Raisonnement à partir de cas dynamique pour la réutilisation contextuelle de l'expérience

Amélie Cordier, Bruno Mascret, Alain Mille
Université Lyon 1, LIRIS, UMR5205, F-69622, France
{prenom.nom}@liris.cnrs.fr

Résumé

Cet article traite du raisonnement à partir de l'expérience tracée (RàPET), un paradigme de raisonnement qui exploite les traces d'interaction laissées par les utilisateurs dans des environnements numériques. Dans le RàPET, les traces d'interaction permettent de conserver les expériences de résolution de problème des utilisateurs « en contexte », leur réutilisation s'en trouve donc facilitée. Par ailleurs, les traces d'interaction peuvent être utilisées comme des sources de connaissances pour découvrir d'autres connaissances utiles au processus de raisonnement. Ce papier décrit les principes du RàPET et propose une architecture générale pour le développement des applications à base de traces. Nous mettons en particulier l'accent sur l'articulation entre le RàPET et les systèmes à base de traces (SBT) dans le contexte du développement d'outils d'assistance aux utilisateurs. Pour cela, nous décrivons les connaissances et les modèles de connaissances impliqués dans les tâches de raisonnement. Nous montrons les avantages que présente l'utilisation des traces comme des conteneurs de connaissances pour la réutilisation de l'expérience. Nous discutons également des questions liées à l'ingénierie des connaissances dynamiques et au rôle du RàPET au regard de cette problématique. Enfin, nous présentons un rapide état des lieux des travaux développés dans ce contexte et nous proposons un plan de travail pour le RàPET, afin d'identifier « ce qu'il reste à faire ». Nous montrons également pourquoi le RàPET peut tirer parti de la dynamique actuelle des développements autour des traces tout en apportant des améliorations aux applications à base de traces.

Mots clés : traces, raisonnement à partir de l'expérience tracée, systèmes d'assistance.

1 Introduction

Le travail présenté dans cet article s'intéresse au défi que présente le développement des systèmes dynamiques et réactifs, capables de s'adapter rapidement, et progressivement, aux changements des besoins et des usages de leurs utilisateurs. Un obstacle majeur s'oppose au développement de telles applications : dans la plupart des cas, les besoins et les modifications des usages ne peuvent pas réellement être anticipés. Ceci complique considérablement la mise en œuvre de mécanismes de raisonnement qui pourraient rendre les systèmes plus souples et plus adaptables. Par exemple, dans le contexte du raisonnement à partir de l'expérience, et en particulier en raisonnement à partir de cas, la capacité du système à s'adapter est limitée par le fait que les modèles de connaissances et les mécanismes de raisonnement sont définis lors de l'étape de conception et sont donc très difficiles à faire évoluer.

Ce papier traite donc du raisonnement à partir de l'expérience tracée (RàPET), paradigme de raisonnement qui vise à apporter des solutions plus dynamiques et plus souples à la problématique de la réutilisation de l'expérience. Nous montrons dans la suite pourquoi s'appuyer sur la notion de « trace » permet de proposer des outils ayant ce type de propriétés. Le RàPET repose sur l'exploitation de *traces d'interaction*. En interagissant avec un système quelconque, l'utilisateur laisse dans l'environnement des traces d'interactions qui constituent des inscriptions numériques de ses propres expériences. Nous considérons que les traces d'interaction sont des conteneurs de connaissances dans lesquels les expériences sont stockées implicitement mais ne sont pas réellement organisées à l'avance. Les expériences passées, appelées *épisodes*, sont seulement identifiées et retrouvées lorsqu'un besoin spécifique apparaît. Ce mécanisme garantit la flexibilité et l'adaptabilité du processus, mais il soulève également des problèmes complexes. Comment retrouver une expérience passée dans une trace ? Quid des connaissances dans un système à base de traces ? Sont-elles toutes contenues dans la trace ou bien doit-on s'appuyer également sur d'autres sources de connaissances ? Quels mécanismes de raisonnement doit-on mettre en œuvre ? Comment faire évoluer les connaissances du système ? Comment valider les connaissances ?

Un autre aspect important dans le RàPET est la notion de contexte. En RàPET, les épisodes sont toujours liés aux traces qui les contiennent. Par conséquent, à tout moment, il est possible de retrouver des éléments contextuels liés à l'épisode courant et de les utiliser pour enrichir le processus de raisonnement. Ainsi, le RàPET permet de manipuler et de réutiliser les expériences de façon beaucoup moins contrainte que si elles étaient représentées sous forme de cas statiques et structurés.

Si l'on s'en tient à la définition du RàPC donnée par Schank [1], alors on peut considérer que le RàPET est une forme de RàPC. En revanche, si l'on cherche à se comparer à la définition donnée par Aamodt & Plaza [2], on peut constater quelques différences non négligeables. En apparence, le principe général du RàPET et celui du RàPC94 (celui d'Aamodt & Plaza, très (trop ?) communément accepté) sont très proches : retrouver une expérience passée, puis l'adapter pour apporter une solution au problème courant. Mais en pratique, les mécanismes mis en œuvre diffèrent. En RàPET, on ne peut plus considérer le raisonnement comme un processus cyclique constitué de cinq étapes successives clairement identifiées. Au contraire, les étapes s'enchevêtrent et les allers-retours entre elles se multiplient rapidement pour préciser la définition du problème ainsi que sa résolution (d'où le côté dynamique). L'étape d'élaboration du RàPC prend ici tout son sens : il s'agit d'identifier clairement le problème puis de construire un ensemble d'indices qui permettront de retrouver dynamiquement un épisode similaire dans la trace. La remémoration consiste à retrouver cet épisode, mais elle est rarement immédiate : des négociations sont nécessaires afin de construire le « bon » épisode, en sélectionnant les « bons » éléments dans la trace. C'est ici que la notion de contexte entre en jeu. C'est elle qui permet de décider si un épisode est bon ou non, étant donné le contexte. L'adaptation est également différente, puisque la stratégie d'adaptation ne dépend plus seulement du problème à résoudre, mais aussi du type d'épisode remémoré. En revanche, la mémorisation du problème résolu devient triviale, puisque tout le processus de résolution de problème est lui aussi stocké dans la trace, par définition, et peut donc être ré-exploité de façon transparente par la suite !

Cet article est organisé de la façon suivante. Dans la section 2, nous revenons sur quelques travaux récents qui visent à développer des applications plus dynamiques et plus adaptables. Ces travaux sont majoritairement issus du champ du raisonnement à partir de cas. Dans la section 3, nous traitons du RàPET en commençant par une rapide description du concept de système à base de traces (SBT) puis en donnant quelques exemples. Ensuite, nous exposons les difficultés soulevées par le développement d'applications à base de traces et nous montrons les bénéfices éventuels d'une telle approche dans le contexte des outils d'assistance à l'utilisateur. L'article se termine par une discussion et une rapide présentation des travaux futurs à explorer dans le contexte du raisonnement à partir de l'expérience tracée.

2 Applications dynamiques et évolutives pour la réutilisation de l'expérience

Depuis quelques temps, plusieurs travaux cherchent à repousser les limites du raisonnement à partir de cas afin de développer des applications plus dynamiques, réactives et adaptées aux utilisateurs. Ces travaux partent tous du constat que les outils actuels sont trop « rigides » et ne sont pas capables d'évoluer pour s'adapter à des besoins non-anticipés ou émergeants. Cette rigidité est notamment due à un ensemble de choix effectués au moment de la conception et qui ne peuvent pas être remis en cause ultérieurement. Par exemple, rares sont les applications de RàPC qui permettent de faire évoluer au fil de l'eau la façon de représenter un cas.

Très récemment, ce problème a été soulevé par Susan Craw dans [3]. Dans cet article, elle propose un défi à la communauté, celui de développer le concept de « *Agile CBR* », que l'on pourrait traduire par « raisonnement à partir de cas agile ». L'idée principale du concept d'*Agile CBR* est de transformer l'approche traditionnelle du raisonnement à partir de cas en une approche dynamique, riche en connaissances, auto-organisée et proposant une méthodologie de résolution de problèmes collaborative. L'idée d'*Agile CBR* s'inspire de la programmation agile qui vise, parmi d'autres objectifs, à s'intéresser aux individus et aux interactions ainsi qu'à proposer une réponse rapide au changement. Ainsi, selon Susan Craw, le côté opportuniste du *Agile CBR* et sa flexibilité visent à fournir la capacité nécessaire pour concevoir des processus dynamiques, capables d'exploiter les informations dont ils disposent en temps réel. Le RàPET revendique les mêmes arguments. Les traces constituent une source de connaissances riche qui devrait être particulièrement utile pour traiter des problèmes de gestion des connaissances soulevées par Susan Craw lorsqu'elle parle des défis posés par l'*Agile CBR*. Par exemple, la question de combiner les expériences de plusieurs utilisateurs pour résoudre le problème spécifique d'un utilisateur particulier peut être efficacement traité par un processus de raisonnement qui s'appuie sur la fusion de traces d'interaction individuelles.

La notion de trace peut également être rapprochée de celle de « provenance ». Nous assistons actuellement à un développement des travaux autour de l'idée de provenance des connaissances. Les travaux décrits dans [4] sont une illustration de cette croissance. Le travail décrit dans cet article s'intéresse donc à la provenance des connaissances mais aussi à la façon d'utiliser les informations sur la provenance pour de nombreuses taches (capturer l'expérience, améliorer le calcul de la similarité, maintenir des bases de cas, etc.). Les auteurs montrent que le raisonnement à partir de cas mémorise toujours les expériences passées sous la forme de cas, mais ne retient aucune information au sujet de la provenance des cas. Or, ils montrent également que cette information sur la provenance des cas a souvent une grande valeur. Avec les traces, les informations de provenance sont toujours conservées puisque les traces contiennent toutes les informations collectées. Les cas ne sont retrouvés que lorsqu'un besoin apparait. Ils sont « construits » à la demande, ce qui permet de choisir, en temps réel, toutes les informations qu'ils doivent contenir, y compris les informations de provenance. Nous pensons que les travaux sur les traces et ceux sur la provenance sont très proches et qu'il serait utile de les combiner.

L'idée de prendre en compte le contexte dans le raisonnement n'est pas nouvelle non plus. Par exemple, [5] montre que combiner les méthodologies du raisonnement à partir de cas et la notion de *context awareness* est une façon nouvelle et performante de modéliser et de raisonner à partir d'informations de contexte. Cette contribution montre comment les comportements des utilisateurs peuvent être appris par une approche « raisonnement à partir de cas », et, pour toutes les raisons citées plus haut, nous pensons que cela pourrait être encore plus efficace avec une approche de RàPET. Illustrant cette idée, [6] utilise des logs d'activités avec des marqueurs temporels. Ces données sont nommées « chemins » (*paths*, dans le texte). Les chemins sont collectés par des applications côté client qui sont implémentées dans plusieurs outils (navigateurs Web, éditeurs de textes, etc.). Ces chemins sont ensuite exploités pour rassembler des informations utilisées pour alimenter des systèmes de recommandation.

Ce que nous qualifions d'*épisode* peut être vu comme un « cas historique » tel que décrit dans [7]. Ce travail propose un environnement pour le développement d'applications de raisonnement à partir de cas historique (*Historical Case-Based Reasoning*). Cet environnement permet à la fois l'expression de données temporelles relatives et absolues, représentant les « histoires » du mode réel sous forme de cas. En HCBR, une base de cas est définie formellement comme une collection d'éléments (temporellement indépendants) couplés à leurs références temporelles. Une approche de RàPET embarque déjà tous ces éléments et y ajoute plusieurs autres possibilités, comme l'extension de contexte, la navigation entre les différents niveaux d'abstraction, la détermination de la provenance, l'élaboration dynamique, etc. Elle est donc plus générale.

D'autres travaux tels que [8], relatifs au partage de l'expérience, peuvent être rapprochés du travail décrit ici. Dans HeyStaks (http://www.heystacks.com), les utilisateurs partagent leurs expériences de recherche avec d'autres utilisateurs. Dans ce travail, les auteurs proposent une approche de classification pour améliorer la pertinence d'un *stak* (une liste de résultats de recherche pertinents dans un contexte donné) tout en palliant les problèmes de bruit dans les résultats (spam, erreurs, résultats périmés, etc.). Il est intéressant de noter que dans ce travail, même si les expériences des utilisateurs sont bien collectées, il reste nécessaire d'aider le système à déterminer quel contexte est réellement en cours d'utilisation. Plutôt que d'utiliser uniquement des classifieurs, il pourrait être possible de combiner l'approche avec une approche à base de traces où la trace serait utilisée pour identifier les éléments nécessaires à la détermination du contexte et pourrait aider à la construction de la *stak*.

À titre d'exemple, pour montrer ce que pourrait être un assistant à base de traces utilisant une approche inspirée du raisonnement à partir de cas, nous pouvons citer [9] qui utilise des techniques de raisonnement à partir de cas pour prédire les objectifs des utilisateurs à partir d'une séquence de ses interactions dans un espace de travail. Une approche à base de trace permettrait de généraliser cette aptitude à assister l'utilisateur, quelque soit sa tâche, en fonction de ce que l'ont peut trouver dans les traces d'interaction de ses expériences précédentes.

Les travaux sur le RàPET en sont encore à l'état prospectif, mais ils ouvrent de nombreuses perspectives applicatives. Dans la suite, nous nous appuyons sur les dernières avancées des travaux autour de la théorie de la trace pour proposer une architecture générale de système mettant en œuvre le RàPET. Afin de rendre nos travaux plus explicites, nous considérons le cas applicatif du développement d'une application d'assistance à l'utilisateur. Nous utilisons cette proposition pour identifier les verrous et proposer des perspectives de recherche.

3 Raisonnement à partir de l'expérience tracée

Dans cette section, nous traitons des principes du raisonnement à partir de l'expérience tracée. Pour cela, il est nécessaire d'introduire les concepts nécessaires à la compréhension de ce qu'est « un système à base de traces », c'est-à-dire un système mettant en œuvre tout ou partie des concepts du raisonnement à partir de l'expérience tracée. Afin de présenter ces concepts de façon plus concrète, nous prenons l'exemple applicatif d'un système à base de traces dédié à l'assistance à l'utilisateur. Nous tenons à attirer l'attention du lecteur sur la distinction qui sera faite plus bas entre « le système à base de traces » et « le système de gestion de bases de traces » (dont le nom a été choisi par analogie avec le système de gestion de bases de données). En fin de section, le lecteur trouvera un exemple visant à clarifier un certain nombre de concepts définis ci-après.

Le système à base de traces. Un système à base de traces (SBT) est constitué de trois principaux composants : un système de gestion de bases de traces (SGBT), une ou plusieurs applications observées et une ou plusieurs applications à base de traces. La figure 1 propose une description de l'architecture générale d'un tel système. La figure est simplifiée. Elle présente un cas où l'on ne considère qu'une application observée et qu'une application à base de traces. Elle illustre la question de la conception d'un assistant à base de traces. Le défi ici est de réutiliser l'expérience pour fournir une assistance aux utilisateurs. Pour proposer une assistance efficace, le système peut soit réutiliser l'expérience de l'utilisateur en s'appuyant sur ses propres traces d'interaction, soit réutiliser et adapter des fragments d'expériences d'autres utilisateurs en explorant leurs traces d'interaction.

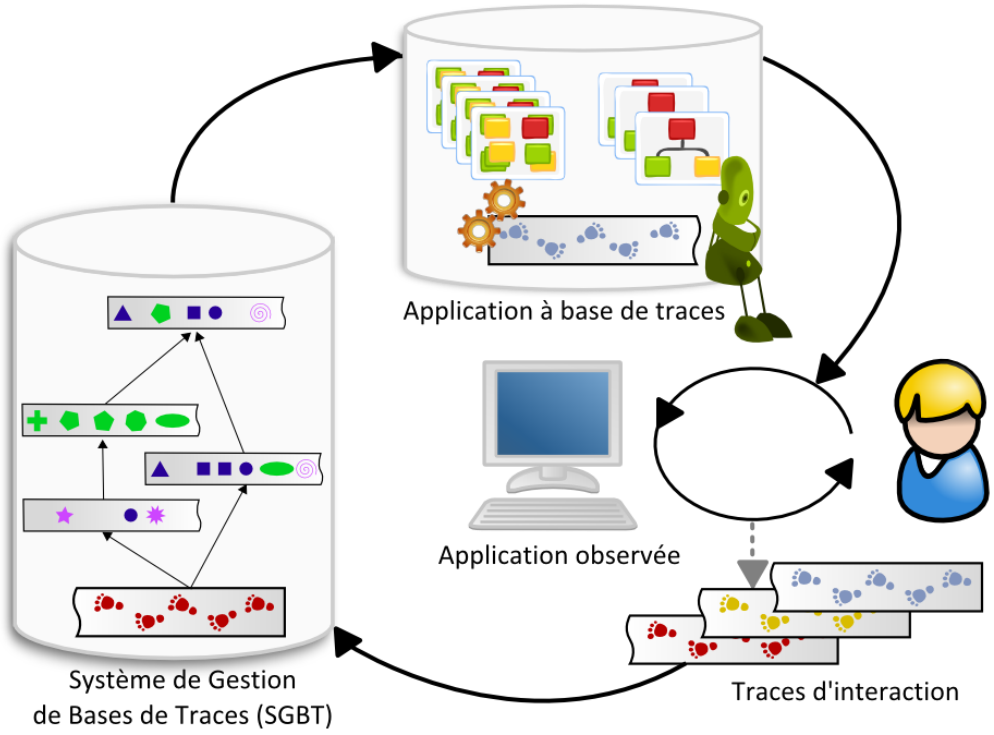

Fig. 1. Architecture générale d'un système à base de traces.

Une façon d'introduire le concept de système de gestion de bases de traces (SGBT) est de faire l'analogie avec un système de fichiers. Le système de fichiers organise, stocke et permet l'accès aux fichiers qu'il contient. Cela dit, sans applications spécifiques pour les lire, les modifier ou les transformer, les fichiers ne présentent que peu d'intérêt pour les utilisateurs qui ne peuvent pas les exploiter directement. Le problème est identique avec les traces. Les traces sont stockées dans le SGBT et si elles ne sont pas

exploitées judicieusement par une application tierce, elles n'ont en général qu'un faible intérêt pour les utilisateurs finaux.

La trace. L'objet central de cette architecture est la trace. Une trace reflète le résultat de l'observation d'une situation donnée. Sur la figure 1, la situation observée est celle de l'utilisation d'une application par un utilisateur (comme c'est souvent le cas). Une trace est un ensemble d'éléments, appelés *obsels* (pour *observed elements*). Une trace modélisée est une trace à laquelle est associé un modèle qui définit formellement la structure et le type des obsels que la trace peut contenir, ainsi que les relations entre ces obsels. Les obsels sont temporellement situés au sein de la trace. Nous pouvons distinguer deux types de traces : les traces premières et les traces transformées. Les traces premières sont le résultat du processus de collecte. Par définition, elles ne sont donc pas transformées. Les traces transformées sont le résultat des opérations de transformation réalisées sur la trace première ou sur une ou plusieurs autres traces transformées. Il existe toujours un lien entre les traces qui ont servi de source à la transformation et la trace résultante (la trace transformée). La figure 2 donne un exemple simple de traces transformées. La trace « du bas » est transformée une première fois, puis cette nouvelle trace est transformée à son tour. Chaque transformation correspond à une abstraction du problème. Le lecteur attentif remarquera que lors du processus d'abstraction, de nouveaux symboles peuvent être introduits afin de représenter des observés (*obsel*) de plus haut niveau. La notion de transformation de traces est détaillée plus bas.

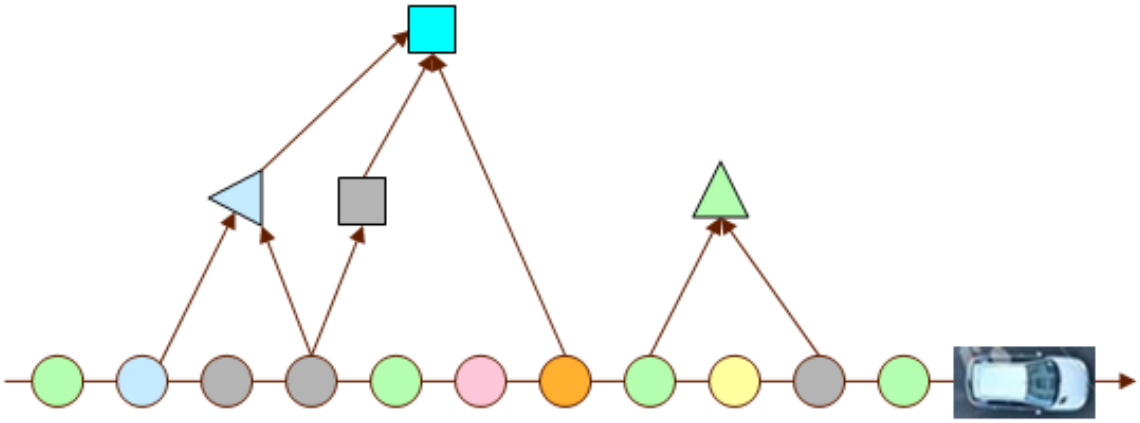

Fig. 2. Une trace et deux niveaux d'abstraction.

Le système de gestion de bases de traces (SGBT). Le SGBT est le composant qui gère les différentes bases de traces. Une base de traces contient à la fois les traces premières ainsi que l'ensemble des traces transformées et leurs modèles associés. Un SGBT fournit des primitives d'entrée/sortie pour manipuler les traces (lecture, écriture, mise à jour, transformations). Les traces sont stockées dans le SGBT au moment du processus de collecte. Ce processus exploite une ou plusieurs sources de collecte qui peuvent être de natures différentes. Soit l'application cible est instrumentée de sorte à ce qu'elle envoie automatiquement au SGBT les éléments à collecter, soit un processus intermédiaire construit la trace première à partir des éléments d'observation disponibles (tels que des fichiers de log des applications observées). Ainsi, une application, un outil, un ensemble de fichiers de logs, une trace d'une autre provenance, etc. constituent autant de sources de collecte. En résumé, le processus de collecte construit la trace primaire qui peut ensuite être transformée en utilisant les opérateurs de transformation existants.

Les transformations de traces. La notion de transformation de traces a une importance capitale dans le RàPET. Une transformation est une opération portant sur une ou plusieurs traces modélisées. Le résultat d'une transformation est une nouvelle trace modélisée, liée aux traces « sources » de la transformation. On peut distinguer plusieurs types de transformations. Nous retiendrons, entre autres, les opérations de filtrage, de fusion et de reformulation. Un opérateur de filtrage, par exemple, produira une nouvelle trace ne contenant que les observés satisfaisant un certain ensemble de critères définis par le filtre. Prenons un exemple très simple dans lequel nous considérons une trace qui contient un ensemble d'observés ayant chacun une propriété « couleur ». Nous définissons un opérateur de filtre imposant de ne garder que les observés dont la couleur est « rouge ». Après application du filtre, la trace transformée ne contiendra que des observés rouges. Cependant, à tout moment, il sera possible de revenir à la trace initiale pour voir, par exemple, quels étaient les voisins immédiats d'un observé donné. Bien entendu, les transformations peuvent être bien plus complexes et peuvent combiner plusieurs opérateurs. Les opérateurs de fusion et de

reformulation fonctionnent de la même façon, mais en plus, ils peuvent être amenés à produire dynamiquement de nouveaux types d'observés pour représenter des observations plus abstraites (comme nous l'avons vu dans l'exemple de la figure 2).

Les opérations de transformation peuvent être manuelles, semi-automatiques ou automatiques. Le SGBT garantit la possibilité, à tout moment, de naviguer entre les traces transformées. En effet, le SGBT conserve en permanence un lien entre la trace transformée et la (ou les) trace(s) source(s) dont elle provient. De plus, des informations sur les opérateurs qui ont permis ces transformations sont également conservées. Autrement dit, le SGBT conserve un treillis de traces transformées afin de garantir la possibilité de navigation. Les traces transformées peuvent être vues comme des représentations différentes d'une même situation à des niveaux d'abstraction différents (cette idée est illustrée dans l'exemple à la fin de cette section). Comme les traces sont liées entre elles, il est toujours possible de se déplacer entre les différents niveaux d'abstraction. Ainsi, on peut considérer un problème à un niveau d'abstraction élevé puis, si cela est nécessaire, redescendre à un niveau d'abstraction plus bas pour obtenir plus de détails. Cette aptitude à naviguer entre les différentes traces modélisées offre donc une flexibilité importante.

Application observée. Le dernier élément de l'architecture est l'outil exploitant les traces. Ce type d'outil peut être connecté au SGBT grâce aux primitives d'entrée/sortie fournies par le SGBT. Les outils peuvent également être connectés avec l'application observée, mais ceci ne rentre pas dans le cadre du travail décrit ici. Ces outils peuvent être de natures diverses : visualisation (interactive) de traces, analyse des usages, assistance à l'utilisateur, etc. Ils doivent s'appuyer sur des connaissances spécifiques pour accomplir certaines tâches (décrites plus bas) en exploitant les traces. Nous appelons ces types de connaissances les connaissances de raisonnement.

Les outils implémentant le paradigme du RàPET doivent donc accomplir plusieurs tâches. La suite de cette section vise à définir ces tâches.

Retrouver un épisode grâce à une signature de tâche. La première tâche consiste à retrouver des épisodes précédents dans la trace en vue de les réutiliser. Pour la recherche d'épisodes, nous nous appuyons sur le concept de signature de tâche. Une signature de tâche contient un ensemble d'éléments distinctifs caractérisant un épisode. À chaque signature de tâche, nous associons un ensemble de mesures de similarité permettant de retrouver dans la trace l'épisode le plus similaire. Nous associons également un ensemble de règles d'adaptation (décrites plus tard). Une autre difficulté vient s'ajouter au problème classique de retrouver des épisodes similaires : la recherche doit s'effectuer parmi les différents niveaux de transformation de la trace. Par conséquent, la recherche doit prendre en compte les différentes représentations possibles d'un même problème. Enfin, cette recherche doit s'effectuer en fonction des informations de contexte disponibles, et ces informations peuvent évoluer au cours du processus. La recherche doit donc être évolutive.

La réutilisation de l'épisode. Une fois retrouvé, l'épisode peut être réutilisé dans beaucoup d'optiques différentes : visualisation, visualisation interactive, rejouage ou adaptation. Dans le cas de la tâche de visualisation, il est nécessaire d'exploiter les différentes connaissances de présentation disponibles de sorte à présenter l'épisode à l'utilisateur de façon claire et compréhensible. La visualisation interactive s'appuie sur le même principe, mais propose en plus à l'utilisateur de naviguer entre les différents niveaux d'abstraction de la trace. La visualisation interactive permet de gérer les différents points de vue sur le même problème de manière efficace et élégante. Pour l'utilisateur, il est possible de considérer le problème à un important niveau d'abstraction puis de redescendre à un niveau plus bas (c'est-à-dire vers une trace transformée plus spécifique) si des détails plus précis sont requis. Le rejouage d'épisodes consiste à retrouver un épisode et à le rejouer à l'identique. Garantir la possibilité de rejouer un épisode à l'identique implique d'importantes contraintes sur la phase de collecte. Notamment, il faut s'assurer que tous les événements nécessaires ont été collectés sous forme d'obels et qu'aucune information ne manquera au moment du rejouage.

L'adaptation d'un épisode. L'adaptation d'un épisode consiste à le réutiliser et à le transformer afin d'apporter une aide à l'utilisateur (assistance, recommandation, exécution automatique d'une tâche, etc.). Dans le contexte du RàPET, il est nécessaire de s'appuyer sur une structure spécifique pour l'adaptation. Cette structure doit offrir un moyen pour le système d'identifier les éléments qui peuvent faire l'objet d'une phase d'adaptation dans l'épisode. Nous nous trouvons ici confrontés à des problèmes similaires à ceux rencontrés en raisonnement à partir de cas et nous avons montré dans un travail précédent [10] qu'il était possible de réutiliser des solutions apportées par les travaux menés dans le contexte du RàPC. En raisonnement à partir de l'expérience tracée, cependant, l'adaptation d'un épisode peut être envisagée à plusieurs niveaux : adaptation de contenu, adaptation de la modalité d'interaction, adaptation de la présentation, navigation entre les différents niveaux d'abstraction (c'est-à-dire adaptation par navigation

entre les traces transformées), etc. Cette diversité parmi les différents types d'adaptation introduit une complexité supplémentaire du point de vue de la représentation des connaissances nécessaires au raisonnement.

Un exemple de RàPET : ABSTRACT. Afin de donner un exemple concret de la façon dont les traces peuvent être collectées, transformées et réutilisées à différents niveaux d'abstraction, nous décrivons brièvement l'application ABSTRACT [11]. ABSTRACT est un projet issu des sciences cognitives et visant à fournir des méthodes et des outils permettant de supporter la découverte de connaissances à partir des traces d'activité. D'un point de vue opérationnel, l'application commence par observer l'activité par différents moyens (collecte de données à partir de différents capteurs, chaque donnée étant marquée par un *timestamp*), ce qui aboutit à la construction de la trace primaire. Par l'intermédiaire des transformations, la trace peut ensuite être manipulée et enrichie avec des symboles plus abstraits. La figure 3 illustre le comportement D'ABSTRACT pour l'analyse du comportement d'un conducteur dans une situation de conduite automobile.

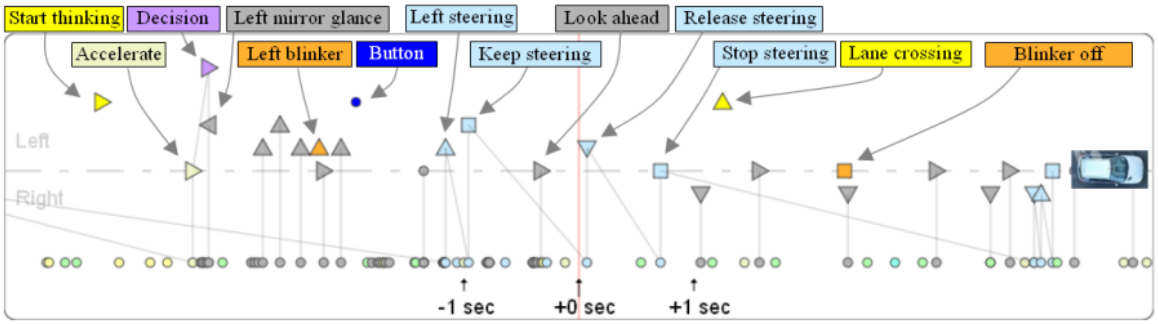

Fig. 3. Transformation des traces dans ABSTRACT.

Résumé. Dans une application à base de traces, le SGBT a un rôle opérationnel : il gère les traces et leurs transformations. Il fournit également les primitives d'entrée/sortie pour manipuler les traces et il permet la navigation entre les différents niveaux de représentation. L'application à base de traces, quant à elle, s'appuie sur des connaissances de présentation et de raisonnement, en plus des traces. Grâce à cela, elle peut atteindre plusieurs objectifs : visualiser un épisode, le réutiliser, l'adapter, etc. Un problème important est celui de l'évolution des connaissances et des modèles de connaissances. L'implémentation de mécanismes pour supporter cette évolution est un problème ouvert. Si la mise à jour de certains conteneurs de connaissances est immédiate (par définition du concept de trace), la mise à jour des modèles de connaissances, en particulier dans le cas des traces modélisées, est plus compliquée et peut nécessiter de faire appel à un processus impliquant un expert. Nous considérons ce problème comme un défi majeur pour le RàPET.

4 Discussion, remarques et conclusion

Le RàPET est un paradigme de résolution de problème dans lequel l'utilisateur et le système collaborent pour résoudre des problèmes mais aussi pour acquérir de nouvelles connaissances. C'est au travers des interactions que l'utilisateur transmet au système les connaissances additionnelles nécessaires pour que celui-ci poursuive son raisonnement. Le système, quant à lui, assiste l'utilisateur en lui apportant des solutions à des problèmes « trop compliqués » pour lui. Au cours de ce processus, la trace constitue un support flexible utilisable à la fois par les humains et par les machines, de façons différentes. Les traces conservent un enregistrement (plus ou moins filtré, en fonction du processus de collecte) des interactions qui ont eu lieu, et donc du processus de résolution de problème. C'est en partie pour cette raison que nous considérons les traces comme une source de connaissances particulièrement riche.

Nous pensons que le raisonnement à partir de l'expérience tracée, parce qu'il offre une approche dynamique et réactive à la question de la collecte et de la réutilisation de l'expérience, est une approche très prometteuse. De plus, les travaux actuels sur les traces sont très actifs et le RàPET pourrait avantageusement tirer profit de cette dynamique. Parmi les recherches sur ce sujet, nous pouvons distinguer deux catégories. La première catégorie concerne les travaux qui traitent des aspects théoriques des traces et de leur gestion (théorie de la trace modélisée, fouille dans les traces pour trouver des motifs pertinents, visualisation

interactive, etc.). La seconde catégorie regroupe les recherches qui visent à produire les applications à base de traces (système à base de traces pour l'analyse de comportement, assistance à l'utilisateur, apprentissage, etc.). Chacun de ces travaux (que nous ne listerons pas ici) a fait l'objet de recherches et d'implémentations plus ou moins indépendantes, mais aucun ne propose actuellement une approche unifiée mettant en œuvre le RàPET comme un élément central permettant à la fois le raisonnement pour la résolution de problèmes, la gestion dynamique des connaissances et la validation des connaissances et du raisonnement. Les réflexions actuelles autour du fameux système à base de traces et du RàPET visent à poser les bases d'une telle approche unifiée.

Parmi les travaux de la première catégorie mentionnée ci-dessus (travaux « théoriques » sur la trace), les travaux traitant des problèmes relatifs aux transformations de traces revêtent une importance toute particulière. En effet, le rôle des transformations est crucial. Les transformations peuvent avoir des objectifs différents : transformer une trace en une trace à un niveau d'abstraction plus élevé, transformer une trace pour en extraire des connaissances ou encore, combiner les deux approches. Cette dernière question soulève un lot de problèmes qui peuvent se résumer aux questions suivantes : est-ce que toutes les connaissances (du point de vue du système) peuvent être représentées sous la forme de transformations, et est-ce que les transformations sont un bon choix pour représenter des connaissances ?

Nous souhaitons également rapprocher les travaux sur les traces et ceux sur la notion de provenance. Nous pensons en effet que ces travaux sont complémentaires. En RàPC, la notion de provenance est utilisée pour améliorer globalement la qualité des résultats produits par le système. Comme nous l'avons montré plus haut, les traces incluent nécessairement ces informations sur la provenance (puisqu'elles gardent une trace de la façon dont les expériences ont été construites). Par conséquent, les informations sur la provenance et l'usage qui en est fait peuvent être remobilisés dans le contexte du RàPET. Les informations sur la provenance peuvent être utilisées de plusieurs façons différentes. Par exemple, dans un système supportant le partage de l'expérience, les informations de provenance peuvent être utilisées pour évaluer la confiance qu'un utilisateur peut avoir en une proposition. La provenance peut aider le système à déterminer quel « type » d'expérience doit être remémorée et réutilisée dans un contexte particulier. Afin d'illustrer ces propos, nous allons détailler l'exemple presque traditionnel du « Copy/Paste » (Copier/Coller, mais la figure est en anglais).

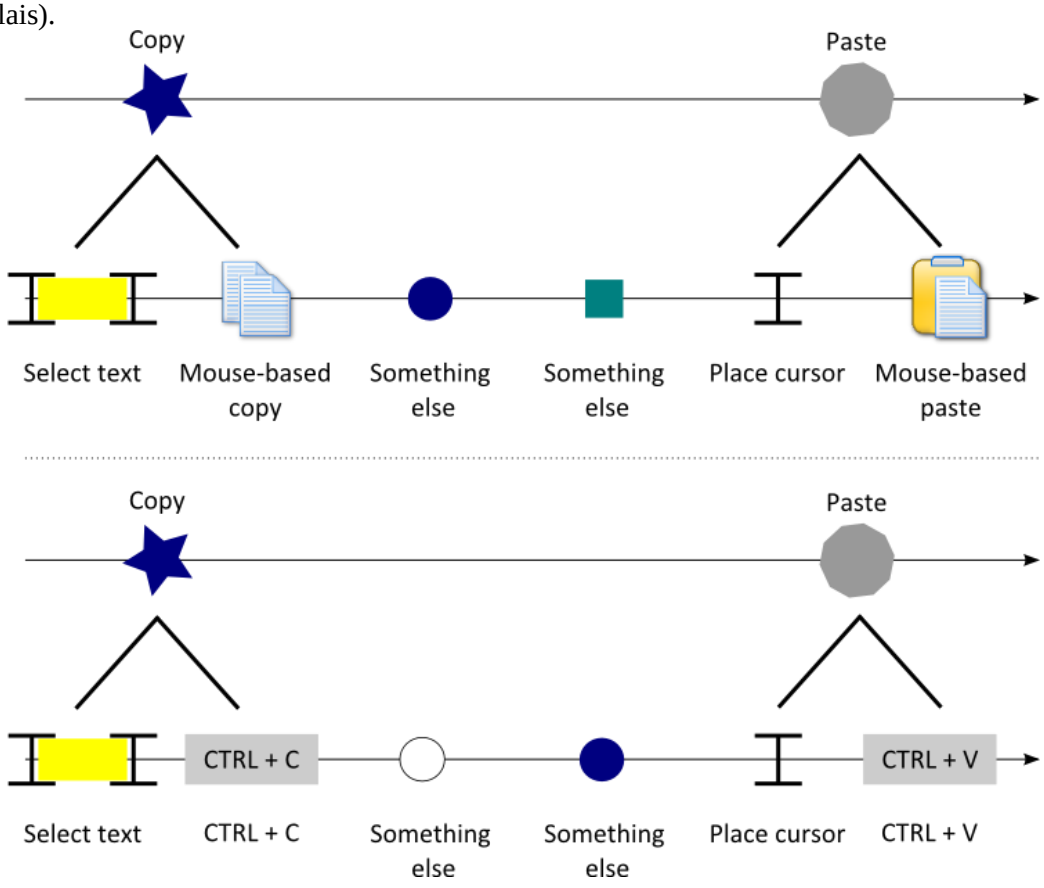

Fig. 4. L'exemple du Copy/Paste.

L'exemple du « Copy/Paste », illustré par la figure 4, montre qu'une même expérience (celle du

Copier/Coller) peut être éprouvée différemment par deux utilisateurs aux usages différents. Supposons qu'un utilisateur aveugle souhaite réutiliser l'expérience de Copier/Coller d'un voyant. Il est raisonnable de penser que la plupart des voyants utilise la souris pour effectuer une telle opération (avec le menu contextuel). Un non-voyant préférera quant à lui n'utiliser que son clavier pour réaliser cette opération. Essayer de partager l'expérience de Copier/Coller entre ces deux utilisateurs en n'utilisant qu'une trace de « bas niveau » est voué à l'échec : les deux utilisateurs ne partagent même pas les mêmes modalités d'interactions avec le système. Il faut donc, dans un premier temps, considérer les traces à un niveau d'abstraction suffisant (c'est-à-dire au niveau ou l'on parle de « Copier/Coller » et non de « Click droit, menu contextuel »). Une fois que le bon niveau d'abstraction est choisi, il s'agit d'adapter la façon d'effectuer l'action. Par exemple, si notre problème est d'aider un non-voyant à faire un Copier/Coller en utilisant comme épisode retrouvé un épisode provenant d'un voyant, il faudra adapter cet épisode en prenant en compte les préférences de l'utilisateur (par exemple, « n'utiliser que le clavier »). Cet exemple soulève également une autre observation que nous trouvons intéressante. Alors que la plupart des utilisateurs semblent préférer la souris pour faire un Copier/Coller (c'est une supposition, surtout pour les besoins de l'exemple), les non-voyants préfèrent probablement une solution « moins commune ». Les informations relatives à la provenance peuvent ici nous aider à trouver les épisodes les mieux adaptés à un utilisateur donné non pas parce qu'ils sont réputés, mais parce que, étant donné le contexte, ils sont plus adaptés. Nous pensons que cette approche est particulièrement importante et prometteuse, en particulier dans le contexte (sans mauvais jeu de mot) des systèmes adaptatifs et des applications d'assistance.

Les perspectives ouvertes par l'implémentation du RàPET sont nombreuses et soulèvent un certain nombre de problèmes. Comment collecter efficacement l'expérience grâce à l'exploitation des traces d'interaction ? Comment définir des mécanismes de transformation pour les traces modélisées et comment faire évoluer ces mécanismes ? Comment adapter les épisodes retrouvés et comment présenter le résultat de cette adaptation aux utilisateurs ? Comment faire évoluer les autres connaissances du système et à partir de quelles sources ? Pour cette dernière question, les traces sont-elles suffisantes ou bien doit-on s'appuyer sur d'autres sources de connaissances, et si oui, comment les intégrer dans le système ?

Toutes ces questions sont autant de pistes de recherche que nous envisageons d'explorer. Nous nous concentrerons en particulier sur le développement de systèmes d'assistance qui soulève des problématiques intéressantes et nombreuses et qui constitue donc un terrain d'expérimentation pertinent.

5 Remerciements

Les auteurs tiennent à remercier les relecteurs de ce papier (ainsi que ceux de la version anglaise) pour leurs commentaires pertinents et leurs questions curieuses (au sens noble). Les éléments de réponses proposés dans ce papier n'étant pas suffisants pour répondre à toutes ces interrogations, nous réfléchissons déjà au prochain article ! Les auteurs remercient également la soigneuse relectrice de cet article.

6 Références

[1] Christopher Riesbeck and Roger Schank. Inside Case-based Reasoning. Northvale, NJ: Erlbaum, 1989.

[2] Agnar Aamodt, and Enric Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. Artificial Intelligence Communications 7, no. 1 (1994): 39-52.

[3] Susan Craw. 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, pages 33-39, Pasadena, CA, 2009.

[4] David B. Leake, Joseph Kendall-Morwick: Four Heads Are Better than One: Combining Suggestions for Case Adaptation. ICCBR 2009: 165-179.

[5] Andreas Zimmermann. Context-awareness in user modelling: Requirements analysis for a case-based reasoning application. Case-Based Reasoning Research and Development, pages 1064–1064, 2003.

[6] Matthew Chalmers. Abstract paths and contextually specific recommendations. Proc. DELOS/NSF Workshop on Personalization and Recommender Systems in Digital Libraries, Dublin, June 2001.

[7] Jixin Ma and Brian Knight. A framework for historical case-based reasoning. Case- Based Reasoning Research and Development, pages 1067–1067, 2003.

[8] Pierre-Antoine Champin, Peter Briggs, Maurice Coyle and Barry Smyth. Coping with Noisy Search Experiences. In 29th SGAI International Conference on Artificial Intelligence (AI-2009). Springer, pages 5-18, Cambridge, December 2009.

[9] Sven Schwarz and Thomas Roth-Berghofer. Towards goal elicitation by user observation. Workshop on Knowledge and Experience Management at GI FGWM, LLWA 2003.

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

[11] Oliver Georgeon, Michael J. Henning, Thierry Bellet, and Alain Mille. Creating Cognitive Models from Activity Analysis: A Knowledge Engineering Approach to Car Driver Modeling. International Conference on Cognitive Modeling, Ann Arbor, MI: Taylor & Francis, pages 43-48.