de cette tâche. L'assistant devra ensuite faire le lien entre les étapes et les obsels à rechercher dans la trace afin d'identifier la signature des traces à rechercher. Une fois la signature déterminée, l'assistant recherchera dans la base de traces les traces correspondantes et, comme pour l'assistance décrite dans le paragraphe précédent, il adaptera les traces retrouvées afin de les faire correspondre au contexte actuel et aux contraintes de l'utilisateur. Avec la trace adaptée, l'assistant pourra ensuite automatiser la tâche pour l'utilisateur.

7 Conclusion

Dans cet article, nous avons présenté Collectra, un outil pour la collecte des traces d'interaction des utilisateurs dans un environnement Web distribué. Collectra permet de collecter les traces d'interaction des utilisateurs de DSMW. Il a été développé comme une extension de MediaWiki et a été interfacé avec le kTBS, une implémentation en Python d'un SGBT. Il constitue la première brique dans la réalisation d'un assistant à base de traces pour les utilisateurs de DSMW.

Plus généralement, nous avons discuté de la réalisation d'un assistant à base de traces. Nous avons montré pourquoi les traces constituaient une source de connaissances riche pour alimenter un moteur d'assistance, et nous avons discuté des éléments à collecter pour être en mesure de fournir cette assistance. Bien que Collectra soit une implémentation spécifique d'un collecteur de traces, une grande partie de ces modèles et ses principes sous-jacents pourraient être réutilisés pour guider la réalisation d'autres collecteurs.

Dans le contexte de Kolflow, les traces sont collectées dans le but d'alimenter le moteur d'assistance aux utilisateurs. Cependant, les traces collectées par des outils de ce type pourraient être exploitées dans de multiples situations. On pourrait par exemple envisager d'étudier les traces collectées pour analyser la dynamique d'une activité collaborative. On pourrait également observer les traces pour détecter les difficultés rencontrées par les utilisateurs en interaction avec l'outil et envisager des améliorations de l'outil visant à atténuer ces difficultés (re-conception). Naturellement, on pourrait exploiter les traces pour donner un retour à l'utilisateur sur sa propre activité. Les usages restent donc à inventer.

Remerciements

Ce travail est réalisé dans le cadre du projet Kolflow, financé par l'ANR (Agence Nationale pour la Recherche) dans le cadre d'un projet de type CONTINT (code : ANR-10-CONTINT-025). Plus d'informations sur le projet Kolflow sont disponibles à l'adresse suivante : `http://kolflow.univ-nantes.fr/`

Références

[1] Nikolaos Avouris, Vassilis Komis, Georgios Fiotakis, Meletis Margaritis, et Eleni Voyiatzaki. Logging of fingertip actions is not enough for analysis of learning activities. In *AIED 2005*, pages 1–8, 2005.

[2] Owen Bell, Mark Allman, et Benjamin Kuperman. On browser-level event logging. Technical Report TR-12-001, International Computer Science Institute, Berkeley, CA, January 2012.

[3] Alexandre Blansché, Hala Skaf-Molli, Pascal Molli, et Amedeo Napoli. Human-machine collaboration for enriching semantic wikis using formal concept analysis. In *SemWiki 2010 - 5rd Semantic Wiki Workshop at the 7th Extended Semantic Web Conference - ESWC 2010*, Heraklion, Greece, May 2010.

[4] Béatrice Cahour et Falzon Pierre. Assistance à l'opérateur et modélisation de sa compétence. *Intellectica*, 12(2) :159–186, 1991.

[5] Pierre-Antoine Champin. ARDECO : an assistant for experience reuse in CAD. In B. Fuchs et A. Mille, editors, *From Structured Cases to Unstructured Problem Solving Episodes For Experience-Based Assistance (Workshop 5 of ICCBR'03)*, June 2003.

[6] Pierre-Antoine Champin, Amélie Cordier, Elise Lavoué, Marie Lefevre, et Hala Skaf-Molli. User Assistance for Collaborative Knowledge Construction. In ACM DL, editor, *WWW 2012 – SWCS'12 Workshop*, pages 1065–1073, April 2012.

[7] Pierre-Antoine Champin, Yannick Prié, Olivier Aubert, Françoise Conil, et Damien Cram. kTBS : Kernel for Trace-Based Systems, 2011. A reference implementation of the notion of Trace Based Management System. Allows to store and compute modeled traces, and access them through a RESTful interface. http ://liris.cnrs.fr/publis/ ?id=5478.

[8] Damien Clauzel, Karim Sehaba, et Yannick Prié. Modelling and visualising traces for reflexivity in synchronous collaborative systems. In *International Conference on Intelligent Networking and Collaborative Systems (INCoS 2009)*, pages 16–23. IEEE Computer Society, November 2009.

[9] M.J. Coombs et J.L. Alty. Expert systems : an alternative paradigm. *International Journal of Man-Machine Studies*, 20 :21–43, 1984.

[10] Amélie Cordier, Marie Lefevre, Stéphanie Jean-Daubias, et Nathalie Guin. Concevoir des assistants intelligents pour des applications fortement orientées connaissances : problématiques, enjeux et étude de cas. In Sylvie Després, editor, *IC 2010 - 21èmes Journées Francophones d'Ingénierie des Connaissances*, pages 119–130. Presses des Mines, June 2010.

[11] Amélie Cordier, Jean Lieber, Pascal Molli, Emmanuel Nauer, Hala Skaf-Molli, et Yannick Toussaint. WikiTaaable : A semantic wiki as a blackboard for a textual case-based reasoning system. In *4th Workshop on Semantic Wikis (SemWiki2009), held in the 6th European Semantic Web Conference*, pages 18–32, May 2009.

[12] Amélie Cordier, Bruno Mascret, et Alain Mille. Extending Case-Based Reasoning with Traces. In *Grand Challenges for reasoning from experiences, Workshop at IJCAI'09*, July 2009.

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

[14] Jean-Mathias Heraud, Jean-Charles Marty, Laure France, et Thibault Carron. Une aide à l'interprétation de traces : application à l'amélioration de scénarios pédagogiques. In *Environnements Informatiques pour l'Apprentissage Humain (EIAH'05)*, pages 237–248, 2005.

[15] Madeth May, Sébastien George, et Patrick Prévôt. Tracking, analyzing, and visualizing learners' activities on discussion forums. In *Proceedings of the sixth conference on IASTED International Conference Web-Based Education - Volume 2*, WBED'07, pages 649–656, Anaheim, CA, USA, 2007. ACTA Press.

[16] Alain Mille. From case-based reasoning to traces-based reasoning. *Annual Reviews in Control*, 30(2) :223–232, October 2006. Journal of IFAC.

[17] Alain Mille, Mick Caplat, et Mick Philippon. Faciliter les activités des utilisateurs d'environnements informatiques : quoi, quand, comment ? *INTELLECTICA*, 2(44) :121–143, December 2006.

[18] Junghoon Oh, Seungbong Lee, et Sangjin Lee. Advanced evidence collection and analysis of web browser activity. *Digital Investigation*, 8(0) :S62–S70, 2011.

[19] Reinhard Oppermann. Adaptively supported adaptability. *International Journal of Human-Computer Studies*, 40 :455–472, March 1994.

[20] Mick Philippon, Alain Mille, et Guy Caplat. Aide à l'utilisateur : savoir quand intervenir. In *IHM*, volume 264 of *ACM International Conference Proceeding Series*, pages 155–162. ACM, 2005.

[21] Charbel Rahhal, Hala Skaf-Molli, Pascal Molli, et Stéphane Weiss. Multi-synchronous collaborative semantic wikis. In *Wise'09 : International Conference on Web Information Systems*, 2009.

[22] E.M. Roth, K.B. Bennett, et D.D. Woods. Human interaction with an intelligent machine. *Cognitive engineering in dynamic worlds*, 20, 1987.

[23] Stuart Schechter, Murali Krishnan, et Michael D. Smith. Using path profiles to predict http requests. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 457–467, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[24] Lotfi Sofiane 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, January 2011.

[25] Hala Skaf-Molli, Gérôme Canals, et Pascal Molli. DSMW : a distributed infrastructure 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. ACM.

[26] S. Weibelzahl. *Evaluation of Adaptive Systems.* Ph.d. thesis, University of Trier, 2002.

[27] D.D. Woods et E.M. Roth. Aiding human performance II : From cognitive analysis to support systems. *Le Travail Humain*, 51(2) :139–172, 1988.

[28] Qiang Yang, Charles X. Ling, et Jianfeng Gao. Mining web logs for actionable knowledge. In *Intelligent Technologies for Information Analysis*, pages 169–192. Springer Heidelberg, 1998.