Modéliser les traces d'interaction pour raisonne a partir de l'expérience tracée ?

Amélie Cordier, Marie Lefevre, Pierre-Antoine Champin, Alain Mille

Modéliser les traces d'interaction pour raisonner à partir de l'expérience tracée ?

Amélie Cordier, Marie Lefevre, Pierre-Antoine Champin, Alain Mille

Université de Lyon, CNRS
LIRIS, UMR5205, Université Lyon 1, France prenom.nom@liris.cnrs.fr

Résumé :
Dans cet article, nous nous intéressons à la problématique de l'ingénierie des connaissances dans le contexte particulier du Raisonnement à Partir de l'Expérience Tracée (RàPET). Le RàPET est un paradigme de raisonnement dans lequel les sources de connaissances principales sont les traces d'interactions modélisées. Les traces modélisées s'avèrent être des représentations particulièrement adaptées à la fois pour collecter facilement des expériences et pour accompagner l'ingénierie des connaissances à partir de ces expériences. Les traces possèdent les bonnes propriétés que l'on attend d'un outil de représentation des connaissances : souples, riches, expressives et calculables. Afin de mettre en lumière les problématiques d'ingénierie des connaissances soulevées par l'utilisation des traces comme sources de connaissances, nous proposons une rapide étude du RàPET en utilisant le cadre descriptif bien connu du raisonnement à partir de cas (RàPC). Nous discutons ensuite des défis soulevés et montrons les bénéfices attendus à développer et à utiliser ces représentations.
Mots-clés : Raisonnement à Partir de Cas (RàPC), Raisonnement à Partir de l'Expérience Tracée (RàPET), Ingénierie de la dynamique des connaissances.

1 Introduction

Avec la popularisation des activités médiées par des environnements numériques, il devient de plus en plus facile de collecter des traces d'interaction des utilisateurs. Ces traces d'interaction sont un bon moyen de capturer les expériences des utilisateurs sous forme d'inscriptions numériques, tout en s'assurant de conserver également la représentation du contexte dans lequel l'expérience est vécue. Cependant, pour que ces expériences capturées soient réutilisables et exploitables au sein d'un processus de raisonnement artificiel, une attention particulière doit être portée à la façon

dont ces traces sont collectées et représentées. En effet, les traces d'interactions ne peuvent devenir des sources de connaissances potentielles que sous réserve qu'elles soient correctement formalisées, et que des méthodes et des outils appropriés soient développés pour les exploiter. Étant donné l'importance grandissante des traces dans notre vie quotidienne, développer des outils pour raisonner sur les traces nous semble donc présenter des enjeux importants.

Le Raisonnement à Partir de l'Expérience Tracée (RàPET) est un paradigme de raisonnement dans lequel les inférences sont effectuées sur des objets bien particuliers qui sont appelés traces modélisées. Les traces modélisées sont définies comme des enregistrements d'événements temporellement situés, observés durant un processus interactif. Le terme RàPET a été introduit dans Mille (2006). Le RàPET (*Trace-Based Reasoning, TBR*, en anglais) était alors décrit comme étant une extension du cycle traditionnel du RàPC, au sens de Aamodt & Plaza (1994). La motivation principale de cette extension était de dépasser les limites connues du RàPC, limites qui résident principalement dans la nature statique de la structure de représentation des cas, principales sources de connaissances mobilisées par le raisonnement. Dans ce papier initiateur sur le sujet, il était mis en évidence que lorsque l'on utilise de telles structures prédéfinies, il est très difficile de prendre en compte le contexte dans le processus de raisonnement. Par contraste avec le RàPC, le RàPET était alors décrit comme une solution permettant de tirer efficacement profit des expériences non structurées capturées dans ses traces. Les expériences étant conservées au sein des traces, elles restent en permanence représentées « en contexte », contexte qui est alors aisément accessible au moment du raisonnement, lui conférant ainsi d'intéressantes propriétés dynamiques.

Depuis l'introduction du terme RàPET en 2006, ce paradigme a fait l'objet d'un certain nombre de travaux de recherche, ainsi que de nombreux débats et discussions. De ces débats, deux questions particulièrement controversées se dégagent. La première concerne la place de *l'expérience* dans le paradigme de raisonnement. Doit-on parler de raisonnement à partir de traces ou de raisonnement à partir de *l'expérience tracée*, et pourquoi ? La seconde question concerne les liens qu'entretiennent le RàPC et le RàPET. Le RàPET doit-il être considéré comme une sorte de RàPC, ou bien est-ce l'inverse ?

Dans cet article, nous nous attachons à répondre à ces deux questions. Nous discutons de la première question en revenant sur les définitions de contexte et d'expérience. Pour aborder la seconde question, nous pro-

posons une très rapide étude du RàPET en adoptant le cadre descriptif bien connu du RàPC, et nous nous attachons à mettre en évidence les différences fondamentales entre les deux paradigmes ainsi que les enjeux spécifiques soulevés par ces différences. Nous montrons en particulier en quoi les propriétés des traces, notamment leurs propriétés séquentielles, nous obligent à inventer de nouveaux mécanismes de raisonnement et à envisager différemment les problématiques d'ingénierie des connaissances.

Dans la section suivante (section 2), nous proposons un ensemble de définitions pour caractériser le raisonnement à partir de l'expérience tracée, et nous effectuons une revue de l'état des recherches autour du RàPET. Dans la section 3, nous proposons une étude comparative du RàPC et du RàPET. Dans la dernière section de l'article, nous rapportons nos conclusions à l'issue de cette étude et nous discutons des défis de recherche à venir.

2 Raisonnement à partir de l'expérience tracée

Dans cette section, nous commençons par discuter des définitions d'expérience et de trace sur lesquelles nous nous appuyons par la suite. Ensuite, nous proposons un résumé des différents concepts mobilisés par le raisonnement à partir de l'expérience tracée. Plus d'informations peuvent être trouvées dans les articles suivants : Mille (2006); Cordier *et al.* (2009); Settouti *et al.* (2009); Cordier *et al.* (2010); Georgeon *et al.* (2012). Lorsque nous donnons des exemples, nous utilisons le contexte du projet de recherche Kolflow [1] [2]. Kolflow est un environnement Web permettant à des agents (qu'il s'agisse d'agents humains ou d'agents artificiels) de collaborer afin de construire des connaissances qui soient à la fois exploitables par les agents humains et par les agents artificiels. Kolflow dispose d'une interface Web qui constitue un espace collaboratif accessible aux différents agents. Cette interface permet d'éditer (corriger et enrichir) des bases de connaissances. Dans l'environnement Kolflow, les traces sont collectées quand les utilisateurs interagissent avec le système, et elles sont utilisées pour fournir aux utilisateurs une assistance appropriée en fonction de leur profil, de la tâche qu'ils exécutent, et du contexte dans lequel ils se trouvent (plus d'informations sur le projet se trouvent dans Champin *et al.* (2012)).

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

2. Ce travail est financé par l'Agence Nationale pour la Recherche (ANR), au travers du projet CONTINT Kolflow (ANR-10-CONTINT-025)

2.1 Raisonnement, expérience, trace

Nous nous intéressons ici à un mode de raisonnement artificiel dans lequel les objets manipulés au sein des inférences sont des traces modélisées. Plus précisément, nous nous intéressons au processus d'élaboration des connaissances à partir de ces objets, connaissances qui seront, pour certaines, remobilisées dynamiquement au cours du raisonnement. Nous soutenons qu'un tel mode de raisonnement permet de repousser les limites liées à la difficulté de prendre en compte le *contexte* d'élaboration des connaissances.

Nous affirmons également que les traces permettent de capturer de manière souple et relativement naturelle l'expérience des utilisateurs. Cette expérience ainsi capturée sert de support à la construction de connaissances. Dans le cadre de cet article, nous considérons l'expérience comme la collection de pratiques, d'usages, et d'expériences, qu'elles soient individuelles ou collectives. C'est donc l'activité de l'utilisateur dans l'environnement tracé qui est source d'expérience, et ce sont les interactions capturées qui témoignent de cette expérience.

En effet, une façon d'observer l'expérience liée à une activité dans un environnement, est d'observer les marques laissées par les interactions effectuées. Ce processus d'observation doit être effectué de manière intentionnelle, par un observateur averti, c'est-à-dire qui sait attribuer une sémantique à ses observations.

2.2 Modèles et traces modélisées

Dans cette section, nous rappelons les définitions des concepts relatifs aux traces modélisées telles que proposées par l'équipe Silex du LIRIS, définitions sur lesquelles nous nous appuyons par la suite. Intuitivement, on a envie de définir une **trace** comme un ensemble d'éléments temporellement situés. Beaucoup d'enregistrements numériques répondent à cette définition (par exemple les logs, ou les flux RSS, pour ne citer qu'eux). Cependant, en l'absence de modèles explicites décrivant le contenu de ces enregistrements, ces « traces » ne sont que très difficilement exploitables (en général, par des processus *ad-hoc*, non réutilisables). Associer un modèle à une trace permet d'en décrire formellement la structure et le contenu et ouvre donc la possibilité de définir des mécanismes d'inférences puissants, et réutilisables, sur les traces modélisées.

Un **modèle de trace** est une description formelle de la structure et du contenu de la trace. Le modèle fournit des informations sur les propriétés

de la trace (comme l'unité de temps utilisée pour dater les éléments, *etc.*), sur les éléments qu'elle contient (les *obsels*, *cf.* plus bas), et sur les relations entre ces éléments.

Un élément observé, appelé **obsel**, est un élément de la trace. L'obsel est l'inscription numérique d'un événement ayant eu lieu dans le monde réel. On parle délibérément d'obsel et non d'événement pour insister sur le fait que l'obsel résulte d'une action volontaire d'observation et d'enregistrement de cet événement. Les obsels sont typés, et les types d'obsels sont décrits formellement dans le modèle de trace. Chaque type est caractérisé par un nom et un ensemble de propriétés (souvent représentées sous la forme de couples attribut-valeur, bien que les représentations puissent être plus complexes), incluant au minimum une propriété *timestamp* (servant à situer temporellement les obsels de ce type).

Une **M-trace** (pour trace modélisée) est une trace associée à son modèle. La figure 1 donne un exemple de M-trace. L'exemple s'inspire des M-traces que l'on rencontre dans le projet Kolflow. La partie gauche de la figure présente un extrait du modèle de trace. Cet extrait montre que dans la trace, on peut trouver au moins deux types d'obsels (*click* et *typeText*). La trace est affichée sur la droite. Dans cet exemple, la trace montre que l'utilisateur a cliqué sur le champ *search*, a saisi le texte *apple* et a cliqué sur le bouton *find*.

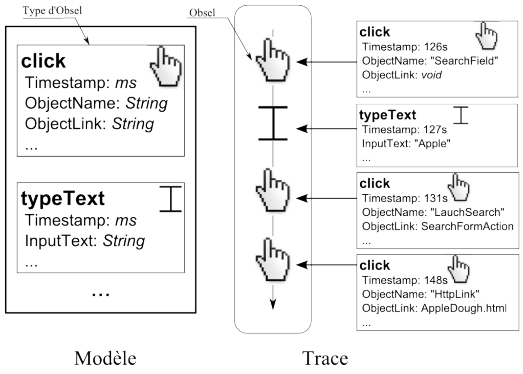

FIGURE 1 – Un exemple de M-trace.

Le **méta-modèle** de traces définit les propriétés que toutes les M-traces et tous les modèles de traces doivent satisfaire pour assurer l'interopérabilité entre les traces. Une description complète du méta-modèle de traces est disponible dans Settouti *et al.* (2009).

2.3 Stocker et manipuler les traces modélisées

Les traces modélisées définies précédemment peuvent être stockées dans un outil que l'on appelle *système de gestion de bases de traces* (SGBT). Un SGBT offre un ensemble d'outils permettant de manipuler des M-traces. La figure 2 présente les fonctionnalités principales d'un SGBT. Les symboles utilisés pour représenter les traces ont été choisis arbitrairement. Les opérations impliquées lors de la manipulation d'une trace sont décrites ci-après.

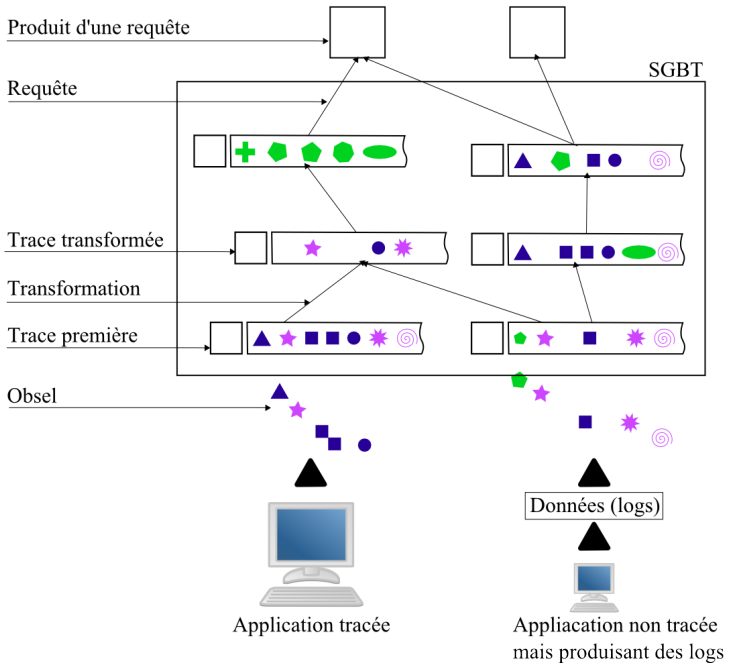

FIGURE 2 – Stockage et manipulation de M-traces dans un SGBT.

Le processus de **collecte** consiste à transmettre les obsels au SGBT et à les stocker dans une trace que l'on appelle *trace première*. Le processus de collecte, mis en œuvre par l'observateur averti dont il était question plus haut, correspond à l'interprétation des marques laissées dans l'environnement par l'activité de l'utilisateur. La trace première, produit de la collecte, est donc construite par l'observateur et correspond à l'état de ses connaissances ainsi qu'au but de son observation. Les traces modélisées peuvent

ensuite être manipulées par le biais de transformations et de requêtes.

Une **transformation** s'applique sur une ou plusieurs M-traces et produit une M-trace, appelée M-trace transformée, pour la distinguer de la M-trace première. Autrement dit, les M-traces transformées sont construites à partir de M-traces existantes. Toutes les M-traces transformées sont stockées dans le SGBT. Les transformations permettent aux utilisateurs d'effectuer de nombreuses manipulations sur les traces, telles que la fusion de plusieurs traces, le filtrage des obsels d'une M-trace, la reformulation de traces, *etc.* Les transformations s'appuient sur des connaissances de transformations dont la nature est discutée plus loin dans cet article. Une transformation est déterministe, ce qui signifie qu'elle donne le même résultat à chaque fois qu'elle est exécutée sur les mêmes traces.

Le **graphe de transformations** représente les relations entre les M-traces. Dans ce graphe, les nœuds représentent les M-traces, et les arcs figurent les transformations appliquées entre les M-traces. Le graphe de transformation est un outil central du SGBT, il permet de naviguer entre les différentes M-traces. Il permet également de conserver des informations sur la provenance des traces, en s'appuyant sur les propriétés déterministes des transformations. Les informations quant à la provenance sont précieuses, en particulier lorsqu'il est nécessaire de retrouver des informations de contexte.

Une **requête** s'applique sur une ou plusieurs M-traces et produit un résultat qui n'est pas une trace (valeur, ensemble de valeurs, texte, document, objet, représentation graphique, fragment d'ontologie, interface de visualisation, *etc.*). Le résultat d'une requête n'est pas stocké dans le SGBT, contrairement au résultat d'une transformation. Cependant, il est toujours possible de garder un lien entre le résultat et la ou les M-traces dont il provient (information de provenance).

Reprenons l'exemple de l'environnement associé au projet Kolflow. Comme mentionné plus haut, les traces de Kolflow enregistrent les interactions entre l'utilisateur et l'environnement web (*click*, *typeText*, *etc.*). Une requête donne, par exemple, le nombre de clicks qu'un utilisateur a fait pendant une période. Les transformations sont utilisées pour exploiter les traces de plusieurs façons différentes. Par exemple, les transformations sont utilisées pour filtrer les traces afin de ne garder qu'une période de temps donnée ou bien pour ne conserver que les obsels représentant certaines actions (par exemple, ne garder que les obsels de type « click » qui renvoient vers une page externe à l'environnement Kolflow). Ces transformations sont aussi utilisées pour réécrire les traces de façon à les présenter

aux utilisateurs avec un vocabulaire adapté dans le cadre de l'assistance. En appliquant de telles transformations aux traces présentées dans la figure 1, on obtient une trace qui contient un nouvel obsel qui est « search for apples ». Comme mentionné plus haut, une trace transformée est aussi une M-trace et a par conséquent son propre modèle.

2.4 RàPET : raisonnement dynamique et interactif

Le RàPET est défini comme un paradigme de raisonnement qui produit des connaissances en effectuant des inférences sur les traces modélisées. Nous affirmons que, étant donné les propriétés décrites plus haut, le RàPET permet un processus de raisonnement interactif, dynamique, et accessible à l'utilisateur. Nous identifions trois étapes principales dans le processus de RàPET : élaboration, remémoration et réutilisation.

L'objectif de l'étape d'**élaboration** est de produire une *signature d'épisode*. Un **épisode** est une M-trace représentant une expérience ou une tâche donnée. Une **signature d'épisode** est une spécification des contraintes qu'un épisode doit satisfaire (schéma, durée, *etc.*). Il existe de nombreuses façons d'exprimer des signatures d'épisodes (règles, ensemble de contraintes, M-traces et contraintes s'appliquant sur ces M-traces, automates, machines à états finis, *etc.*). Par conséquent, il existe également de nombreuses méthodes pour définir ces signatures. Par exemple, si l'on cherche à retrouver une situation passée similaire à la situation courante (afin de fournir de l'assistance à l'utilisateur), alors la signature d'épisode peut être créée à partir d'un fragment de la trace courante de l'utilisateur. L'utilisateur visualise sa trace courante et spécifie les éléments qui sont pertinents et significatifs pour générer la signature d'épisode.

Dans Kolflow, les signatures d'épisodes sont définies de plusieurs façons différentes. Une signature est un sous-ensemble du modèle de trace associé à un ensemble de contraintes. Par exemple, si l'on souhaite retrouver tous les épisodes qui correspondent à la modification d'une page pendant les trois dernières heures, on construit une signature d'épisode comme une transformation recherchant les sous-ensembles de traces commençant pas un obsel du type « edit page » et se finissant avec un obsel du type « save page », avec au moins un obsel « typeText » dans la séquence.

L'étape de **remémoration** consiste à trouver un ensemble d'épisodes qui correspondent à la signature exprimée. Des contraintes peuvent s'appliquer sur le nombre d'épisodes à remémorer, sur l'algorithme de recherche à utiliser, sur la certitude avec laquelle la solution proposée correspond à la signature d'épisode, *etc.*

La phase de **réutilisation** consiste à utiliser l'épisode retrouvé pour résoudre le problème en cours de résolution. Durant cette étape, l'utilisateur et/ou le mécanisme de raisonnement peuvent appliquer des requêtes et des transformations sur les traces pour en exploiter le contenu. De nouveau, en fonction de la tâche en cours d'exécution, il existe de nombreuses façons de réutiliser un épisode. On peut tout simplement afficher l'épisode à l'utilisateur. On peut également rejouer l'épisode à l'identique, ou avec des adaptations contextuelles. L'utilisateur peut également appliquer des requêtes spéciales sur l'épisode pour produire de nouvelles connaissances (comme des informations statistiques, des informations à propos des utilisateurs, etc.).

Durant tout le processus de raisonnement, le RàPET exploite des traces et des transformations stockées dans le SGBT, mais aussi des connaissances supplémentaires représentées dans la base de connaissances associée au moteur de raisonnement (mesures de similarités, stratégies d'adaptation, connaissances de transformations *etc.*). Les trois étapes de raisonnement du RàPET sont étroitement liées et gravitent autour de l'utilisateur. Ce dernier peut interagir avec le processus de raisonnement à n'importe quel moment, et obtenir un feedback immédiat *via* les traces.

La figure 3 donne un aperçu d'une application implémentant du RàPET en adoptant un point de vue « représentation des connaissances ». L'utilisateur interagit avec un système (appelé « système observé ») pour effectuer une tâche donnée. Les traces d'interaction sont collectées et stockées dans le SGBT (flèches fines sur la figure). Le RàPET s'appuie sur toutes les bases de connaissances disponibles. Les lignes épaisses montrent les impacts possibles du RàPET. Le RàPET peut être utilisé par le système observé pour effectuer : de l'automatisation de tâches, des recommandations, de l'assistance à l'utilisateur, de la visualisation de traces, *etc.* (voir (1) sur la figure). De plus, le RàPET peut être utilisé pour effectuer de l'enrichissement dynamique et au fil de l'eau des bases de connaissances du système (2). Il peut faciliter les tâches telles que la classification, la découverte de connaissances, *etc.* De la même façon, le RàPET peut contribuer à enrichir les bases de connaissances externes (3). Ces connaissances externes peuvent être utilisées dans plusieurs buts, dont la réingénierie du système entier. Enfin, les utilisateurs peuvent, au sein d'un processus réflexif, apprendre de nouvelles connaissances produites par le processus du RàPET (4) (Ollagnier-Beldame (2011)).

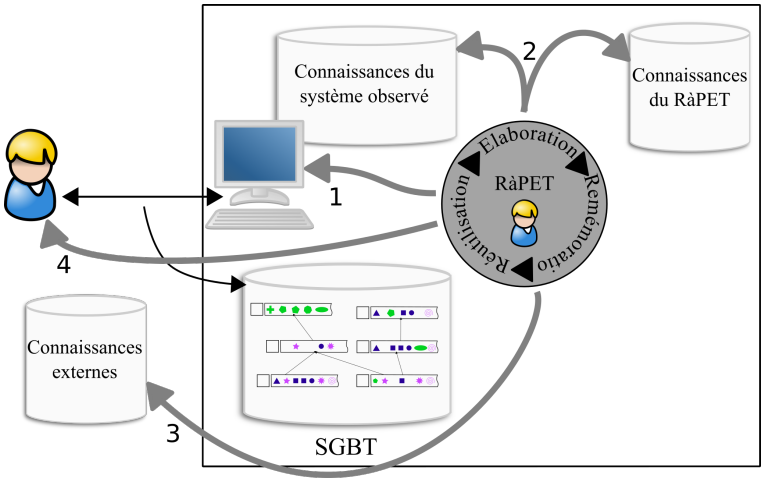

FIGURE 3 – Mise en œuvre du RàPET.

3 Le RàPET du point de vue du RàPC

Cette section a pour objectif de comparer le RàPC et le RàPET. Pour cela, nous adoptons le point de vue du RàPC et nous suivons le cycle proposé par Aamodt & Plaza (1994). Nous nous positionnons également par rapport aux conteneurs de connaissances du RàPC identifié par Richter & Aamodt (2005).

3.1 Une comparaison du point de vue des processus

Nous comparons ici les trois étapes du RàPET (Élaboration, Remémoration et Réutilisation) avec les quatre étapes du cycle proposé par Aamodt & Plaza (1994) (Remémoration, Réutilisation, Révision, Mémorisation).

La phase d'élaboration, bien qu'absente du cycle original d'Aamodt et Plaza, existe malgré tout dans le RàPC, et a notamment été décrite par Mille (2006). Elle consiste à construire le cas cible, qui décrit le problème pour lequel nous recherchons une solution. Souvent, ce cas cible a la même structure que n'importe quel autre cas, mais la partie solution est manquante. En RàPET, cette définition du cas cible ne s'applique pas. La nature du cas cible est en effet très dépendante de la tâche en cours (assistance à l'utilisateur, acquisition de connaissances, résolution de problèmes, *etc.*). C'est pour pallier cette difficulté que la notion d'épisode a été introduite. Une signature d'épisode décrit les propriétés des épisodes que l'on

doit trouver dans les traces ainsi que les contraintes sur le processus de remémoration. La signature d'épisode est la sortie de l'étape d'élaboration. Cette étape d'élaboration, souvent omise en RàPC car jugée triviale, prend donc une importance majeure dans le RàPET. Comme discuté dans la section précédente, plusieurs approches peuvent être utilisées pour construire une signature d'épisode, en fonction du type de problème que l'on traite. Ces approches doivent prendre en compte les caractéristiques de la trace, parmi lesquelles les propriétés séquentielles de celle-ci, sa faible structuration, et le grand nombre d'éléments qu'elle contient. De plus, ces approches bénéficient des mécanismes de transformation fournis par le SGBT.

Après cette étape (explicite ou non) d'élaboration, les deux cycles passent à l'étape de remémoration. En RàPC, la remémoration s'appuie sur l'utilisation de mesures de similarités qui comparent le cas cible aux cas sources dans le but de déterminer quels sont les cas les plus similaires. En RàPET, on ne peut pas se contenter de mesures de similarité, car la signature d'épisode n'est pas directement comparable aux traces. Elle sert au contraire à identifier des épisodes qui la satisfont. Il faut donc trouver de nouveaux mécanismes permettant cette extraction d'un épisode depuis les traces disponibles.

L'étape suivante du cycle du RàPC est l'étape de réutilisation, durant laquelle le cas est réutilisé et éventuellement adapté de sorte à résoudre le problème courant. Cette étape est similaire en RàPET. L'épisode retrouvé est réutilisé afin de fournir une solution au problème courant. Toutefois, étant donné le grand nombre de situations dans lesquelles le RàPET peut être appliqué, l'étape de réutilisation peut être faite de plusieurs façons différentes. Les épisodes peuvent être affichés à l'utilisateur, ils peuvent être transformés pour accomplir un but particulier, ils peuvent être transformés pour être adaptés à un contexte particulier, *etc.* En fonction de la façon dont les signatures d'épisodes sont réutilisées, différentes stratégies s'appliquent et différents conteneurs de connaissances sont impliqués (visualisation, stratégies d'adaptation, *etc.*).

Le RàPC comporte aussi une étape de révision et une étape de mémorisation. Durant l'étape de révision, la solution est corrigée si elle n'est pas satisfaisante. Une fois que la solution est réparée, le cas est considéré comme un cas résolu et il est mémorisé dans la base de cas pour des réutilisations futures. En RàPET, ces étapes n'existent pas, car elles sont naturellement effectuées. En effet, puisque la solution proposée est mise en œuvre dans le système tracé, avec les éventuelles révisions ap-

portées par l'utilisateur, elle est tracée (et donc mémorisée) comme n'importe quelle autre interaction.

3.2 Une comparaison du point de vue des connaissances mobilisées

Dans Richter & Aamodt (2005), quatre conteneurs de connaissances sont identifiés pour caractériser les connaissances mobilisées par le RàPC. Ces conteneurs sont : le vocabulaire du domaine, la base de cas, les mesures de similarité, et les règles d'adaptation.

Les M-traces agissent comme des cas dans le sens où elles contiennent des expériences de résolution de problème. Cependant, à la différence des cas, elles enregistrent également le contexte dans lequel ces expériences se sont déroulées. Le modèle de la trace peut être comparé à la structure du cas en RàPC (construite grâce au vocabulaire du domaine). Ce modèle donne une première sémantique pour l'interprétation de la trace considérée. Il informe le raisonneur sur ce qu'il peut s'attendre à trouver dans la trace et permet donc la définition de mécanismes d'inférences qui peuvent s'appliquer sur la trace. Cependant, une différence importante réside dans le fait que le cas est généralement représenté avec une structure prédéfinie, statique (et souvent très simple), alors que dans les traces, les expériences ne sont pas formalisées explicitement : elles sont immergées au sein d'un ensemble d'informations contextuelles. Par conséquent, les expériences sont plus difficiles à retrouver. Il n'est plus simplement question de définir une mesure de similarité entre de objets structurellement identiques (ou au moins comparables), mais il faut désormais concevoir de nouvelles opérations pour repérer les éléments significatifs d'une expérience au milieu d'un flot d'informations. Cependant, bien que l'approche soit plus complexe, les bénéfices sont nombreux. Cela permet un meilleur accès aux informations contextuelles (puisque les expériences sont conservées dans leur contexte) à n'importe quel moment. Cela permet également de construire dynamiquement les signatures des épisodes que l'on souhaite retrouver, ce qui permet notamment de retrouver différent types d'expériences dans le même ensemble de trace, en fonction des besoins, qui ne sont pas nécessairement identifiés à l'avance.

Les mesures de similarité telles qu'envisagées par Richter ne s'appliquent pas véritablement dans le contexte du RàPET, même si elles sont parfois sollicitées dans une sous-partie du processus de recherche d'épisodes passés. Afin de retrouver des épisodes correspondant à une signature d'épisode donnée, le RàPET utilise des stratégies de recherche variées et des connaissances qui sont stockées dans la base de connais-

sances du RàPET. Les mesures de similarité en font partie, bien entendu, mais d'autres formes de connaissances, telles que les requêtes ou les transformations peuvent aussi être mobilisées pour participer au processus de recherche d'épisodes.

En RàPET, l'adaptation peut également s'appuyer sur de nombreuses formes de connaissances. La forme la plus naturelle est bien entendu la transformation. En effet, une transformation paramétrée permet facilement d'adapter une trace, témoin d'une expérience, à un contexte particulier.

Comme cela a été expliqué précédemment, il est important de noter que, étant donné la dynamique du processus de RàPET, les utilisateurs interagissent à tous les niveaux du processus de raisonnement, et participent ainsi à l'enrichissement des différentes bases de connaissances.

4 Discussion

Lorsque nous avons débuté ce travail, notre objectif était de situer le RàPET par rapport au RàPC. Sur ce plan, nos conclusions ne sont pas tranchées. En effet, le RàPC et le RàPET partagent beaucoup de points communs, mais le RàPET s'attaque à des problèmes plus larges, et ouvre de nouveaux défis, principalement liés à la nature dynamique et séquentielle des connaissances qu'il manipule. Le RàPET ne répond pas à la définition classique du RàPC proposée par Aamodt & Plaza (1994). En revanche, il partage d'intéressants points communs avec la définition du RàPC telle que proposée par Schank (1982). Le système de *mémoire dynamique* de Schank, qui se réorganise elle-même, en continu, et qui stocke dynamiquement des instances d'événements est très similaire au fonctionnement d'un SGBT. La notion d'épisode, quant à elle, rappelle les scripts, qui sont regroupés en MOPS (les MOPS sont des *Memory Organization Packets* que l'on a envie d'assimiler aux d'épisodes) et expliqués par les TOPS (*Thematic Organization Packets*) qui modélisent la connaissance stable et indépendante du domaine.

L'évaluation d'applications implémentant un processus de RàPET est en dehors du périmètre de cet article. Cependant, nous pouvons donner quelques exemples d'applications, dans différents domaines, qui implémentent tout ou partie du processus de RàPET et qui ont prouvé leur efficacité. Par exemple, Mathern *et al.* (2010) ont appliqué un processus de RàPET pour faire de l'extraction dynamique de connaissances à partir de données séquentielles. Dans cet article, ils ont montré que les séquences sur lesquelles ils travaillaient pouvaient être transformées en M-

traces, et que le RàPET appliqué à ces M-traces permettait d'améliorer les résultats du processus d'extraction de connaissances. Georgeon *et al.* (2012) décrivent l'implémentation d'un système de RàPET permettant de modéliser l'activité de conduite automobile à partir des traces collectées grâce à un véhicule instrumenté. Ils ont ensuite montré comment le RàPET pouvait être exploité pour faciliter l'activité d'analyse et de modélisation des processus. Plusieurs exemples ont montré que les traces sont des objets réflexifs : les utilisateurs ont tendance à s'approprier très facilement leurs traces. A titre d'exemple, Zarka *et al.* (2011) proposent une application Web dans laquelle les utilisateurs peuvent visualiser la trace de leurs actions et l'utiliser pour mieux comprendre leur activité. Le RàPET s'avère également être un outil efficace pour proposer de l'assistance aux utilisateurs, comme cela est discuté par Cordier *et al.* (2010) et comme cela est montré dans le cadre du projet Kolflow. Le RàPET pour l'assistance aux utilisateurs a également été étudié dans le cadre des systèmes pour l'apprentissage humain. L'application Visu, par exemple, illustre l'utilisation des traces réflexives dans un contexte d'apprentissage collaboratif (Bétrancourt *et al.* (2011)).

Dans cet article, nous avons montré l'importance de modéliser les traces pour les mobiliser efficacement dans un processus de raisonnement. Nous soutenons que le RàPET, rendu ainsi possible, possède des propriétés intéressantes pour l'ingénierie de la dynamique des connaissances et la réflexivité. Ces propriétés sont discutées dans la suite.

L'originalité du RàPET est qu'il aborde le problème de **l'ingénierie de la dynamique des connaissances**. Dans de nombreux systèmes à base de connaissances, les connaissances sont représentées de façon totalement décontextualisée et sans aucune information quant à leur provenance. Par conséquent, il est impossible (ou très difficile) d'expliquer ou de réviser les connaissances utilisées de façon éclairée. Par opposition, les traces modélisées offrent une solution élégante pour capturer les expériences des utilisateurs ainsi que le contexte dans lequel se déroulent ces expériences. Les traces modélisées apparaissent ainsi comme des riches conteneurs de connaissances dans lesquels il est possible de naviguer afin de mobiliser les éléments de contexte lorsque cela s'avère nécessaire. En effet, les épisodes (représentant des expériences spécifiques) étant toujours liés aux traces dont ils proviennent, il est toujours possible de récupérer les explications (transformation de traces) sur leur provenance.

Grâce au mécanise flexible des transformations, le RàPET constitue un mécanisme de **raisonnement dynamique**. En cela, les propositions du

RàPET peuvent être rapprochées de la notion de RàPC agile proposée par Craw (2009). Dans cet article, Craw suggère que le RàPC devrait être plus agile en exploitant les connaissances contenues dans les cas de façon opportuniste afin de procéder à des mises à jour successives d'une solution qui serait alors évolutive. Dans un processus de RàPET, les mécanismes sous-jacents de transformations facilitent cette dynamicité dans le raisonnement. Les transformations peuvent être sollicitées, combinées et même créées au fil de l'eau afin de répondre aux besoins spécifiques du raisonnement. Fournir une telle flexibilité ouvre bien sûr un certain nombre de défis. Le premier défi s'adresse aux ingénieurs de la connaissance qui se doivent de définir soigneusement les modèles de traces de sorte à ce qu'ils soient suffisamment riches pour permettre différents usages, parfois difficiles à prévoir. C'est un problème qui s'avère particulièrement important au moment de la définition du processus de collecte. Le second défi est lié aux transformations et à leur reproductibilité. Afin de garantir cette propriété, il est nécessaire de s'assurer que les connaissances de transformations sont stables au cours du temps. Par conséquent, si les connaissances de transformation évoluent, il faut pouvoir en assurer la traçabilité. Ceci pose un problème intéressant de gestion des connaissances de transformation. Doivent-elles être gérées par le SGBT ? Comment en garantir l'évolution ? Ceci soulève également la question de la définition des frontières du RàPET. Est-ce que le RàPET doit se limiter à la production et à la manipulation de M-traces, ou bien doit-il couvrir également les processus conduisant à la production d'autres ressources et connaissances, non représentables par des M-traces ?

Nous pensons que le défi majeur soulevé par le RàPET est celui de la **réflexivité**. Bien que la manipulation et le traitement des traces modélisées soient des problèmes important à résoudre, seule l'intégration quasi-transparente des traces dans les applications en permettra une appropriation effective par les utilisateurs finaux. Tout comme les bases de données sont aujourd'hui facilement intégrées dans les systèmes pour assurer des services de stockage et d'interrogation, nous imaginons que les traces deviendront aussi facilement intégrées et apporteront de l'assistance aux utilisateurs, notamment en facilitant des processus réflexif d'appropriation des environnements.

Références

AAMODT A. & PLAZA E. (1994). Case-based reasoning foundational issues, methodological variations, and system approaches. *AI Communications*, **7**(1),

39–59.

BÉTRANCOURT M., GUICHON N. & PRIÉ Y. (2011). Assessing the use of a Trace-Based Synchronous Tool for distant language tutoring. In *Computer Supported Collaborative Learning 2011*, p. 486–493.

CHAMPIN P.-A., CORDIER A., LAVOUÉ E., LEFEVRE M. & SKAF-MOLLI H. (2012). User Assistance for Collaborative Knowledge Construction. In *WWW 2012 - SWCS'12 Workshop*, p. 1065–1073 : ACM DL.

CORDIER A., MASCRET B. & MILLE A. (2009). Extending Case-Based Reasoning with Traces. In *Grand Challenges for reasoning from experiences, Workshop at IJCAI'09*.

CORDIER A., MASCRET B. & MILLE A. (2010). Dynamic Case Based Reasoning for Contextual Reuse of Experience. In C. MARLING, Ed., *Provenance-Awareness in Case-Based Reasoning Workshop. ICCBR 2010*, p. 69–78.

CRAW S. (2009). Agile case-based reasoning : A grand challenge towards opportunistic reasoning from experiences. In *Proceedings of the IJCAI-09 Workshop on Grand Challenges in Reasoning from Experiences*, p. 33–39, Pasadena, CA.

GEORGEON O., MILLE A., BELLET T., MATHERN B. & RITTER F. (2012). Supporting activity modelling from activity traces. *Expert Systems, 29(3), 261-275*.

MATHERN B., BELLET T. & MILLE A. (2010). An Iterative Approach to Develop a Cognitive Model of the Driver for Human Centred Design of ITS. In *European Conference on Human Centred Design for Intelligent Transport Systems*, Proceedings of European Conference on Human Centred Design for Intelligent Transport Systems, p. 85–95 : HUMANIST publications.

MILLE A. (2006). From case-based reasoning to traces-based reasoning. *Annual Reviews in Control*, **30**(2), 223–232. Journal of IFAC.

OLLAGNIER-BELDAME M. (2011). The use of digital traces : a promising basis for the design of adapted information systems ? *International Journal on Computer Science and Information Systems*.

RICHTER M. & AAMODT A. (2005). Case-based reasoning foundations. *The Knowledge Engineering Review*, **20**(03), 203–207.

SCHANK R. C. (1982). *Dynamic Memory : A Theory of Reminding and Learning in Computers and People*. New York, NY, USA : Cambridge University Press.

SETTOUTI L. S., PRIÉ Y., CRAM D., CHAMPIN P.-A. & MILLE A. (2009). A Trace-Based Framework for supporting Digital Object Memories. In *1st International Workshop on Digital Object Memories (DOMe'09) in the 5th International Conference on Intelligent Environments (IE 09)*.

ZARKA R., CORDIER A., EGYED-ZSIGMOND E. & MILLE A. (2011). Rule-Based Impact Propagation for Trace Replay. In A. RAM & N. WIRATUNGA, Eds., *International Case-Based Reasoning Conference (ICCBR 2011)*, LNAI 6880, p. 482–495 : Springer-Verlag Berlin Heidelberg.