Spécification et conception de services d'analyse de l'utilisation d'un environnement informatique pour l'apprentissage humain

Diem Pham Thi Ngoc

Thèse de Docteur de l'Université du Maine

Spécialité

Informatique

présentée par

PHAM THI NGOC Diem

Pour l'obtention du titre de

Docteur de l'Université du Maine

Spécification et conception de services d'analyse de l'utilisation d'un environnement informatique pour l'apprentissage humain

Soutenue publiquement le 25 Novembre 2011 devant le jury composé de :

Rapporteurs :	
Mme Agathe MERCERON	Professeur, Beuth University of Applied Sciences
M. Sébastien GEORGE	Maître de Conférences HDR, INSA de Lyon
Examinateurs :	
M. Thibault CARRON	Maître de Conférences HDR, Université de Savoie
Mme Dominique PY	Professeur des Universités, Université du Maine
Directeur :	
M. Christophe CHOQUET	Professeur des Universités, Université du Maine
Co-Encadrant :	
M. Sébastien IKSAL	Maître de Conférences, Université du Maine

Année 2011
Numéro d'Ordre :

Réingénierie des EIAH Dirigée par les Modèles

Remerciements

Je tiens à remercier Mme Agathe MERCERON, Professeur à Beuth University of Applied Sciences - Berlin et M. Sébastien GEORGE, Maître de conférences Habilité à Diriger des Recherches à l'INSA Lyon, de m'avoir fait l'honneur d'être rapporteurs de la thèse.

Je tiens à remercier les membres de mon jury de thèse de doctorat, M.Thibault CARRON, Maître de conférences Habilité à Diriger des Recherches à l'Université de Savoie, Mme Dominique PY, Professeur des Universités à l'Université du Maine.

Je remercie sincèrement M. Christophe CHOQUET, Professeur des Universités à l'Université du Maine d'avoir dirigé cette thèse, pour sa relecture, ses retours, ses conseils toujours très constructifs.

Je remercie infiniment M. Sébastien IKSAL, Maître de conférences à l'Université du Maine, qui a assuré l'encadrement scientifique de cette thèse. Je le remercie pour avoir dirigé cette thèse, pour sa confiance constante et pour sa relecture, ses retours, son soutien et son aide durant la durée de mes recherches.

J'adresse également tous mes remerciements à tous les membres de l'équipe REDiM sur le site de l'IUT de Laval en particulier les autres doctorants, mais aussi Clara et Clément, ainsi que ceux qui sont sur le site du Mans, en particulier Aina, pour leur soutien et leur aide scientifique.

Merci à toutes les personnes qui travaillent à l'IUT de Laval : enseignants du département SRC (Services et Réseaux de Communication) et enseignants du département Informatique pour leurs conseils au niveau des enseignements, merci en particulier à Nathalie, Brigitte, Sylvie et Thierry pour leur soutien administratif, merci au personnel du service informatique.

Enfin, je remercie particulièrement mes amis, tous mes collègues à Cantho et toute ma famille pour leurs encouragements et leur soutien et surtout mon mari Dien et ma fille Minh pour leur soutien moral tout au long de ma thèse.

Résumé

Notre travail de recherche s'inscrit dans le cadre du projet de recherche REDiM (Réingénierie des EIAH Dirigée par les Modèles) qui porte sur la réingénierie d'un scénario pédagogique. Il se focalise plus précisément sur l'analyse de traces collectées en session par un EIAH pour fournir à l'enseignant des indicateurs calculés.

Dans notre contexte de travail, UTL (Usage Tracking Language) permet de définir des indicateurs sous une forme proche des patrons de conception. Il a été conçu pour répondre aux questions de capitalisation et de réutilisation. Par contre, UTL ne disposait initialement pas de moyens pour spécifier formellement la façon de calculer l'indicateur à partir des traces collectées. De plus, les approches par développement ad hoc d'indicateurs ne permettent pas de modéliser de façon formelle la méthode de calcul. En général, les patrons de conception se limitent à la description, ils ne peuvent donc pas être automatisés. Les patrons de données UTL sont analogues. Des descriptions textuelles dans UTL pour produire un indicateur à partir des traces ne permettent pas de générer automatiquement les valeurs d'un indicateur.

Notre principal objectif de recherche a donc été de définir des modèles, des méthodes et des outils pour la formalisation et l'automatisation du calcul d'indicateurs. Nous proposons une nouvelle version d'UTL qui permet de modéliser des indicateurs sous une forme capitalisable, automatisable et réutilisable afin de fournir des indicateurs signifiants à l'enseignant/concepteur. Ces indicateurs peuvent être calculés en temps réel ou après une session, respectivement dans un contexte de tutorat ou de réingénierie du scénario pédagogique. Pour cela, nous avons élaboré une nouvelle version d'UTL qui intègre un langage de combinaison de données nommé DCL4UTL.

L'originalité de notre approche réside dans le fait que cette version permet non seulement de capitaliser des savoir-faire sur les techniques d'analyse d'usage d'un EIAH, mais aussi, avec le langage DCL4UTL (1) de décrire formellement dans une forme générique des méthodes de modélisation et de calcul d'indicateurs à partir des traces collectées par un EIAH, (2) d'intégrer des fonctions externes (qui proviennent d'autres outils d'analyse), et (3) de créer des données intermédiaires paramétrées facilitant la modélisation et la réutilisation de la méthode de calcul d'indicateurs. Nous avons également développé un outil d'analyse pour calculer les indicateurs modélisés.

Cette version est le résultat d'une étude théorique et d’une analyse de l’état de l’art, mais aussi de travaux exploratoires sur la modélisation d’indicateurs et l’analyse de traces. L’approche et le langage ont été validés par plusieurs expérimentations avec plusieurs EIAH existants.

Mots clés : calcul d’indicateurs, langage de combinaison de données, analyse de traces, EIAH, donnée intermédiaire paramétrée.

Sommaire

1 Introduction générale

Sommaire

L'analyse de traces ou de données d'interaction entre un système et ses utilisateurs est un domaine très actif. Elle concerne plusieurs terrains d'application comme le e-commerce, les Systèmes d'Informations (SI), les Environnements Informatiques pour l'Apprentissage Humain (EIAH), etc. Cette analyse concerne l'exploitation et l'interprétation des données d'utilisation d'outils informatiques. Son objectif est d'aider à identifier des motifs récurrents (usage particulier) ou extraire des informations significatives sur l'activité ou le comportement des utilisateurs.

Le travail présenté dans ce document concerne l'analyse de traces dans le domaine particulier des EIAH. Il porte précisément sur la modélisation et le calcul d'indicateurs à partir des traces d'interaction collectées lorsque des apprenants utilisent un système d'apprentissage informatisé. Ce chapitre présente les principaux éléments nécessaires à la compréhension de ce travail. Il s'agit ici de donner un aperçu global du contenu de la thèse. On y trouvera donc une présentation du contexte de recherche, de la problématique étudiée et de notre proposition, ainsi qu'une description des outils et de leurs expérimentations.

1.1 Introduction

Le travail présenté dans cette thèse s'inscrit spécifiquement dans le domaine des EIAH. Ce sont des *"environnements informatiques conçus dans le but de favoriser l'apprentissage humain, c'est-à-dire la construction de connaissances chez un apprenant"* [Tchounikine et al. (2004)]. Ces environnements relèvent de l'informatique, mais aussi d'autres disciplines comme la pédagogie, la psychologie, etc.

Depuis quelques années, la recherche en EIAH s'intéresse à l'étude, la conception et la réalisation de ces environnements. Ces activités se regroupent sous le terme ''ingénierie''. Il s'agit de *"l'ensemble des activités nécessaires à la définition, la conception et la réalisation de projets centrés sur la conception d'artefacts"* [Tchounikine (2009)]. Nous citons ici la définition de Tchounikine [Tchounikine (2002)] qui considère l'ingénierie des EIAH comme *"les travaux visant à définir des concepts, méthodes et techniques reproductibles et /ou réutilisables facilitant la mise en place (conception – réalisation – expérimentation – évaluation – diffusion) d'environnements de formation ou d'apprentissage (dans leur articulation avec les dispositifs informatiques d'aujourd'hui) en permettant de dépasser le traitement ad hoc des problèmes"*. Selon cette définition, l'activité d'ingénierie implique cinq étapes : la conception, la réalisation, l'expérimentation, l'évaluation et la diffusion, la conception est alors une étape indispensable pour la mise en œuvre des étapes suivantes.

Dans les EIAH, le suivi et l'évaluation en session sont des activités importantes. Le suivi des apprenants permet à l'enseignant/tuteur d'observer, d'évaluer et de réguler les activités réalisées par les apprenants en session. L'enseignant/tuteur peut également apprécier la connaissance que les apprenants acquièrent, et les assister dans leur travail en faisant des interventions appropriées (par exemple : en donnant des conseils pour résoudre des problèmes). Dans le cadre particulier des environnements de Formation Ouverte et À Distance (FOAD), les activités effectuées par les apprenants ne sont pas directement visibles par l'enseignant/tuteur. Il est donc nécessaire d'aider les enseignants/concepteurs à comprendre ce qui se passe en session d'apprentissage. L'observation est très souvent utilisée pour comprendre ce qui se passe lors de l'utilisation d'un environnement informatique. Il s'agit d'une action de suivi permettant de recueillir des informations, par exemple sur les comportements des apprenants [De Ketele et Roegiers (2009)]. Dans les sessions d'apprentissage en face à face, l'observation peut être facilement réalisée parce que l'enseignant/tuteur et les apprenants sont réunis au même moment, dans un même lieu. L'enseignant/tuteur peut donc directement observer et superviser les activités de ses apprenants, il peut alors en permanence évaluer, réagir selon ce qu'il voit pour réguler la

session d'apprentissage de la manière la plus appropriée [Romero et Ventura (2007)] [Zinn et Scheuer (2006)]. Dans le cadre des environnements de FOAD, la modalité d'apprentissage ne se limite ni dans l'espace ni dans le temps. Les apprenants et l'enseignant/tuteur ne sont pas obligés de se réunir dans une salle. L'enseignant/tuteur ne peut donc pas directement observer les activités des apprenants. L'enseignant ne sait donc pas ce qui est fait par chaque étudiant, ni ce qui se passe durant la session. Souvent, l'enseignant/tuteur manque des outils appropriés afin de superviser les activités des apprenants dans ce type d'EIAH [Hijon et Velazquez (2006)].

Récemment, plusieurs travaux s'intéressent à l'expression des besoins d'observation [Zendagui (2010)] par l'enseignant/concepteur ainsi qu'à la perception de ce qui se passe en session. Un besoin d'observation est une description de ce que l'enseignant/concepteur veut savoir sur le déroulement d'une session d'apprentissage. Ces besoins doivent être explicités par l'enseignant/concepteur parce que c'est ce dernier (1) qui définit et décrit ce qui doit être réalisé en session, et (2) qui décide de ce qui est nécessaire et important dans l'observation et l'étude d'une session d'apprentissage. Ces besoins d'observation sont utilisés pour la définition des indicateurs [Dimitracopoulou (2004b)], ces derniers aident l'enseignant/concepteur à suivre le déroulement d'une session d'apprentissage, pendant ou après la session. En fait, la plupart des EIAH permettent d'enregistrer les actions et/ou l'activité de l'utilisateur. L'observation et la supervision d'une session d'apprentissage sont très souvent basées sur l'analyse de grands volumes de données collectées pendant ou après la session d'apprentissage [Romero et Ventura (2007)]. Cette analyse peut se composer de différents traitements comme par exemple le filtrage des données, la visualisation ou le calcul d'indicateurs.

La notion d'indicateur est utilisée dans de nombreux domaines : économie, chimie, biologie, éducation, santé, etc. Dans le domaine des EIAH, ils sont utiles pour l'évaluation, la régulation ou la réingénierie de l'activité d'apprentissage ainsi que dans le cas de la réflexivité pour l'apprenant. Dans le projet ICALTS, l'indicateur ‘‘*action collaborative”* (Collaborative Action function indicator CA) [Avouris et al. (2003)] fournit une évaluation quantitative de la collaboration dans un groupe d'utilisateurs pendant un intervalle de temps donné. Pour Bousbia, l'indicateur ‘‘*typologie de navigation”* [Bousbia et al. (2009b)] défini dans un contexte éducatif basé sur le WEB permet de détecter les styles d'apprentissage des étudiants. Parfois les indicateurs sont très visuels, May propose de visualiser l'indicateur ‘‘*lecture d'un message dans un forum”* [May (2008)] qui correspond à une interprétation des différentes actions de lecture réalisées par les apprenants. Nous présentons en détail quelques indicateurs dans le chapitre suivant.

Dans le domaine des EIAH, l'analyse des traces est devenue un axe de recherche très actif. La plupart des EIAH intègrent un outil de collecte des traces. Ces traces sont ensuite utilisées pour différents objectifs. Il s'agit par exemple de découvrir des comportements d'apprenants en session par l'application des techniques de data-mining sur ces traces [Castro et al. (2007), Romero et Ventura (2007)]. L'interprétation de ces traces par des outils de visualisation [France et al. (2006), Heraud et al. (2005), Mazza et Dimitrova (2003)] permet à l'enseignant/tuteur de comprendre ce qui se passe réellement, pour pouvoir réagir avec pertinence et suivre l'activité des apprenants en session. Les traces sont utilisées aussi pour produire des outils réflexifs [May (2009)]. Elles permettent également la supervision [Després et Coffinet (2004), Guéraud et al. (2004)] et l'élaboration d'indicateurs utilisés pour le tutorat [Lekira (2010), Teutsch et al. (2004)].

Depuis quelques années, plusieurs travaux ont porté sur le calcul des indicateurs, par exemple ceux présentés dans le projet ICALTS [Dimitracopoulou (2004b)], ou bien des indicateurs concernant la communication médiatisée (chat et forum) [Anjewierden et al. (2007), Bratitsis et Dimitracopoulou (2005), Avouris et al. (2005)], etc. Ces indicateurs sont souvent faits d'une manière *ad hoc.* Ils sont généralement conçus spécifiquement pour un EIAH sans volonté explicite de réutilisation des méthodes de modélisation d'indicateurs. Les langages de modélisation pédagogique comme IMS Learning Design [IMS-LD (2003)] et Learning Design Language (LDL) [Ferraris et al. (2007)] permettent le partage d'informations sur l'enseignement et leur réutilisation sur différents systèmes. Ils intègrent des concepts liés à l'observation comme le concept ''*observable*'' dans LDL. Il s'agit de points d'observation (par exemple l'état d'interaction (visible, démarré, arrêté, etc.), le déroulement du participant (l'apprenant ou l'enseignant) dans l'activité) et des sous-éléments qui permettent par exemple de construire des tableaux de bord pour les enseignants ou encore des traces à consulter par les apprenants. Ces langages de scénarisation pédagogique ne fournissent pas de méthodes pour la scénarisation de l'observation. Ils se limitent à la description d'observables, ce qui ne permet pas l'exploitation des traces, ni le calcul d'indicateurs. Ils ne permettent donc pas la capitalisation et la réutilisation du savoir-faire en observation.

Actuellement, les travaux concernant le calcul d'indicateurs en EIAH, la capitalisation de méthodes de modélisation d'indicateurs à partir des traces collectées, et leur réutilisation ne sont pas pris en compte. Nous nous intéressons donc particulièrement à ce problème. La modélisation d'indicateurs automatiquement calculables est une tâche complexe qui nécessite des connaissances informatiques. Cette tâche ne peut donc pas être réalisée par l'enseignant non informaticien. De plus, la plupart des systèmes actuels permettant de calculer des indicateurs sont fermés, c'est-à-dire que ces indicateurs doivent être modélisés

et définis avant le développement du système. Lors de l'ajout d'un nouvel indicateur, le système doit être modifié pour pouvoir intégrer cet indicateur. Ces systèmes ne prennent pas en compte les besoins d'observation exprimés par l'enseignant. L'enseignant/concepteur doit alors se contenter des indicateurs prédéfinis. Nous allons prendre en compte ces besoins d'observation dans notre proposition.

Cette thèse s'inscrit dans le contexte de l'analyse de traces générées par les EIAH pour produire des indicateurs. Plus précisément, nous nous intéressons à la modélisation et au calcul d'indicateurs à partir des traces dans une perspective de capitalisation des descriptions génériques et réutilisables de ces indicateurs dans différents contextes. Comme la modélisation des indicateurs est une tâche complexe pour l'enseignant/concepteur, elle demande beaucoup d'expertise ainsi que des compétences techniques particulières. L'objectif principal de cette thèse est de proposer des méthodes permettant de capitaliser la modélisation et le calcul d'indicateurs, ainsi que des outils d'analyse afin d'exécuter ces indicateurs en temps réel ou après une session d'apprentissage. Nous n'imposons pas d'indicateurs, nous nous basons sur les besoins d'observation spécifiés par l'enseignant. L'outil d'analyse proposé exécute ces indicateurs pour adresser leurs résultats à l'enseignant qui pourra les interpréter et les exploiter.

1.2 Définition des principaux termes utilisés

Cette partie présente une définition de certaines notions fréquemment employées dans ce document : trace, trace brute, observable, observé, moyen d'observation, etc. Ces définitions sont présentées par [Choquet (2007)].

- **La trace** : tout utilisateur d'un système interactif est susceptible de laisser des traces de son utilisation. Lorsque l'enregistrement de cette trace par le système informatique est intentionnel, la structure de données générée est une séquence d'actions temporellement situées les unes par rapport aux autres. Communément, ces traces sont enregistrées dans des fichiers de logs. Certains systèmes, comme par exemple LISTEN[1] prétraitent ces traces pour organiser les données collectées dans une base de données (relationnelle ou XML). Certains dispositifs d'apprentissage mettent en place d'autres vecteurs que l'EIAH pour collecter des traces d'utilisation, tels des enregistrements vidéos, des enquêtes par questionnaire. Les traces sont un ensemble de ces données collectées par un dispositif d'apprentissage.

[1] http://www.cs.cmu.edu/~listen/

- **La donnée brute** : par définition, la nature et le format d'une trace sont dépendants du dispositif d'apprentissage considéré. Dans une perspective de capitalisation et de réutilisation des techniques d'analyse de ces traces, un des objectifs de la modélisation de l'observation consiste à représenter ces traces dans un format indépendant du dispositif d'apprentissage qui les a produites. De plus, seule une partie des données constituant une trace est significative pour l'analyse de l'observation. La modélisation de l'observation doit donc également s'attacher à définir des méthodes permettant de repérer et d'extraire des traces ces données significatives. La donnée brute est une telle donnée, obtenue par transformation de la trace : localisée dans la trace par la modélisation de l'observation, extraite de cette trace après la collecte et représentée dans un format indépendant du dispositif d'apprentissage.
- **L'observable** et **l'observé** : Lorsque l'on s'attache à considérer la modélisation de l'observation et l'analyse de l'observation, toute donnée d'observation existe d'abord par la spécification de son obtention avant d'exister comme une donnée tangible, obtenue par l'observation. L'observable est toute variable définie comme devant être évaluée par l'observation de l'utilisation d'un EIAH. L'observé est toute valeur d'observable.
- **Le moyen d'observation** : Modéliser l'observation consiste à définir l'ensemble des techniques d'analyse, automatiques ou non, permettant d'établir un observable. Analyser l'observation consiste à appliquer ces techniques sur les traces de manière à obtenir un observé. Le moyen d'observation est une telle technique. Un moyen d'observation est une fonction permettant d'obtenir un observable d'un ensemble d'observables.
- **L'indicateur** : C'est une variable généralement calculée ou établie à l'aide de données observées, témoignant du mode, du processus ou de la qualité de l'interaction.
- **Le besoin d'observation** : C'est une description de ce que l'enseignant/concepteur veut savoir sur le déroulement d'une session d'apprentissage [Zendagui (2010)].

1.3 Position du travail dans le projet REDiM

Cette thèse s'inscrit dans le cadre du projet de recherche **R**éingénierie des **E**IAH **Di**rigée par les **M**odèles (REDiM) mené au Laboratoire d'Informatique de l'Université du Maine (LIUM[2]),

[2] http://www-lium.univ-lemans.fr

qui porte sur l'étude de l'ingénierie et de la réingénierie des EIAH en adoptant une approche dirigée par les modèles.

- L'approche dirigée par les modèles est une approche de développement des systèmes informatiques centrée sur les modèles. Dans le cadre du projet REDiM, ces modèles décrivent des situations d'apprentissage, plus particulièrement des scénarios pédagogiques [Pernin et Lejeune (2004)]. C'est ''*le résultat du processus de conception d'une activité d'apprentissage, processus s'inscrivant dans un temps donné et aboutissant à la mise en œuvre du scénario. Dans un scénario, on trouve donc des objectifs, une planification des activités d'apprentissage, un horaire, une description des tâches des étudiants, des modalités d'évaluation*'' [Daele et al. (2002)]. Pernin et Lejeune [Pernin et Lejeune (2004)] définissent un scénario pédagogique comme ''*la description du déroulement d'une situation d'apprentissage en termes de rôles, d'activités et d'environnement nécessaire à sa mise en œuvre, mais aussi en termes de connaissances manipulées*''. Ces acteurs distinguent deux types de scénario pédagogique. Un scénario prédictif est décrit *a priori* par un enseignant/concepteur en vue de la mise en place d'une situation d'apprentissage. Par contre, un scénario descriptif décrit *a posteriori* le déroulement de la situation d'apprentissage en y incluant en particulier les traces de l'activité des acteurs et leurs productions. L'approche dirigée par les modèles dans le projet REDiM utilise le modèle de représentation du scénario prédictif pour l'ingénierie et le modèle de représentation du scénario descriptif pour la réingénierie.

De par sa définition, un EIAH est lui-même un système informatique. Il peut donc évoluer ou être éventuellement amélioré, car (1) les besoins des utilisateurs ne sont pas satisfaits de façon optimale ; (2) certaines fonctionnalités sont mal conçues ; (3) de nouvelles méthodes/techniques plus adaptées sont disponibles. En Génie Logiciel, ce processus s'appelle la réingénierie, dont l'intérêt est de diminuer le coût de la maintenance. [Chikofsky et Cross (1990)] définissent la réingénierie d'un système comme ''*le processus d'examen et d'altération d'un système afin de le reconstituer sous une nouvelle forme*''. La réingénierie consiste donc à représenter et implémenter ce qui a été conçu dans une démarche d'ingénierie en utilisant des méthodes et des techniques plus adaptées. Pour les EIAH, on dénombre deux catégories de réingénierie :

i) La réingénierie de la plateforme qui limite son champ d'étude d'un point de vue fonctionnalités et ergonomie.

ii) La réingénierie du scénario pédagogique qui considère la conception des situations d'apprentissage d'un point de vue qualitatif, d'usage, ou encore d'observation, par exemple l'amélioration éventuelle du scénario prédictif en utilisant le scénario descriptif.

Le projet REDiM du LIUM porte précisément sur la deuxième catégorie de réingénierie : la réingénierie du scénario pédagogique. Il considère trois axes de recherche :

- La définition d'un modèle d'organisation du processus d'ingénierie et de réingénierie d'un EIAH et d'un ensemble de termes à adresser aux concepteurs pour leur permettre d'expliciter les éléments méthodologiques d'un tel processus [Corbière (2006)].
- La définition d'outils permettant à des enseignants/concepteurs de capitaliser et d'exploiter les connaissances susceptibles d'aider à l'ingénierie et la réingénierie du scénario pédagogique [El-Kechaï (2008), Zendagui (2010)].
- La prise en compte des moyens d'observation d'une situation d'apprentissage afin d'adresser aux enseignants/concepteurs des indicateurs signifiants [Iksal et al. (2004), Randriamalaka et Iksal (2006)]. Dans cet axe de recherche, le langage Usage Tracking Laguage (UTL) [Choquet et Iksal (2007a)] est un résultat intéressant et important du projet REDiM. UTL a été proposé pour définir et modéliser des données d'observation ainsi que les besoins d'observation associés. Nous présentons en détail ce langage dans le chapitre suivant.

Notre travail concerne l'axe de recherche traitant de l'observation et plus particulièrement le langage UTL. Ce langage peut être utilisé dans toutes les étapes du processus de réingénierie d'un scénario pédagogique. Il s'agit d'un processus itératif qui se compose généralement de trois étapes : la conception d'un scénario pédagogique avant la session, la mise en place du scénario dans une plate-forme et l'analyse des traces liées au déroulement du scénario dans la plate-forme.

- Dans la première étape, l'enseignant/concepteur spécifie le scénario pédagogique. Il décrit les activités, les besoins d'observation et les ressources nécessaires pour modéliser ses intentions pédagogiques. Ce scénario est appelé le scénario prédictif. A ce niveau, UTL est un moyen permettant d'exprimer les besoins d'observation de l'enseignant.
- La deuxième étape correspond au déroulement de la situation d'apprentissage. Les apprenants, les tuteurs et les autres acteurs éventuels utilisent un dispositif d'apprentissage lors de la mise en œuvre du scénario prédictif. Le scénario réalisé

par les apprenants dans cette étape est appelé le scénario descriptif. Les informations que les différents acteurs laissent durant la session d'apprentissage, ainsi que leurs interactions avec l'environnement sont enregistrées par la plate-forme. Puis, ces traces collectées sont restructurées en utilisant UTL, dans une forme indépendante des formats de traces générées par le dispositif. Elles sont les données nécessaires pour l'analyse et le calcul des indicateurs.

- Ensuite, l'étape d'analyse des données collectées par le dispositif d'apprentissage peut être réalisée pendant ou après la session d'apprentissage. Cette tâche utilise plusieurs types de traitements pour obtenir des informations signifiantes et pertinentes (le filtrage [LISTEN (2011)], la transformation [Settouti et al. (2006)], le calcul d'indicateurs [Bratitsis et Dimitracopoulou (2005), Djouad et al. (2009)], la visualisation [Mazza et Dimitrova (2003), May (2008)], etc.) . Les résultats de cette étape sont utilisés par l'enseignant/concepteur lors de l'évolution du scénario pédagogique pour le cycle suivant du processus de réingénierie. Ici, UTL permet de définir les indicateurs et les données nécessaires pour les obtenir.

Au sein de ce processus de réingénierie, notre travail se focalise sur une phase particulière : l'analyse. Comme nous l'avons présenté, UTL permet de définir des besoins d'observation ainsi que des indicateurs. Cependant, il manque des outils et des méthodes afin d'aider les enseignants/concepteurs lors de la spécification de leurs besoins d'observation. Une étude sur la spécification des besoins d'observation est donc réalisée [Zendagui et al. (2008), Zendagui et al. (2009a), Zendagui et al. (2009b)] pour assister les enseignants/concepteurs à spécifier leurs besoins d'observation lors de la phase de conception d'un scénario pédagogique. UTL ne permet pas d'automatiser le calcul des indicateurs car il n'intègre pas la formalisation des méthodes de calcul. Notre travail suit l'étape de spécification des besoins d'observation et correspond donc à la modélisation du calcul d'indicateurs.

Le processus d'analyse reçoit les traces produites en session. Son objectif est de modéliser et de calculer des indicateurs en utilisant la spécification des besoins d'observation produite par l'étape de conception. Ces indicateurs sont utilisés pour l'évaluation, la régulation, la réingénierie et la réflexivité. On utilisera UTL pour modéliser ces indicateurs, afin d'en obtenir une description indépendante des formats de traces du dispositif d'apprentissage et du langage de scénarisation pédagogique. Cependant, dans la version actuelle d'UTL, cette description du calcul des indicateurs est textuelle, ce qui n'est pas automatiquement interprétable par la machine. Avant nos contributions, le processus d'analyse instrumenté par UTL ne pouvait donc pas produire d'indicateurs pour les adresser aux enseignants/concepteurs. Nous avons modifié UTL pour répondre à ce problème.

Comme nous l'avons présenté, notre travail porte principalement sur la réingénierie des scénarios pédagogiques. Nous avons également contribué au projet PEDALO mené aussi au LIUM. Ce projet se focalise sur la régulation de l'activité de tutorat. Il propose une approche qui s'appuie sur des indicateurs calculés à partir des traces d'interactions pour aider le tuteur à réguler l'activité de l'apprenant, mais aussi à réguler sa propre activité [Lekira (2010)]. Mais cette approche manquait de méthodes et d'outils génériques pour la modélisation et le calcul d'indicateurs. Nous avons collaboré avec ce projet pour lui fournir les indicateurs calculés. Nous présentons plus loin dans ce document une expérimentation réalisée en collaboration avec ce projet.

Dans la phase initiale du projet REDiM, les participants ont prévu un processus d'analyse pour l'ingénierie, ce qui a permis d'établir la première version d'UTL. Nous nous sommes intéressés à l'instrumentation des acteurs pendant la session d'apprentissage. Nous travaillons donc afin d'étendre UTL et de permettre le calcul d'indicateurs en temps réel. Nous verrons en conclusion que nous avons actuellement des perspectives pour ce langage concernant l'adaptation, et la modélisation en session de nouveaux indicateurs.

1.4 Aperçu de la problématique

Les travaux existants concernant la spécification et le calcul des indicateurs peuvent être regroupés en deux catégories. La première propose des indicateurs pour l'enseignant. Ces indicateurs sont définis par des chercheurs ou des développeurs de solutions informatiques pour EIAH. Ils sont souvent conçus spécifiquement et faits de manière *ad hoc*. La réutilisation des méthodes de calcul est donc difficile. La deuxième catégorie vise à fournir des méthodes/outils permettant de calculer des indicateurs. Ces indicateurs sont formalisés à partir des besoins d'observation des enseignants. Ils peuvent être produits pendant ou après la session. Notre travail se focalise sur cette deuxième catégorie.

Nous observons que la plupart des travaux proposant des méthodes et des outils permettant de calculer des indicateurs sont dirigés par le système ; C'est-à-dire qu'à chaque fois qu'il y a une évolution/modification, le système informatique doit être modifié. La capitalisation et la réutilisation du savoir-faire en observation d'une session d'apprentissage ne sont que peu ou pas abordées. La capitalisation et la réutilisation facilitent l'observation et l'analyse d'une session. Nous nous intéressons particulièrement à ce problème, parce que la modélisation des indicateurs et leurs méthodes de calcul sont hors de portée de l'enseignant non informaticien.

En ce qui concerne les méthodes permettant la modélisation du calcul des indicateurs, nous considérons deux questions :

i) Comment capitaliser et réutiliser des méthodes de modélisation du calcul des indicateurs ?
ii) Comment automatiser le calcul des indicateurs modélisés ?

Le patron de conception au sens d'Alexander [Alexander et al. (1977)] est une solution pour répondre à des problèmes spécifiques et récurrents. Ce type de patron est souvent utilisé dans le processus de conception d'un logiciel ainsi que dans différents domaines parce qu'il apporte des solutions en matière de capitalisation et de réutilisation. Dans notre contexte de travail, UTL permet de définir des indicateurs sous une forme proche des patrons de conception. Il est conçu pour répondre aux questions de capitalisation et de réutilisation. Par contre, UTL ne dispose pas de moyens pour spécifier formellement la façon de calculer l'indicateur à partir des traces collectées. De plus, les approches par développement ad hoc d'indicateurs ne permettent pas de modéliser de façon formelle la méthode de calcul. Un premier élément de notre problématique de recherche porte donc sur cette représentation formelle de la méthode de calcul des indicateurs. La question posée ici est :

> Comment étendre UTL pour modéliser formellement la méthode de calcul des indicateurs ?

En ce qui concerne la deuxième question, en général les patrons de conception se limitent à la description, ils ne peuvent donc pas être automatisés. Les patrons de données UTL sont analogues. Des descriptions textuelles dans UTL pour produire un indicateur à partir des traces ne permettent pas de générer automatiquement les valeurs d'un indicateur. La prise en compte du langage formel ci-dessus est une première étape qui n'est pas suffisante. La seconde question que nous posons donc est :

> Quels sont les moyens/outils nécessaires pour automatiser le calcul des indicateurs modélisés ?

Nous proposons alors une première problématisation de notre recherche comme suit : nous considérons la définition des modèles, méthodes et outils pour la formalisation et l'automatisation du calcul d'indicateurs. Nous souhaitons proposer une nouvelle version

d'UTL qui doit permettre de modéliser des indicateurs sous une forme capitalisable, automatisable et réutilisable afin de fournir les indicateurs signifiants à l'enseignant/concepteur. Ces indicateurs doivent être calculés en temps réel ou après une session, dans un contexte de réingénierie de scénario pédagogique et de tutorat.

1.5 Travaux réalisés

Notre travail vise à définir des méthodes ou outils qui permettent la modélisation et le calcul des indicateurs sous une forme automatisable, dans une perspective de capitalisation et de réutilisation du savoir-faire en analyse d'une session d'apprentissage. Pour cela, nous travaillons à l'élaboration d'une nouvelle version d'UTL qui intègre un langage de combinaison de données nommé DCL4UTL. Cette version permet de capitaliser des savoir-faire sur les techniques d'analyse d'usage d'un EIAH. L'approche consiste à décrire formellement dans une forme générique des méthodes de modélisation et de calcul d'indicateurs à partir des traces collectées par les EIAH. Nous trouverons plus loin dans ce document notre proposition concernant la spécification de cette version.

Nous proposons un processus d'analyse incluant l'enseignant, l'analyste et le développeur. Nous posons pour hypothèse que l'enseignant est l'auteur du scénario pédagogique et qu'il est capable d'exprimer ses besoins d'observation. Toutefois, la transformation du besoin en la spécification d'un indicateur nécessite des compétences particulières. L'analyste aide alors l'enseignant à modéliser ses indicateurs, tandis que le développeur programme des sondes logicielles à intégrer dans l'EIAH cible pour répondre aux exigences du scénario d'observation.

Pour automatiser le calcul des indicateurs, nous avons développé un interpréteur intégré dans un outil d'analyse. Pour valider notre proposition, nous présentons tout d'abord une utilisation de DCL4UTL avec un indicateur connu du projet ICALTS. Puis, dans le domaine des EIAH, nous avons utilisé des traces générées par un environnement réalisé au laboratoire Systèmes Communicants (SysCom[3]) de l'Université de Savoie pour modéliser et calculer des indicateurs après une session. Ensuite, dans un domaine autre que les EIAH, nous avons travaillé en collaboration avec l'Equipe Lavalloise Handicaps et Innovations Technologiques (ELHIT) pour modéliser et calculer des indicateurs après une session avec des traces générées par le système VAPS (Virtual Action Planning Supermarket). Ce dernier a été développé pour l'aide au diagnostic des dommages d'attaques cardio-vasculaires. Finalement, nous avons fait une expérimentation avec les traces générées par le dispositif

[3] http://www.syscom.univ-savoie.fr/

d'apprentissage Hop3x. C'est un environnement qui permet aux étudiants d'apprendre la programmation orientée objet avec le langage Java. Ce dispositif d'apprentissage est utilisé au département informatique de l'Université du Maine. Dans cet environnement, les indicateurs sont calculés en temps réel.

1.6 Présentation du document

La suite de ce document se compose de sept chapitres décrivant notre positionnement scientifique, notre problématique et nos objectifs, nos propositions et la mise en place de nos propositions.

- Le chapitre 2 est consacré à la présentation de notre positionnement scientifique. Dans cette partie, nous présentons des travaux existants menés en analyse des traces dans les EIAH, ainsi que les différentes approches qu'intègrent ces travaux. Puis, nous concluons ce chapitre par une discussion sur l'inadéquation de ces travaux par rapport à notre travail.
- Le chapitre 3 du document détaille notre problématique, présente notre objectif de recherche en s'appuyant sur le positionnement scientifique. Des questions de recherche sont posées et l'approche méthodologique est présentée.
- Le quatrième chapitre de ce document présente notre proposition. Nous détaillons dans ce chapitre le processus d'analyse des traces que nous proposons dans le contexte du projet REDiM.
- Le chapitre 5 présente une nouvelle version d'UTL qui intègre le langage de combinaison de données pour UTL, appelé DCL4UTL. Nous spécifions comment DCL4UTL est utilisé pour modéliser de nouveaux indicateurs incluant leur méthode de calcul à partir de différentes données UTL par un exemple avec un indicateur connu du projet ICALTS. Cette instrumentation aide à comprendre nos propositions. Cette section présente également comment ces méthodes de calcul peuvent être réutilisées dans d'autres contextes.
- Le chapitre 6 concerne l'aspect technique et informatique de la thèse. Il porte sur l'architecture générale de l'outil d'analyse qui exécute le langage DCL4UTL. Cet outil est conçu pour faciliter l'activité d'observation de l'enseignant pendant la session. Nous détaillons l'architecture de l'interpréteur DCL4UTL qui est une partie et aussi le noyau de l'outil d'analyse. Nous proposons également l'intégration de fonctions externes dans cet outil d'analyse. Finalement, deux applications de cet outil pour calculer des indicateurs après la session sont décrites avec les traces générées par le système VAPS et un environnement réalisé au laboratoire SysCom.

- Le septième chapitre présente essentiellement la mise en œuvre de notre proposition dans le contexte du DUT Services et Réseaux de Communication, avec des étudiants en première année, dans lequel des indicateurs sont calculés en temps réel à partir des traces générées par la plateforme Hop3x. Les résultats de ces indicateurs sont utilisés pour aider le tuteur à réguler l'activité de l'apprenant, mais aussi afin de réguler sa propre activité puis à améliorer le scénario d'apprentissage. Dans cette section, nous discutons également de la possibilité du langage DCL4UTL à répondre aux besoins d'observation de l'enseignant, mais aussi de la performance du système implémenté par rapport aux objectifs que nous avons posés.
- La dernière partie de ce document présente un bilan de notre travail, des résultats obtenus et aussi des limites de nos propositions. Ce chapitre se conclut avec quelques perspectives en termes de travaux que nous envisageons de poursuivre.

2 Positionnement scientifique

Sommaire

Dans ce chapitre, nous situons notre approche dans le cadre général des travaux menés en analyse des traces collectées par les EIAH pour formaliser et calculer des indicateurs. Dans notre travail, ces indicateurs sont modélisés et formulés à partir de besoins d'observation définis par les enseignants/concepteurs lors de la conception d'un scénario pédagogique. Nous nous intéressons à ces travaux dans l'optique de la modélisation du calcul d'indicateurs pour la réingénierie du scénario pédagogique et pour le tutorat. La première partie du chapitre sera consacrée à une présentation générale de l'analyse de traces. La deuxième partie présente des langages qui peuvent être utilisés pour la modélisation des indicateurs. La troisième partie présente les travaux sur la définition des indicateurs spécifiques dans un contexte particulier. Dans une dernière partie, nous nous intéressons aux travaux sur la modélisation des indicateurs et sur la méthode générique pour les calculer. Nous discutons ensuite de la pertinence de ces travaux. A partir de cette discussion, nous posons le cadre de notre travail et situons notre proposition scientifique.

2.1 L'analyse de traces

Dans le domaine des EIAH, l'observation est une activité importante parce qu'elle aide les enseignants/tuteurs à comprendre ce qui se passe dans une session d'apprentissage, à percevoir comment l'apprenant réalise son activité ou à évaluer une session. Selon [Postic et De Ketele (1988)], "l'observation *est un processus dont la fonction première immédiate est de recueillir de l'information sur l'objet pris en considération".* Dans les EIAH, ces informations recueillies peuvent être des données enregistrées dans les fichiers de logs ou des enregistrements vidéo. Toutes ces données sont volumineuses et peuvent provenir de différentes sources. Elles sont difficiles à exploiter et à comprendre par l'utilisateur. Une étape d'analyse sur ces données est donc nécessaire afin de produire les informations signifiantes et utiles pour l'utilisateur.

L'analyse de données en général, ou l'analyse de traces en particulier concerne le traitement et l'organisation des données brutes pour extraire ou produire des informations utiles. En fait, ces données brutes peuvent prendre différentes formes comme des mesures (poids, climat, etc.), des réponses à une enquête, des observations, des fichiers de log, etc. Elles peuvent être très utiles, mais aussi difficiles à comprendre et à utiliser. L'analyse de ces données brutes est donc réalisée pour fournir à l'utilisateur des informations utiles. Par exemple, un fichier de log qui enregistre toutes les activités des apprenants dans une session est difficile à exploiter pour les enseignants/tuteurs. Par contre, ces derniers peuvent interpréter les résultats de l'analyse de ce fichier (par exemple le parcours d'un étudiant, le nombre d'apprenants qui ont participé au test, est-ce que les apprenants ayant réussi le test ont eu recours à des ressources extérieures ? etc.).

En informatique, l'analyse de données concerne plusieurs axes comme l'analyse de logs des systèmes ou des sites Web, l'analyse de données des systèmes d'e-commerce, l'analyse de traces des systèmes d'apprentissage, etc. L'analyse consiste souvent en trois étapes principales : le recueil des traces, le traitement des traces en appliquant des techniques d'analyse sur ces traces et l'interprétation des résultats.

Dans la plupart des systèmes informatiques, le fichier de log est une approche de collecte de données. Un fichier de log est "*un enregistrement électronique des interactions qui ont eu lieu entre un système et ses utilisateurs*" [Jansen et al. (2009)]. Ces fichiers de logs peuvent provenir de différents systèmes comme des sites web, des systèmes d'information, des blogs, des e-journaux, etc. L'analyse de ces fichiers est une méthode de recherche permettant l'étude du comportement de l'utilisateur. Cette méthode est utilisée depuis très longtemps [Meister et Sullivan (1967)].

Selon [Gaudioso et Talavera (2006)], les deux principales approches d'analyse des données sont : *l'analyse des données dirigée par l'hypothèse* et *l'analyse des données dirigée par la découverte*. Comme le montre la figure 2-1, dans l'approche d'analyse des données dirigée par l'hypothèse, l'utilisateur commence à partir d'une question et explore des données pour confirmer son intuition. Au contraire, dans l'approche d'analyse des données dirigée par la découverte (ou data-mining), l'hypothèse est automatiquement extraite à partir des données pour trouver des motifs plus complexes qui se rapportent à différents aspects des données.

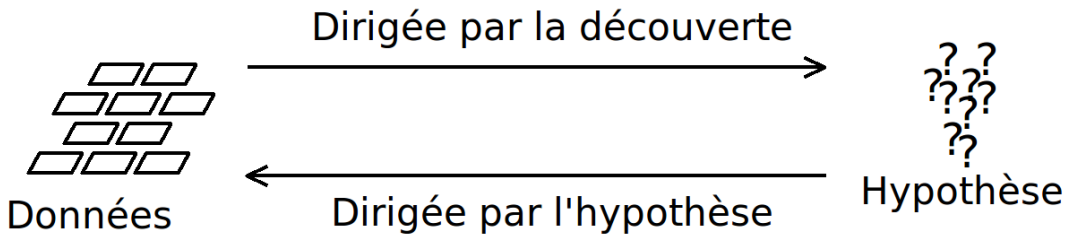

Figure 2-1 Deux approches d'analyse des données

Le data-mining est défini comme un "*processus d'exploration et d'analyse, par des moyens automatiques ou semi-automatiques, d'un large volume de données afin de découvrir des tendances ou des règles*" [Berry et Linoff (1997)]. Il s'agit d'appliquer des techniques d'apprentissage automatique intelligentes sur les données collectées par des systèmes informatiques pour extraire ou découvrir des informations signifiantes pour l'utilisateur.

Aujourd'hui, le Web est devenu une source d'information incontournable et les moteurs de recherche sont des outils indispensables pour la recherche d'informations sur le web. Trois des méthodes les plus couramment utilisées pour aider les utilisateurs en ligne à repérer les éléments pertinents sont les moteurs de recherche, les taxonomies et, plus récemment, les systèmes de recommandation [Parsons et al. (2004)].

Les interactions des utilisateurs avec des sites Web sont souvent appelées des transactions log [Jansen et Pooch (2001)]. L'analyse de ces données est un domaine de recherche qui évolue sans cesse. Cette analyse peut être classée sous le domaine de recherche de Web-mining [Etzioni (1996)] ce qui utilise le data-mining ainsi que d'autres techniques similaires pour extraire/découvrir des ressources, des motifs et des connaissances à partir du Web et des données liées au Web. Les travaux sur l'analyse de transactions log peuvent être regroupés en plusieurs catégories, par exemple (1) l'analyse des transactions comprenant des requêtes de recherche soumises à un moteur de recherche général comme Google ou à

un site Web/système spécifique comme un e-gouvernement, un e-bibliothèque ; (2) l'analyse des traces générées par un site Web/système spécifique comme un site d'e-commerce, etc.

Plusieurs travaux de recherche portent sur l'analyse de transactions log générées par des moteurs de recherche [Jansen et al. (1998), Silverstein et al. (1999), Jansen et Pooch (2001), Beitzel et al. (2004)]. Ces fichiers comprennent souvent un très grand nombre de requêtes de recherche. La plupart de ces travaux se focalisent sur l'analyse des requêtes des utilisateurs des moteurs de recherche pour produire des statistiques comme le nombre de sessions, le nombre de requêtes, le nombre de requêtes par session, le nombre de mots par requête, le pourcentage des requêtes utilisant des opérations booléennes, le nombre de pages de résultats consultées par chaque utilisateur, etc. Ces résultats permettent aux chercheurs de comprendre le comportement des utilisateurs. Ils aident également les chercheurs à comparer leurs résultats à travers les différents types de moteurs de recherche à différents moments, ainsi qu'à améliorer éventuellement la conception du moteur de recherche.

En ce qui concerne l'analyse des requêtes de recherche collectées par un site Web spécifique, la plupart des travaux de ce type se focalisent sur l'analyse des requêtes de recherche sur un seul site Web qui n'est pas un moteur de recherche. [Chau et al. (2005)] réalisent une analyse sur les requêtes collectées par le site Utah[4], un site du gouvernement de l'Utah. L'analyse de ces requêtes vise à produire des statistiques comme le nombre de requêtes uniques, le nombre de requêtes répétées, le nombre de requêtes vides, le nombre moyen de requêtes par session, le nombre de mots par requêtes, le nombre maximum de mots par requête, le pourcentage de requêtes avec un mot, avec deux mots, avec trois mots, etc. Ces résultats aident les développeurs à améliorer la performance des services disponibles sur le site Utah et aident les chercheurs à mener des recherches plus profondes dans ce domaine. [Jones et al. (1998), Jones et al. (2000)] prennent en compte l'analyse quantitative et qualitative de requêtes collectées pendant plus d'un an (de janvier 1998 à mai 1999) par une e-bibliothèque en Nouvelle-Zélande. Ils obtiennent quelques résultats comme le nombre de mots par requête et ils constatent qu'environ 80% des requêtes sont composées d'un, de deux ou trois mots. Leur étude indique également que la plupart des utilisateurs (73%) adoptent les paramètres par défaut du moteur de recherche sans aucune modification.

Plusieurs travaux portent sur la construction des systèmes de recommandation [O'Connor et Herlocker (1999), Herlocker et al. (2004), Parsons et al. (2004), Takács et al. (2009)] à partir de données collectées par un site Web/système comme un site e-commerce, une e-

[4] http://www.utah.gov/

bibliothèque. Ces travaux utilisent souvent des techniques de data-mining comme le *clustering* [O'Connor et Herlocker (1999)] sur un ensemble de données pour recommander les éléments pertinents (par exemple un livre, un film, un jeu, un produit, un restaurant, etc.) à l'utilisateur. Différentes approches peuvent être utilisées pour recommander [Parsons et al. (2004)], par exemple la recommandation des éléments sur la base du comportement passé de l'individu, la recommandation des éléments sur la base du comportement passé des utilisateurs similaires, la recommandation des objets similaires à ce que l'utilisateur est en cours de considérer, etc.

Nous pouvons constater que les travaux sur l'analyse des requêtes des moteurs de recherche sont souvent dirigés par l'hypothèse, ce qui nous intéresse dans notre hypothèse de recherche. Les chercheurs analysent des données pour répondre à leurs besoins statistiques. Dans notre contexte, ces statistiques sont des indicateurs, tandis que les autres travaux visant à l'analyse de données des systèmes comme l'e-commerce, utilisent souvent l'approche dirigée par la découverte. Certains travaux de ce type présentent la notion d'indicateur. [Parsons et al. (2004)] ont proposé une méthode *ad hoc* pour calculer l'indicateur ''*viewing time*''. Cet indicateur est utilisé pour déduire la préférence de l'utilisateur sur les attributs des articles et le système de recommandation se base sur cet indicateur.

Dans le domaine des EIAH, l'analyse de traces est également un axe de recherche très actif. Les données générées par les EIAH sont nombreuses et elles se composent de plusieurs informations signifiantes à exploiter [Mostow et Beck (2006)]. Depuis quelques années, plusieurs travaux portent sur l'analyse de traces générées par les EIAH pour répondre à différents objectifs. Ces travaux servent à aider l'enseignant ou le tuteur à comprendre les activités faites par l'étudiant pendant une session d'apprentissage. L'enseignant/tuteur peut donc observer, évaluer, réguler des activités des apprenants et notamment améliorer/adapter le scénario pédagogique qu'il a conçu. En fait, l'analyse des traces peut être réalisée pendant ou après la session d'apprentissage en utilisant plusieurs traitements (la collecte, le filtrage, la transformation, le calcul, la visualisation, etc.) pour obtenir les informations signifiantes et pertinentes.

La collecte des traces. Les traces collectées par les EIAH peuvent prendre différentes formes. La plupart des EIAH permettent actuellement de collecter automatiquement des traces brutes. Elles peuvent être enregistrées directement dans une base de données, dans les fichiers de log et/ou structurées selon différents formats ou modèles (Document Type Definition - DTD) d'un fichier XML, d'un fichier de log au standard Common Log Format (CLF). D'autres moyens comme l'enregistrement vidéo ou le questionnaire peuvent également être utilisés pour collecter ces données d'interactions.

L'organisation et la gestion des traces. Les traces collectées peuvent être réorganisées pour faciliter la gestion et l'exploration des traces. Plusieurs travaux de recherche portent sur ce thème. [Broisin et Vidal (2007)] ont proposé un modèle UML générique de traces qui permet de structurer et d'ajouter une sémantique claire aux données observées. [Choquet et Iksal (2007a)] ont proposé le langage UTL pour la structuration et l'analyse de traces. Le système Reading Tutor [LISTEN (2011)], Logic-ITA [Yacef (2005)] et le système DIAS [Bratitsis et Dimitracopoulou (2005)] collectent les traces, puis les restructurent dans une base de données relationnelle. Le système CourseVis [Mazza et Dimitrova (2003)] organise les données collectées sous la forme de fichiers XML.

La visualisation des traces. Les traces collectées sont souvent volumineuses et difficiles à exploiter et à comprendre pour l'utilisateur. La visualisation des traces peut être donc utilisée pour aider éventuellement l'utilisateur (l'enseignant/tuteur, l'apprenant, le chercheur) à comprendre le contenu des traces. [France et al. (2006)] utilisent des figures de Chernoff, des liens et des bulles pour permettre au tuteur de voir en temps réel les activités des apprenants, les parcours effectués et les activités réalisées ou en cours de réalisation au sein de l'EIAH, ainsi que d'adapter l'activité en interagissant directement sur l'interface de visualisation. Le travail de [Heraud et al. (2005)] est similaire, ils utilisent la barre d'ombre pour aider l'enseignant à comprendre l'activité des apprenants grâce à la notion de clarté. [May (2008)] utilise des boules avec différentes couleurs pour visualiser en temps réel la lecture d'un message par personne ainsi que le temps passé pour la lecture d'un message dans un forum de discussion. [Bratitsis et Dimitracopoulou (2005)] utilisent différents types d'histogrammes pour afficher le résultat des indicateurs calculés. [Mazza et Dimitrova (2003), Mazza et Milani (2005)] utilisent différentes techniques de visualisation pour aider l'enseignant/tuteur à obtenir une compréhension des aspects sociaux, cognitifs et comportementaux des apprenants dans une session d'apprentissage à distance.

Le calcul des indicateurs. La modélisation et le calcul d'indicateurs sont devenus un thème de recherche aujourd'hui dans les EIAH.

La littérature a proposé plusieurs définitions pour le terme ‘‘indicateur’’. Dans notre contexte de recherche, nous partageons entièrement la définition des projets ICALTS [Dimitracopoulou (2004b)] et DPULS. Le projet ICALTS se centre sur la spécification des besoins d'analyse de l'interaction complexe entre les acteurs d'un dispositif d'apprentissage collaboratif. Ce projet a introduit et défini le concept d'Indicateur d'Analyse de l'Interaction (Interaction Analysis Indicator) comme étant ‘‘*une variable généralement calculée ou établie à l'aide de données observées, témoignant du mode, du processus ou de la qualité de l'interaction*’’.

Le projet DPULS [Choquet (2005)] se focalise sur l'étude de l'analyse de l'usage d'un EIAH pour aider les enseignants/concepteurs à définir des situations pédagogiques adaptées aux usages observés. Sur la base de la définition d'un indicateur de l'interaction par le projet ICALTS, le projet DPULS a défini plus largement le concept d'indicateur pédagogique : "*c'est une variable signifiante sur le plan pédagogique, calculée ou établie à l'aide de données observées, et témoignant de la qualité de l'interaction, de l'activité et de l'apprentissage dans un EIAH*".

De par sa définition, un indicateur est une donnée calculée à partir des traces collectées par un EIAH. La modélisation et le calcul des indicateurs sont des processus complexes. Plusieurs approches pour l'élaboration d'indicateurs sont considérées. Certains travaux portent sur la transformation des traces pour produire des indicateurs [Settouti et al. (2006), Djouad et al. (2009)]. Certains travaux utilisent des techniques de data-mining [Merceron et Yacef (2003)], tandis que d'autres considèrent les langages de requête comme un outil pour calculer des indicateurs simples [Merceron et Yacef (2004a), Heiner et al. (2004)]. Le processus de calcul d'indicateurs peut également nécessiter la combinaison de ces approches [Romero et al. (2007)].

Dans les travaux actuels, différents types d'indicateurs sont considérés. Un indicateur peut être l'indicateur d'interaction [Dimitracopoulou (2004b)], l'indicateur de collaboration [Gendron (2010)], l'indicateur d'activité [Zorrilla et al. (2010)], etc. En ce qui concerne nos travaux, nous considérons la modélisation et le calcul des indicateurs pédagogiques, ce qui est défini par le projet DPULS.

Le calcul d'un indicateur fournit un ensemble de valeurs qui peuvent être quantitatives [Reffay et Lancieri (2006)] ou qualitatives. Dans les EIAH, ces valeurs sont notamment adressées à l'enseignant, au tuteur ou au concepteur pédagogique, mais aussi à l'apprenant. Elles sont utilisées pour différents objectifs comme l'évaluation, la régulation, l'adaptation, la réingénierie. Dans notre contexte d'expérimentation, ces valeurs sont utilisées par l'enseignant pour la réingénierie d'un scénario pédagogique et pour faire le tutorat. Nous considérons le calcul des deux types d'indicateurs quantitatifs et qualitatifs.

En fait, la modélisation du calcul d'indicateurs concernent plusieurs étapes : la collecte des traces nécessaires, le traitement des traces, la formalisation des méthodes de calcul et sa visualisation ou son utilisation. Dans toutes ces étapes, l'étape centrale est la modélisation et le calcul d'indicateurs. C'est l'étape à laquelle nous nous intéressons particulièrement dans notre contexte.

Dans le domaine des EIAH, plusieurs travaux portent sur la modélisation et le calcul d'indicateurs. Ces travaux peuvent en général être regroupés en deux types. Le premier concerne la proposition des indicateurs spécifiques dans un contexte particulier. Le deuxième concerne la proposition des méthodes génériques ou outils pour le calcul d'indicateurs. Le reste du chapitre présente les approches orientées vers le problème de la modélisation et du calcul d'indicateurs selon ces deux types de travaux.

2.2 Les langages pour l'analyse de traces et le calcul d'indicateurs

Un langage est très souvent conçu pour des objectifs spécifiques. Cette partie présente des langages qui peuvent être utilisés dans le calcul des indicateurs comme les langages de requête, les langages d'intelligence artificielle, les langages de transformation de modèles.

2.2.1 Les langages de requête

Les langages de requête (SQL, XQuery[5], SPARQL[6], etc.) sont très souvent utilisés pour effectuer des requêtes sur des bases de données. Les deux langages de requête les plus connus sont SQL (Structured Query Language) et Xquery. Ils permettent différents types d'opérations sur une base de données relationnelle (SQL) ou sur des documents XML (XQuery).

2.2.1.1 Les langages de requête SQL

SQL est un langage déclaratif. Il se compose de trois parties : le langage de définition de données, le langage de manipulation de données et le langage de contrôle de données. Le langage de définition de données (*Data Definition Language*) est utilisé pour la définition des éléments d'une base de données (tables, colonnes, clefs, index, contraintes, etc.). La manipulation des données est réalisée en utilisant le langage de manipulation de données (*Data Manipulation Language*). Ce langage permet de récupérer un ensemble de données parmi celles disponibles dans la base et de mettre à jour (insertion, modification et suppression) des données dans la base. Finalement, c'est le langage de contrôle de données (*Data Control Language*) qui s'occupe de gérer les droits d'accès aux données. Les requêtes SQL sous la forme de SELECT-FROM-WHERE sont utilisées pour l'interrogation sur des bases de données.

Par exemple, soit une trace de type ''*Sélection question*'' de la façon suivante :

```
<E K="SQ" T="1265805988386" H="13:46:28,386"><NQ>1</NQ></E>
```

[5] http://www.w3.org/TR/xquery/
[6] http://www.w3.org/TR/rdf-sparql-query/

Cette trace est extraite d'un fichier qui enregistre toutes les activités d'un apprenant, ici dénommé Clément, pendant la session. Elle concerne le choix d'une question à réaliser. Elle peut être transformée en une table SELECTION_QUESTION d'une base de données qui se compose de quatre colonnes de la façon suivante :

SELECTION_QUESTION	**Apprenant**	**T**	**H**	**NQ**
	Clément	1265805988386	13:46:28,386	1

Tableau 2-1 Exemple d'une table dans la base de données

La requête SQL dans le tableau 2-2 permet de calculer l'indicateur ‘‘le nombre de questions réalisé par apprenant ’’ :

```
SELECT Apprenant,count(DISTINCT NQ) as nombre-de-question
FROM  SELECTION_QUESTION
GROUP BY Apprenant
```

Tableau 2-2 Exemple d'une requête SQL

2.2.1.2 Les langages de requête XQuery

XML est de plus en plus utilisé aujourd'hui pour échanger des données entre des sources diverses et variées. Il était nécessaire d'avoir un langage simple pour pouvoir effectuer des requêtes sur ces documents. Pour lire des données XML et produire un résultat au format XML, XQuery est disponible. XQuery est compatible avec XSLT, le langage de transformation XML le plus connu et avec XPath, un langage pour extraire des nœuds dans un arbre XML. Ce sont des spécifications du W3C[7]. XQuery est un langage de requête permettant d'extraire des informations d'un document ou une collection de documents XML, de combiner les valeurs, les variables d'opérateurs et des fonctions pour générer un résultat. Il permet aussi d'effectuer des calculs complexes à partir des informations extraites pour produire de nouveaux documents XML. XQuery opère sur les données XML et joue un rôle similaire à celui du langage SQL conçu pour les données relationnelles. Une requête XQuery est formalisée à partir de cinq clauses : *for, let, where, order by* et *return* (ou FLWOR). Une requête peut contenir une ou plusieurs clauses *for* et *let*. La clause *let* facultative permet d'associer des valeurs à une ou plusieurs variables. La clause *where* optionnelle permet de filtrer les tuples produits par les clauses *for* et *let.* La clause *order by* facultative précise respectivement si le tri ainsi que le critère de tri (croissant ou décroissant) se font. La clause *return* construit le résultat comprenant des tuples satisfaisant le filtre, en respectant l'ordre dans lequel les tuples ont été produits.

[7] http://www.w3.org/

Par exemple, soit une trace ci-dessous :

```
1150291858979;Jenny;test 1
```

Cette trace spécifie le moment où l'apprenant Jenny commence à faire le test 1. Elle peut être transformée en donnée XML de la façon suivante :

```
<rawDatum type= "Exemple-Activite" >
      <date-de-debut>1150291858979</date-de-debut>
      <acteur>Jenny</acteur>
      <activite>test 1</activite>
</rawDatum>
```

Tableau 2-3 Exemple d'une donnée XML

Soit l'indicateur ''*le nombre d'activités par apprenant* '', la requête XQuery ci-dessous (le tableau 2-4) permet de calculer cet indicateur :

```
for $i in distinct-values (//rawDatum/acteur)
let $v:=//rawDatum[@type="Exemple-Activite" and acteur=$i]
return <apprenant nom= "{data($i)}"
                        nombre= "{data(count($v/activite))}"/>
```

Tableau 2-4 Exemple d'une requête XQuery

2.2.2 Les langages d'intelligence artificielle

Si les traces et les indicateurs sont considérés comme les objets, et que la méthode de calcul d'indicateurs est considérée comme un ensemble de règles de calcul, les langages d'intelligence artificielle peuvent être un choix pertinent. CLIPS (C Language Integrated Production System) est un langage déclaratif et logique conçu spécifiquement pour les domaines de l'intelligence artificielle. Il est utilisé couramment pour la construction des systèmes experts et en particulier pour la gestion des systèmes de production. CLIPS est rapide et efficace. Ce langage est aussi prévu pour de la déduction logique. L'utilisation de CLIPS demande la définition de faits et de règles sur ces faits. En ce qui concerne le calcul des indicateurs à partir des traces, les traces et les indicateurs doivent être représentés comme des faits et la méthode de calcul doit être représentée comme un ensemble de règles de calcul.

Par exemple, nous prenons l'exemple ''*Sélection question*'' dans la section 2.2.1.1 :

```
<E K="SQ" T="1265805988386" H="13:46:28,386"><NQ>1</NQ></E>
```

Cette trace peut être définie et représentée sous CLIPS de la façon suivante :

```
(deftemplate sq
       (slot apprenant  (type STRING))
       (slot t  (type STRING))
       (slot h  (type STRING))
       (slot nq (type INTEGER)))

(assert (sq (apprenant "Clément")(t "1265805988386")
                                    (h "13:46:28,386") (nq 1)))
```

Tableau 2-5 Exemple d'un fait en CLIPS

Le code en CLIPS ci-dessous (le tableau 2-6) permet de calculer l'indicateur "*le nombre de questions réalisé par apprenant*". Nous présentons le code pour le cas simple, c'est-à-dire le nombre de questions peut se composer de questions identiques. Cette méthode est plus complexe que celle décrite en SQL ou XQuery.

```
;déclarer une liste d'apprenant
(deftemplate liste-apprenant
      (slot nom (type STRING))
)
;définir le format de l'indicateur
(deftemplate indicateur
      (slot nom (type STRING))
      (slot nombre (type INTEGER))
)
;définir une fonction qui calcule le nombre de questions
(deffunction count-question()
   ; récupérer l'apprenant à partir des faits sq
   ; et l'ajouter  dans la liste liste-apprenant
   (do-for-all-facts((?fct sq))
       TRUE
       (assert (liste-apprenant (nom ?fct:apprenant)))
   )
   ;calculer le nombre de questions
   (do-for-all-facts((?fc liste-apprenant))
     TRUE
     (bind ?count (length(find-all-facts((?f sq))
                              (eq ?f:apprenant ?fc:nom))))
      ; le résultat
     (assert (indicateur (nom ?fc:nom) (nombre ?count)))
   )
)
```

Tableau 2-6 Exemple d'une méthode de calcul d'un indicateur en CLIPS

2.2.3 Les langages de transformation de modèles

Le calcul d'un indicateur peut être considéré comme une transformation de modèles [Settouti et al. (2006)], à partir des modèles des traces jusqu'au modèle de l'indicateur. En théorie, cela peut être réalisé en utilisant l'approche de transformation de modèles [Miller et Mukerji

(2003)]. Selon l'architecture MDA (Model Driven Architecture) [Miller et Mukerji (2003)], la transformation de modèles peut être un processus qui prend en entrée un ensemble de modèles sources et fournit en sortie des modèles cibles en utilisant un ensemble de règles de transformation. Pour la transformation, chaque modèle doit être décrit par un méta-modèle. La transformation de modèles se fait par (1) la spécification des règles de transformation exprimant la correspondance entre les concepts du méta-modèle source et les concepts du méta-modèle cible; (2) puis par l'application de ces règles au modèle source pour produire le modèle cible. ATL (ATLAS Transformation Language) est un langage de transformation développé dans le cadre du projet ATLAS au LINA[8] à Nantes et fait partie du projet Eclipse M2M[9]. C'est le langage permettant de réaliser des transformations de modèles dans le cadre de l'architecture MDA proposé par l'OMG[10]. ATL permet de définir des modules et des requêtes. Un module est une transformation d'un modèle vers un autre modèle tandis qu'une requête est une transformation d'un modèle vers un type primitif, des éléments d'un modèle, des collections, etc.

2.2.4 Discussion

Plusieurs travaux de recherche concernant l'analyse de traces ont utilisé les langages de requête sur les traces pour extraire un jeu de données dans la base ou pour calculer des indicateurs simples [Heiner et al. (2004), Merceron et Yacef (2004a), Bratitsis et Dimitracopoulou (2005)]. [Romero et al. (2007)] utilisent SQL pour extraire des données avant d'appliquer les techniques de data-mining. Dans le travail de [Broisin et Vidal (2007)], le langage de requête CQL (CIM Query Language) peut être utilisé pour l'exploitation des traces. Dans le système Tatiana [Dyke et al. (2010)], les scripts écrits en XQuery sont employés pour transformer un fichier d'un corpus de traces en *rejouables* (une trace particulière qui peut être transformée, visualisée, enrichie et synchronisée). Les requêtes de base de SQL ou Xquery sont souvent suffisantes pour calculer des indicateurs simples comme *le nombre de bonnes questions réalisées par étudiant*. Par contre, les langages de requête ne sont pas utilisés pour calculer les indicateurs complexes qui nécessitent plusieurs traitements comme *l'indicateur de typologie de navigation* [Bousbia et al. (2009b)], *l'indicateur associé aux actions collaboratives* [Dimitracopoulou (2004b), Avouris et al. (2003)] (Collaborative Action function indicator), etc. Les langages de requête ne permettent pas de décrire des algorithmes complexes. Ils sont conçus particulièrement pour l'interrogation des données sur des bases de données. Ces langages ne fournissent pas une solution permettant de capitaliser le savoir-faire dans la formalisation des requêtes sur les

[8] http://www.sciences.univ-nantes.fr/lina/atl/
[9] http://www.eclipse.org/m2m/
[10] http://www.omg.org/

traces pour calculer des indicateurs. Ils ne disposent pas d'une possibilité d'intégrer une fonction ou une méthode écrite dans d'autres langages provenant d'autres outils d'analyse.

CLIPS est rapide et efficace pour les traitements sur des faits et des règles. Les traces et les indicateurs doivent donc être représentés sous la forme de faits et les méthodes de calculs d'indicateurs sont les règles. L'un des points forts de CLIPS est de disposer d'une possibilité d'intégrer des fonctions externes écrit en C, Ada ou Fortran. Les langages d'intelligence artificielle comme CLIPS, Prolog (un langage de programmation logique qui est utilisé dans plusieurs systèmes d'intelligence artificielle et dans le traitement des langages naturels) sont peu utilisés pour calculer des indicateurs et manipuler sur les données, car ils sont conçus spécifiquement pour les déductions logiques, pas pour la combinaison et l'extraction des données. L'utilisation de ces langages pour interroger les données est complexe (cf. l'exemple dans le tableau 2-6), car nous devons indiquer la méthode pour acquérir des données. Ils ne fournissent pas des caractéristiques/fonctionnalités permettant de réaliser cette tâche comme les langages SQL et XQuery.

En ce qui concerne le calcul d'indicateurs, les langages de transformation de modèles comme ATL peuvent théoriquement être pertinents. En fait, ils sont souvent utilisés dans le domaine du Génie Logiciel. Ces langages de transformation de modèles sont très peu ou pas encore utilisés pour l'analyse de traces, et par conséquent pour le calcul d'indicateurs. L'utilisation d'ATL nécessite la définition des méta-modèles de traces et d'indicateurs. Ces méta-modèles doivent être validés avant leur utilisation dans les règles de transformation. De plus, les transformations sont souvent lourdes. Si le calcul d'un indicateur correspond donc à une transformation, le calcul d'un ensemble d'indicateurs prendra du temps. Les langages de transformation de modèles ne sont donc pas une bonne solution pour produire des indicateurs en temps réel.

2.3 Ingénierie des indicateurs dans les EIAH

2.3.1 Introduction

Un indicateur est souvent utilisé pour assister l'enseignant/tuteur dans l'observation des activités des apprenants dans les EIAH. Plusieurs travaux de recherche ont été proposés dans la gestion des traces et l'élaboration des indicateurs. Certains visent à définir des indicateurs et des outils d'analyse spécifiques pour résoudre des problèmes dans un contexte particulier, tandis que d'autres recherches portent sur la définition des méthodes génériques pour la gestion, pour la modélisation et pour l'analyse des traces. Différentes approches d'analyse des traces sont utilisées dans ces travaux comme par exemple l'approche data-mining, l'approche par patron de conception, l'approche par les modèles,

etc. Cette partie est consacrée à la présentation de différents travaux concernant la modélisation, la gestion des traces ainsi que le calcul d'indicateurs.

Dans un premier temps, nous présentons les travaux portant sur l'approche data-mining. Puis, les travaux concernant la proposition des indicateurs spécifiques sont abordés. La troisième partie est consacrée à la présentation des travaux sur la modélisation et le calcul des indicateurs. La dernière partie aborde le problème de la capitalisation et de la réutilisation du savoir-faire en analyse de traces.

2.3.2 Les travaux sur l'approche data-mining

Ces dernières années, les techniques de data-mining ont été utilisées dans plusieurs domaines comme le e-commerce [Srivastava et al. (2000)], la bioinformatique [Srivastava et al. (1998)], la médecine [Richards et al. (2001)], etc. L'utilisation des techniques de data-mining dans l'exploration de données éducatives est également devenue populaire, en témoigne la conférence EDM[11] (Educational Data Mining) qui est organisée pour discuter des travaux dans cette discipline. Plusieurs travaux sont publiés autour de l'utilisation du data-mining pour découvrir des informations utiles assistant les chercheurs, les apprenants, les autres acteurs concernés dans les EIAH et notamment les enseignants/tuteurs dans le processus d'enseignement. L'application de techniques de data-mining sur les traces générées par des EIAH permet (1) d'identifier des motifs utiles concernant le comportement des étudiants [Talavera et Gaudioso (2004)] ; (2) de savoir de quelle façon les étudiants apprennent dans les EIAH [Zaïane et Luo (2001)] ; (3) de découvrir les associations entre les besoins des étudiants et une liste de matériels d'apprentissage [Romero et al. (2004)] ; (4) de connaître la cause des problèmes dans le système d'apprentissage [Nilakant et Mitrovic (2005)] ; (5) pour proposer des expériences et des activités d'apprentissage personnalisables aux étudiants [Tang et McCalla (2005)] ; (6) de classifier les étudiants en se basant sur les caractéristiques extraites à partir des données enregistrées afin de prévoir leurs notes finales [Minaei-Bidgoli et Punch (2003)] ; (7) de décrire le processus d'apprentissage de l'étudiant avec l'objectif de classifier les étudiants pour leur donner différents conseils en se basant sur leurs compétences et sur d'autres caractéristiques [Hamalainen et al. (2004)] ; etc.

Un processus de data-mining sur les données collectées par des EIAH est un cycle itératif qui comprend les quatre mêmes étapes comme dans le processus de data-mining général [Romero et Ventura (2007)] :

[11] http://educationaldatamining.org/

- La collecte des données : les données d'interaction en session entre l'apprenant et un EIAH sont stockées par exemple, dans une base de données [Romero et al. (2007), Merceron et Yacef (2003)].
- Le prétraitement des données : les données stockées sont nettoyées et transformées en un format approprié pour être exploitées.
- L'application de techniques de data-mining : les algorithmes de data-mining sont appliqués pour extraire/découvrir des informations signifiantes pour l'utilisateur (l'enseignant, l'apprenant, etc.). Pour cela, des outils de data-mining spécifiques ou généraux peuvent être utilisés.
- L'interprétation, l'évaluation et le déploiement des résultats : les résultats obtenus sont interprétés et utilisés par l'utilisateur. Par exemple, des informations découvertes sont utilisées pour prendre des décisions sur des activités de l'apprenant afin d'améliorer l'apprentissage de l'apprenant [Romero et al. (2007)].

[Romero et al. (2007)] appliquent ce processus sur les traces générées par la plate-forme Moodle[12]. Ces auteurs utilisent des outils de data-mining disponibles comme (1) GISMO [Mazza et Milani (2005)] pour visualiser le nombre d'accès total par les étudiants à toutes les ressources d'un cours ; (2) l'outil Weka[13] [Witten et Frank (2005)] pour regrouper les apprenants d'un cours spécifique dans différents groupes en fonction des activités réalisées; (3) l'outil Keel[14] [Alcalá-Fdez et al. (2008)] pour classer les apprenants dans différents groupes avec la même note finale en fonction des activités réalisées ; etc.

Le data-mining est également utilisé pour le calcul d'indicateur. TADA-Ed [Merceron et Yacef (2004b), Merceron et Yacef (2005)] est un outil qui permet d'exploiter les traces générées par le dispositif d'apprentissage Logic-ITA [Merceron et Yacef (2003), Merceron et Yacef (2004b), Yacef (2005), Merceron (2009)] et pour calculer certains indicateurs décrits ci-dessous.

Logic-ITA est un outil tutoriel basé sur le Web utilisé à l'Université de Sydney depuis 2001. Son objectif est d'aider les étudiants à pratiquer des exercices de preuve formelle en logique des propositions et d'informer l'enseignant sur le progrès des étudiants, leurs problèmes, etc. La partie "*Logic Tutor*" de cet outil propose des exercices à des étudiants. Chacun se compose d'un ensemble de formules propositionnelles : les prémisses plus une autre formule et la conclusion. Les étudiants décrivent la conclusion à partir de prémisses. Chaque dérivation comprend en général plusieurs étapes. Les traces générées par chaque étudiant

[12] http://moodle.org
[13] http://www.cs.waikato.ac.nz/ml/weka/
[14] http://www.keel.es/

sont stockées dans un fichier individuel et récupérées lorsque l'étudiant se connecte à nouveau. Ces données concernent toutes les étapes, y compris le niveau de l'exercice, la réponse à chaque étape de chaque exercice comprenant les messages d'erreurs et les règles de logique utilisées, l'heure, etc. De cette façon, le système connait l'historique de l'étudiant, les erreurs qu'il a faites, et est capable de choisir des exercices adaptés au besoin de l'étudiant.

En ce qui concerne l'exploration des traces, ces fichiers sont ensuite récupérés, les traces brutes sont structurées et stockées dans une base de données. L'analyse de ces dernières peut être réalisée par deux moyens :

- Des requêtes SQL simples sur la base permettent d'établir certains indicateurs, notamment les messages d'erreurs les plus fréquents, les règles de logique avec lesquelles les étudiants commettent le plus d'erreurs, les exercices avec le plus d'erreurs, les exercices non réussis, etc.
- L'outil appelé TADA-Ed (Tool for Advanced Data Analysis in Education) a été développé pour aider à trouver des motifs intéressants dans un ensemble de données. Il s'agit d'une plate-forme d'exploration de données dédiée aux enseignants, leur permettant de visualiser et d'explorer des données concernant les exercices des apprenants. Il permet d'appliquer les techniques de data-mining comme les règles d'association, la segmentation (en anglais *classification*), et le regroupement ou classification (en anglais *clustering*) sur les traces collectées par Logic-ITA afin de découvrir des motifs pertinents. Les données stockées dans la base de données ont besoin d'une transformation avant pour être utilisées avec ces techniques afin de pouvoir produire des résultats signifiants pour l'enseignant. A travers cet outil, certains indicateurs peuvent être calculés comme les erreurs souvent associées (en utilisant les règles d'association), la prédiction des notes selon des erreurs (en utilisant la segmentation), etc. [Merceron et Yacef (2005)]. Ces indicateurs sont utilisés surtout par l'enseignant/concepteur pour améliorer l'enseignement.

Dans ce travail, ces indicateurs aident en général l'enseignant à surtout améliorer son cours. Ils permettent à l'enseignant de se concentrer sur les erreurs commises le plus souvent et sur les règles les plus utilisées de manière incorrecte afin de (1) modifier une ou plusieurs parties du cours et (2) d'adapter son enseignement aux différents niveaux des apprenants. Ces indicateurs sont également un moyen pour améliorer le système d'apprentissage. TADA-Ed permet aux enseignants/tuteurs de choisir et de paramétrer les indicateurs disponibles à exécuter. Ils peuvent également exprimer leurs indicateurs simples par des

requêtes SQL, ce qui est similaire au projet LISTEN[15] [Mostow et al. (2005)]. Mais Logic-ITA et TADA-ED ne disposent pas d'un outil d'interrogation. L'enseignant/tuteur peut utiliser le système de gestion de la base (ici Access[16]) pour définir des indicateurs. TADA-Ed utilise les techniques de data-mining pour calculer des indicateurs. La sélection d'un indicateur à exécuter et ses paramètres, ainsi que l'interprétation de ses résultats sont donc difficiles pour l'enseignant non-informaticien. La capitalisation des méthodes de calcul d'indicateurs et du savoir-faire en observation et en analyse d'une session n'est pas encore considérée dans ce travail.

Nous observons que les techniques de data-mining dans le domaine éducatif sont souvent complexes. Leurs implémentations et utilisations pour un EIAH spécifique prennent du temps [Iksal et al. (2010)]. Ces techniques sont souvent utilisées pour détecter, découvrir des motifs intéressants ou pour prévoir par exemple, le niveau de connaissance des apprenants à partir des données collectées. Selon [Romero et Ventura (2007), Romero et al. (2007)], la plupart des outils de data-mining courants sont trop complexes. Leurs utilisations et l'interprétation de leurs résultats sont souvent difficiles pour les enseignants/concepteurs. Les techniques de data-mining dépassent ce que les enseignants/concepteurs peuvent généralement appréhender. En conséquence, les chercheurs ou les analystes des EIAH interprètent des résultats ou produisent des rapports utilisés ensuite par les enseignants/concepteurs pour prendre des décisions de réingénierie, d'évaluation, de régulation ou de réflexivité.

Comme nous l'avons abordé précédemment, notre travail repose sur le point de vue suivant : l'enseignant/concepteur exprime ses besoins d'observation et interprète les résultats des indicateurs. L'analyse de traces a non seulement pour objectif la découverte ou la prédiction, mais aussi le fait de répondre aux besoins attendus de l'enseignant. Nous prenons donc en compte l'approche d'analyse des données dirigée par l'hypothèse dans notre travail.

2.3.3 Les travaux sur la définition des indicateurs spécifiques

Cette partie présente des travaux qui portent sur la définition des indicateurs dans un contexte particulier (un forum, un chat, un EIAH, etc.). Les indicateurs présentés dans cette partie ne sont pas définis par l'enseignant. Ils sont proposés à l'enseignant pour faciliter la détection des styles d'apprentissage, l'assistance directe aux utilisateurs dans un forum, l'évaluation de l'activité de l'apprenant, etc.

Des indicateurs pour la détection des styles d'apprentissage

[15] http://www.cs.cmu.edu/~listen/
[16] C'est un logiciel de Microsoft

Les travaux de [Bousbia et al. (2009b), Bousbia et al. (2009a), Bousbia et Labat (2007)] se situent dans le contexte d'environnements d'apprentissage basés sur le Web. Ces auteurs s'intéressent à la détection des styles d'apprentissage par l'analyse de comportements de l'apprenant. Le processus de détection des styles d'apprentissage se compose de trois étapes [Bousbia et al. (2009a)] : (1) le *choix d'indicateur* permet à l'enseignant de choisir l'indicateur à partir d'un ensemble d'indicateurs proposé ; (2) l'*observation* vise à collecter des données sur l'interaction des différents acteurs d'une situation d'apprentissage ; (3) l'*analyse et l'interprétation* consistent à calculer l'indicateur choisi, puis à détecter les styles d'apprentissage et à adresser les résultats à l'enseignant.

Les indicateurs dans ces travaux sont classifiés en trois niveaux : bas, intermédiaire et haut, ce qui est proposé dans le projet ICALTS [Dimitracopoulou (2004b)]. Cette classification a pour objectif de faciliter l'utilisation et le calcul des indicateurs. En fait, les seuls indicateurs de haut niveau sont proposés à l'enseignant. Les indicateurs intermédiaires et bas sont nécessaires pour le calcul d'indicateur de haut niveau. Ces indicateurs sont calculés selon trois critères : page, contenu et session. L'*indicateur de typologie de navigation* [Bousbia et al. (2009b)] est un indicateur de haut niveau utilisé pour la détection des styles d'apprentissage. Cet indicateur est calculé à partir d'un ensemble d'indicateurs intermédiaires (les durées de consultation des pages du cours, le type de consultation, la forme de parcours réalisé, le lien sémantique entre les pages consultées). En plus, des indicateurs comme *le temps passé sur une page de contenu*, *la durée totale de consultation du cours*, *le pourcentage d'activités réalisées*, *l'intérêt de la page*, etc. visent à aider l'enseignant à percevoir et interpréter le comportement de l'apprenant durant les sessions d'apprentissage [Bousbia et Labat (2007)].

Des indicateurs concernant la communication médiatisée

Dans un environnement à distance, les outils de communication comme le chat et le forum sont devenus des outils importants. Ces derniers permettent aux apprenants et aux tuteurs d'échanger leurs problèmes ou idées concernant une situation d'apprentissage. Les activités de communication entre les différents acteurs (l'apprenant, le tuteur) dans une session d'apprentissage à distance sont les données nécessaires à la compréhension du comportement des apprenants. L'exploitation de ces activités devient un champ de recherche dans le domaine des EIAH.

Dans le cadre de sa thèse, [May (2008)] propose une approche permettant de collecter et d'exploiter des traces de communication d'un forum pour fournir du soutien aux apprenants, aux tuteurs ainsi qu'aux enseignants et chercheurs pendant et après leurs activités. Sa

proposition porte sur deux points : (1) le traçage des activités de communication et (2) la proposition d'un outil permettant aux tuteurs et aux apprenants d'analyser et de visualiser en temps réel les traces obtenues. Ces dernières sont divisées en trois types d'interactions : les interactions homme-machine, les interactions humain-humain par l'utilisation de machines et les interactions machine-machine lors de la communication entre deux utilisateurs. Ces interactions sont collectées par : le serveur et la machine de l'utilisateur. Ces traces sont représentées sous un format général de traces [May (2009)]. Ce format est exprimé sous la forme d'un schéma XML et se compose des éléments principaux :

- <Tool> : l'outil de communication,
- <Activity> : l'activité d'interaction entre un utilisateur et l'outil de communication,
- <User> : l'utilisateur réalisant une activité,
- <HCI> : la description des actions et les interactions d'un utilisateur pendant la réalisation d'une activité,
- <Time> et <Date> : l'heure et la date d'une activité,
- <Comment> : le commentaire.

La figure 2-2 présente un exemple d'une trace de communications conforme au format proposé.

```
<TraceHCI id="3">
      <Tool>Forum</Tool>
      <Activity>Send a new message</Activity>
      <User>Mathed</User>
      <HCI id="1">
            <Title>Write a message content</Title>
            <Object>Input</Object>
            <TimeHCI>12:09:10 PM</TimeHCI>
            <Atributte/>
            <HCIContent>
               <Content>Hello there,how are you doing?</Content>
               <ContentAttribute>IDMsg=7,IDForum=4</ContentAttribute>
            </HCIContent>
      </HCI>
      <Time>12:15:30 PM</Time>
      <Date>10/03/2007</Date>
      <Comment/>
</TraceHCI>
```

Figure 2-2 Un extrait d'une trace de communications [May (2009)]

Dans son travail, [May (2008)] a proposé le calcul de certains indicateurs à partir des traces décrites selon ce schéma. Ces indicateurs sont *"lecture d'un message"*, "*le profil d'un utilisateur"*, l'indicateur concernant deux activités *"Poster un nouveau message et Répondre à un message"*, etc. Ils ont été calculés et représentés sous une forme graphique en temps réel. La figure 2-3 présente le résultat de l'indicateur "*lecture d'un message*" dans un forum.

Comme le montre la figure 2-3, les sphères (chacune correspond à la visualisation d'un message par un utilisateur) avec quatre couleurs (chacune correspond à un type d'action faite par l'utilisateur) sont utilisées pour visualiser en temps réel la lecture d'un message par personne ainsi que le temps passé (le diamètre de la sphère) pour la lecture d'un message dans un forum de discussion. La distance entre les sphères correspond au temps écoulé entre deux lectures. Cette visualisation permet une interprétation en détail des différentes actions de lecture faites par les apprenants.

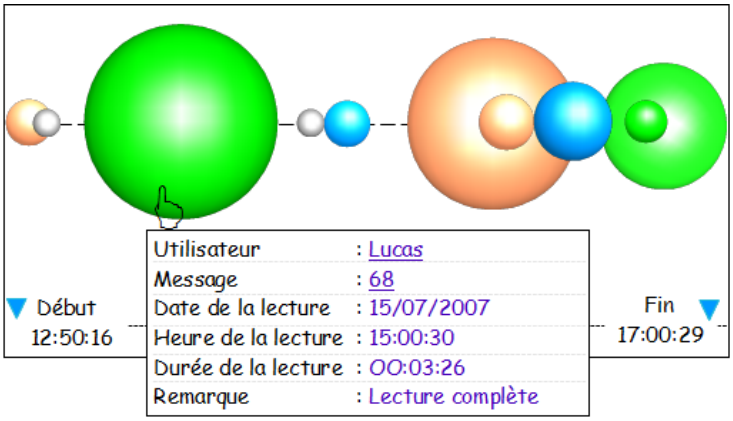

Figure 2-3 La visualisation de l'indicateur ''*lecture d'un message*'' [May (2008)]

Le système DIAS [Bratitsis et Dimitracopoulou (2005)] est aussi un travail concernant les activités de communication. DIAS (Discussion Interaction Analysis System) a été développé par le laboratoire LTEE (Learning Technology and Educational Engineering), de l'Université d'Aegean. L'outil DIAS est personnalisable pour l'analyse de l'interaction dans les forums de discussion. Il permet d'enregistrer les interactions des utilisateurs dans une base de données relationnelle et fournit plusieurs fonctionnalités afin de faciliter la participation des utilisateurs (apprenants, tuteurs, enseignants).

Pour l'analyse des données collectées par le système, cet outil a proposé un ensemble d'environ quatre-vingts indicateurs [Bratitsis et Dimitracopoulou (2008)] (y compris toutes les variations possibles de ces indicateurs) : des indicateurs statistiques simples aux indicateurs cognitifs et métacognitifs complexes. Ces indicateurs sont divisés en trois catégories : les indicateurs individuels, les indicateurs de groupe et les indicateurs généraux. Certains indicateurs sont pertinents pour les enseignants, tandis que d'autres sont prévus pour les apprenants participant au forum. Les indicateurs proposés peuvent également être utilisés par les chercheurs. L'objectif principal du projet est d'offrir une assistance directe aux utilisateurs (notamment les apprenants travaillant en collaboration) afin d'activer leurs processus métacognitifs, leur permettant d'autoréguler leurs activités, et aussi offrir aux enseignants la possibilité d'identifier des situations et des difficultés qui nécessitent des

interventions régulatrices. Tous les indicateurs sont affichés graphiquement ou sous forme textuelle.

La figure 2-4 est un exemple de visualisation d'indicateurs. Cette figure représente l'*indicateur d'activité.* Chaque bulle colorée par une couleur correspond à un apprenant. La taille de la bulle représente le nombre de types de messages utilisés. L'axe X montre le nombre de messages postés. L'axe Y correspond au nombre de messages lus.

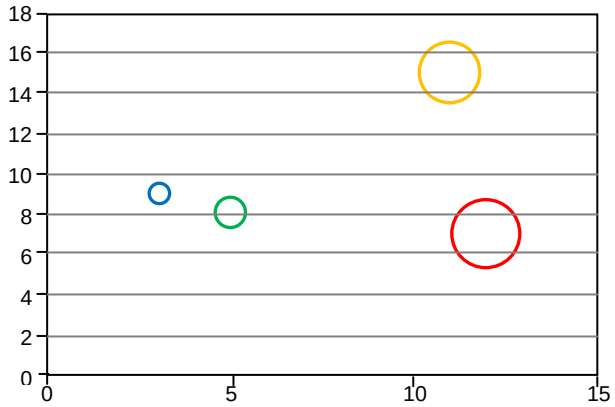

Figure 2-4 La visualisation de l'indicateur d'activité par l'outil DIAS [Bratitsis et Dimitracopoulou (2005)]

Les travaux de [Pendergast (2006)] et [Chan et al. (2004)] concernent également les activités de communication. [Pendergast (2006)] propose un outil qui permet aux enseignants/tuteurs d'évaluer l'activité des apprenants dans l'utilisation des forums. Cet outil fournit certains indicateurs comme le nombre de messages envoyés, le nombre de messages reçus, la longueur des messages, etc. Pour [Chan et al. (2004)], ils définissent un indice de la participation des étudiants en utilisant cinq paramètres correspondant à cinq actions d'un étudiant : le nombre de pages vues, le nombre de questions lues dans le forum, le nombre de questions envoyées dans le forum, le nombre de sessions de chat et le nombre de messages de chat soumis.

Des indicateurs d'activité

Dans les environnements en face à face, la présence des apprenants en classe et leur contribution aux cours peuvent être considérées comme des activités utilisées pour évaluer la participation des apprenants. Pour les cours en ligne, cette évaluation ne peut être que réalisée en se basant sur les activités des apprenants [Zorrilla et al. (2010)] concernant l'accès au cours, l'écriture et la lecture des messages dans les forums de discussion, la réalisation des quizz en ligne, l'écriture dans un wiki, etc. Ces données ne sont pas souvent

interprétables par l'enseignant/tuteur. Plusieurs travaux s'intéressent donc au calcul des indicateurs à partir de ces activités pour faciliter cette évaluation.

[Juan et al. (2009)] proposent un système pour surveiller en ligne la performance et l'activité des apprenants en utilisant des données à partir de fichiers de log et une base de données. Ce système fournit trois indicateurs qui sont calculés en fonction du nombre d'événements (les messages envoyés ou lus dans les forums, les e-mails envoyés ou lus, les tests réalisés en ligne, les documents téléchargés ou chargés, etc.). Le premier est l'indicateur de classification des étudiants. Cet indicateur est défini comme le nombre d'événements par étudiant au cours de la semaine par rapport au nombre moyen d'événements par étudiant par semaine. Le deuxième est l'indicateur de suivi d'étudiant individuel qui surveille le niveau d'activité de chaque étudiant tout au long du cours. Le dernier est l'indicateur de niveau de participation surveillant le pourcentage d'étudiants qui ont fini les tests.

[Zorrilla et al. (2010)] proposent et calculent des indicateurs d'activité. Ces indicateurs sont fondés sur un paramètre qui regroupe les critères de l'enseignant/tuteur afin de mesurer l'activité de son cours (temps passé, résultat ou une combinaison des deux). Ils sont obtenus pour chaque apprenant dans chaque ressource (pages de contenu, forums, wiki, etc.) par rapport à l'activité réalisée par ses camarades de classe/groupe. Ces indicateurs seront affichés périodiquement à la fois pour les apprenants et les tuteurs, pour que chaque apprenant puisse observer l'effort qu'il/elle a fait par rapport au reste du groupe, et pour que le tuteur puisse évaluer la qualité de l'activité et la participation de chaque apprenant dans le cours. La figure 2-5 présente *l'indicateur dans les pages de contenu pour les trois étudiants*. Dans cette figure, l'activité réalisée par les trois apprenants dans les pages de contenu montre qu'ils se comportent différemment.

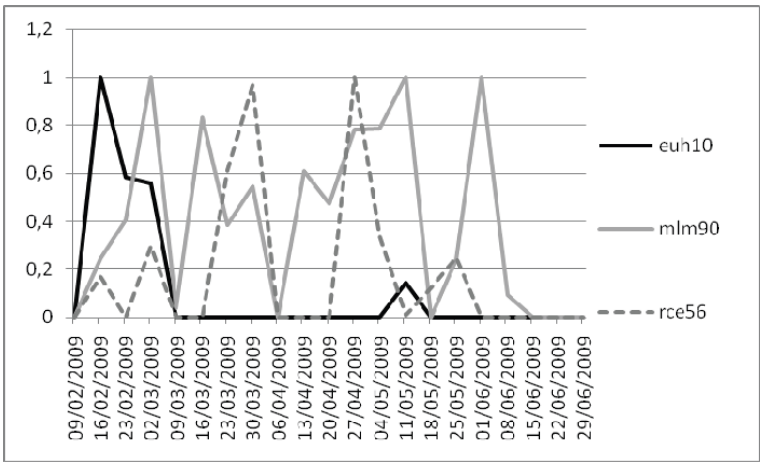

Figure 2-5 La visualisation de l'indicateur dans les pages de contenu pour les trois étudiants

Discussion

Les travaux présentés dans cette partie portent sur la définition des méthodes et sur le développement d'outils pour l'élaboration d'indicateurs spécifiques dans un contexte particulier. Les travaux de [May (2008)] et [Bratitsis et Dimitracopoulou (2005)] proposent des indicateurs sur l'utilisation d'un outil particulier, il s'agit d'un forum. [Bousbia et al. (2009b)] considèrent le calcul d'indicateurs spécifiques dans un contexte particulier. C'est le cas d'un cours basé sur le Web. Ce cours doit être organisé sous une forme arborescente comprenant plusieurs activités qui peuvent se décomposer en sous activités comme la consultation d'un contenu hypermédia, la réalisation d'un travail, etc. Les travaux de [Zorrilla et al. (2010), Juan et al. (2009)] prennent en compte le calcul d'indicateurs spécifiques pour répondre aux besoins d'observation particuliers : l'évaluation de la participation des apprenants aux cours.

Dans les travaux ci-dessus, les indicateurs sont disponibles pour l'enseignant/tuteur [Zorrilla et al. (2010), May (2008), Bousbia et al. (2009a), Juan et al. (2009), Bratitsis et Dimitracopoulou (2005)] ou l'apprenant [Zorrilla et al. (2010), May (2008), Bratitsis et Dimitracopoulou (2005)]. Tous ces indicateurs sont prédéfinis. L'utilisateur (l'enseignant/tuteur ou l'apprenant) peut choisir et paramétrer les indicateurs disponibles. Les travaux de [Zorrilla et al. (2010), Bousbia et al. (2009a), Bratitsis et Dimitracopoulou (2005)] permettent à l'enseignant/tuteur de décider quels paramètres et quelles ressources il veut utiliser pour mesurer l'activité, la fréquence avec laquelle les indicateurs sont générés. Dans un contexte particulier, les indicateurs proposés sont pertinents. Par contre, les méthodes proposées et les outils développés par ces travaux ne permettent pas à l'enseignant/tuteur d'exprimer lui-même ses indicateurs.

Nous observons que la plupart des indicateurs présentés ci-dessus, à l'exception de ceux proposés par Bousbia, concernent les activités d'interaction réalisées via le forum, le chat, le wiki, etc. Ces indicateurs sont représentés graphiquement, cela facilite donc l'interprétation par l'enseignant/tuteur. Certains indicateurs simples comme le nombre de messages envoyés, le nombre de messages lus, etc. sont pris en compte dans la plupart des démarches [Pendergast (2006), Chan et al. (2004), May (2008), Juan et al. (2009)] portant sur l'analyse d'activités de communication. Nous observons que nombre de ces indicateurs ont été étudiés dans le travail de [Reffay et Lancieri (2006)] portant sur l'analyse quantitative des forums de discussion et ayant pour objectif d'identifier et de qualifier les usages dans les groupes d'utilisateurs. Les travaux de [Juan et al. (2009)] et [Chan et al. (2004)] utilisent ces indicateurs pour élaborer d'autres indicateurs. [May (2008)] s'intéresse au calcul du temps

passé pour la lecture d'un message, ce qui est similaire au travail de [Bousbia et Labat (2007)] considérant le calcul du temps passé sur une page du contenu.

Avec le même indicateur, par exemple le nombre de messages envoyés, chaque approche propose une méthode propre pour le calculer. Cette méthode ne peut pas être réutilisée pour calculer le même indicateur dans d'autres travaux. En effet, ces indicateurs sont calculés de manière *ad hoc* et les travaux ci-dessus ne prennent pas en compte une méthode de calcul d'indicateurs générique, capitalisable et réutilisable.

2.3.4 Les travaux sur la modélisation et le calcul d'indicateurs

Depuis quelques années, plusieurs travaux de recherche visent à définir et à proposer des méthodes génériques permettant l'analyse des traces et la modélisation des indicateurs. Cette partie présente les approches concernant cet axe de recherche. Nous abordons d'abord les projets du Réseau d'Excellence Européen Kaleidoscope[17] centrés sur l'analyse des interactions des acteurs dans les environnements d'apprentissage collaboratifs. Puis, nous présentons les travaux concernant l'approche de patron de conception en analyse de traces pour la modélisation d'indicateurs. La troisième partie est consacrée aux travaux portant sur l'approche par les modèles pour l'analyse et la gestion des traces.

2.3.4.1 Les projets ICALTS, IA et CAViCoLA

La littérature sur les EIAH présente plusieurs travaux de recherche destinés à l'observation et à l'analyse des interactions entre les acteurs (apprenants, tuteurs, enseignants, etc.) dans les environnements d'apprentissage collaboratifs assistés par ordinateur. Les projets du Réseau d'Excellence Européen Kaleidoscope comme ICALTS (*Interaction & Collaboration Analysis supporting Teachers & Students' Self-regulation*) [Dimitracopoulou (2004b)], IA (*Interaction Analysis* - Supporting teachers & Students' self-regulation) [Dimitracopoulou (2005)] et CAViCoLA (*Computer based Analysis and Visualization of Collaborative Learning Activities*) [CAViCoLA (2006)] ont porté sur ce champ.

Les projets ICALTS en 2004 et IA en 2005 se focalisent sur l'analyse de l'interaction complexe entre les acteurs d'un dispositif d'apprentissage collaboratif. Le projet IA est une suite du projet ICALTS. Il porte sur des questions de recherche abordées dans le projet ICALTS et vise à proposer des solutions pour permettre l'interopérabilité des outils et des méthodologies de mise en place des indicateurs dans un contexte d'autorégulation des apprenants et des enseignants. Les partenaires de ces projets ont proposé un cadre pour

[17] http://www.noe-kaleidoscope.org/telearc/

décrire le processus d'analyse de l'interaction et les concepts de base concernant l'analyse de l'interaction assistée par ordinateur.

- Le processus d'analyse de l'interaction se compose de quatre étapes : (1) la sélection ou le filtrage des données, (2) le traitement de données, (3) l'application des méthodes de traitement de données sur les données traitées pour produire un ou plusieurs indicateurs et (4) la visualisation ou la représentation des valeurs des indicateurs aux utilisateurs. Ces valeurs doivent être adressées directement aux étudiants ou aux enseignants sous une forme appropriée (textuelle, numérique, ou graphique).
- Dans le projet ICALTS, la notion d'indicateur[18] a été étudiée en détail. Le projet ICALTS a collecté et présenté quarante-cinq indicateurs et différents outils d'analyse permettant d'analyser les interactions sur différents dispositifs d'apprentissage collaboratifs. Comme nous l'avons présenté dans la section 2.3.3 (page 42), ces indicateurs sont hiérarchisés selon trois niveaux : (1) les indicateurs de haut niveau sont ceux qui ont une valeur interprétable et souvent calculée par un processus complexe à partir des données brutes ; (2) les indicateurs intermédiaires sont ceux élaborés à partir des données brutes et nécessaires pour le calcul d'indicateurs de haut niveau ; (3) les indicateurs de bas niveau sont ceux qui n'ont pas une valeur interprétable et qui sont souvent extraits directement à partir des données brutes. Ces indicateurs fournissent des informations utiles au chercheur ainsi qu'à l'apprenant et notamment à l'enseignant/tuteur. Ils aident ce dernier à comprendre les activités des apprenants, la façon dont l'apprenant réalise son activité, résout des problèmes, travaille en collaboration dans un groupe, etc.

Le projet CAViCoLA a hérité des résultats obtenus des projets ICALTS et IA. Les travaux de recherche de ce projet portent sur la question de l'adaptation dynamique des représentations des données analysées et de leur visualisation en fonction des centres d'intérêt de l'usager de ces données. Pour la communication entre différents outils d'analyse provenant de différents EIAH, CAViCoLA a proposé un format commun de traces. Ce format se compose d'une description DTD XML.

Comme le montre la figure 2-6, la structure de ce format général se compose d'un ensemble d'*actions* et éventuellement d'un élément *preamble*. Ce dernier permet de décrire des

[18] Un indicateur d'analyse de l'interaction est defini dans le projet ICALTS comme '' *'something' related to the 'quality' of individual activity (e.g. variables that he/she change, order of significant actions, etc.), the mode or the quality of the collaboration (e.g. division of labor, participation rates, categories of specific contributions), the process or the quality of the collaborative product. These variables have to be interpreted, taking into account, the learning activity, the profile of the participants and the context of interaction, etc.*''

informations sur le scénario de l'activité qui a produit la donnée d'interaction telles que les acteurs, les rôles, etc. Chaque *action* est décrite par trois éléments requis et certains éléments facultatifs. Les éléments requis sont la date de l'action, le type d'action (par exemple ‘‘envoyer un message’’), les acteurs de l'action et le rôle qu'ils assument dans l'action. Les autres éléments d'une action sont optionnels comme le contenu décrivant l'action et les objets sur lesquels l'action s'applique.

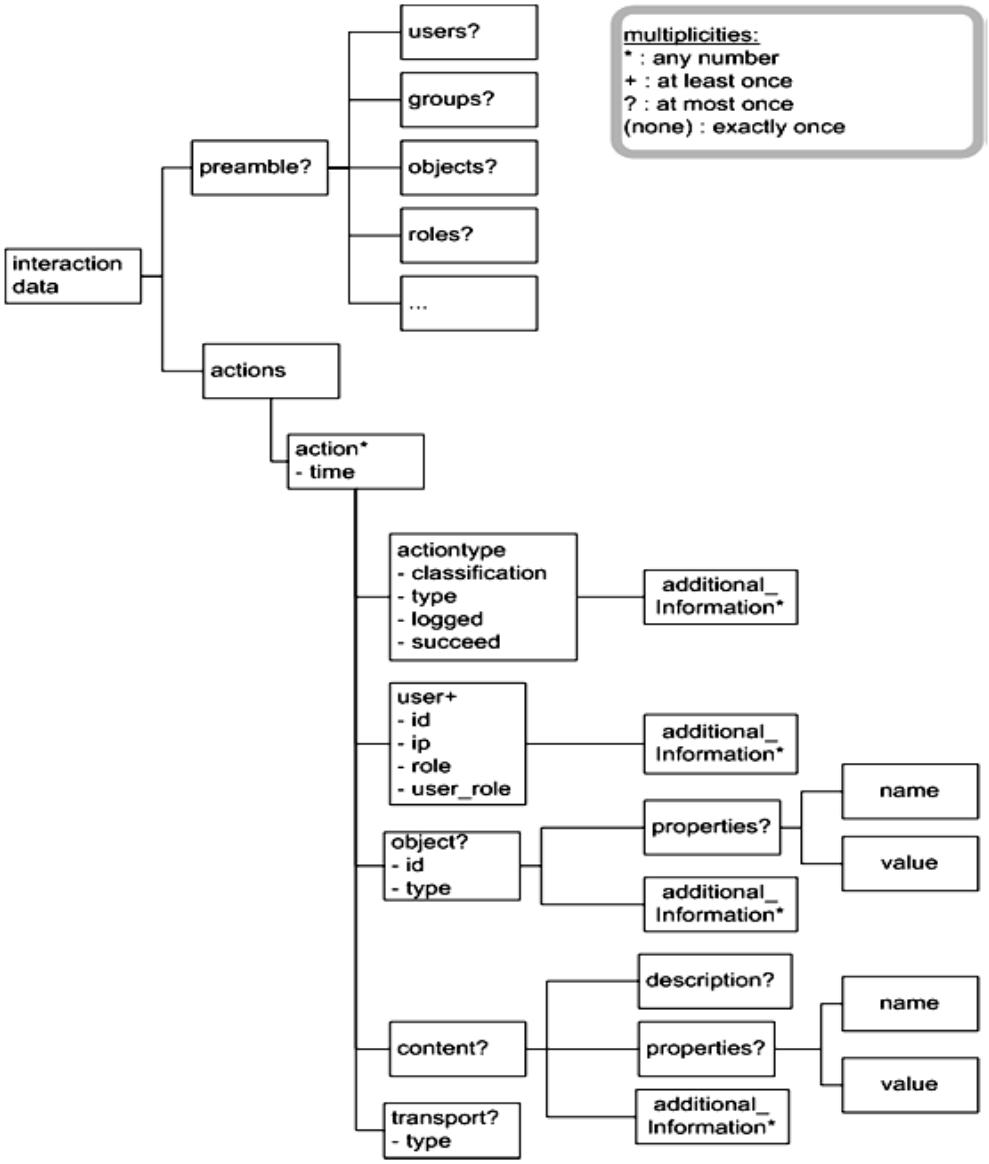

Figure 2-6 Structure simplifiée du format commun de traces proposé par CAViCoLA

Ce format est utilisé comme un moyen intermédiaire de communication entre un EIAH et un outil d'analyse. La figure 2-7 décrit la communication entre un EIAH et un outil d'analyse via le format commun : EIAH → XSLT → DTD du format commun → XSLT → outil d'analyse. Cette communication est réalisée par l'application de transformations XSL sur le format de sortie de l'EIAH et le format d'entrée de l'outil d'analyse.

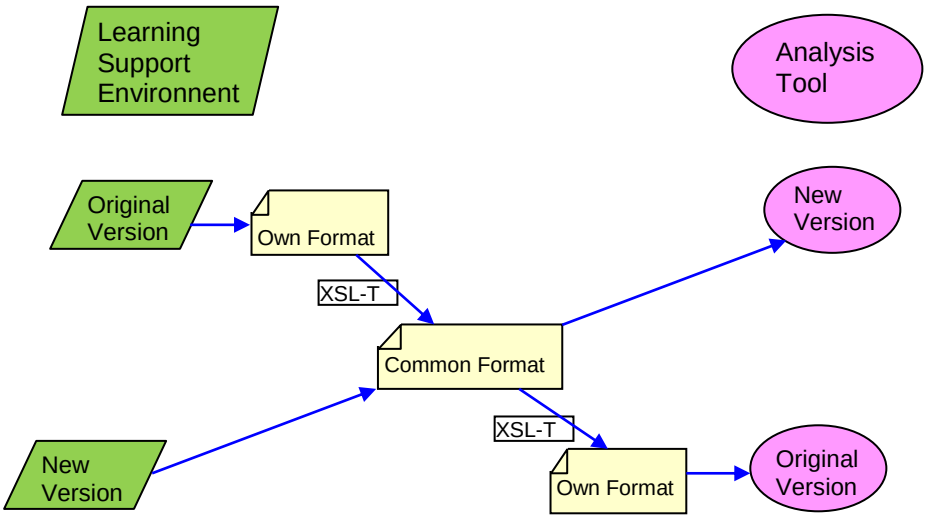

Figure 2-7 Communication entre un EIAH et un outil d'analyse via un format commun DTD

2.3.4.2 L'approche par patron de conception en analyse de traces

2.3.4.2.1 Introduction

Un patron de conception (design pattern en anglais) est utilisé pour décrire une solution générale réutilisable pour répondre à des problèmes récurrents dans notre environnement [Alexander et al. (1977)]. Cette approche est aujourd'hui largement utilisée dans plusieurs domaines : en architecture [Alexander et al. (1977)], en génie logiciel [Gamma et al. (1996), Buschmann et al. (1996), Schmidt et al. (2000)], en gestion de projet [Völter (2004)], en communication [Hohpe et Woolf (2003)], mais aussi dans le domaine éducatif et les EIAH. Nous présentons quelques travaux concernant les patrons de conception dans le domaine de l'éducation et des EIAH.

Depuis ces dernières années, l'approche par patron de conception est considérée comme une solution pour le problème de partage et de réutilisation du savoir-faire et de la connaissance experte concernant l'enseignement et l'apprentissage. Le projet PPP (Pedagogical Patterns Project) [PPP (2000)] est un des projets connus dans la création et la collecte des patrons pédagogiques pour l'enseignement et l'apprentissage. Ce projet a proposé un ensemble de patrons concernant la mise en place de séminaires [Fricke et Voelter (2000)], la mise en place d'un cours [Eckstein (2000)] et le développement de cours d'informatique [Bergin (2000)]. En plus de ces patrons, le projet collecte aussi de nombreux types de patrons portant sur l'enseignement et sur l'apprentissage, par exemple les patrons pour l'apprentissage actif, les patrons pour l'enseignement selon différentes perspectives, les patrons pour le feedback, les patrons pour l'apprentissage par l'expérience. Cette collection se compose de patrons basés sur des travaux antérieurs, révisés et réécrits dans la forme

de patron proposé par Alexander [Alexander et al. (1977)]. En général, chaque patron se compose des éléments suivants :

- D'un nom,
- D'une partie pour la description du problème à résoudre et éventuellement des contraintes,
- D'une partie pour la description de la solution du problème décrit,
- D'une partie pour la description des exemples et/ou des discussions, et une partie pour lier le patron à d'autres, ce qui a été proposé par [Gamma et al. (1996)].

Tous les patrons proposés ou collectés sont décrits en langage naturel et sont exprimés dans une forme similaire à celle utilisée par Alexander.

Si le projet PPP s'adresse particulièrement à l'enseignant ou au formateur dans le cadre d'une formation générale, le projet E-LEN [E-LEN (2003)] s'adresse au concepteur pédagogique dans le cadre d'une formation en ligne ou à distance. C'est un projet Européen dont un des objectifs est d'identifier et de collecter des expériences dans la conception et la mise en œuvre des environnements d'apprentissage basés sur le réseau afin de construire un ensemble de patrons pour le e-learning. Ces patrons ont été classifiés en quatre groupes d'intérêt (Special Interest Groups) : les ressources d'apprentissage et les systèmes de gestion de l'apprentissage (LMS - Learning Management Systems) [Avgeriou et al. (2003)], l'apprentissage tout au long de la vie, l'apprentissage collaboratif et l'apprentissage adaptatif. Le format de chaque patron proposé par ce projet est une variante de celui proposé par Alexander. Il se compose des éléments suivants :

- D'un nom,
- D'une partie pour la description du problème à résoudre,
- D'une partie pour l'explication des origines du problème, éventuellement avec un exemple pour mieux communiquer,
- D'une partie pour la description de la solution proposée par ce patron,
- D'une partie pour la description des exemples du patron dans LMS réel,
- D'une partie pour la spécification d'autres patrons qui sont liés à celui-ci.

Dans le cadre du projet PLAIT (Pattern Language for Architectures of Intelligent Tutors), [Devedzic et Harrer (2005)] ont fourni aux concepteurs informatiques ou aux informaticiens un ensemble de patrons portant sur la conception d'architectures logicielles des ITS (Intelligent Tutoring Systems). Contrairement aux descriptions assez générales des patrons des projets PPP et E-LEN, les patrons proposés dans ce travail sont décrits de façon un peu plus détaillée. Par exemple, ce travail présente plusieurs architectures d'ITS comme

l'architecture classique des ITS, l'architecture de systèmes d'apprentissage collaboratif, l'architecture de systèmes utilisant des agents pédagogiques, etc. Pour chaque architecture d'ITS, la partie de solution décrit les modules de base, l'élément de diagramme illustre le diagramme entre les modules présentés dans la partie de solution et la partie de variante décrit les modules supplémentaires qui pourraient être ajoutés afin d'enrichir le système. Mais ces patrons se limitent également à la description, ils n'indiquent pas comment implémenter la solution proposée.

Contrairement aux patrons ci-dessus concernant l'enseignant et l'apprentissage ou la conception d'architectures logicielles des ITS, nous présentons ensuite les travaux utilisant l'approche par patron de conception pour le partage et la réutilisation des savoir-faire et des techniques de la communauté en analyse des traces pour l'élaboration des indicateurs. La suite de cette partie présente des travaux portant sur la création de patrons pour la capitalisation et la réutilisation de savoir-faire en analyse de traces.

2.3.4.2.2 Le projet DPULS

Le projet DPULS [Choquet (2005), Pozzi (2005), Verdejo et al. (2005), Delozanne et al. (2005), Delozanne et al. (2007)] (Design Patterns for collecting and analysing Usage of Learning Systems) est une action du Réseau d'Excellence Européen Kaléidoscope. Ce projet se base sur l'idée que le savoir-faire et l'expérience des enseignants/concepteurs en collecte et analyse des usages mis en place pour un EIAH peuvent être capitalisés et réutilisés dans la conception d'autres EIAH. Donc l'objectif principal du projet est de fournir aux concepteurs pédagogiques, aux enseignants et aux tuteurs des expériences et des patrons de conception réutilisables pour les aider à résoudre des problèmes récurrents dans l'observation et le suivi des activités des apprenants dans les EIAH.

Ce projet a proposé une typologie des données d'observation et fourni un ensemble de patrons de conception. La figure 2-8 illustre l'ensemble des types de données de la typologie. Chaque type est défini en fonction des objectifs d'utilisation et de sa provenance. Comme le montre la figure 2-8, ce projet distingue deux types de donnée (*Data*) : la donnée primaire (*Primary data*) et la donnée dérivée (*Derived data*). Cette dernière est calculée ou établie à partir de données primaires ou d'autres données dérivées. Au contraire, une donnée primaire n'est pas calculée ou déduite à partir d'autres données. Une donnée primaire peut être :

- Une donnée brute (*Raw data*) collectée avant, pendant ou après la session d'apprentissage ;

- Une donnée additionnelle (*Additional data*), qui peut être contextuelle (*Contextual data*), concerne la description de la situation d'apprentissage et est disponible avant la session (par exemple, un scénario prédictif, la méta-donnée d'une ressource, une taxonomie décrivant le domaine d'apprentissage), ou prédictive (*Predictive data*), qui est liée aux résultats produits par les acteurs de la situation d'apprentissage pendant la session (par exemple, un rapport d'un étudiant à évaluer, le compte-rendu d'activité d'un tuteur).
- Une donnée subjective (*Subjective data*) est obtenue par l'analyste de la session (par exemple, les réponses aux questions dans un questionnaire, une donnée obtenue par l'analyse textuelle).
- Un indicateur (*Indicator*) est une caractéristique de la donnée (*Data*) et est défini comme une variable signifiante sur le plan pédagogique, calculée ou établie à l'aide de données observées, et témoignant de la qualité de l'interaction, de l'activité et de l'apprentissage dans un EIAH pédagogique.

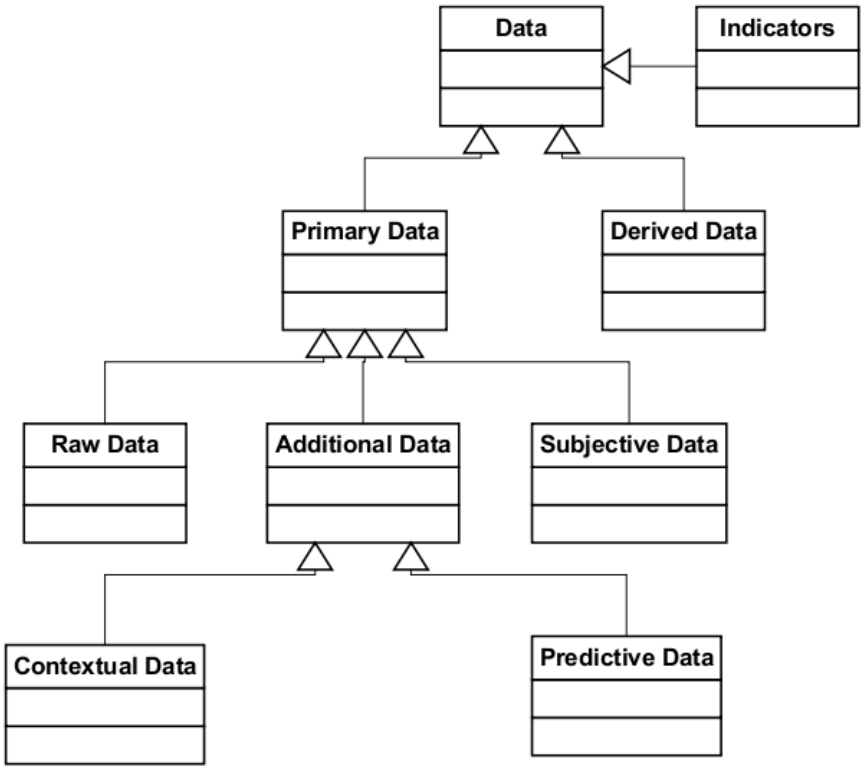

Figure 2-8 Typologie des données d'observation proposée par DPULS

Le deuxième résultat du projet est un ensemble de patrons de conception permettant à l'enseignant/concepteur et au tuteur de pouvoir capitaliser et réutiliser le savoir-faire de collecte et d'analyse d'usage d'un EIAH vers un autre. Chaque patron comprend cinq parties (cf. figure 2-9) qui permettent d'identifier un patron, son problème ainsi que ses solutions :

- La partie ‘‘*General*’’ est utilisée pour décrire le problème comprenant un contexte d'application caractérisé par un type d'EIAH, un type de situation pédagogique et les acteurs cibles de l'analyse des usages prescrite par le patron.
- La partie ‘‘*Problem*’’ décrit le problème d'analyse des usages traité par le patron.
- La partie ‘‘*Set of Solutions”* décrit la (ou les) solution(s) au problème applicable(s) dans le contexte d'application du patron, notamment en définissant les indicateurs à établir et en caractérisant la méthode d'analyse permettant de les établir.
- La partie ‘‘Set *of Related Patterns*’’ décrit les autres patrons définis qui sont liés à celui-ci. Ils peuvent être des patrons du projet DPULS ou des patrons d'autres langages de patrons.
- La partie ‘‘*Pattern Identification*’’ permet d'indexer le patron, d'en gérer les versions et d'y associer des références bibliographiques.

General	Name	
	Abstract	
	Category	
	Context	Type of System
		Type of Situation
		Actors
		General Description
Problem	Statement	
	Tracking Focus	
	Analysis	
Set of Solutions	Solution Name	
	Objective	
	Requisites	Indicators
		Methods
	Description	
	Discussion	
	Example	
Set of Related Patterns	Related Pattern	Related Pattern name
		Related Pattern type
		Relationship
Pattern Identification	Author	
	Date	
	Version	
	Bibliographic References	

Figure 2-9 La structure d’un patron de conception dans DPULS

2.3.4.2.3 Le langage UTL

Le langage UTL [Choquet et Iksal (2007a)] est conçu avec l'objectif de décrire les différentes données nécessaires à l'analyse des traces dans une forme indépendante du langage de scénarisation et du format de représentation des traces. Concrètement, il permet de :

- Capitaliser des savoir-faire en analyse de trace et en observation d'une session,
- Définir le besoin d'observation et, avec l'assistance de l'informaticien, de spécifier les moyens techniques pour l'acquisition des données à observer,
- Structurer des données générées par le dispositif d'apprentissage depuis la donnée brute jusqu'à l'indicateur dans une forme indépendante du format des traces générées.

Pour arriver à cet objectif, UTL a repris les deux types principaux de données du projet DPULS : la *donnée primaire* (Primary datum: PD) et la *donnée dérivée* (derived datum: DD). La *donnée primaire* n'est pas calculée ou établie avec l'aide d'autres données et elle se décompose en la *donnée brute* (raw datum: RD), la *donnée de production* (content datum : CD) et la *donnée additionnelle* (additional datum : AD). La *donnée intermédiaire* (intermediate datum : ID) et l'*indicateur* (indicator : I) constituent le type *''donnée dérivée''.* Cette dernière est calculée ou établie à l'aide des *données primaires* et/ou d'autres *données dérivées*. La figure 2-10 présente le modèle conceptuel du langage.

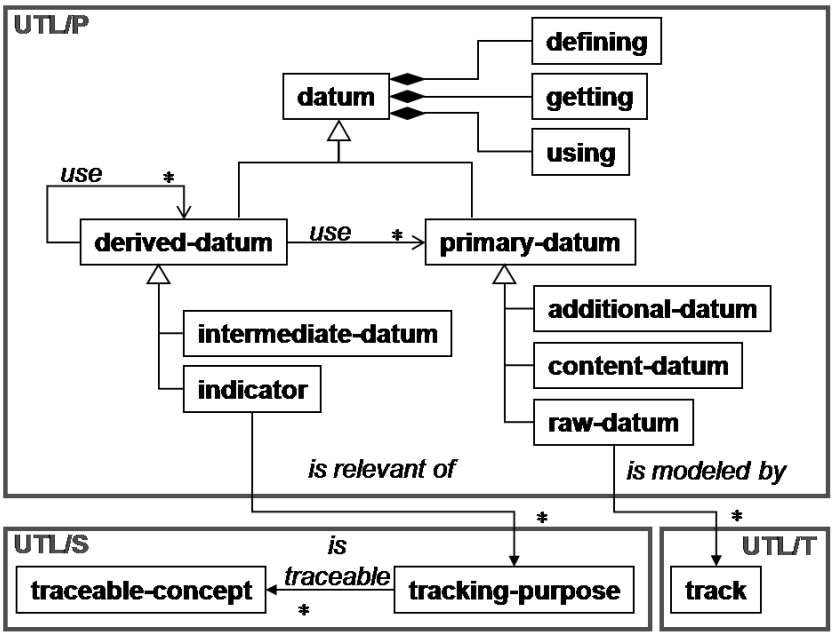

Figure 2-10 Le modèle conceptuel d'UTL

- Une *donnée brute* (RD) est extraite de la trace après la collecte et représentée dans un format indépendant du dispositif d'apprentissage. Elle peut être collectée avant,

pendant ou après la session d'apprentissage par le dispositif d'apprentissage comme, par exemple, un fichier de ‘‘logs’’ enregistré par le système, un enregistrement vidéo de l'apprenant pendant la session, les informations recueillies par un questionnaire avant ou après la session, ou l'ensemble des messages postés dans un forum.

- Une *donnée de production* (CD) est produite volontairement par les acteurs de la session d'apprentissage (apprenant, tuteur et/ou enseignant). Ces données sont principalement des travaux réalisés par les étudiants et destinés à être évalués, mais peuvent également être, par exemple, un rapport établi par le tuteur sur la qualité de l'activité ou d'une ressource pédagogique.
- Une *donnée additionnelle* (AD) est une donnée liée à la situation pédagogique et utilisée pour la production des données dérivées. Ces données sont de natures très variables : la valeur d'un champ d'une métadonnée caractérisant une ressource, une taxonomie, une donnée ad hoc, etc.
- Un *indicateur* (I) est un observable signifiant sur le plan pédagogique, calculé ou établi à l'aide d'observés, et témoignant de la qualité de l'interaction, de l'activité et de l'apprentissage dans un EIAH.
- Une *donnée intermédiaire* (ID), au contraire d'un indicateur, n'a pas de signification pédagogique en soi. Elle est cependant nécessaire à la construction d'un ou de plusieurs indicateurs.

Chaque type de donnée UTL est défini selon trois facettes : Defining – Getting - Using

- La facette définition (Defining) permet de modéliser le besoin d'observation,
- La facette obtention (Getting) de la trace permet de modéliser le moyen d'observation,
- La facette utilisation (Using) précise l'utilité de la donnée d'observation et décrit l'utilisation qui en sera faite.

Le modèle d'information détaillé des types de donnée UTL selon ces trois facettes est présenté dans [Choquet et Iksal (2007a)]. Nous présentons particulièrement ici le modèle d'information détaillé de l'indicateur. Son modèle d'information est montré dans la figure 2-11.

- La facette *Defining* de l'indicateur est caractérisée par trois éléments : le titre (*Title*) de l'indicateur, la cardinalité (*Cardinality*) de l'indicateur représentant le nombre de valeurs que l'on peut avoir et une éventuelle description (*Description*) détaillée de l'indicateur.

- La facette *Getting* est décrite par deux éléments principaux : *Component* et *Method*. Elle caractérise la construction de la donnée, c'est-à-dire la manière dont la donnée est générée à partir d'autres données appelées composants (*Component*). Chaque composant est décrit par un titre (*Title*) et un type (*Type*) qui est une *donnée primaire* ou une *donnée dérivée*. C'est cet élément *Component* qui définit l'ensemble des observables nécessaires à la spécification de l'indicateur. L'élément méthode (*Method*) est obligatoire. Il peut être du type (*Type*) manuel, semi-automatique ou automatique. Si un acteur humain intervient dans l'établissement de l'indicateur, il faut indiquer son rôle (*Role-involved*). Si la méthode est faite de manière automatique ou semi-automatique, l'outil support (*Tool*) doit être indiqué grâce à sa localisation (*Location*) dans la mesure du possible ou par une description (*Description*) et quelques exemples (*Example*). L'outil a toujours un titre (*Title*). La facette *Getting* est également composée d'un titre (*Title*). Elle peut avoir une *Description* pour compléter la donnée et faciliter la compréhension, une *Discussion* pour préciser la nature de la donnée désirée et différents exemples (*Exemple*) décrivant des méthodes à réaliser pour l'acquisition de la donnée.
- La facette *Using* comporte quatre éléments. L'élément *Data* permet de stocker les valeurs de l'indicateur, une fois calculé ou établi. *Used-by* est proposé pour faciliter la navigation dans un graphe de dépendances des données. *Format* décrit comment la donnée est représentée (par exemple, son schéma XML), une fois obtenue. L'élément *Pedagogical-context* permet de définir un ou plusieurs objectifs d'observation (*Tracking-purpose*) de l'indicateur. Chaque objectif d'observation est décrit par quatre éléments. L'élément concepts traçables (*Traceable-concept*) qui permet d'associer l'indicateur à un élément du scénario pédagogique. Il comporte deux éléments. *Concept* est un concept utilisé par l'enseignant pour définir son scénario pédagogique (e.g. activité, ressource). *Relationship* indique les relations avec les autres concepts traçables. Son titre (*Title*) est utilisé pour décrire la sémantique de la relation. L'élément *Type* indique un type d'exploitation. Il y a quatre types d'exploitation définis : réingénierie, régulation, évaluation et réflexivité. Un ou plusieurs rôles (*Recipient-role*) précisent le destinataire de l'indicateur. Une *Description* peut être ajoutée pour décrire de façon détaillée l'objectif d'observation.

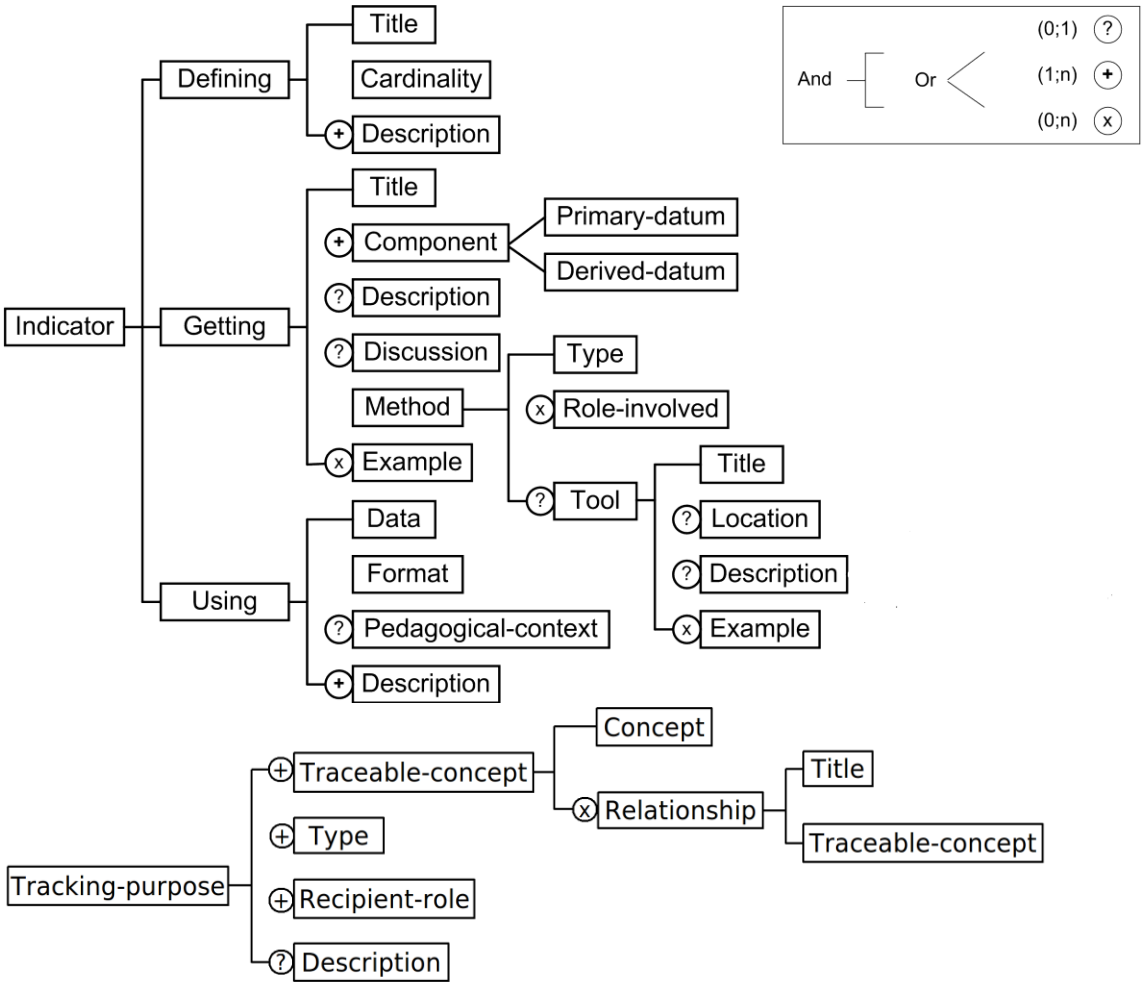

Figure 2-11 Le modèle d'information de l'indicateur

2.3.4.2.4 Les patrons d'Indicateur Réutilisables pour la supervision de l'apprentissage

Dans un objectif de capitalisation et de réutilisation des savoir-faire en termes de définition des indicateurs, [Diagne (2009)] a proposé un formalisme pour décrire les *Patrons d'Indicateur Réutilisable*. Ce type de patron est proche de ceux qui sont proposés par le projet DPULS. Comme le montre la figure 2-12, en se basant sur le modèle général d'un patron de conception présenté par [Gamma et al. (1996)], chaque patron d'indicateur réutilisable est une structure comprenant quatre parties :

- *Descriptif de l'indicateur* : cette partie concerne la description générale de l'indicateur en spécifiant son nom, le type d'indicateur (cognitif, social, etc.) et sa description détaillée ;
- *Problème* : cette partie permet de décrire les objectifs de régulation pour lesquels l'indicateur a été utilisé, le contexte de supervision dans lequel ces objectifs ont été atteints et les contraintes spécifiques de réutilisation de l'indicateur ;

- *Solution* : consiste à décrire la *fonction "indicateur"* et son domaine de définition. Cette fonction peut être un algorithme, une formule mathématique ou un programme écrit en langage de programmation qui décrit la méthode d'acquisition de l'indicateur. La *fonction "indicateur"* doit obligatoirement être décrite dans le patron pour pouvoir être réutilisée. Un modèle d'indicateur se compose de trois parties : *sélection, analyse* et *visualisation*. La *sélection* consiste à récupérer des traces nécessaires à l'établissement de l'indicateur à partir d'une ou de plusieurs sources de traces générées par une plateforme d'apprentissage. L'*analyse* permet de structurer et de traiter les traces brutes pour obtenir un ensemble de données pertinentes répondant au problème défini. La *visualisation* représente les données obtenues dans la phase d'*analyse* sous la forme d'une présentation textuelle ou en utilisant des techniques de visualisation (par exemple un graphe). C'est la *fonction "indicateur"* qui constitue la partie d'*analyse*.
- *Autres patrons* : cette partie permet d'indiquer les autres patrons définis dans le même contexte.

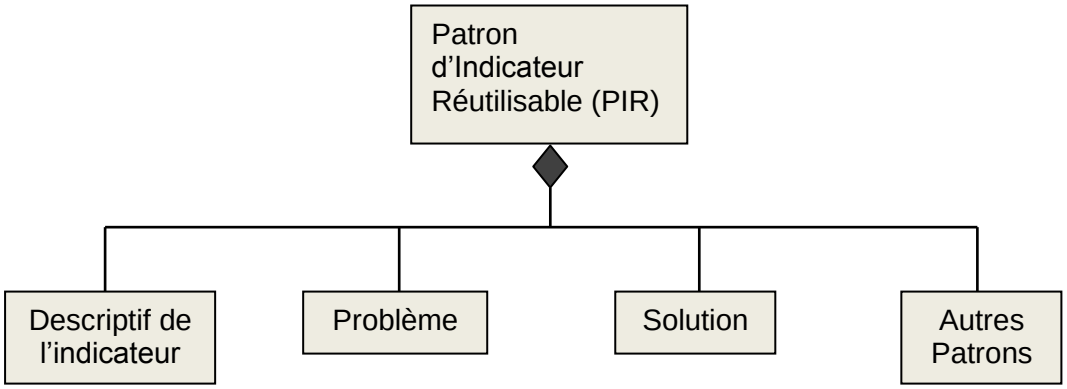

Figure 2-12 La structure d'un patron d'indicateur réutilisable

2.3.4.2.5 Le patron d'un indicateur de collaboration

L'apprentissage collaboratif prend de plus en plus de place dans l'apprentissage assisté par ordinateur. Plusieurs travaux de recherche s'intéressent à l'observation ou à l'analyse de ce type d'activité. Le travail de Gendron [Gendron (2010)] porte sur l'amélioration des activités collaboratives par la mise en place d'indicateurs. Pour cela, ce travail a proposé un cadre conceptuel pour l'élaboration d'indicateurs de collaboration. Chaque indicateur est décrit par trois vues : une *carte d'identité*, un *patron* et une *vitrine*. La figure 2-13 présente ces trois vues.

La carte d'identité comprend des informations comme le nom (Name), la description (Desc), la date de création et de modification (Creation date et Update Date) ainsi que le créateur et

les modificateurs (Creator et Updator(s)). Ces informations permettent d'identifier, de décrire et de gérer des indicateurs.

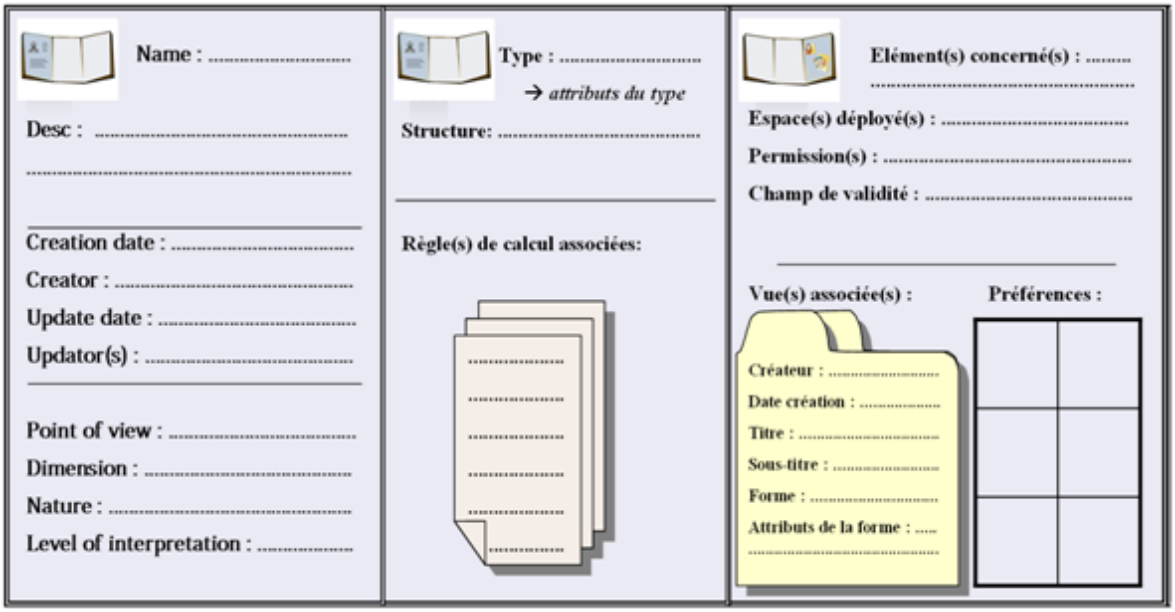

Figure 2-13 La structure d'un indicateur de Collaboration

Le patron intitulé *Patron d'Indicateur de Collaboration* permet de spécifier la méthode de calcul d'indicateurs. Ce patron se compose d'un type, d'une structure et de règles de calcul.

- *Type* : il s'agit du type de donnée d'un indicateur. Les types peuvent être numériques, alphanumériques et structurés. Les types numériques et alphanumériques utilisés se composent des types *Nombre* (par exemple : le nombre de compilations par question) et *Seuil* (par exemple : le niveau de chaque apprenant). Les types structurés se composent des types *Tableau* et *Intervalle Discret*.
- *Structure* : il s'agit du type de production d'un indicateur. Dans ce contexte, deux types sont considérés : l'indicateur *élémentaire* et l'indicateur *composite*. Un indicateur *élémentaire* est calculé à partir des traces, tandis qu'un indicateur composite est calculé à partir d'autres indicateurs.
- *Règles de calcul* : il s'agit d'appliquer un ensemble de règles sur les traces et/ou les indicateurs pour obtenir les valeurs de l'indicateur.

La vitrine contient les informations concernant l'instanciation des indicateurs.

En ce qui concerne ces règles de calcul, Gendron a proposé un langage de calcul basé sur l'approche ''*système à base de règles*''. Le langage se compose des faits et des règles. Les traces et les indicateurs sont les faits. Une règle comprend les faits et les actions opérées sur les faits pour obtenir la valeur de l'indicateur.

Toutes les traces contiennent une action, une ressource, un acteur, une date et une valeur (cf. figure 2-14).

```
<trace>        : :=
                    <trace_action>
                    <trace_content>
                    <trace_actor>
                    <trace_date>
                  [ <trace_value> ]

<trace_action>   : := action <property>
<trace_content> : := content <property>
<trace_actor>    : := actor <property>
<trace_date>     : := date <property>
<trace_value>    : := value <property>
```

Figure 2-14 La syntaxe d'une trace

Tous les indicateurs ont le même format qui se compose d'un nom, d'une date de création et de dernière modification, d'un état, d'un ou plusieurs créateurs, d'un objet, d'un type et d'une valeur (cf. figure 2-15).

```
<indicator> : :=
                <indic_name>
                <indic_date_calculation>
                <indic_date_creation>
                <indic_state>
                <indic_creators>
               [ <indic_object> ]
                <indic_value>
                <indic_type>

<indic_name>                : := name <property>
<indic_date_calculation>    : := date_calculation <property>
<indic_date_creation>       : := date_creation <property>
<indic_state>               : :=  state  <property>
<indic_creators>            : := creators <property>
<indic_object>              : :=  object  <property>
<indic_value>               : := value <property>
<indic_type>                : :=
                              <indic_type_name>
                              <indic_type_date>
                              <indic_type_property>*

<indic_type_name> : := type_name « Number" |
                       " Discrete Interval"  | « Threshold" | "Table"
<inidc_type_date> date_begin <prop_value>
                  date_end <prop_value>
<indic_type_ property> : := <indic_type_prop_name> <property>
<indic_type_prop_name>: := "period "  | "thresholds"
```

Figure 2-15 La syntaxe d'un indicateur

Chaque règle est identifiée par un nom unique et toujours accompagnée d'un couple <conditions/actions>. La partie ‘‘condition’’ permet de décrire des éléments qui déclenchent le calcul de la valeur de l'indicateur. Ces éléments peuvent être les traces, les indicateurs et les opérateurs comme le filtrage. La partie ‘‘action’’ décrit les opérations (comme affichage, calcul mathématique, composition) effectuées sur la valeur d'un ou plusieurs d'indicateurs. La partie ‘‘*target*’’ décrit des indicateurs modifiés par la règle. La figure 2-16 présente la syntaxe d'une règle de calcul et la figure 2-17 en est un exemple.

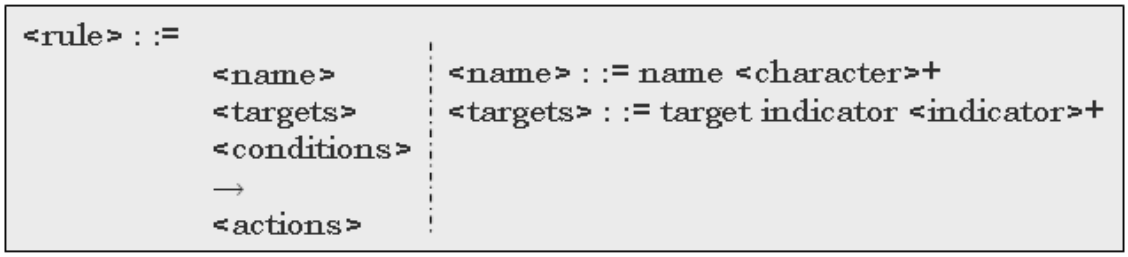

Figure 2-16 La syntaxe d'une règle

name ''Construction de l'indicateur Adhésion à un espace''

targets

name ''=='' ''Adhésion à un espace''
state ''=='' ''actif''
object >> indic_obj
value >> val

filters

action ''=='' ''consultation''
OR
action ''=='' ''contribution''
actor >> trace_actor
date >> d

criteria test(indic_obj==trace_actor.groupe)

→

val.incrémenter(d)

Figure 2-17 Exemple d'une règle de calcul utilisant la base de règles

2.3.4.3 L'approche par les modèles

2.3.4.3.1 Calcul des indicateurs dirigé par les modèles de trace SBT

Le Système à Base de Traces (SBT) [Settouti et al. (2006), Settouti et al. (2007)] est un travail qui hérite du passé du LIRIS (Laboratoire d'InfoRmatique en Image et Systèmes d'information) autour de MUSETTE [Champin et al. (2003)]. Un SBT est défini comme *un système informatique dont le fonctionnement implique à des degrés divers, la gestion, la transformation et la visualisation de traces modélisées explicitement en tant que telles* [Settouti et al. (2006)]. Il facilite donc l'exploitation des traces. Il permet de collecter les traces en session, de faire des transformations sur ces traces pour produire des indicateurs, pour ensuite les visualiser. Dans un système à base de traces, une trace est considérée comme *une séquence temporelle d'observés* et est toujours accompagnée de son modèle de traces. Il s'agit d'*un ensemble d'objets étiquetés représentant le vocabulaire de la trace*. Une trace est donc liée à son modèle ainsi qu'à ses règles de transformation. Ces dernières sont

appliquées sur une ou plusieurs traces pour atteindre la trace significative pour l'utilisateur. Les deux types de transformation dans un SBT sont la transformation manuelle et la transformation automatique [Settouti et al. (2007)].

- La transformation manuelle est réalisée par l'utilisateur et contient toute modification impliquant un changement dans la composition des éléments de la trace (ajout, suppression ou modification des observés).
- La transformation automatique est réalisée moyennant des modèles de transformation. Un modèle de transformation est un ensemble de règles exprimant des filtres de sélection ou des réécritures de motifs. La transformation automatique se compose de trois opérateurs de base représentés dans la figure 2-18 :

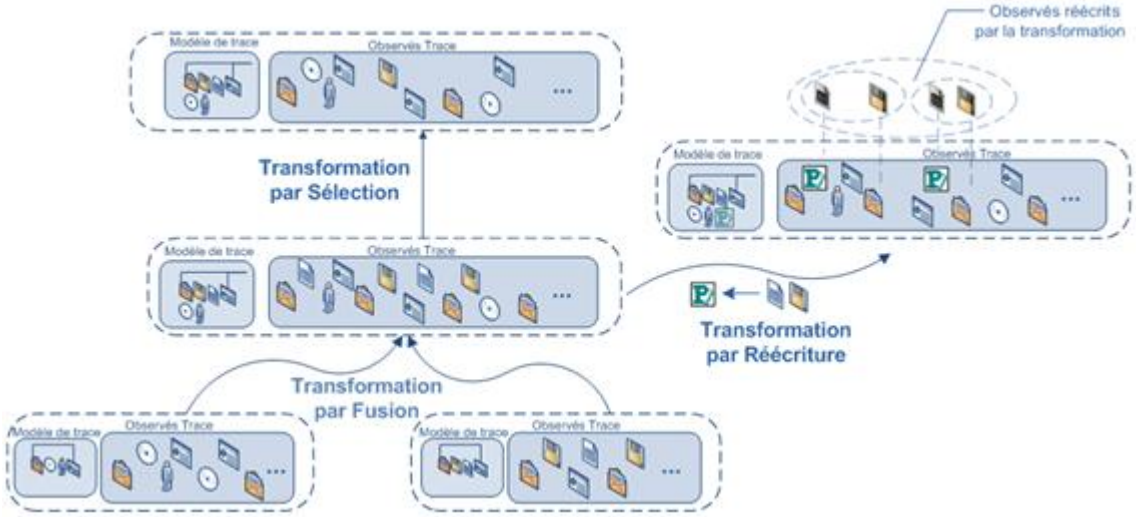

Figure 2-18 La transformation de traces modélisées [Settouti et al. (2007)]

i) Sélection : cet opérateur sélectionne une partie d'un observé d'une trace. Il produit une nouvelle trace contenant tous les observés respectant un filtre de sélection donné.

ii) Réécriture de motif : cet opérateur permet de remplacer un ou plusieurs observés par un autre observé.

iii) Fusion temporelle : cet opérateur concatène des observés de traces sources de même modèle pour obtenir une nouvelle trace.

Dans leur récent travail, [Settouti et al. (2009)] ont également proposé un langage nommé TQML [Settouti (2011)] (Transforming and Querying Modelled traces Langage) permettant l'interrogation et la transformation des traces modélisées.

En utilisant ces transformations automatiques, [Djouad et al. (2009)] proposent les règles de calcul des indicateurs s'appliquant au modèle de traces. Dans leur travail, un indicateur est

une donnée qui peut être calculée à partir d'une ou plusieurs traces modélisées. Il s'agit (1) de récupérer et de restructurer les traces brutes issues des sources de traçage pour produire de nouvelles traces modélisées nommées traces premières et (2) ensuite d'appliquer les opérateurs de transformation sur la trace première afin de calculer des reformulations de trace et des indicateurs qui fassent sens dans le cadre d'une activité collaborative. Le calcul d'un indicateur ‘‘I’’ avec l'aide d'un SBT comprend les quatre étapes illustrées dans la figure 2-19 :

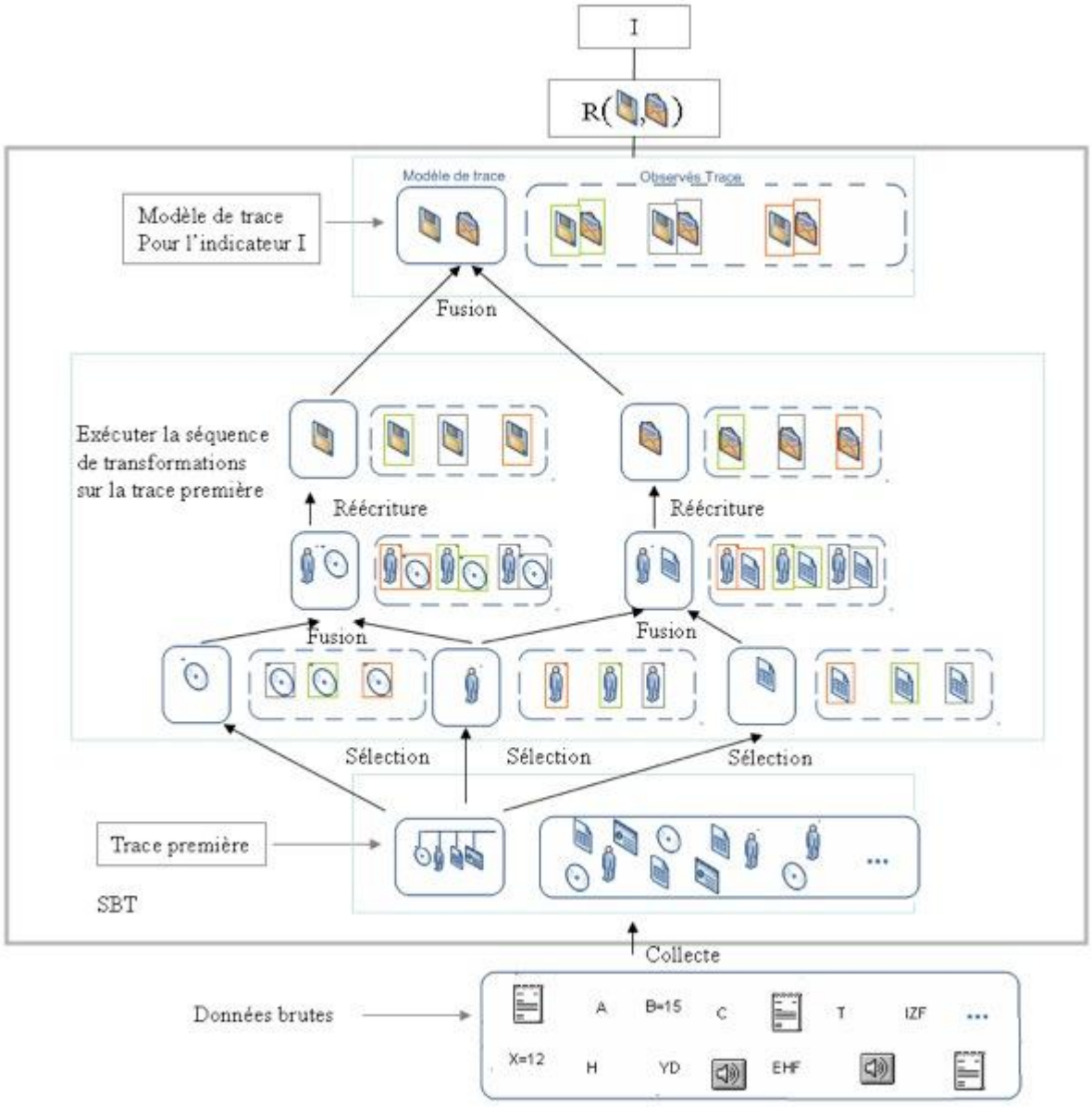

Figure 2-19 La séquence de transformations sur la trace première [Djouad et al. (2009)]

1. La première étape consiste à définir un modèle de trace pour ‘‘I’’ ainsi que ses transformations.

2. La deuxième étape consiste à définir une séquence de transformations pour établir l'indicateur ‘‘I’’. Cette séquence permet de faire le passage de la trace première modélisée vers le modèle de l'indicateur ‘‘I’’.
3. La troisième étape a pour but de collecter les observés de la trace première à partir des données générées par le dispositif d'apprentissage.
4. Cette étape consiste à exécuter la séquence de transformations définie dans la deuxième étape. Son résultat est l'indicateur ‘‘I’’ conforme à son modèle de trace défini dans la première étape.

Pour faciliter le calcul d'indicateurs, [Djouad et al. (2009)] ont défini deux opérateurs *compter* et *filtrer*. Ces opérateurs ne sont pas des transformations. Ils ne sont pas intégrés dans le SBT.

- Compter : permet de compter le nombre d’instances d'observés d'un certain type dans une trace. Le résultat est un nombre entier positif ou nul.
- Filtrer : permet de filtrer et de trier un ensemble d'observés selon le temps, les types d'outils et/ou les types des observés.

2.3.4.3.2 L’approche de traçage conduite par les modèles

En utilisant l'approche de modélisation CIM (Common Information Model) [DMTF (2005)], Broisin et Vidal [Broisin et Vidal (2007)] ont proposé une approche conduite par les modèles pour la gestion des traces. Leurs contributions portent sur la création d'un modèle UML de traces et sur la proposition d’une architecture distribuée et décentralisée associée à ce modèle afin de favoriser le partage et la réutilisation des traces collectées par différents outils et dispositifs d’apprentissage.

Le modèle UML proposé est générique et est une extension du méta-modèle CIM. Il permet de structurer et d’ajouter une sémantique aux traces produites par les activités explicites des utilisateurs dans un système d’apprentissage instrumenté par le Web. Comme le montre la figure 2-20, la modélisation UML vise à présenter quatre composants principaux. Le composant *EIAH_System* modélise les systèmes d’apprentissage en ligne. Le concept *EIAH_Resource* représente les ressources intégrées dans ces systèmes. En ce qui concerne les utilisateurs, ce concept n'est pas modélisé dans la figure 2-20, pour des raisons de simplification. Le composant *EIAH_ActivityOnSystem* modélise les activités sur les systèmes, tandis que le composant *EIAH_ActivityOnResource* représente les activités sur les ressources. Ces éléments dans le modèle UML sont utilisés pour la description des traces à collecter sur l’utilisation du système d’apprentissage, sur l’utilisation des ressources

par les utilisateurs notamment les apprenants et sur la réalisation des activités pendant la session par les apprenants.

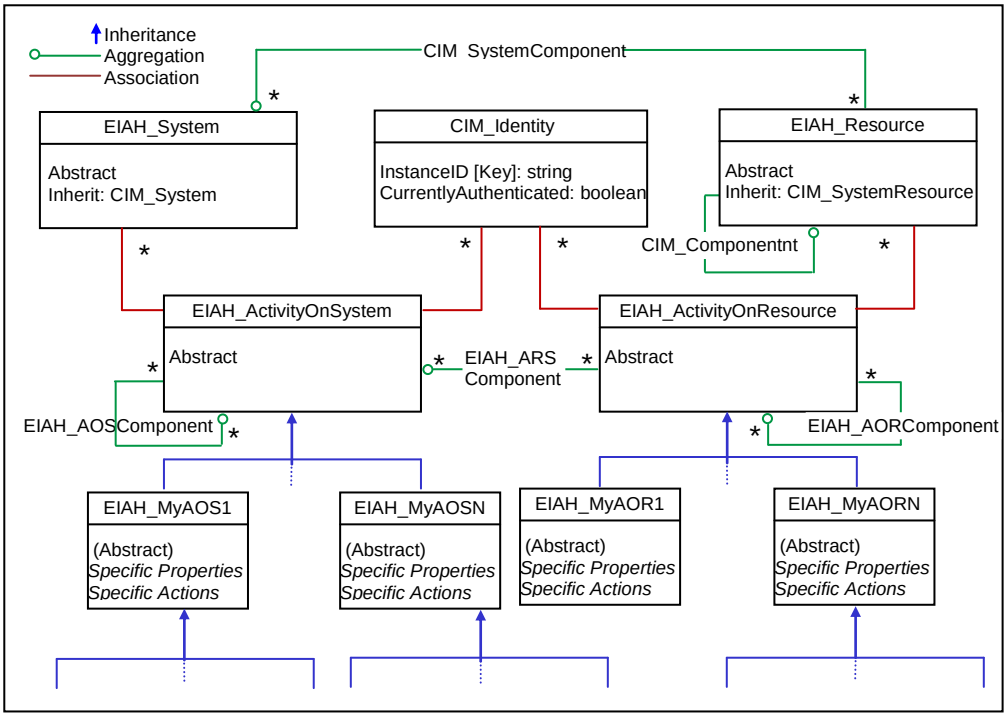

Figure 2-20 Un modèle UML générique de traces proposé par [Broisin et Vidal (2007)]

L'architecture distribuée et décentralisée associée à ce modèle UML est conforme au WBEM (Web Based Entreprise Management). Cette architecture permet de collecter des traces des utilisateurs d'un système d'apprentissage, de stocker, de visualiser et d'exploiter ces traces. L'exploitation de ces traces peut être réalisée par des requêtes CQL (CIM Query Language).

2.3.4.4 Discussions

Les travaux présentés dans cette partie concernent des propositions d'approches, de méthodes ou d'outils qui permettent de modéliser les traces et/ou les indicateurs et leurs méthodes de calcul.

Le projet ICALTS vise à l'analyse des traces d'interaction entre les apprenants dans les EIAH. Ce processus d'analyse est similaire au processus d'application de data-mining sur les données collectées par des EIAH présenté par [Romero et Ventura (2007)] ci-dessus. Le processus d'analyse proposé dans ce projet considère que les besoins d'observation, les points de vue des enseignants/tuteurs ou chercheurs sont des facteurs importants pour formaliser les indicateurs. C'est-à-dire que l'approche d'analyse de traces par l'hypothèse est prise en compte. Nous partageons entièrement ce point de vue. Les partenaires du projet ont

collecté et présenté plusieurs indicateurs et différents outils d'analyse de traces provenant de plusieurs laboratoires de recherche en Europe. Chacun de ces outils est développé pour répondre à des problèmes spécifiques d'un EIAH qui génère souvent des traces dans un format propre. Il est donc difficile de réutiliser un outil prévu pour un EIAH afin d'analyser les traces d'un autre EIAH. Le projet CAViCoLA s'est intéressé à ce problème. Ce projet propose un format unifié de traces d'interaction. Cette approche permet donc la communication et la réutilisation des outils d'analyse dans les différents EIAH. Cependant, ces projets ne disposent pas d'une méthode pour le développement d'un outil générique permettant de calculer les indicateurs pour les différents EIAH. Ils ne considèrent également pas le problème de la capitalisation et de la réutilisation des savoir-faire sur les techniques d'analyse de l'observation d'une session d'apprentissage. Ce problème est plus ou moins pris en compte dans plusieurs travaux : le projet DPULS, le projet REDiM, le travail de [Diagne (2009)] et [Gendron (2010)].

Le projet DPULS a proposé des solutions pour répondre aux pratiques d'analyse diverses dans une structure commune réutilisable. La plupart des champs d'un patron sont décrits en langage naturel ce qui permet d'être compréhensible par l'enseignant/concepteur. De plus, par la définition d'indicateurs, chaque patron comprend également les descriptions techniques concernant un problème à résoudre ce qui peut permettre d'implémenter la solution proposée par le patron. Cependant, les patrons du projet DPULS se limitent à la description en détail des solutions génériques permettant de produire des indicateurs. Ce projet ne propose en aucun cas des outils d'analyse automatiques et génériques qui sont réutilisables pour exécuter les patrons proposés.

En se basant sur les résultats des projets du Réseau d'Excellence Européen Kaleidoscope, [Choquet et Iksal (2007a)] ont proposé le langage UTL. Comme nous l'avons indiqué précédemment, UTL est utilisé pour définir et modéliser des données d'observation ainsi que les besoins d'observation associés. Il permet de définir des indicateurs sous une forme proche des patrons de conception. Il propose une possibilité de capitaliser des savoir-faire sur les techniques d'analyse de l'observation d'une session d'apprentissage. Par contre, UTL ne dispose pas de moyens pour spécifier formellement la façon de calculer l'indicateur à partir des traces collectées. Les indicateurs décrits par UTL ne peuvent donc pas être automatisés sans une modification importante du langage.

En ce qui concerne également la réutilisation des indicateurs, le travail de Gendron [Gendron (2010)] porte sur l'amélioration des activités collaboratives par la mise en place d'indicateurs. Cette approche est intéressante, parce qu'elle permet une modélisation qui prend en compte la réutilisation, la personnalisation et l'adaptation des indicateurs en fonction des utilisateurs

et/ou du contexte de l'activité. Un des points forts de cette approche est de gérer différentes versions des règles de calcul d'un indicateur. C'est-à-dire que ce travail considère que les règles de calcul d'un indicateur peuvent évoluer au fil du temps, il permet donc d'enregistrer ces évolutions. Les traces collectées sont transformées en un format général (comme le schéma XML proposé par [May (2009)], ce qui est similaire au format unifié étudié dans le projet CAViCoLA) avant d'y appliquer les règles de calcul permettant de produire les valeurs d'un indicateur. Le calcul d'indicateurs est donc indépendant du format de traces générées par le dispositif d'apprentissage. La partie de patron d'un indicateur est exécutée pour produire ses valeurs. Cela est fait par un outil d'analyse basé sur le système à base de règles. Cette approche propose également un format général pour la représentation des indicateurs. Le format général de trace proposé avec cinq éléments (l'action, la ressource, l'acteur, la date et la valeur) peut être pertinent dans le contexte des activités collaboratives, mais il ne permet pas de décrire les traces comprenant plus que ces cinq éléments. Les trois parties pour la description d'un indicateur ne disposent pas d'une description de méthode de calcul d'indicateurs compréhensible ou interprétable par l'enseignant/tuteur. Les indicateurs considérés dans cette approche se limitent aux types de données numériques, alphanumériques, de tableau et d'intervalle discret. Cela peut s'avérer insuffisant pour modéliser par exemple des indicateurs dont le type de données est une structure avec plusieurs champs.

En ce qui concerne également la réutilisation des savoir-faire en analyse de traces, le travail de [Diagne (2009)] propose une approche de patron d'indicateur réutilisable pour la capitalisation et la réutilisation d'indicateurs. Nous nous intéressons à certains points positifs de ce travail. Premièrement, la question de la réutilisation des indicateurs est traitée par la proposition d'un patron d'indicateurs réutilisables, ce qui est proche des patrons proposés par le projet DPULS. Cette approche décrit en détail des solutions (dans le même sens que le projet DPULS) et la façon d'implémenter ces solutions par la définition de fonction d'indicateur. Deuxièmement, ce travail propose un environnement multi-agents (EM-AGIIR) permettant d'exécuter ces solutions pour les réutiliser à partir de nouvelles sources de traçage, ce qui n'est pas résolu dans le projet DPULS. Cet outil permet également à l'enseignant/tuteur de définir ou de modifier lui-même ses indicateurs à n'importe quel moment. Finalement, ce travail utilise aussi une approche d'analyse de traces par l'hypothèse. Par contre, les patrons sont proposés pour un contexte particulier : la supervision de l'apprentissage. Cette approche considère que la sélection des traces nécessaires pour calculer un indicateur ne constitue pas la méthode d'analyse pour obtenir l'indicateur. La réutilisation des méthodes de calcul des indicateurs ne prend donc pas en compte cette sélection. Cette sélection est donc toujours dépendante du format de traces

généré par des EIAH. Le format des traces et des indicateurs ne sont également pas explicitement décrits. Les indicateurs sont réutilisés dans des situations d'apprentissage similaires, mais cette approche ne permet pas de réutiliser une partie de la méthode de calcul d'indicateurs, ce qui est traité dans le travail de [Djouad et al. (2009)].

L'approche de calcul d'indicateurs de [Djouad et al. (2009)] se base sur les modèles de traces SBT. Nous estimons que l'approche SBT [Settouti et al. (2006)] propose plusieurs points intéressants. Le système SBT permet l'intégration de la plupart des traitements sur les traces (collecte, gestion, transformation, visualisation, etc.). Ces traitements de SBT sont proches de ceux du système Tatiana [Dyke et al. (2010)]. Ce dernier permet également aux chercheurs de créer, d'exploiter, de transformer, d'analyser, de visualiser et de synchroniser les *rejouables* (un *rejouable* est une trace particulière qui peut être transformée, visualisée, enrichie et synchronisée). Mais Tatiana repose sur les *rejouables*, tandis que SBT manipule les traces modélisées. SBT peut fonctionner avec plusieurs traces aux différents formats, provenant de plusieurs sources (par exemple fichier de log, vidéo, etc.) comme l'outil ColAT [Avouris et al. (2005)]. C'est un des objectifs auxquels plusieurs travaux de recherche concernant l'analyse de traces dans les EIAH s'intéressent. Dans SBT, chaque trace est associée à un modèle de traces exprimé explicitement. Cela facilite la modélisation de plusieurs types de trace, ce qui est limité dans la proposition de [May (2009)] et [Gendron (2010)]. Nous partageons ce point de vue. Une autre proposition dans SBT [Settouti et al. (2009)] est également intéressante. C'est le langage de l'interrogation et la transformation des traces modélisées. Cependant, il s'agit d'une spécification formelle du langage, ce dernier n'est pas encore implémenté. Il est donc difficile d'évaluer les possibilités du langage dans la formalisation du calcul d'indicateurs.

Par contre, le travail de [Djouad et al. (2009)] permet de calculer les indicateurs par des transformations automatiques [Settouti et al. (2007)] sur les traces modélisées. Nous nous intéressons à ce travail sur trois points de vue. Premièrement, il s'agit de proposer une méthode générique pour la modélisation du calcul d'indicateurs. Deuxièmement, ces transformations sont réalisées indépendamment du format de traces générées par le dispositif d'apprentissage. Troisièmement, toutes les transformations associées à un indicateur et les modèles de traces produits par ces transformations sont capitalisés. Cela permet donc de réutiliser ces transformations pour calculer un nouvel indicateur dans le même contexte ou dans différents contextes. Par contre, en ce qui concerne le processus de calcul des indicateurs, nous pensons que trois types de transformation (sélection, réécriture, fusion) et deux opérateurs (compter, filtrer) ne sont pas suffisants pour le calcul d'indicateurs. Certains indicateurs, comme par exemple les indicateurs pour le tutorat dans

notre contexte (nous présentons ces indicateurs dans le chapitre 7), nécessitent des traitements ou des combinaisons complexes sur tout ou une partie du contenu des traces. Dans notre contexte, les indicateurs sont utilisés pour la réingénierie du scénario pédagogique ou pour le tutorat. Nous pensons donc qu'il est nécessaire d'associer chaque indicateur aux concepts du scénario et aux objectifs d'exploitation de la donnée pour faciliter l'amélioration du scénario ainsi que la réalisation d'intervention en session.

En ce qui concerne également la collecte, le stockage, la visualisation et l'exploitation des traces comme SBT, [Broisin et Vidal (2007)] proposent un modèle UML générique pour structurer les traces produites par les activités explicites des utilisateurs dans un système d'apprentissage instrumenté par le Web. Ces auteurs ont développé une architecture associée à ce modèle pour superviser les activités apprentissage des apprenants. L'utilisateur peut exprimer ses indicateurs par les requêtes CQL (CIM Query Language). Cependant, cette approche ne dispose pas d'une méthode de modélisation générique du calcul d'indicateur ainsi que d'une méthode de capitalisation du savoir-faire des utilisateurs dans l'expression des indicateurs. Ce problème n'est également pas considéré dans le système Tatiana [Dyke et al. (2010)].

2.3.5 Les travaux sur la capitalisation et la réutilisation des savoir-faire en analyse de traces

Depuis ces dernières années, le concept de capitalisation et de réutilisation de savoir-faire ou d'expérience est devenu un thème de recherche dans plusieurs domaines, notamment dans le domaine du Génie de Production [Tichkiewitch (2008), Bissay et al. (2008), Ruet (2002)]. Les travaux concernant la capitalisation et la réutilisation portent souvent sur l'utilisation de patrons de conception et l'approche de raisonnement (à partir de cas, à partir de règles).

Dans les EIAH, plusieurs travaux/projets portent sur ce thème. En fait, les projets PPP, E-LEN, DPULS, UTL dans le projet REDiM que nous avons présentés ci-dessus ont également utilisé la solution du patron de conception pour le problème de la capitalisation et de la réutilisation de savoir-faire concernant l'enseignement et l'apprentissage. Mais ces patrons ne sont pas automatisés. Cela est considéré dans le projet TeTraKap [Quénu-Joiron et Condamines (2009)] et le travail de [Hadj M'tir et al. (2008)] en utilisant l'approche basée sur le Raisonnement à Partir de Cas.

L'approche basée sur le Raisonnement à Partir de Cas *résout les nouveaux problèmes par l'adaptation des solutions qui ont été utilisées pour résoudre des problèmes passés* [Riesbeck et Schank (1989)]. Elle permet de développer des outils intelligents facilitant la

capitalisation et la réutilisation des connaissances. Un système de raisonnement à partir de cas comprend une base de cas, un système d'indexation pour organiser les cas, un algorithme qui assure un ‘‘*matching*’’ par exemple la méthode du voisin le plus proche et une méthode d'adaptation.

Le projet TeTraKap (TEacher TRAIning by Knowledge cAPitalization) a pour objectif de concevoir, de développer un système d'apprentissage basé sur le web visant à la formation continue des professeurs des écoles. Ce projet s'intéresse particulièrement à la capitalisation des savoir-faire d'enseignants experts, comme par exemple la gestion de la classe. Ces connaissances sont ensuite partagées et réutilisées entre les enseignants novices et experts dans un but de formation. Les savoir-faire sont capitalisés dans une base de connaissances sous forme de cas. Pour faciliter la réutilisation des savoir-faire, l'approche basée sur le Raisonnement à Partir de Cas est utilisée et intégrée au système d'apprentissage qui dispose d'un espace personnalisé pour chaque enseignant. Ce dernier peut exploiter la base de cas pour y rechercher des informations. Ce système dispose également d'un forum permettant aux enseignants de résoudre des problèmes. Si un problème est résolu, la solution peut être capitalisée dans la base sous forme d'un nouveau cas.

[Hadj M'tir et al. (2008)] utilisent également l'approche basée sur le Raisonnement à Partir de Cas pour aider l'apprenant à réutiliser des expériences d'apprentissage dans un système d'apprentissage coopératif. Ces auteurs considèrent la capitalisation et la réutilisation des expériences d'apprentissage pour améliorer le système d'apprentissage coopératif. La capitalisation consiste à modéliser la situation d'apprentissage d'un apprenant donné. Ce modèle comprend le profil de l'apprenant ainsi que les caractéristiques du processus d'apprentissage. La réutilisation porte sur l'exploitation des expériences passées afin d'offrir à l'apprenant courant la meilleure expérience adaptée à ses besoins.

Ces travaux sont pertinents dans le contexte donné. Ils proposent des solutions et des outils permettant la capitalisation et la réutilisation des savoir-faire des enseignants et des apprenants dans les EIAH. Ces deux travaux [Quénu-Joiron et Condamines (2009)] et [Hadj M'tir et al. (2008)] considèrent particulièrement la capitalisation de l'expérience en réalisation d'une tâche, par exemple la gestion d'une classe ou la pédagogie. Ces expériences ne nécessitent pas de techniques informatiques ou d'algorithmes complexes à implémenter pour réaliser une tâche. Ces savoir-faire sont similaires à ceux de l'enseignant dans la spécification de ses besoins d'observation. Tandis que notre travail s'intéresse à la proposition d'une méthode de modélisation du calcul d'indicateurs. Cette méthode doit être réalisée sous une forme capitalisable, automatisable et réutilisable des savoir-faire en

analyse d'une session pour produire les valeurs des indicateurs répondant aux besoins d'observation de l'enseignant.

2.4 Conclusions

Nous avons présenté et discuté ci-dessus plusieurs travaux concernant la modélisation et le calcul des indicateurs. Nous considérons ces travaux selon deux questions :

1. Est-ce que ces travaux proposent une méthode de modélisation du calcul d'indicateurs dans une forme capitalisable, réutilisable et automatisable ?

2. Est-ce que ces travaux permettent à l'enseignant/concepteur d'exprimer ses besoins d'observation, ce qui est utilisé pour formaliser des indicateurs ?

Nous constatons que plusieurs travaux considèrent le calcul d'indicateurs notamment les indicateurs concernant la communication médiatisée ou l'activité d'interaction. La plupart des travaux visent la proposition d'indicateurs spécifiques pour l'enseignant sans volonté explicite de réutilisation des méthodes de modélisation d'indicateurs. Peu de travaux permettent à l'enseignant d'exprimer ses indicateurs ainsi que de capitaliser et de réutiliser des méthodes de modélisation d'indicateurs pédagogiques pour la réingénierie et le tutorat. Nous pensons que la formalisation d'un indicateur dépend en général de deux problèmes. Le premier concerne l'expression des besoins d'observation des enseignants, comme par exemple la relation entre ces besoins et les concepts du scénario. Le deuxième porte sur la méthode de calcul d'indicateur. Dans notre contexte, nous pensons que la capitalisation et la réutilisation du savoir-faire devraient porter sur ces deux points. C'est-à-dire qu'ils permettent de capitaliser le savoir-faire de l'enseignant en utilisant ses besoins d'observations en session d'apprentissage et le savoir-faire dans la modélisation du calcul des indicateurs. Nous nous intéressons à ces deux points dans notre travail.

3 Problématique et approche méthodologique

Sommaire

A partir de l'analyse précédente, un élément clé qui semble émerger porte sur la modélisation du calcul d'indicateurs dans une forme capitalisable, réutilisable et automatisable. Nous avons déjà souligné précédemment dans ce document que la capitalisation et la réutilisation du savoir-faire devraient porter sur deux points que nous rappelons ici :

- Le savoir-faire des enseignants/concepteurs dans l'expression de leurs besoins d'observation.
- Le savoir-faire des analystes dans la formalisation des méthodes de calcul d'indicateurs dans une forme exécutable.

Plusieurs travaux ont porté sur la capitalisation du savoir-faire des enseignants et du calcul d'indicateurs. Néanmoins, nous pensons que ces travaux ne sont pas toujours satisfaisants par rapport aux deux points ci-dessus. Dans ce chapitre nous commençons par dresser un bilan des travaux présentés dans le chapitre précédent et indiquons les raisons de cette insatisfaction. Ensuite, nous définissons notre problématique. Nous présentons nos objectifs de recherche en nous appuyant sur le positionnement scientifique. Ce chapitre sera conclu par la méthodologie de recherche adoptée.

3.1 Définition de la problématique

Comme nous l'avons annoncé dans le chapitre 1, notre travail porte sur l'analyse de traces collectées en session par des EIAH, plus précisément sur la modélisation du calcul d'indicateurs. Cette dernière a pour objectif de produire des indicateurs qui sont utilisés par l'enseignant/concepteur pour l'aider dans la réingénierie d'un scénario pédagogique, mais aussi dans son activité de tutorat en session d'apprentissage.

Il est nécessaire d'intégrer une phase d'analyse des usages dans le processus de développement d'un EIAH pour aider les enseignants à comprendre la qualité de la situation pédagogique mise en œuvre. L'enseignant/concepteur devrait donc être étroitement associé à la phase d'analyse [Choquet et Iksal (2007b)]. Les résultats du processus d'analyse doivent être utilisés et interprétés par l'enseignant/tuteur. Nous avons présenté dans le chapitre 2 plusieurs travaux de recherche concernant l'observation et l'analyse des activités d'interaction d'une session d'apprentissage en se basant sur les traces générées dans des EIAH. Plusieurs approches et outils sont proposés par ces travaux. Nous considérons ces approches et ces outils sur trois critères : (1) l'expression des indicateurs par l'enseignant, (2) la prise en compte de la capitalisation et de la réutilisation des méthodes de calcul d'indicateurs et (3) la possibilité de capitaliser les savoir-faire de l'enseignant dans l'association de ses besoins d'observation aux concepts du scénario et aux objectifs d'observation. Dans notre contexte de recherche, ces approches et ces outils n'apportent pas de solutions complètement satisfaisantes pour la modélisation de l'observation et du calcul d'indicateurs en vue de l'amélioration du scénario pédagogique et de la régulation d'activités de l'apprenant.

Premièrement, les travaux concernant l'utilisation des techniques de data-mining sont directement adressés aux chercheurs ou à des développeurs de solutions informatiques pour EIAH et non aux enseignants/tuteurs. Ce sont généralement les chercheurs ou des informaticiens qui analysent et interprètent des résultats pour découvrir des informations signifiantes. Les travaux attachés aux méthodes de data-mining sont principalement appliqués aux EIAH pour évaluer la connaissance des apprenants, non pour superviser et réguler des activités de l'apprenant.

Généralement, le groupe de travail visant notamment à la proposition d'indicateurs spécifiques n'implique pas le rôle de l'enseignant/concepteur dans le processus d'analyse des traces d'interactions d'une session d'apprentissage. Ces travaux portent souvent sur l'analyse de traces d'interactions générées par des outils de communication (chat, forum, wiki, etc.). Les indicateurs d'analyse d'interaction de ces travaux sont adressés à

l'enseignant/tuteur, mais ils ne sont pas proposés par ce dernier. Ce sont des chercheurs ou développeurs de solutions informatiques pour EIAH qui les définissent. Par conséquent, ces indicateurs peuvent ne pas tout à fait répondre aux besoins des enseignants qui jouent un rôle clé dans le processus de réingénierie. Ils sont de plus souvent développés de manière *ad hoc* pour répondre à des problèmes spécifiques d'un EIAH, leurs méthodes de calcul sont donc difficiles à réutiliser.

Par contre, les recherches concernant la modélisation des traces et/ou des indicateurs permettent à l'enseignant/tuteur ou au chercheur d'exprimer ses indicateurs. Elles sont souvent orientées vers la définition d'un modèle générique pour la gestion des traces et la réutilisation de la méthode de calcul des indicateurs. La modélisation du calcul d'indicateurs de certains de ces travaux comme celui de [Broisin et Vidal (2007)] est réalisée par l'utilisateur (l'enseignant/tuteur ou chercheur) et n'est pas capitalisée. Les autres travaux dans ce groupe sont centrés sur la capitalisation de la méthode de calcul des indicateurs. Mais ces travaux ne prennent pas en compte le problème de la capitalisation du savoir-faire de l'enseignant dans l'association de ses besoins d'observation aux concepts du scénario et aux objectifs d'observation.

Dans un contexte de réingénierie, le projet REDiM du LIUM a proposé UTL dédié à la modélisation des indicateurs ainsi qu'à toute donnée impliquée dans la modélisation de l'observation et à leur capitalisation sous une forme proche des patrons de conception, ce qui permet de réutiliser le savoir-faire de la communauté en analyse et en observation d'une session. Il est conçu pour l'homme et est prévu aussi pour la machine. C'est-à-dire que l'enseignant/concepteur peut le comprendre et l'utiliser pour l'expression de ses besoins et qu'il peut être exécuté par la machine. Mais la version d'UTL [Choquet et Iksal (2007a)] existante avant nos travaux n'est pas encore exécutée par la machine. Elle permet par contre une description structurée des indicateurs (des données sont formalisées à partir des besoins d'observation des enseignants) dans une forme indépendante des formats de traces générées par un dispositif d'apprentissage. Elle permet aussi d'associer chaque besoin d'observation aux objectifs d'observation et aux concepts du scénario.

A partir de l'analyse ci-dessus, nous nous focalisons sur les éléments essentiels pour lesquels nous proposons notre approche :

- Dans notre contexte de réingénierie et de tutorat, ce sont (1) les enseignants/concepteurs qui apportent des améliorations au scénario pédagogique qu'ils ont conçu, et (2) les enseignants/tuteurs qui régulent les activités des

apprenants. Ils sont impliqués dans le processus de modélisation de l'observation et de l'analyse des traces d'une session d'apprentissage. Donc le résultat de ce processus devrait être adressé directement aux enseignants/tuteurs et non pas à un chercheur de la communauté EIAH ou à un développeur de solutions informatiques pour EIAH. Ce sont les enseignants/concepteurs qui définissent leurs besoins d'observation, qui interprètent et utilisent les résultats de ce processus d'analyse. Ces besoins d'observation sont utilisés par l'analyste pour modéliser les indicateurs, qui doivent être adressés à l'enseignant.

- Comme nous l'avons présenté dans le chapitre 1, le processus de conception et de réingénierie d'un scénario pédagogique dans notre contexte est un processus itératif. La conception du scénario est progressive [El-Kechaï (2008)] et adaptable pendant la session [Ouraiba et al. (2010)]. Donc, les indicateurs doivent pouvoir être calculés pendant et/ou après une session.
- Les connaissances en modélisation ainsi qu'en définition des indicateurs doivent pouvoir être capitalisées. Ceci facilite la réutilisation du savoir-faire en analyse et observation d'une session d'apprentissage.
- Ces indicateurs devraient être formellement exprimés pour qu'ils puissent être calculés automatiquement.

Au regard de ces spécificités, le travail que nous poursuivons dans cette thèse porte donc sur l'extension d'UTL concernant la méthode de calcul d'indicateurs. Cette méthode doit être explicite, formelle et exécutable. Notre travail devra donc répondre à la question suivante qui résume notre problématique : *à partir d'UTL, comment fournir des indicateurs aux enseignants pendant ou après une session pour qu'ils puissent évaluer leurs besoins d'observation afin d'apporter des améliorations au scénario pédagogique qu'ils ont conçu, ainsi qu'afin de réguler les activités des apprenants en session ?*

Ceci nous amène à répondre aux questions sous-jacentes suivantes :

1. Comment intégrer une méthode de modélisation formelle du calcul dans les indicateurs UTL ?

2. Comment automatiser ces indicateurs ?

Autrement dit, ***nous voulons proposer un langage permettant de modéliser la méthode de calcul d'indicateurs complexes, d'intégrer des outils externes (qui proviennent d'autres outils d'analyse) et de faciliter la génération de l'outil d'analyse pour calculer les indicateurs modélisés. La description de ces méthodes de calcul doit être***

représentée dans une forme capitalisable. Les moyens (le langage et l'outil d'analyse) que nous proposons ont pour but d'aider l'enseignant/concepteur à bien comprendre la lecture et l'écriture de méthodes de modélisation et de calcul d'indicateurs. La modélisation du calcul d'indicateurs est une tâche compliquée et qui nécessite des compétences informatiques. Elle est donc hors de portée de l'enseignant non informaticien. Puisque nous ne pouvons pas directement instrumenter l'enseignant avec un langage, nous allons proposer un langage qui permet aux analystes de décrire les méthodes de calcul d'indicateurs de manière capitalisable et réutilisable. Ce qui va permettre dans un deuxième temps à l'enseignant de réutiliser des indicateurs.

À partir de cette problématique, notre travail se base sur l'hypothèse suivante : UTL peut être enrichi pour permettre de décrire formellement la méthode du calcul d'indicateurs, ce qui permet alors la production automatisée de ces indicateurs.

3.2 Objectifs de notre travail

Comme nous l'avons énoncé dans la section 1.3, le troisième axe de recherche du projet REDiM est de définir les moyens d'observation d'une situation d'apprentissage afin d'adresser aux enseignants/concepteurs des indicateurs signifiants. C'est notre objectif général. En ce qui concerne notre objectif opérationnel, nous étudions comment réaliser un outil d'analyse des traces générées par les EIAH permettant aux enseignants/concepteurs d'utiliser une banque d'indicateurs pédagogiques afin d'étudier et d'observer une session d'apprentissage, comme par exemple de faire émerger les différences entre le scénario d'apprentissage prescrit et ceux qui ont été observés lors de la session. Plus précisément, l'outil devra être capable :

- D'exécuter les indicateurs modélisés à partir des besoins d'observation définis par les enseignants/concepteurs portant sur le déroulement d'une situation d'apprentissage dans les EIAH. Cette exécution facilite l'interprétation par les enseignants/concepteurs de ce qui s'est passé pendant une session. Elle leur permet d'améliorer éventuellement leurs scénarios pédagogiques pour la session suivante ainsi que de réaliser des actions tutorales appropriées, d'aider à la supervision de session, de fournir des conseils aux apprenants pendant ou après une session d'apprentissage.
- De permettre le partage et la réutilisation entre les enseignants/concepteurs de leurs compétences et de leurs expériences dans l'observation d'une session

d'apprentissage. Le processus de définition de leurs besoins d'observation doit permettre aux enseignants/concepteurs de consulter, puiser ou réutiliser les indicateurs définis dans différentes situations d'apprentissage. Pour cela, ces indicateurs doivent être capitalisés dans une base d'indicateurs.

- D'exécuter de nouveaux indicateurs ou de ré-exécuter des indicateurs prédéfinis à tout moment pendant la session ou pendant l'utilisation de l'outil.
- D'intégrer et d'exécuter certaines fonctions externes qui sont définies dans d'autres outils d'analyse ou dans notre processus de réingénierie d'un scénario pédagogique.

3.3 Approche méthodologique

Comme nous l'avons exprimé, nous ne proposons pas les indicateurs. L'enseignant définit ses besoins d'observation, qui sont utilisés ensuite pour formaliser les indicateurs. Notre formalisme devra répondre à ces besoins d'observation.

Dans notre contexte, le sujet cible du processus de réingénierie d'un scénario pédagogique et de l'action tutorale est l'enseignant/concepteur. Le processus d'analyse du déroulement d'une situation d'apprentissage démarre à partir des besoins d'observation des enseignants et a pour but de leur adresser les résultats de ce processus. De plus, nous prenons également en compte une description formelle du calcul des indicateurs pour que les indicateurs puissent être automatisés. Nous nous inscrivons donc dans une démarche qui allie une approche langage pour la description formelle du calcul d'indicateurs et une approche d'analyse des données par l'hypothèse pour produire les valeurs des indicateurs à partir des traces. Dans cette optique, UTL est considéré comme un cadre théorique et technologique sur lequel nos propositions s'appuient. Notre méthodologie de recherche est illustrée dans la figure 3-1.

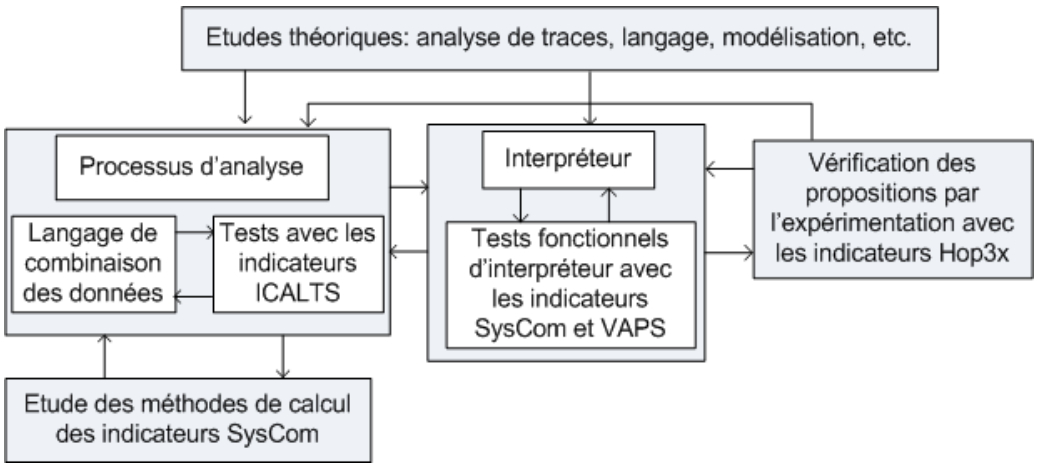

Figure 3-1 La méthodologie de recherche

Dans notre contexte de travail, nous avons disposé des traces collectées par un environnement réalisé au laboratoire SysCom et des indicateurs considérés sur ces traces [Iksal et al. (2007)]. Nous avons donc étudié la description formelle des méthodes de calcul de ces indicateurs. A partir de cette étude, nous avons proposé le langage de combinaison des données UTL pour produire les indicateurs. Ce langage est nommé DCL4UTL. Il est utilisé dans le processus d'analyse de traces. Ce dernier est présenté dans le chapitre suivant de ce document.

Afin de valider notre langage, nous avons sélectionné et modélisé des indicateurs connus présentés dans le projet ICALTS en utilisant DCL4UTL. Avec cette modélisation, le langage DCL4UTL a été enrichi. Nous avons ensuite développé un interpréteur pour le langage DCL4UTL. Cet interpréteur a été intégré dans un outil d'analyse de traces. Ce dernier permet de transformer les traces, de calculer et de capitaliser les indicateurs, de consulter les indicateurs et les données nécessaires pour les calculer. La figure 3-2 illustre le processus de réalisation du calcul des indicateurs par l'interpréteur. Une description des moyens d'observation des besoins est tout d'abord écrite selon les règles de la grammaire du langage DCL4UTL. Cette description est ensuite analysée afin de déceler d'éventuelles erreurs. Enfin, cette description est exécutée. Dans l'étape d'exécution présentée dans la figure 3-2, nous avons adopté l'approche d'analyse des données dirigée par l'hypothèse. Le processus d'analyse de traces commence par les besoins d'observation des enseignants. Il exploite les traces pour trouver des données appropriées répondant aux besoins et adresse ces données à l'enseignant, qui interprète les résultats.

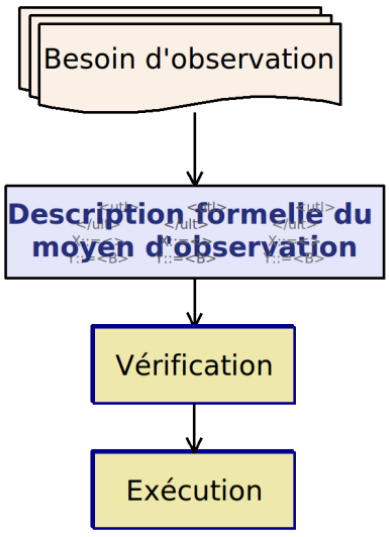

Figure 3-2 Le processus de réalisation de la description formelle des moyens d'observation

Pour éprouver le langage et l'outil sur la réponse aux besoins d'observation de l'enseignant, nous avons mené le calcul de plusieurs indicateurs après la session. Dans le domaine EIAH, nous avons testé notre langage avec les traces d'un environnement réalisé au laboratoire SysCom. Nous avons élargi notre contexte dans d'autres domaines que les EIAH. Pour cela nous avons utilisé les traces générées par le système VAPS (Virtual Action Planning Supermarket) pour tester le calcul d'indicateurs avec le langage DCL4UTL. Nous présentons plus loin dans ce document ces systèmes.

Puis, nous avons élargi notre contexte pour prendre en compte le calcul d'indicateurs en temps réel. Nous avons intégré cette fonctionnalité dans le langage DCL4UTL. Afin d'effectuer une vérification de nos propositions, nous avons travaillé avec un enseignant/concepteur de l'Université du Maine pour spécifier ses besoins d'observation et avec deux tuteurs de l'IUT de Laval (Université du Maine) pour superviser des sessions d'apprentissage. Nous avons utilisé la plate-forme d'apprentissage Hop3x pour collecter des traces d'interactions. Nous avons formalisé les indicateurs à partir des besoins d'observation de l'enseignant. Ces indicateurs ont été calculés en temps réel pour les adresser aux tuteurs. Cette expérimentation est présentée à la fin de ce document.

Notre démarche consiste donc à (1) indiquer un processus d'analyse des traces impliquant l'enseignant/concepteur, (2) concevoir un langage dédié à la spécification formelle des méthodes de calcul d'indicateurs, (3) construire un outil pour ce langage et (4) valider notre proposition. Dans le reste de ce document, nous détaillons les différentes étapes de la démarche que nous venons de présenter. Tout d'abord, nous décrivons le processus d'analyse des traces dans le contexte du projet REDiM. Nous présentons le langage dédié pour la description formelle des moyens d'observation appelé le langage de combinaison de données pour UTL : depuis l'analyse d'utilisation jusqu'à l'implémentation d'un interpréteur. Nous présentons les différentes applications et expérimentations que nous avons réalisées afin de valider nos propositions.

4 Processus d'analyse de traces

Sommaire

Notre travail porte sur le processus d'analyse de traces du cycle de réingénierie du scénario pédagogique et de l'action tutorale. Dans cette partie, nous définissons le processus d'analyse de traces collectées en session d'apprentissage mis en place dans le contexte du projet REDiM. Il s'agit d'un ensemble d'activités interactives qui transforment les traces et les besoins d'observation de l'enseignant en indicateurs. Ce processus a pour but de faciliter la modélisation et le calcul d'indicateurs. Il comprend quatre processus : (1) la modélisation de données, (2) le prétraitement de données, (3) la combinaison de données traitées pour produire un ou plusieurs indicateurs et (4) la capitalisation et la représentation des valeurs des indicateurs aux enseignants/concepteurs.

4.1 Introduction

Comme nous l'avons abordé dans le chapitre 1, le processus de conception et de réingénierie d'un scénario pédagogique est divisé en trois étapes : la conception d'un scénario pédagogique avant une session, la mise en place du scénario dans une plate-forme et l'analyse des traces d'utilisation pendant ou après une session d'apprentissage. Notre travail se centre précisément sur l'analyse des traces dans un contexte de réingénierie, mais aussi dans le cadre de l'aide à l'action tutorale. Notre objectif est d'adresser à l'enseignant/concepteur les indicateurs calculés pour l'aider à observer et à évaluer une session d'apprentissage afin d'améliorer éventuellement son scénario pédagogique ou de réguler l'activité de l'apprenant. Un processus d'analyse de trace est en général divisé en quatre étapes comme celles proposées par [Romero et Ventura (2007)] et [Dimitracopoulou (2004b)] qui sont présentées dans le chapitre 2. Le processus d'analyse de traces que nous considérons est proche de ces processus. La différence porte notamment sur la méthode de modélisation et la capitalisation des données. Notre approche se focalise sur la méthode de modélisation des données (à partir des données brutes jusqu'aux indicateurs) et plus précisément sur la méthode de modélisation du calcul d'indicateurs. Il se centre également sur la capitalisation des méthodes de calcul ; mais aussi des données produites par ces méthodes. Il se compose de quatre processus : (1) la modélisation de données, (2) le prétraitement de données, (3) le calcul, c'est-à-dire la combinaison des données traitées pour produire un ou plusieurs indicateurs et (4) la capitalisation puis la représentation des valeurs des indicateurs pour les enseignants/concepteurs. C'est l'enseignant/concepteur qui utilise et interprète ces valeurs. Il joue donc un rôle important dans la conception du scénario, mais aussi dans l'analyse de traces. Dans ce chapitre, nous présentons d'abord les rôles participant au processus d'analyse de traces, ensuite ce processus est décrit en détail.

4.2 Les rôles participant au processus d'analyse des traces

Comme nous l'avons présenté dans le chapitre 3, un de nos objectifs est de fournir des indicateurs signifiants à l'enseignant/concepteur pour l'aider à étudier une session d'apprentissage. L'enseignant/concepteur joue donc un rôle clé dans le processus de réingénierie. Nous considérons que le processus d'analyse de traces implique la collaboration entre trois rôles : l'enseignant/concepteur, l'analyste et le développeur. La figure 4-1 montre les rôles considérés dans ce processus.

L'enseignant/concepteur. Dans la situation de conception avant la session, l'enseignant/concepteur définit son scénario pédagogique. Pour l'aider à comprendre ce qui se passe pendant la session, il lui est également demandé d'indiquer ce qu'il veut observer

pendant la session sous la forme de besoins d'observation [Zendagui et al. (2008)]. L'enseignant/concepteur doit aussi spécifier les informations qu'il souhaite obtenir dans le résultat une fois que ces besoins ont été évalués.

En fait, la définition des besoins d'observation est également un domaine de recherche [Zorrilla et al. (2010)], [Zendagui et al. (2008)], [Gendron (2010)], etc. La spécification des besoins d'observation dépend de la capacité et de l'expérience de l'enseignant/concepteur. Dans notre contexte d'étude, nous ne considérons donc pas ces problèmes, ni de quelles façons l'enseignant/concepteur définit ses besoins. Nous nous intéressons au calcul d'indicateurs formalisés à partir de ces besoins.

Une fois que les indicateurs sont calculés, leurs instances sont produites pendant ou après la session d'apprentissage. L'enseignant/tuteur utilise ces instances selon ses objectifs comme, par exemple l'amélioration du scénario pédagogique, l'évaluation d'une session d'apprentissage, le tutorat, etc.

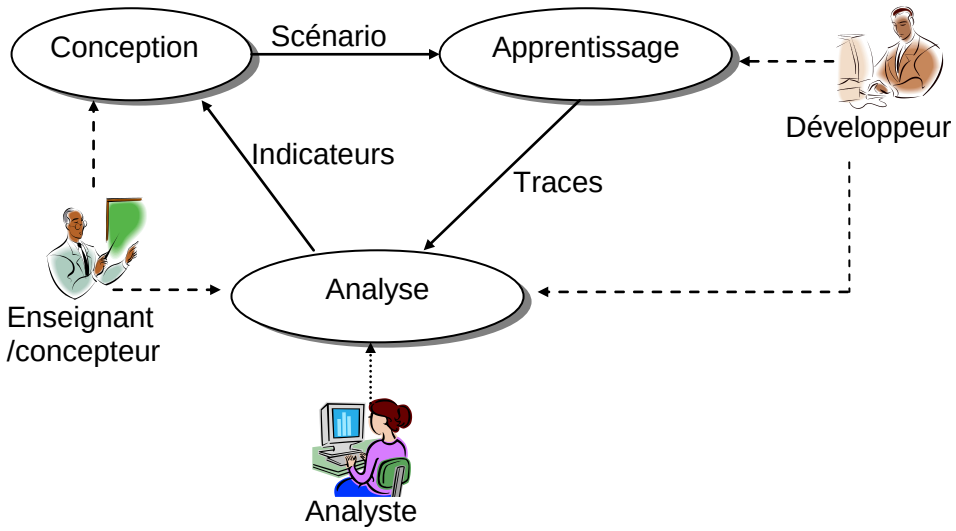

Figure 4-1 Les rôles concernés dans le cycle de conception et de réingénierie d'un scénario pédagogique

L'analyste. Un analyste est un spécialiste de l'analyse de données qui possède des compétences informatiques (par exemple un ingénieur d'étude). L'analyste joue le rôle d'un assistant qui aide l'enseignant/concepteur dans la transformation de ses besoins en indicateurs calculables par la machine. Avant la session, l'analyste formalise les indicateurs avec un langage de modélisation à partir des besoins d'observation définis par l'enseignant/concepteur. Il spécifie également la méthode de calcul d'indicateurs avec un langage formel, un langage de requête ou un langage de programmation afin de pouvoir automatiser l'obtention des valeurs.

Comme nous l'avons abordé ci-dessus, l'enseignant/concepteur spécifie les informations qu'il souhaite obtenir dans le résultat une fois que l'indicateur est calculé. Ensuite, l'analyste exploite ces informations afin de créer le format des résultats de l'indicateur (par exemple en XML). Il utilise l'élément *Using.data.format* (voir la figure 2-11) de l'indicateur. Il doit aussi définir les modèles de données primaires qui seront ensuite utilisés pour transformer les traces et les autres données générées par la plate-forme sous la forme de données primaires UTL.

Le développeur. En ce qui concerne le déroulement de la situation d'apprentissage, le scénario pédagogique est mis en œuvre dans une plate-forme. Cette dernière génère des traces d'interaction concernant l'activité de l'apprenant en session d'apprentissage. Un développeur est un informaticien qui est responsable du déploiement d'un environnement d'apprentissage selon le scénario pédagogique et les besoins d'observation de l'enseignant. Par exemple, il peut ajouter des sondes dans le dispositif d'apprentissage pour répondre aux exigences du scénario d'observation.

Dans notre contexte, nous considérons que la méthode de calcul d'indicateurs peut être formalisée en utilisant des fonctions externes. Ces dernières peuvent provenir d'autres outils existants (data mining par exemple). L'utilisation de ces fonctions externes permet d'éviter la déclaration des fonctions trop complexes. Le développeur peut également être amené à produire des fonctions externes nécessaires à la formalisation d'indicateurs.

4.3 Le processus d'analyse des traces

Dans notre contexte de réingénierie et de tutorat, nous considérons qu'un processus d'analyse de traces se compose des activités produisant les indicateurs à partir des traces de l'EIAH et des besoins d'observation de l'enseignant. Ce processus a pour but de faciliter les tâches concernant la modélisation et le calcul d'indicateurs. Nous groupons ces activités en un ensemble de processus qui se compose du processus de modélisation de données, du processus de prétraitement de données, du processus de calcul et du processus de capitalisation de données [Pham-Thi et al. (2009b)]. Ce regroupement est réalisé selon les tâches incluses dans chaque processus. La modification interne d'un processus n'a aucune influence sur le fonctionnement des autres processus. Cela facilite les modifications de chaque processus.

Le processus de modélisation de données dont toutes les tâches sont réalisées manuellement et sont nécessaires pour le fonctionnement des autres processus. Il utilise les besoins d'observation de l'enseignant et fournit les modèles de données UTL et notamment

la méthode formelle de calcul d'indicateurs. Ses tâches sont faites par l'analyste avec l'aide de l'enseignant et/ou du développeur.

Le processus de prétraitement proposé a pour objectif de construire les données brutes à partir des traces générées par l'outil externe (l'EIAH). Ce processus est réalisé automatiquement.

Le processus de calcul a pour but de générer les valeurs des indicateurs. Il est basé sur l'interpréteur du langage DCL4UTL que nous proposons. Il est réalisé automatiquement et ne dépend pas de l'EIAH qui génère les traces. L'interpréteur ne calcule que les indicateurs et les données intermédiaires. La mise en forme des valeurs et leur capitalisation ne font pas partie des fonctionnalités d'un interpréteur.

Le processus de capitalisation de données transforme le résultat du calcul de façon à le rendre conforme au format de la donnée dérivée et stocke ces données dans une base de données UTL. Ce processus stocke également les productions des processus de modélisation de données et de prétraitement de données. C'est pourquoi ce processus ne peut pas être associé directement au processus de calcul.

Nous regroupons tous ces processus dans un processus que nous désignons par ''*processus d'analyse*''. Ce processus est représenté par la figure 4-2. Dans cette figure, chaque processus est représenté par un paquetage. Un processus produit et consomme des artefacts. Ces derniers caractérisent des flux de communication entre les différents processus. Chaque flux est représenté sous forme d'une interaction entre deux processus (un processus émetteur et un processus récepteur). Nous utilisons également ici les conventions graphiques du profil UML pour ECA (Entreprise Collaboration Architecture) présenté par le consortium OMG [OMG (2004)] pour représenter le processus.

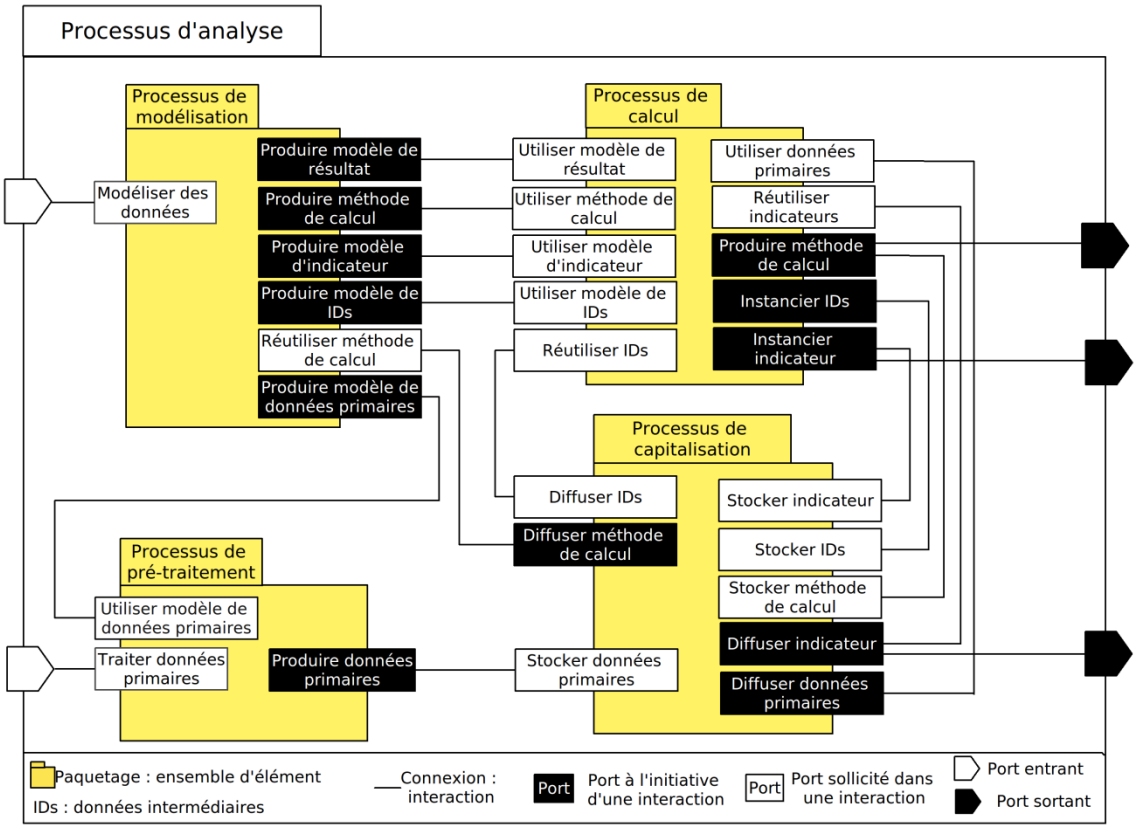

Figure 4-2 Les quatre processus du processus d'analyse des traces

4.3.1 Le processus de modélisation de données

Ce processus n'est pas spécifié explicitement dans les processus d'analyse de trace proposés par [Romero et Ventura (2007)] et [Dimitracopoulou (2004b)]. Il concerne la spécification des données nécessaires pour formaliser des indicateurs à partir des besoins d'observation de l'enseignant. Dans notre contexte, son but est de produire des modèles de données UTL. Il consiste à modéliser les données primaires UTL en prenant en compte les traces collectées en session. Ce processus comprend également la modélisation des données dérivées, en particulier des indicateurs (au sens d'UTL), en prenant en compte les besoins d'observation de l'enseignant ainsi que les autres données UTL modélisées. Cette activité est réalisée par l'analyste. Il modélise les données UTL selon les trois facettes du modèle DGU. Il doit spécifier dans quel format ces données sont stockées, cela consiste en la définition du champ *Using.format* pour chaque donnée UTL. L'analyste doit décrire comment les données brutes peuvent être obtenues à partir des traces, ce qui signifie la méthode d'acquisition de la donnée dans le champ *Getting.method* pour chaque donnée brute UTL. Les autres données primaires comme les données de production et les données additionnelles peuvent éventuellement être ajoutées. Pour les indicateurs, l'analyste utilise

un formalisme particulier permettant de spécifier formellement la manière dont les indicateurs peuvent être calculés à partir d'autres données UTL. Nous proposons le langage DCL4UTL, qui est présenté dans le chapitre suivant. Dans ce processus, le développeur peut ajouter les fonctions externes nécessaires aux différents calculs. DCL4UTL est conçu pour permettre cette intégration. Ce processus peut réutiliser des données modélisées et capitalisées (notamment les indicateurs) par le processus de capitalisation de données.

Le processus de modélisation participe donc à six interactions avec les autres processus :

Quatre productions pour le processus de calcul :

1. Des méthodes de calcul.
2. Des modèles de résultat d'indicateur.
3. Des modèles d'indicateur.
4. Des modèles de données intermédiaires.

Une production pour le processus de prétraitement :

5. Des modèles de données primaires.

Une interaction avec le processus de capitalisation :

6. Il réutilise des méthodes de calcul qui sont stockées par le processus de capitalisation.

4.3.2 Le processus de prétraitement de données

Si le processus de modélisation de données produit des modèles de données primaires UTL, ce processus a pour but de préparer les données primaires nécessaires à la production des valeurs d'indicateurs. Il permet de transformer le contenu des traces collectées (par exemple : fichiers de log, base de données, etc.) en données primaires UTL. Le contenu informationnel de la trace est stocké dans le champ *Using.data* selon le modèle de données (*Using.format*) défini dans le processus de modélisation de données. Cette transformation est réalisée par un programme qui prend en compte la méthode d'acquisition des données primaires notamment les données brutes. La figure 4-3 présente ce processus.

Le processus de prétraitement participe donc à deux interactions :

1. Il utilise des modèles de données primaires qui sont produits par le processus de modélisation.

2. Il produit des données primaires qui sont stockées par le processus de capitalisation.

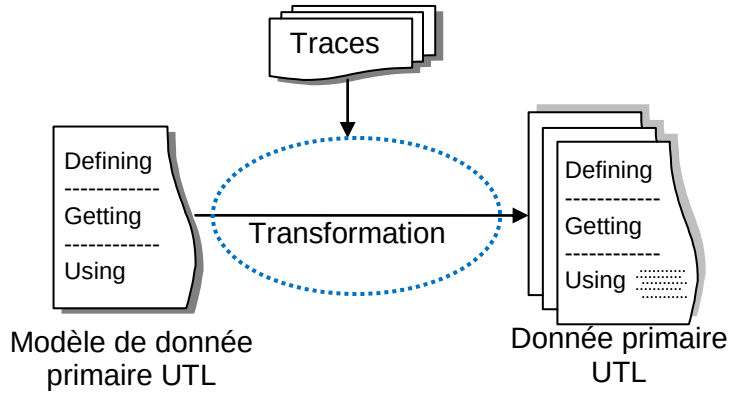

Figure 4-3 Le processus de prétraitement de données

4.3.3 Le processus de calcul d'indicateurs

Comme le montre la figure 4-4, ce processus a pour but de combiner les données primaires produites dans le processus de prétraitement de données et éventuellement les autres données dérivées pour produire les valeurs des indicateurs spécifiés dans le processus de modélisation. En général, le calcul d'un indicateur peut nécessiter des données intermédiaires. Le processus de calcul d'indicateurs génère alors les valeurs des indicateurs, mais aussi les valeurs des données intermédiaires correspondantes. Ce processus est réalisé automatiquement en utilisant la description formelle de la méthode de calcul d'un indicateur décrite par l'analyste dans le processus de modélisation.

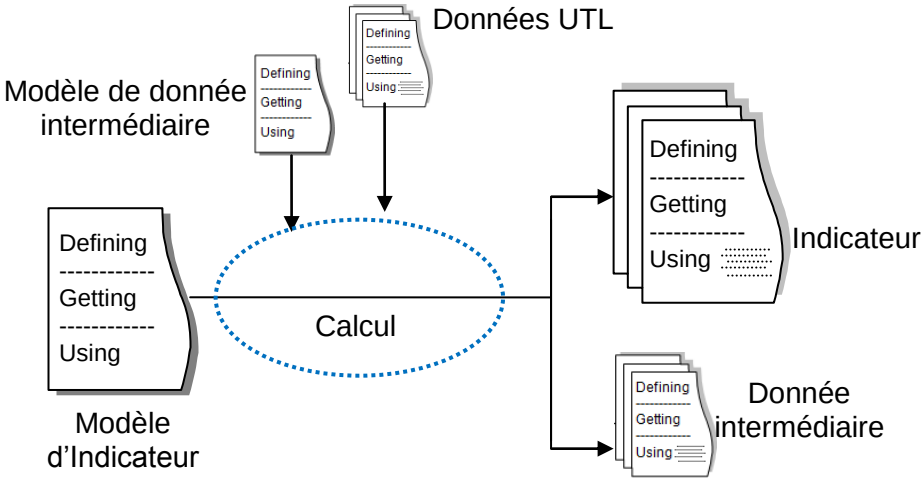

Figure 4-4 Le processus de calcul d'indicateurs

Ce processus de calcul participe donc à dix interactions :

Quatre interactions avec le processus de modélisation :

1. Il utilise les méthodes de calcul.
2. Il utilise les modèles de résultat d'indicateur.
3. Il utilise les modèles d'indicateur.
4. Il utilise les modèles de données intermédiaires.

Six interactions avec le processus de capitalisation :

5. Il utilise les indicateurs.
6. Il utilise les données intermédiaires.
7. Il utilise les données primaires.
8. Il produit les instances des indicateurs.
9. Il produit les instances des données intermédiaires.
10. Il produit des méthodes de calcul.

4.3.4 Le processus de capitalisation de données

Ce processus a pour but de capitaliser toutes les productions d'autres processus et d'adresser les valeurs des indicateurs à l'enseignant/concepteur. Pour chaque session d'apprentissage, il stocke d'abord tous les modèles de données UTL définis dans le processus de modélisation. Puis, il capitalise les instances des données primaires produites dans les processus de prétraitement. Il fournit ces données au processus de calcul pour calculer les indicateurs et capitalise les valeurs de ces indicateurs ainsi que les données intermédiaires correspondantes, dans le champ *Using.data* selon le format de stockage défini dans le processus de modélisation de données.

Ces valeurs peuvent ensuite être utilisées pour élaborer d'autres indicateurs. Ce processus permet donc de capitaliser les modèles de données UTL, leurs valeurs et leurs méthodes d'acquisition ensemble. Il fournit les données nécessaires pour le calcul d'indicateurs ainsi que les modèles de données pour la modélisation d'autres indicateurs. Les indicateurs et les définitions des données dérivées sont utilisés comme des patrons pour l'analyse d'autres sessions d'apprentissage. Ce processus de capitalisation participe donc à huit interactions :

Six interactions avec le processus de calcul :

1. Il stocke les indicateurs.
2. Il stocke les données intermédiaires.
3. Il stocke les méthodes de calcul.
4. Il diffuse les indicateurs.
5. Il diffuse les données intermédiaires.
6. Il diffuse les données primaires.

Une interaction avec le processus de prétraitement :

7. Il stocke les données primaires produites par le processus de prétraitement.

Une interaction avec le processus de modélisation :

8. Il diffuse les méthodes de calcul utilisées par le processus de modélisation.

4.4 Conclusion

Le travail que nous poursuivons dans cette thèse se focalise sur l'analyse de traces pour le calcul d'indicateurs. Ces indicateurs sont formalisés à partir des besoins d'observation de l'enseignant/concepteur et sont à destination de l'enseignant et éventuellement d'autres acteurs de la session d'apprentissage (tuteur, apprenant, etc.). Nous souhaitons mettre l'accent sur le fait que l'enseignant/concepteur joue le rôle principal dans le processus de réingénierie, mais aussi dans l'action tutorale. Nous considérons donc également ce rôle de l'enseignant dans le processus d'analyse de traces que nous avons proposé dans ce chapitre. Dans ce processus, l'enseignant/concepteur travaille en collaboration avec l'analyste et le développeur pour mettre en place un scénario pédagogique dans un EIAH. Ce processus reçoit les traces collectées en session et les besoins d'observation de l'enseignant comme entrées et les transforme en indicateurs comme sorties. Il est divisé en quatre processus : le processus de modélisation de données, le processus de prétraitement de données, le processus de calcul et le processus de capitalisation de données. Cela facilite la modification et la réalisation des activités concernant la modélisation et le calcul d'indicateurs à partir des traces, parce que toutes les tâches d'un processus sont indépendantes des tâches des autres processus. La modification interne d'un processus n'a aucune influence sur le fonctionnement des autres processus. Ces processus produisent et consomment des artefacts (les modèles de données UTL, les méthodes de calcul d'indicateur, les données UTL, etc.). C'est dans le processus de modélisation de données

qu'un indicateur est formalisé avec le langage DCL4UTL dans une forme capitalisable et réutilisable. Cet indicateur est ensuite calculé grâce au processus de calcul en utilisant les traces transformées par le processus de traitement et capitalisé par le processus de capitalisation. Ces processus sont réalisés automatiquement à l'exception du processus de modélisation de données. Les chapitres 5 et 6 présentent les moyens permettant de réaliser ces processus. Le langage DCL4UTL est présenté dans le chapitre suivant.

5 Spécification du langage DCL4UTL

Sommaire

Dans cette partie, nous présentons le langage de combinaison de données pour UTL appelé DCL4UTL. Notre objectif est de produire des patrons de données exécutables et réutilisables. Comme nous l'avons présenté dans le chapitre 3, nous nous basons sur les indicateurs d'une séance de travaux pratiques (TP) avec le laboratoire SysCom et du système VAPS au laboratoire ELHIT pour construire le langage DCL4UTL. Nous décrivons d'abord ces activités. Puis, le langage DCL4UTL et la méthode de modélisation de calcul d'indicateurs avec DCL4UTL sont présentés. Ensuite, nous illustrons une utilisation du langage en modélisant un indicateur du projet ICALTS. Cette modélisation spécifie la méthode de calcul de l'indicateur à partir de différentes données UTL avec DCL4UTL. L'idée est que ces méthodes de calcul soient réutilisables dans d'autres contextes. Nous présentons donc également un exemple d'utilisation qui illustre la façon de modéliser la méthode de calcul d'un indicateur du projet VAPS et nous réutilisons cette méthode pour modéliser un indicateur lié à une séance de TP avec le laboratoire SysCom.

5.1 Introduction

En ce qui concerne l'analyse des traces, la compétence et la connaissance de l'enseignant dans l'analyse d'une session d'apprentissage devraient pouvoir être partagées et réutilisées par la communauté pour faciliter l'analyse du déroulement d'une situation d'apprentissage. C'est dans ce but que le projet DPULS [Choquet (2005)] a fourni des patrons de conception permettant à l'enseignant et au tuteur de pouvoir capitaliser et réutiliser le savoir-faire en collecte et en analyse d'utilisation d'un EIAH vers un autre. Cependant, ces patrons se limitent à la description et ce projet n'a pas proposé d'outil d'analyse automatique réutilisable pour exécuter ces patrons.

En se basant sur les idées et les résultats de ce projet et d'autres projets européens tels qu'ICALTS [Dimitracopoulou (2004a)], [Choquet et Iksal (2007a)] ont proposé le langage de modélisation UTL. Il est principalement dédié à la modélisation des indicateurs et à leur capitalisation sous une forme proche des patrons de conception. Ces indicateurs sont décrits dans une forme indépendante du langage de scénarisation et du format de représentation des traces. Le calcul de ces indicateurs a pour objectif d'aider l'enseignant dans l'observation et l'évaluation d'une session d'apprentissage et donc de lui permettre d'améliorer éventuellement son scénario pédagogique pour la prochaine session. Nous considérons également que les patrons de données UTL devraient pouvoir être exécutés. Cependant la description des moyens d'observation de l'indicateur (l'élément *Getting.Method*, cf. figure 2-11) n'est pas formelle. C'est une description textuelle qui ne permet pas d'être interprétée par la machine à travers des outils d'analyse automatique. De plus, ce type de méthode d'analyse ne permet pas la réutilisation.

Comme nous l'avons présenté dans le chapitre 2, en ce qui concerne le domaine de l'analyse des traces générées par les EIAH, il existe plusieurs outils qui permettent d'analyser les données collectées par des systèmes d'apprentissage. Ils exploitent les traces dans un contexte particulier (chat, forum, etc.) et sont limités à un ensemble fermé d'indicateurs. Ces indicateurs sont définis avant la mise en œuvre de l'outil. Cependant, comme nous l'avons vu dans le chapitre précédent, la définition des indicateurs dépend des compétences des enseignants. Les outils d'analyse existants proposent des indicateurs prédéfinis et non modifiables. Les besoins d'observation des enseignants sont variés et évolutifs. Ils changent d'une session d'apprentissage à une autre. Il est donc nécessaire de proposer une solution évolutive d'observation. Nous nous intéressons à la capacité de modification ou d'ajout des indicateurs tout au long de l'utilisation de l'outil d'analyse avec peu de modifications par rapport à l'existant. Avec la même idée, l'approche de Diagne sur

les patrons d'indicateur réutilisable [Diagne (2009)] utilise directement un langage de programmation pour décrire la méthode de calcul des indicateurs. Chaque indicateur est encapsulé dans un agent pour être intégré dans un logiciel de supervision. Mais cette approche dépend du format des traces de la plate-forme d'apprentissage ainsi que de l'architecture de la base de données et du langage de programmation. Donc, si le format de traces évolue, la méthode de calcul d'indicateur doit être modifiée. L'outil proposé par [Gendron (2010)] permet également de calculer et de gérer la modification des indicateurs. Nous partageons ce point de vue. Par contre, ce travail ne considère que les indicateurs concernant les activités collaboratives et les valeurs de ces indicateurs se limitent aux types de données numériques, alphanumériques, de tableau et d'intervalle discret. Ces types sont alors insuffisants pour modéliser des indicateurs de type complexe, par exemple lorsque le type de données d'un indicateur est un objet avec plusieurs champs. De plus, comme nous l'avons présenté dans le chapitre 2, cette approche ainsi que le travail de [Djouad et al. (2009)] ne permettent pas d'associer l'indicateur avec des éléments du scénario pédagogique et à des types d'exploitation de la donnée (par exemple réingénierie, régulation, évaluation et réflexivité). L'enseignant/tuteur se voit systématiquement proposer des indicateurs préconçus, il a par conséquent des difficultés à les comprendre et à les utiliser (par exemple, il s'interroge sur la situation d'apprentissage pour laquelle l'indicateur a été prévu, ou encore sur l'objectif d'observation). En effet, ces travaux ne permettent pas à l'enseignant/concepteur d'exprimer ses besoins d'observation, et la sémantique de l'indicateur n'est pas présente dans les outils. Dans notre travail, nous prenons en compte cette association « besoin d'observation – indicateur – sémantique ». Tous les travaux sur les indicateurs utilisent des traces comme base. Les différences entre notre travail et les autres travaux résident dans le fait que, (1) notre approche permet à l'enseignant/concepteur d'exprimer ses besoins d'observation ; (2) nos indicateurs sont formalisés à partir de ces besoins exprimés et non des traces ; (3) nos indicateurs portent toujours une sémantique. Cette dernière spécifie la signification de l'indicateur, le lien entre l'indicateur et les éléments/concepts du scénario pédagogique embarqué dans le dispositif d'apprentissage ainsi que l'objectif d'observation de l'indicateur ; (4) nos indicateurs sont directement adressés à l'enseignant/tuteur car conçus en fonction de ses besoins. Ce dernier peut donc choisir et ré-utiliser des indicateurs selon différentes situations d'apprentissage et en fonction de ses objectifs d'observation. Cela facilite aussi l'amélioration, l'évolution du scénario ainsi que les activités du tuteur en session.

En ce qui concerne aussi l'analyse de traces, il existe plusieurs fonctions/opérateurs (par exemple : les fonctions de data mining) provenant d'outils d'analyse disponibles (par

exemple : Keel [Alcalá-Fdez et al. (2008)]). Nous pouvons réutiliser ces fonctions/opérateurs avec peu de modifications pour nous adapter à chaque contexte d'utilisation. Nous prenons donc en compte la réutilisation de ces fonctions dans notre travail.

Afin de prendre en compte ces besoins, le langage de combinaison de données (Data Combination Language - DCL) basé sur UTL est donc proposé comme une extension d'UTL (aussi appelé DCL4UTL) pour faciliter la génération et la réutilisation des outils d'analyse. C'est un langage formel s'appuyant sur le langage UTL. Il permet de décrire des méthodes de calcul des indicateurs dans un format indépendant du format de représentation des traces, du langage de scénarisation, mais aussi de l'architecture de la base de données. Cette nouvelle version d'UTL permet donc de modéliser, de capitaliser et de produire des indicateurs automatisables. Elle permet d'exprimer la sémantique des données UTL, de spécifier le lien entre l'indicateur, le scénario pédagogique et l'objectif d'observation, ainsi que la méthode de calcul des indicateurs d'une manière compréhensible par l'homme.

Dans les sections suivantes, dans un premier temps, nous décrivons les deux expérimentations (une séance de travaux pratiques avec le laboratoire SysCom et le système VAPS) utilisées dans les exemples du chapitre. Dans un second temps, nous présentons notre langage DCL4UTL. Ensuite, les exemples d'utilisation du langage sur les indicateurs du projet ICALTS sont présentés. Finalement, nous décrivons les fonctionnalités du langage DCL4UTL et leurs exemples d'utilisation sur les traces du laboratoire SysCom et du système VAPS.

5.2 Les expérimentations préliminaires

Cette partie présente brièvement les deux expérimentations (SysCom, VAPS) qui vont servir d'exemples dans le reste de ce chapitre. Les traces collectées et les indicateurs considérés dans ces expérimentations ont permis de créer et de tester le langage DCL4UTL. Nous présentons également les indicateurs du projet ICALTS qui sont utilisés dans ce chapitre à des fins de test du langage.

5.2.1 L'expérimentation au laboratoire SysCom

Cette expérimentation a eu lieu en juin 2006 au département SeRéCom (Services et Réseaux de Communication) au sein de l'Université de Savoie [Iksal et al. (2007)].

La session d'apprentissage concernée est un TP avec une classe de 15 étudiants, en première année de DUT, département SeRéCom. L'activité d'apprentissage de la session concernait la modélisation orientée objet. Dans une session de trois heures, les étudiants

doivent décrire un projet avec le support d'un logiciel de création de diagrammes UML nommé Softeam Objecteering/UML. Les étudiants effectuent leur TP de la manière habituelle, un par machine. Ils avaient quatre types de diagrammes à réaliser dans l'ordre de leur choix (Exercice Rouge, Exercice Jaune, Exercice Bleu, Exercice Vert) et deux autres activités (la rédaction du compte-rendu de TP et la possibilité de se mettre en pause). Le code des couleurs est utilisé pour ne pas influencer l'ordre d'exécution des exercices.

Toutes les activités des étudiants sont collectées par un environnement développé pour l'observation des activités d'apprentissage [Carron et al. (2006)]. Ces activités sont enregistrées dans un seul fichier de log. Le contenu de ce fichier est décrit dans le chapitre 6.

5.2.2 Le système VAPS au laboratoire ELHIT

Le système VAPS (Virtual Action Planning Supermarket) a été développé par [Klinger (2006)]. Il s'agit d'un supermarché virtuel (cf. figure 5-1) de taille moyenne avec de nombreux rayons pour plusieurs types de produits comme les boissons, la viande, les poissons, les aliments salés, les aliments sucrés, les conserves, les produits d'entretien, etc.

Ce système VAPS a été développé pour l'aide au diagnostic des dommages causés par des accidents vasculaires cérébraux ainsi qu'au réapprentissage chez les patients parkinsoniens. Le patient participant à la session apprend d'abord à utiliser VAPS. Puis, il fait des courses lors d'une séance. Avant de commencer la séance, les consignes orales sont données au patient. Ce dernier sait donc ce qu'il doit faire pendant la séance. Une consigne peut être : ''*Dans le supermarché, vous achèterez : une baguette ; des pommes vertes ; un baril de lessive de 2kg ; 1kg de farine ; un tee-shirt pour enfant ; deux artichauts ; et des chaussettes beiges. Pour payer, il vous suffira de cliquer sur un porte-monnaie qui se trouvera sur l'écran*'' [Klinger (2006)].

Figure 5-1 Le supermarché virtuel - Vue de l'entrée, [Klinger (2006)]

Finalement, la séance démarre. La session commence quand le participant se trouve au point d'entrée du VAPS. Le patient réalise librement ses tâches. La séance s'arrête automatiquement dès que le VAPS reconnaît que le patient a atteint l'aire de sortie du VAPS. Si le patient se perd dans le VAPS, la session peut se terminer avant d'arriver à l'aire de sortie. Le temps de la session est illimité.

Le système VAPS enregistre toutes les activités des patients dans plusieurs fichiers de log (un pour chaque patient). Nous détaillons ces fichiers dans le chapitre 6.

5.2.3 Le projet ICALTS

Les indicateurs d'une séance de travaux pratiques (TP) avec le laboratoire SysCom et du projet VAPS nous ont permis de créer le langage DCL4UTL. Nous avons souhaité donc tester ce langage avec d'autres indicateurs. Comme nous l'avons présenté dans le chapitre 2, le projet ICALTS a collecté plusieurs indicateurs provenant de différents laboratoires de recherche en Europe. C'est pourquoi nous avons choisi des indicateurs de ce projet : l'indicateur ''*Collaborative Action Function*'', l'indicateur ''*Agent"* et l'indicateur ''*Interaction"*. Les deux indicateurs *"Agent"* et ''*Interaction"* sont nécessaires pour le calcul de l'indicateur ''*Collaborative Action Function*''. Nous détaillons ces indicateurs dans la section 5.4.2

5.3 Spécification du langage

Nous travaillons sur la combinaison d'un ensemble de traces pour produire des indicateurs. Dans notre contexte de travail, nous disposons du langage UTL qui permet de modéliser des indicateurs à partir des traces. Or ce langage manque de méthodes et d'outils génériques opérationnalisés permettant de modéliser le calcul d'indicateurs. Il est donc plus approprié d'utiliser les langages de requêtes comme SQL, XQuery pour formaliser des indicateurs. Cependant, comme nous l'avons présenté dans le chapitre 2, ces langages de requêtes ne permettent pas de capitaliser la méthode d'acquisition de données avec ses résultats dans une forme réutilisable. Ils ne permettent également pas d'intégrer des fonctions externes. De plus, ils dépendent de l'architecture de la base de données. Par exemple, les requêtes SQL dépendent du schéma de la base de données, si le schéma évolue, ces requêtes sont difficilement réutilisables. UTL est imposé comme contexte de notre travail, nous avons donc développé un son extension (DCL4UTL) pour la description formelle des méthodes de calcul d'indicateurs.

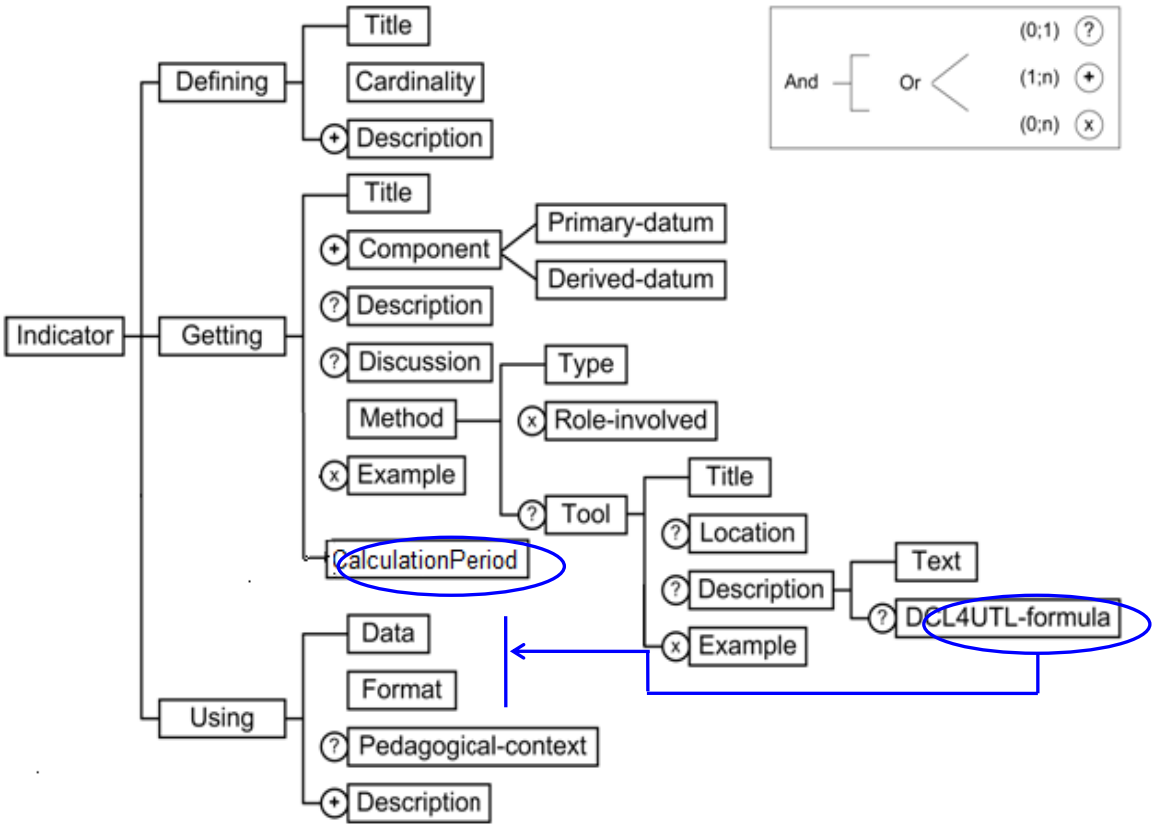

Figure 5-2 Le modèle d'information de l'indicateur

Les objectifs du langage DCL4UTL sont de combiner les données UTL pour produire une nouvelle donnée dérivée, en particulier l'indicateur (au sens d'UTL) et de capitaliser l'expertise de calcul des indicateurs pour la réutiliser. Les langages de requêtes classiques (SQL, Xquery) permettent différents types d'opérations sur une base de données (relationnelle ou XML). Ils peuvent être utilisés pour produire des indicateurs simples. Notre proposition se base donc sur ces langages. En plus, nous ajoutons dans notre langage des fonctionnalités qui n'existent pas comme la capitalisation des méthodes de calcul ou l'intégration des fonctions externes, etc.). De plus, il s'agit d'un langage fondé et intégré dans UTL. Nous utilisons le champ *Method.Tool.Description* de la facette *Getting* (cf. figure 5-2) pour décrire formellement la façon d'obtenir une donnée dérivée à partir de données UTL avec DCL4UTL. Pour cela, nous ajoutons deux champs : *Description.Text* et *Description.DCL4UTL-formula*. Le champ *Description.Text* est destiné à un utilisateur humain. Ce champ exprime la méthode de calcul de manière compréhensible par l'homme. Le champ *Description.DCL4UTL-formula* est interprétable par la machine. Nous avons également ajouté le champ *Getting.CalculationPeriod* permettant de spécifier le moment de calcul de l'indicateur pour une utilisation en temps réel. Ce calcul peut être déclenché (1)

quand un événement spécifique est généré par le dispositif d'apprentissage, (2) après un intervalle de temps prédéfini ou (3) quand la demande est effectuée par l'utilisateur.

Le langage permet à l'analyste de décrire une solution de combinaison des données spécifiées dans les éléments *Getting.Component* de la facette *Getting* pour acquérir une nouvelle donnée. Ce langage opère aussi sur la facette *Using* des données UTL. La facette *Using* de chaque donnée a un champ *Format* et un champ *Data*. Le champ *Format* définit la manière dont les instances des données (dans la signification d'UTL) sont représentées une fois qu'elles sont produites. Le champ *Data* stocke les instances calculées en prenant en compte ce format.

5.3.1 Les éléments principaux du langage DCL4UTL

La figure 5-3 présente les éléments principaux du langage DCL4UTL. Ce dernier contient des structures de contrôle qui combinent des expressions. Nous considérons trois types de structures : la répétition (for), la condition (if) et l'affectation. Comme d'autres langages de requête, DCL4UTL se compose également de mots-clés.

Puisque DCL4UTL est un langage de combinaison de données. Il opère donc sur un ensemble de données. Le calcul d'un indicateur doit parcourir cet ensemble pour produire ses valeurs. C'est pour cette raison que nous ajoutons la boucle *for* au langage. Le résultat des indicateurs peut être produit selon des critères/conditions. C'est pourquoi une structure conditionnelle comme le *if* est nécessaire. Comme dans tous les langages, nous utilisons des variables et par conséquent l'affectation.

Chaque expression est normalement une combinaison entre des opérateurs et des opérandes. La combinaison est pratiquée sur des données UTL, en particulier sur les éléments de la facette *Using.data*. Ces éléments peuvent être une valeur numérique ou alphanumérique. Les opérateurs comme +, -, *, / sont donc nécessaires pour la combinaison des valeurs numériques et la fonction de chaîne de caractères a pour but de combiner les valeurs alphanumériques. Comme nous l'avons indiqué ci-dessus, les valeurs des indicateurs peuvent être produites selon certains critères, les opérateurs relationnels (<, <=, >, >=, ==, =!) et les opérateurs logiques (and, or) doivent alors être présents pour évaluer une condition. Nous n'avons pas proposé la négation car il est possible utiliser l'opérateur différent '= !'.

Puisque le langage travaille sur un ensemble de données, le calcul d'indicateurs peut nécessiter le filtrage pour obtenir un élément ou un sous-ensemble d'éléments dans la liste

d'éléments. De plus, le calcul d'indicateurs concerne souvent les statistiques sur l'ensemble (ou ce sous-ensemble filtré) de données. C'est pour cela que nous proposons les fonctions de position, les fonctions de filtre et de tri de données ainsi que les fonctions statistiques.

Il est difficile de prévoir toutes les fonctions pour le calcul d'indicateurs, parce que cela dépend de la complexité de l'indicateur à calculer. Dans le processus de modélisation d'indicateurs, certaines fonctions complexes peuvent s'avérer utiles, elles n'existent pas dans DCL4UTL, donc ce dernier dispose d'une possibilité d'intégrer des fonctions externes. Ces fonctions peuvent être définies par le développeur ou provenir d'autres outils existants (data mining par exemple).

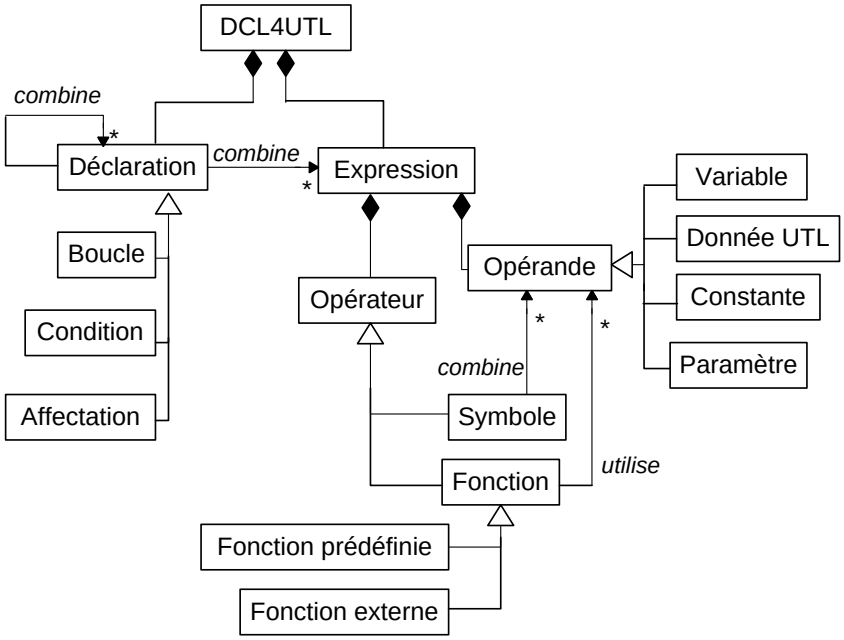

Figure 5-3 La structure du langage DCL4UTL

5.3.1.1 Les mots-clés

Notre langage dispose également des mots-clés suivants comme la plupart des langages de programmation ou de requête :

- ''*cal*'' permet de déclarer la combinaison ;
- ''*as*'' permet de déclarer l'organisation spécifique des données (filtrage, tri, etc.)
- ''*for each*'' est une structure de contrôle pour une boucle.
- ''*if*'' et ''*else*'' correspondant à la structure de contrôle conditionnelle.
- ''*where*'' permet de déclarer des alias pour les données.
- ''*parameter*'' permet de déclarer des paramètres pour les données intermédiaires.

- ‘‘*extern*’’ permet de faire appel à une fonction externe.
- ‘‘*true*’’ et ‘‘*false*’’ sont les constantes booléennes.

5.3.1.2 Les opérandes du langage

Nous considérons quatre types principaux d’opérandes : les données UTL, les constantes, les variables et les paramètres. Une donnée UTL correspond à l'un des éléments du champ *Using.data* des données UTL, par exemple *Using.data.indicator.user*. Une constante peut être une valeur numérique ou alphanumérique. Les variables sont utilisées comme les variables des langages de programmation. Les paramètres sont les variables utilisées pour créer des données intermédiaires. Nous détaillons ces données dans la section suivante. Tous les opérandes de DCL4UTL sont décrits dans sa grammaire (voir l'annexe A).

5.3.1.3 Les opérateurs du langage

Pour les opérateurs, nous proposons deux types d’opérateurs : le symbole (par exemple : +, -, *, /, etc.) et la fonction (par exemple : sort, filter, compare, etc.). Nous avons sélectionné les principaux symboles suivants :

- Les opérateurs arithmétiques : +, -, *, /.
- Les opérateurs logiques : and, or.
- Les opérateurs de comparaison ou relationnels : <, <=, >, >=, ==,=!.

Les opérateurs logiques et relationnels ne sont utilisés que dans les expressions conditionnelles. La valeur d'une expression conditionnelle est *true* ou *false*.

L'opérateur représenté par une fonction se compose aussi de paramètres. Ces paramètres peuvent être un opérande ou une expression conditionnelle. Deux types de fonctions sont proposés : la fonction prédéfinie et la fonction externe.

5.3.1.3.1 La fonction prédéfinie

Les fonctions prédéfinies (ou bien fonctions internes) sont celles du langage. Elles se composent de fonctions concernant les statistiques, la chaîne de caractère, la position, le filtrage, le tri. Les principales fonctions prédéfinies sont les suivantes :

Les fonctions de position : ***first***(*element*), ***end***(*element*), ***index***(*element*, *index*).

- L’opérateur ***first***(*element*) renvoie le premier élément d’un ensemble d'éléments de données.

- L'opérateur ***end***(*element*) renvoie le dernier élément d'un ensemble d'éléments de données.
- L'opérateur ***index***(*element*, *index*) renvoie l'élément à la position *index* dans un ensemble d'éléments de données, la valeur d'index commence à partir de 1 qui correspond au premier élément d'un ensemble d'éléments.

Les fonctions statistiques : ***sum***(*element*), ***sumDis***(*element*), ***min***(*element*), ***max***(*element*), ***count***(*element*), ***countDis***(*element*).

- La fonction ***sum***(*element*) (respectivement ***sumDis***(*element*)) retourne la somme de toutes les valeurs (respectivement la somme des valeurs distinctes) d'un élément numérique *element*.
- Les fonctions ***min***(*element*) et ***max***(*element*) sont utilisées pour chercher une valeur minimale et maximale dans l'ensemble d'occurrences d'un élément numérique *element*.
- La fonction ***count***(*element*) (respectivement ***countDis***(*element*)) comptera le nombre d'occurrences (respectivement d'occurrences distinctes) d'un élément *element* contenant des données UTL.

La fonction de chaîne de caractères : ***concat***(*str1*, *str2*). Cet opérateur concatène une chaîne de caractères *str2* à une autre *str1*. *str1* peut être une chaîne de caractères ou un élément du champ *Using.data* de type de chaine de caractères (*String*).

La fonction de filtre de données : ***filter***(*crit*). Cet opérateur extrait des données dans un ensemble d'éléments de données UTL selon le critère *crit* qui est une condition de sélection sur les données.

La fonction de tri de données : ***sort***(*crit*). Cet opérateur va trier des données selon un ou plusieurs éléments avec un critère croissant (*asc*) ou décroissant (*desc*).

Ces deux opérateurs ***filter*** et ***sort*** peuvent être utilisés pour faire des filtres ou des tris sur les données avant de calculer les éléments d'une donnée dérivée.

Exemple :

```
count(R.E as filter(R.user == "u" and R.session == "s") )
```

Tableau 5-1 Exemple d'utilisation des fonctions internes

Dans cet exemple, la méthode de calcul est : (1) de sélectionner (*filter*) les données brutes du type *R* pour lesquelles le nom d'utilisateur (*R.user*) est *u* et le nom de session (*R.session*) est *s* ; (2) avec le résultat de cette sélection, de compter le nombre d'éléments *E* des données *R*.

5.3.1.3.2 La fonction externe

Dans le chapitre 2, nous avons présenté plusieurs outils qui permettent le calcul d'indicateurs à partir des données collectées par les systèmes d'apprentissage. Nous considérons utile d'intégrer des fonctions externes disponibles pour la combinaison des données dans notre processus d'analyse. Cela permet de pouvoir réutiliser des outils d'analyse de données qui existent dans le domaine des EIAH. C'est alors moins coûteux que de développer une nouvelle fonction. C'est pour cette raison que DCL4UTL a été conçu afin de permettre d'intégrer des opérateurs et/ou fonctions externes [Pham-Thi (2010), Pham-Thi et al. (2010b)] qui proviennent d'autres outils d'analyse, et/ou sont créés par le développeur qui joue l'un des trois rôles principaux (enseignant, analyste et développeur) dans le processus d'analyse d'une session d'apprentissage.

L'utilisation d'opérateurs externes dans la combinaison des données nécessite de fournir le nom de la fonction, une liste d'arguments et éventuellement le nom de la classe (et son chemin pour un langage objet), selon la syntaxe suivante :

```
extern [<Class>] <Fonction> <Arguments>
```

Tableau 5-2 La syntaxe d'un appel d'une fonction externe

Exemple : le code suivant fait un appel à une fonction externe `effacerEspace` qui efface tous les espaces dans une chaîne de caractères `str`. Cette fonction est définie dans la classe `FonctionExterne`.

```
$str= «voici un exemple»
$trim=extern FonctionExterne effacerEspace(str)
```

Tableau 5-3 Exemple d'utilisation d'une fonction externe

Où :

- ***extern*** : est un mot-clé,
- *FonctionExterne* : est le nom de la classe,
- *effacerEspace*: est le nom de la méthode.

Actuellement, DCL4UTL permet de déclarer une variable qui peut prendre une valeur de type simple (une chaîne de caractères, une valeur numérique) ou de type complexe (un tableau, un objet). Les règles de production suivantes de la grammaire permettent la déclaration et l'affectation d'une variable.

- La déclaration de n'importe quelle variable :

```
<VariableDeclaration> ::= "$" <Id>
```

Tableau 5-4 La règle de production pour la déclaration d'une variable

Exemple : *$str*

- Trois types d'affectation d'une variable sont considérés. L'affectation d'une variable avec une valeur numérique ou alphanumérique, avec un tableau et avec un objet. Dans un premier temps, nous avons décidé de nous focaliser sur ces types. Les autres types peuvent ensuite être ajoutés en se basant sur ces types.

```
<AssignmentStatements> ::= <AssignmentStatement> " ;"
<AssignmentStatement> ::=  <SimpleAssignment>
                         | <ArrayAssignment>
                         | <StructureAssignment>
```

Tableau 5-5 La règle de production pour les trois types d'affectation d'une variable

i) L'affectation d'une variable avec une valeur numérique ou alphanumérique

```
<SimpleAssignment> ::= <Declaration> "=" <Expression>
<Declaration> ::= <VariableDeclaration> | <Id>
```

Tableau 5-6 La règle de production pour l'affectation d'une variable avec une valeur

Exemple str = "abd" ; *// str est déjà déclaré.*
ou $i = 0 ;

ii) L'affectation d'une variable avec un tableau

```
<ArrayAssignment> ::= <Declaration>"="(<Array>
                                |<TwoDimensionsArray>)
```

Tableau 5-7 La règle de production pour l'affectation d'une variable avec un tableau

+ Tableau à une dimension

```
<Array> ::= "[" <Expression>
                          {"," <Expression>} "]"
```

Tableau 5-8 La règle de production pour la spécification des valeurs d'un tableau à une dimension

+ Tableau à deux dimensions

```
<TwoDimensionsArray> ::=
                  "[" <Array> {"," <Array>} "]"
```

Tableau 5-9 La règle de production pour la spécification des valeurs d'un tableau à deux dimensions

Exemple : $tab1 = [1, 2, 3] ;
$tab2 = [[1, 2, 5], [4, 5.1, 6]] ;

iii) L'affectation d'une variable avec un objet.

```
<StructureAssignment> ::= <Declaration> "=" <Structure>
<Structure> = "[" <StructureName>  <Field> {"," <Field>} "]"
<Field> = <Id> ":" (<Expression> | <Structure>)
```

Tableau 5-10 La règle de production pour l'affectation d'une variable avec un objet

Exemple : $adresse= [MonPackage.Adresse numero : 1, rue : ‘‘Liberté’’, codePostal : [MonPackage.Code code : 53000, ville : ‘‘Laval’’]] ;

En ce qui concerne les fonctions externes, elles ne peuvent que fonctionner avec des arguments prenant des valeurs de types décrits ci-dessus (une chaîne de caractères, une valeur numérique, un tableau et un objet) ou les éléments *Using.data* d'UTL.

5.3.2 Syntaxe du langage

Nous avons défini une grammaire pour DCL4UTL selon la syntaxe BNF (Backus-Naur Form). Elle est trop longue pour être présentée en extension dans cette partie, donc nous présentons un extrait de la grammaire et de la syntaxe du langage. La grammaire du langage se trouve dans l'annexe A. Un extrait de la grammaire est présenté dans le tableau 5-11.

```
<DCL4UTL> ::= <CompilationUnit>
<CompilationUnit> ::= "cal" <Statements> ["as" <ConditionList>]
"where" <AliasList>
["parameter" <FormalParamsList>]

<Statements> ::= "{" (<Statement>)+ "}"
<Statement> ::= <ForStatement> | <AssignmentStatements>
    | <IfStatement>

<ConditionList> ::= (<Filter> [ <Sort> ] ) | <Sort>
<Filter> ::= "filter" "(" <Expression> ")"
<Sort> ::= "sort" "(" <SortList> ")"
<SortList> ::= <SortCondition> { "," <SortCondition> }
<SortCondition> ::= <UTLElement>["[" <Id> "]"] ["asc" | "desc"]

<AliasList> ::= <Alias> { "," <Alias> }[ <RealParameterList> ]
<Alias> ::= <Id> "=" <UTLElement>

<RealParameterList> ::= "["<RealParameter>
                                        {","<RealParameter>} "]"
<RealParameter> ::= <Id> "=" <Expression>

<FormalParamsList> ::= <FormalParameter> {","<FormalParameter> }
<FormalParameter> ::= <Id> "=" <Id>
[...]
```

Tableau 5-11 Extrait de la grammaire DCL4UTL

La syntaxe de base du langage est la suivante :

cal <*Expression*> **where** <*Data*> [**parameter** <*ParameterList*>]

Tableau 5-12 La syntaxe de base du langage DCL4UTL

Où :

- *Expression* : sont les expressions concernant les éléments du champ *Using.data* à calculer. Elles sont éventuellement constituées d'opérandes, d'opérateurs, de fonctions internes et externes et de structures de contrôle.
- *Data* : sont les données UTL nécessaires (données primaires et/ou données dérivées) pour la combinaison des données.
- *ParameterList* : est une liste utilisée pour passer des paramètres entre les données à calculer.
- **cal**, **where**, **parameter** : sont des mots-clés.
- Les éléments encadrés par des crochets ([]) sont optionnels.

Exemple d'utilisation du langage :

```
cal {
    for each $i in AD.utilisateur do {
       I.tauxRealisation[patient=i]=
           (I1.nombreBonnesActions[patient==i] *100)/12;
    }
}
where
    I = VAPS-I-Taux-Realisation.using.data.indicator,
    I1= VAPS-I-BA.using.data.indicator,
    AD= VAPS-AD-Utilisateur.using.data.additionalDatum
```

Tableau 5-13 La méthode de calcul de l'indicateur VAPS ''*Le taux de réalisation*''

Le code ci-dessus décrit la méthode de calcul de l'indicateur ''*Le taux de réalisation*'' (*VAPS-I-Taux-Realisation*) du système VAPS, c'est-à-dire le pourcentage de bonnes actions réalisées par chaque patient. Nous distinguons deux parties dans ce code : la méthode (la partie entre le mot-clé *cal* et le mot-clé *where*) et les données utilisées dans la méthode (la partie après le mot-clé *where*). Cet indicateur (*VAPS-I-Taux-Realisation*) est calculé avec l'aide d'un indicateur ''*Le nombre de bonnes actions*'' (*VAPS-I-BA*) qui calcule le nombre de bonnes actions par patient et d'une donnée additionnelle qui décrit la liste des patients (*VAPS-AD-Utilisateur*). Nous déclarons donc les trois données dans la partie *where*. I, I1, AD sont les alias. *VAPS-I-Taux-Realisation*, *VAPS-I-BA*, *VAPS-AD-Utilisateur* sont les noms des données UTL. L'élément *using.data* spécifie que les éléments de données utilisés dans la méthode sont extraits de cet élément. Dans la partie correspondant à la méthode de calcul, le taux de réalisation est calculé pour chaque patient. La boucle *for each* permet donc de parcourir la liste des patients : *for each $i in AD.utilisateur do*. Ensuite, pour chaque patient *i*, le taux de réalisation du patient *i* (*I.tauxRealisation[patient=i]*) est calculé par l'expression :

```
I.tauxRealisation[patient=i]=
          (I1.nombreBonnesActions[patient==i] *100)/12;          (5.1)
```

Où : *I1.nombreBonnesActions[patient==i]* est le nombre de bonnes actions réalisées par le patient *i*. Dans ce cas, on suppose que le nombre maximal de bonnes actions est douze. L'expression (5.1) retourne un résultat qui a le format décrit dans le tableau 5-14. Ce format doit être défini dans la partie *Using.format* de l'indicateur *VAPS-I-Taux-Realisation*.

```
<indicator type="VAPS-I-Taux-Realisation">
    <tauxRealisation patient="string">float</tauxRealisation>
</indicator>
```

Tableau 5-14 Le format de l'indicateur ''*Le taux de réalisation*'' (VAPS-*I-Taux-Realisation)*

5.4 Méthode de modélisation de calcul d'indicateurs avec UTL et DCL4UTL

Dans cette partie, nous présentons d'abord la méthode de modélisation d'indicateurs UTL. Puis, nous montrons un exemple d'utilisation de cette méthode avec les indicateurs du projet ICALTS.

5.4.1 Modélisation d'indicateurs

Comme nous l'avons présenté dans le chapitre 2, les indicateurs sont calculés ou établis à partir de données primaires (la donnée brute (RD), la donnée additionnelle (AD) et la donnée de production (CD)) et/ou d'autres données dérivées (l'indicateur (I) et la donnée intermédiaire (ID)). Ces indicateurs sont formalisés à partir des besoins d'observation des enseignants. Ces besoins sont nécessairement identifiés par les enseignants lors de la définition de leur scénario. En nous appuyant sur ces informations, nous proposons une méthode générique pour la modélisation et le calcul d'indicateurs. Elle se compose de quatre étapes :

- La première étape consiste à modéliser les données brutes (dans la signification d'UTL) à partir du contenu des fichiers de logs générés par le dispositif d'apprentissage.
- La deuxième étape concerne la création si nécessaire des données additionnelles et/ou des données de production.

Ces deux premières étapes peuvent être considérées comme les étapes de préparation des données [Djouad et al. (2009)].

- La troisième étape se compose de la définition et de la modélisation des indicateurs [Pham-Thi et al. (2010a), Iksal et al. (2010)] et éventuellement des données intermédiaires nécessaires au calcul. Dans cette étape, UTL est employé comme un langage de modélisation pour décrire et définir des indicateurs. DCL4UTL est utilisé comme un langage de formalisation de la méthode de calcul des indicateurs à partir des données UTL.

Les tâches dans ces trois premières étapes sont effectuées par des analystes de données. Ces tâches sont incluses dans le processus de modélisation de données (cf. chapitre 4).

- La dernière étape concerne le calcul automatique des indicateurs et de leur capitalisation. Le tableau 5-15 montre l'algorithme de calcul d'un indicateur. Le calcul

d'indicateurs est réalisé par l'interpréteur de DCL4UTL présenté dans le chapitre suivant. Ce calcul est réalisé par le processus de calcul présenté au chapitre 4.

L'algorithme pour calculer un indicateur i : calculer(i)

Entrée : Données primaires et/ou données dérivées décrites dans le champ *Getting.component*.

Sortie : Instances de l'indicateur représentées en prenant en compte le champ *Using.format*.

Algorithme :

```
Récupérer les données nécessaires Ds pour calculer i à partir de
champ Getting.component;
Pour chaque donnée dérivée d ∈ Ds
{
     si d n'est pas encore calculée
     calculer (d) ;
}
Récupérer la formule de calcul de l'indicateur i à partir de
champ Getting.method.tool.description.DCL4UTL-formula;
Analyser la formule et l'exécuter ;
Capitaliser des instances de l'indicateur dans le champ
Using.data conforme au format exprimé dans le champ Using.format;
```

Tableau 5-15 L'algorithme de calcul d'un indicateur

5.4.2 Modélisation de l'indicateur ''Collaborative Actions Function'' avec UTL et DCL4UTL

Cette partie présente la façon dont un nouvel indicateur peut être modélisé avec UTL et DCL4UTL. Nous avons choisi l'indicateur ''*Collaborative Action Function*'' (CA) [Dimitracopoulou (2004a)] présenté dans le projet ICALTS pour spécifier sa modélisation [Iksal et al. (2010)].

5.4.2.1 Introduction

Le calcul de cet indicateur est basé sur deux autres indicateurs : *Indicateur d'Agent* et *Indicateur d'Interaction*. Les trois indicateurs ont été définis et implémentés dans la plateforme MODELLINGSPACE [Avouris et al. (2003)].

- *L'indicateur d'Agent* présente le nombre d'utilisateurs (apprenants, tuteurs, etc.) qui ont utilisé au moins une chaîne d'interaction ou de collaboration pendant une session collaborative dans un intervalle de temps donné.

- *L'indicateur d'Interaction* présente le nombre d'actions d'interaction réalisées dans une chaîne d'interaction (chat, forum, etc.) pour un intervalle de temps donné.

L'indicateur *CA* combine ces deux indicateurs pour fournir une évaluation quantitative de la collaboration dans un groupe d'utilisateurs pendant un intervalle de temps donné. Si on considère une session de collaboration dans l'intervalle *[t0-tm]*, cet intervalle de temps est quantifié en utilisant un paramètre *n* de la façon suivante : *ti=t0+i*d*, avec *d=(tm-t0)/n*. Alors, l'indicateur *CA* est calculé selon la formule suivante :

$$CA(ti) = \sum_{k=1}^{k_{max}} A(k, ti) * I(k, ti)$$

Avec :

- *A(k, ti)* : le nombre d'utilisateurs qui ont interagi avec au moins une chaîne d'interaction *k* dans l'intervalle de temps *[ti-1, ti]*.
- *I(k, ti)* : le nombre d'actions faites dans une chaîne d'interaction *k* durant l'intervalle de temps *[ti-1, ti]*.

5.4.2.2 La modélisation de l'indicateur CA avec UTL et DCL4UTL

Dans cette section, nous modélisons l'indicateur *CA* selon les trois facettes du modèle DGU d' UTL. Le tableau 5-16 correspond à la facette *Defining*, le tableau 5-17 illustre la facette *Using* de l'indicateur *CA* et le tableau 5-18 présente la facette *Getting*.

Defining	Title	Indicateur associé aux actions collaboratives(CA)
	Cardinality	n, *le nombre de valeurs de cet indicateur est* ***n*** *parce que l'indicateur est calculé après chaque intervalle de temps*.
	Description	Cet indicateur indique le niveau des actions collaboratives dans un groupe d'apprenants durant un intervalle de temps donné.

Tableau 5-16 La facette *Defining* de l'indicateur CA

Using	Data	*N'est pas utilisé dans la modélisation*
	Format	`<indicator type="ICALTS-I-CA"> <ca time="string"> float </ca> </indicator>`
	Used-by	

Tableau 5-17 La facette Using de l'indicateur CA

5.4.2.3 La méthode de calcul de l'indicateur CA

Dans cette partie, nous ne décrivons pas la modélisation de toutes les données UTL nécessaires pour la modélisation de l'indicateur CA, mais les formats des données primaires seront présentés. Pour les données dérivées, nous donnons leurs formats et leurs formules de calcul.

<table>
<tr><td rowspan="8">Getting</td><td>Title</td><td colspan="4">Calcul d'indicateur associé aux actions collaboratives</td></tr>
<tr><td>Component</td><td>Addition-aldatum</td><td colspan="3">ICALTS-AD-C</td></tr>
<tr><td>Component</td><td>Indicator</td><td colspan="3">ICALTS-I-A</td></tr>
<tr><td>Component</td><td>Indicator</td><td colspan="3">ICALTS-I-I</td></tr>
<tr><td rowspan="4">Method</td><td>Type</td><td colspan="3">Automatic</td></tr>
<tr><td rowspan="3">Tool</td><td>Title</td><td colspan="2">Fonction de combinaison déclarative basée sur les données des agents et des interactions</td></tr>
<tr><td rowspan="2">Description</td><td>Text</td><td>$CA(ti) = \sum_{k=1}^{k_{max}} A(k,ti) * I(k,ti)$</td></tr>
<tr><td>Formula-DCL4UTL</td><td>Voir le tableau 5-31</td></tr>
</table>

Tableau 5-18 La facette *Getting* de l'indicateur CA

Afin d'établir l'indicateur CA à partir des traces générées par la plate-forme d'apprentissage, nous supposons que cette dernière fournissait des traces contenant au moins les éléments suivants :

{temps, codeAction, acteur, codeRessource}

Où :

- *temps* concerne la date de début d'une action.
- *codeAction* concerne une action d'interaction effectuée.
- *acteur* concerne une personne qui fait une action.
- *codeRessource* concerne une chaîne d'interaction utilisée.

Nous formalisons la formule de calcul de l'indicateur CA à partir de ces traces selon les étapes suivantes :

1. Nous créons une donnée brute (dans la signification d'UTL) appelée *ICALTS-RD-Action* pour décrire le contenu des traces générées par la plate-forme.

```
<rawDatum type="ICALTS-RD-Action">
      <heure> string </heure>
      <action> string </action>
      <acteur> string </acteur>
      <ressource> string </ressource>
</rawDatum>
```

Tableau 5-19 Le format de la donnée brute *ICALTS-RD-Action*

Le tableau 5-20 présente les correspondances entre les éléments de la trace et de la donnée brute *ICALTS-RD-Action*.

Trace	ICALTS-RD-Action
temps	<heure>
codeAction	<action>
acteur	<acteur>
codeRessource	<ressource>

Tableau 5-20 Les éléments de la donnée brute correspondant aux éléments de la trace

2. Nous décrivons deux données additionnelles (dans la signification d'UTL) représentant le nombre d'intervalles et la liste des chaînes d'interaction :

   i) La donnée *ICALTS-AD-P* correspond au paramètre *n* dans la formule *ti=t0+i*d*, avec *d=(tm-t0)/n*, qui est le nombre d'intervalles. Le format de cette donnée est le suivant :

```
<additionalDatum type="ICALTS-AD-P">
      <n> integer </n>
</additionalDatum>
```

Tableau 5-21 Le format de la donnée additionnelle *ICALTS-AD-P*

   ii) La donnée *ICALTS-AD-C* correspond à un ensemble de chaînes d'interaction. Son format est le suivant :

```
<additionalDatum type="ICALTS-AD-C">
      <channel id="integer">string</channel>
</additionalDatum>
```

Tableau 5-22 Le format de la donnée additionnelle *ICALTS-AD-C*

3. Deux données intermédiaires sont modélisées afin de calculer l'indicateur CA.

i) La donnée *ICALTS-ID-TDF* présente la date de début et la date de fin de la session. Son format est le suivant :

```
<intermediateDatum type="ICALTS-ID-TDF">
      <t0> time </t0>
      <tm> time </tm>
</intermediateDatum>
```

Tableau 5-23 Le format de la donnée *ICALTS-ID-TDF*

La méthode de calcul de cette donnée est décrite comme suit :

```
cal{
           ID.t0 = first(R.heure as sort(R.heure));
           ID.tm = end(R.heure as sort(R.heure)) ;
      }
where ID = ICALTS-ID-TDF.using.data.intermediateDatum,
           R = ICALTS-RD-Action.using.data.rawDatum
```

Tableau 5-24 La méthode de calcul de la donnée *ICALTS-ID-TDF*

Cette donnée (*ICALTS-ID-TDF*) est calculée avec l'aide d'une donnée brute de type ICALTS-RD-Action. La méthode de calcul est : (1) de trier les instances de cette donnée brute selon la date de début d'une action (*sort(R.heure)*) ; (2) avec le résultat de ce tri, de sélectionner le premier élément *heure* (*first*) correspondant à la date de début (*t0*) et le dernier élément *heure* (*end*) correspondant à la date de fin (*tm*).

ii) La donnée *ICALTS-ID-D* calcule le paramètre *d=(tm - t0)/n*. Son format est le suivant :

```
<intermediateDatum type="ICALTS-ID-D">
      <d> float </d>
</intermediateDatum>
```

Tableau 5-25 Le format de la donnée *ICALTS-ID-D*

Selon l'expression *d=(tm - t0)/n*, cette donnée (*ICALTS-ID-D*) est calculée avec l'aide d'une donnée intermédiaire de type *ICALTS-ID-TDF* et d'une donnée additionnelle de type *ICALTS-AD-P*. La méthode de calcul de cette donnée est la suivante :

```
cal {
      ID1.d = (ID2.tm - ID2.t0)/AD.n ;
      }
where ID1 = ICALTS-ID-D.using.data.intermediateDatum,
      ID2 = ICALTS-ID-TDF.using.data.intermediateDatum,
      AD = ICALTS-AD-P.using.data.additionalDatum
```

Tableau 5-26 La méthode de calcul de la donnée *ICALTS-ID-D*

4. L'indicateur CA est calculé à partir de deux indicateurs : *ICALTS-I-A* et *ICALTS-I-I*.

i) L'indicateur *ICALTS-I-I* représente le nombre d'actions faites dans une chaîne d'interaction *k* durant l'intervalle de temps *[ti-1, ti]*. La description de l'indicateur est la suivante :

```
<indicator type="ICALTS-I-I">
      <interactions ti="time"
            channel="integer">integer</interactions>
</indicator>
```

Tableau 5-27 Le format de l'indicateur *ICALTS-I-I*

La méthode de calcul de cette donnée est présentée dans le tableau 5-28.

```
cal {
      for each $k in AD.channel[id] do {
          for each $value in [1;A.n-1] do {
                $ti1 = ID1.t0 + value * ID2.d;
                $ti0 = ID1.t0 + (value-1) * ID2.d;
                I.interactions[time=ti1][channel=k] =
              count(R.action as filter(R.heure <= ti1
              and R.heure > ti0 and R.ressource == k));
          }
        }
      }
where  I = ICALTS-I-I.using.data.indicator,
      A = ICALTS-AD-P.using.data.additionalDatum,
      D = ICALTS-AD-C.using.data.additionalDatum,
      ID1 = ICALTS-ID-TDF.using.data.intermediateDatum,
      ID2 = ICALTS-ID-D.using.data.intermediateDatum,
      R = ICALTS-RD-Action.using.data.rawDatum
```

Tableau 5-28 La méthode de calcul de l'indicateur *ICALTS-I-I*

Cette donnée (*ICALTS-I-I*) est produite à partir de cinq données :

- la donnée additionnelle *ICALTS-AD-P* (pour récupérer le paramètre *n* qui est nécessaire pour le calcul *ti*),
- la donnée additionnelle *ICALTS-AD-C* (pour récupérer les chaînes d'interaction),
- la donnée intermédiaire *ICALTS-ID-TDF* (pour récupérer *t0*),
- la donnée intermédiaire *ICALTS-ID-D* (pour récupérer *d*),
- la donnée brute *ICALTS-RD-Action* (pour calculer le nombre d'actions faites)

Dans la partie correspondant à la méthode de calcul de l'indicateur *ICALTS-I-I* :

- la boucle *for each $k in AD.channel[id] do* parcourt la liste des chaînes d'interaction.
- la boucle *for each $value in [1;A.n-1] do* a pour but de calculer les intervalles de temps *[ti-1, ti]* pour chaque chaine d'interaction *k*.
- la ligne *$ti1 = ID1.t0 + value * ID2.d* est *ti* dans l'expression *[ti-1, ti]*.
- la ligne *$ti0 = ID1.t0 + (value-1) * ID2.d* est *ti-1* dans cette expression.
- le nombre d'actions faites durant chaque intervalle de temps *[ti-1, ti] (I.interactions[time=ti1][channel=k])* est calculé par la fonction *count* dans l'expression (5.2) :

```
I.interactions[time=ti1][channel=k] =
     count(R.action as filter(R.heure <= ti1
     and R.heure > ti0 and R.ressource == k));          (5.2)
```

ii) L'indicateur *ICALTS-I-A* calcule le nombre d'utilisateurs qui ont interagi avec au moins une chaîne d'interaction *k* dans l'intervalle de temps *[ti-1, ti]*. La description de l'indicateur est la suivante :

```
<indicator type="ICALTS-I-A">
     <nbagents time="string"
          channel="integer">integer</nbagents>
</indicator>
```

Tableau 5-29 Le format de l'indicateur *ICALTS-I-A*

La méthode de calcul de cette donnée est présentée dans le tableau 5-30. De façon similaire, cet indicateur est également calculé à partir de cinqs données ci-dessus. Le nombre d'utilisateurs qui interagi avec au moins une chaîne d'interaction *k* dans l'intervalle de temps *[ti-1, ti]* (*I.nbagents[time=ti1][channel=k]*) est calculé par la fonction *count* dans l'expression (5.3):

```
I.nbagents[time=ti1][channel=k] =
        countDis(R.acteur as filter(R.heure <= ti1
         and R.heure > ti0 and R.resource == k) );          (5.3)
```

```
cal {
     for each $k in AD.channel[id] do {
          for each $value in [1;A.n-1] do {
               $ti1 = ID1.t0 + value * ID2.d;
               $ti0 = ID1.t0 + (value-1) * ID2.d ;
               I.nbagents[time=ti1][channel=k] =
            countDis(R.acteur as filter(R.heure <= ti1
            and R.heure > ti0 and R.resource == k) );
          }
      }
    }
where I = ICALTS-I-A.using.data.indicator,
     A = ICALTS-AD-P.using.data.additionalDatum,
     AD = ICALTS-AD-C.using.data.additionalDatum,
     ID1 = ICALTS-ID-TDF.using.data.intermediateDatum,
     ID2 = ICALTS-ID-D.using.data.intermediateDatum,
     R = ICALTS-RD-Action.using.data.rawDatum
```

Tableau 5-30 La méthode de calcul de l'indicateur *ICALTS-I-A*

Finalement, nous calculons l'indicateur *CA* par la définition de la méthode de calcul de l'indicateur *ICALTS-I-CA* dans le champ *Getting.method.tool.description. DCL4UTL-formula* de la facette *Getting* (cf. tableau 5-18). La méthode de calcul de l'indicateur est décrite dans le tableau 5-31. Cet indicateur est calculé en se basant sur les deux indicateurs calculés *ICALTS-I-A* et *ICALTS-I-I* et est produit selon les *ti*. Ces *ti* sont déjà calculés dans deux indicateurs *ICALTS-I-I et ICALTS-I-A*.

```
    cal {
         for each $ti in I1.interactions[time] do {
            $sum = 0;
            for each $k in AD.channel[id] do {
               sum = sum +
                  (I1.interactions[time==ti][channel==k]*
                    I2.nbagents[time==ti][channel==k]);
             }
            I.ca[time=ti]=sum; // CA(ti) = ∑_{k=1}^{kmax} A(k,ti) * I(k,ti)
         }
       }
    where I = ICALTS-I-CA.using.data.indicator,
         I1 = ICALTS-I-I.using.data.indicator,
         I2 = ICALTS-I-A.using.data.indicator,
         AD = ICALTS-AD-C.using.data.additionalDatum
```

Tableau 5-31 La méthode de calcul de l'indicateur *CA*

Dans la partie correspondant à la méthode de calcul de l'indicateur *ICALTS-I-CA* :

- la boucle *for each $ti in I1.interactions[time] do* parcourt la liste des *ti* dans l'indicateur *ICALTS-I-I*. Cette liste peut être également extraite à partir de l'indicateur *ICALTS-I-A*.
- la boucle *for each $k in AD.channel[id] do* parcourt la liste des chaînes d'interaction afin de produire la somme $CA(ti) = \sum_{k=1}^{k_{max}} A(k,ti) * I(k,ti)$ pour chaque *ti*. Cette somme est calculée à traves de l'expression (5.4) :

```
sum = sum +
  (I1.interactions[time==ti][channel==k]*
     I2.nbagents[time==ti][channel==k]);
```
(5.4)

- dans l'expression (5.4) : *I1.interactions[time==ti][channel==k]* récupère le nombre le nombre d'actions faites dans une chaîne d'interaction *k* durant l'intervalle de temps *[ti-1, ti]*. *I2.nbagents[time==ti][channel== k]* récupère le nombre d'utilisateurs ayant interagi avec au moins une chaîne d'interaction *k* dans l'intervalle de temps *[ti-1, ti]*.

5.5 Les fonctionnalités du langage

DCL4UTL est conçu en particulier pour combiner les données UTL afin de produire les données dérivées UTL et les indicateurs. Un autre objectif du langage est de faciliter la capitalisation et la réutilisation des méthodes de calcul obtenues lors de la modélisation d'indicateurs. Cette section présente les possibilités des langages UTL et DCL4UTL dans ce sens.

5.5.1 Réutilisation de la méthode de calcul d'un indicateur

En plus des compétences et expériences requises pour la définition des indicateurs, la description de la méthode de calcul de certains indicateurs peut prendre du temps [Diagne (2008)]. Donc, notre objectif est de faciliter cette description en réutilisant des méthodes de calcul d'indicateurs dans différentes situations, activités ou plateformes d'apprentissage. Cette question est également traitée dans le travail de [Diagne (2008)]. Cette dernière propose qu'un indicateur soit élaboré à partir de trois vues : sélection, analyse et visualisation. Mais une seule partie de l'analyse est réutilisée et un indicateur n'est réutilisé que dans des situations d'apprentissage similaires. L'approche proposée par [Djouad (2008)] utilise une séquence de transformations des M-Traces [Settouti et al. (2006)] à partir de la M-Trace première vers le modèle de l'indicateur à élaborer. Toute séquence peut être réutilisée pour recalculer le même indicateur dans d'autres plateformes d'apprentissage. Une partie de cette séquence peut aussi être réutilisée pour calculer un nouvel indicateur dans la même

plateforme. Cependant cette approche propose un ensemble limité d'opérations (le matching, la sélection, l'élagage de modèle, la réécriture d'une trace, la fusion) [Djouad et al. (2009)].

Notre langage DCL permet de réutiliser la méthode de calcul des indicateurs dans une optique similaire à [Djouad (2008)] (c'est-à-dire qu'une partie de la méthode est réutilisable dans une même plate-forme et toute méthode peut être réutilisée dans d'autres plateformes en modifiant légèrement un indicateur existant).

Dans notre approche, nous décrivons des indicateurs par l'utilisation de données intermédiaires. Les différents indicateurs peuvent partager une ou plusieurs données intermédiaires. Ainsi, la méthode de calcul de chaque donnée intermédiaire est considérée comme une partie réutilisable de la méthode de calcul d'un indicateur existant pour calculer un nouvel indicateur dans la même plate-forme. Nous présentons maintenant un exemple de réutilisation de toute la méthode de calcul d'un indicateur dans d'autres plateformes.

Nous notons M, M_I et M_{ID} les méthodes de calcul correspondant à une donnée dérivée, un indicateur et une donnée intermédiaire. Ces méthodes sont formalisées à partir de deux éléments : un ensemble des combinaisons C et un ensemble des données D. Les combinaisons C manipulent les données D :

$$M = f(C, D) \tag{5.5}$$

L'ensemble des données D est défini pour chaque donnée dérivée :

$$D = \{Rd, Ad, Cd, Id, I\} \tag{5.6}$$

Avec :

- *Rd* : un ensemble de données brutes UTL transformées à partir de traces générées par un EIAH,
- *Ad* : un ensemble de données additionnelles UTL,
- *Cd* : un ensemble de données de production UTL,
- *Id* : un ensemble de données intermédiaires produites à partir de la combinaison,
- *I* : un ensemble d'indicateurs produits à partir de la combinaison.

Dans la définition (5.5), l'ensemble des combinaisons C peut éventuellement être réutilisé pour la modélisation des données dérivées par n'importe quel EIAH. Tandis que l'ensemble

de données *D* de type donnée brute d'UTL devrait être adaptée à chaque EIAH au travers de son composant ''track'' (c'est un élément de la donnée brute UTL, cf. l'annexe D).

Nous considérons la modélisation d'un même indicateur, mais dans deux contextes différents A et B (c'est-à-dire deux dispositifs d'apprentissage A et B). Deux environnements d'apprentissage différents utilisent souvent des formats de traces différents. La partie suivante indique comment un indicateur I_b dans un contexte B peut être modélisé en réutilisant la méthode de calcul du même indicateur existant I_a d'un contexte A.

Nous supposons qu'I_a est existant. Selon (5.5), la méthode de calcul de l'indicateur I_a modélisée à partir des traces générées par un EIAH A est formellement décrite comme suit :

$$M_{Ia} = f(C, D_A) \tag{5.7}$$

Selon la définition (5.6) :

$$D_A = \{Rd_A, Ad_A, Cd_A, Id_A, I_A\} \tag{5.8}$$

De façon similaire, la méthode de calcul du même indicateur nommé I_b modélisé à partir des traces générées par un EIAH *B,* est formellement décrite comme suit :

$$M_{Ib} = f(C, D_B) \tag{5.9}$$

Où :

$$D_B = \{g(Rd_B), Ad_B, Cd_B, Id_B, I_B\} \tag{5.10}$$

Avec :

g : est une fonction qui transforme un ensemble de données brutes Rd_B en données comme données brutes Rd_A.

Seules les données brutes *Rd* sont à adapter, les autres données (Id_B, I_B) sont inchangées car elles reposent sur les *Rd*. Les données Ad_B et Cd_B peuvent être modélisées avec le même format que les données Ad_A ainsi que Cd_A, parce que ces données sont souvent créées par l'analyste.

En utilisant les définitions (5.7), (5.8), (5.9) et (5.10), nous spécifions la méthode de modélisation de l'indicateur I_b en réutilisant la même méthode de calcul que l'indicateur I_a mais sur un ensemble de données initiales différentes. Plus concrètement, nous présentons maintenant un exemple qui décrit cette réutilisation.

Exemple

Dans cette partie, nous considérons le même indicateur *‘‘le parcours par apprenant’’* dans deux EIAH différents : le système VAPS et un environnement réalisé au laboratoire SysCom. Nous commençons par décrire la méthode de calcul de cet indicateur dans le système VAPS. Ensuite, cette méthode est réutilisée pour modéliser le même indicateur dans un environnement réalisé au laboratoire SysCom.

La modélisation de l’indicateur ‘‘*le parcours par apprenant*’’ dans le système VAPS

Le système VAPS génère des traces pour les actions de chaque patient dans un fichier de log. Chaque trace se compose de trois parties : *l’action*, *l’instant de réalisation* de l’action par le patient et *l’évaluation d’action*. Ces traces sont représentées de la façon suivante :

```
Démarrage  00m 00s 525ms
Intrusion  00m 50s 168ms   mauvaise
Prise des artichauts  00m 57s 593ms   bonne
Prise des pommes01m 31s 391ms   bonne
```

Tableau 5-32 Exemple de traces VAPS concernant des actions d’un patient

Pour la modélisation de l’indicateur *‘‘le parcours par apprenant’’* (nommé *VAPS-I-Parcours*), ces traces sont transformées en une donnée brute UTL nommée *‘‘VAPS-RD-Action’’* sous la forme :

```
<rawDatum type="VAPS-RD-Action">
          <patient>string</patient>
          <action>string</action>
          <pointDeControle>string</pointDeControle>
          <evaluation>string</evaluation>
</rawDatum>
```

Tableau 5-33 Le format de la donnée brute *VAPS-RD-Action*

L’indicateur nommé *VAPS-I-Parcours* est calculé avec l'aide de cette donnée brute (*VAPS-RD-Action*) et d’une donnée additionnelle (*VAPS-AD-Utilisateur*) qui décrit la liste des patients. Nous déclarons donc les trois données dans la partie *where* (cf. tableau 5-35). I, R, AD sont des alias. *VAPS-I-Parcours*, *VAPS-RD-Action*, *VAPS-AD-Utilisateur* sont les noms des données UTL. Le format du résultat de cet indicateur est le suivant :

```
<indicator type="VAPS-I-Parcours">
     <parcours identifiant="string"> string
     </parcours>
</indicator>
```

Tableau 5-34 Le format de l'indicateur *VAPS-I-Parcours*

Sa méthode de calcul est décrite de la façon suivante :

```
cal { for each $u in AD.utilisateur do  {
                 $parcours =""  ;
                 for each  $a in R.action as
                                  filter(R.patient==u) do
                   {
                       concat(parcours, a) ;
                       concat(parcours, "-->") ;
                   }
                I.parcours[identifiant=u]= parcours ;
         }
      }
Where
    I = VAPS-I-Parcours.using.data.indicator,
    R=  VAPS-RD-Action.using.data.rawDatum,
    AD= VAPS-AD-Utilisateur.using.data.additionalDatum
```

Tableau 5-35 La méthode de calcul de l'indicateur *VAPS-I-Parcours*

Dans la partie correspondant à la méthode de calcul du tableau 5-35, le parcours par apprenant est calculé pour chaque patient. Il s'agit de regrouper les données *VAPS-RD-Action* par patient et d'en combiner les actions. La boucle *for each* permet donc de parcourir la liste des patients : *for each $u in AD.utilisateur do.* Ensuite, pour chaque patient *u* (*identifiant=u*), le parcours (I.*parcours[identifiant=u]*) est calculé par l'expression :

```
I.parcours[identifiant=u]= parcours
```
(5.11)

Dans cette méthode (le tableau 5-35), la fonction ''*concat*'' prédéfinie de DCL4UTL est utilisée pour concaténer les actions d'un patient.

La modélisation de l'indicateur ''*le parcours par apprenant*'' dans un environnement réalisé au laboratoire Syscom

L'environnement réalisé au laboratoire Systèmes Communicants génère des traces pour les activités de tous les étudiants dans un même fichier de log comprenant deux types de traces. Le premier spécifie que l'étudiant est en train d'effectuer une action. Le deuxième

décrit le changement d'activité par un étudiant. La figure 5-4 donne un exemple de ces deux traces.

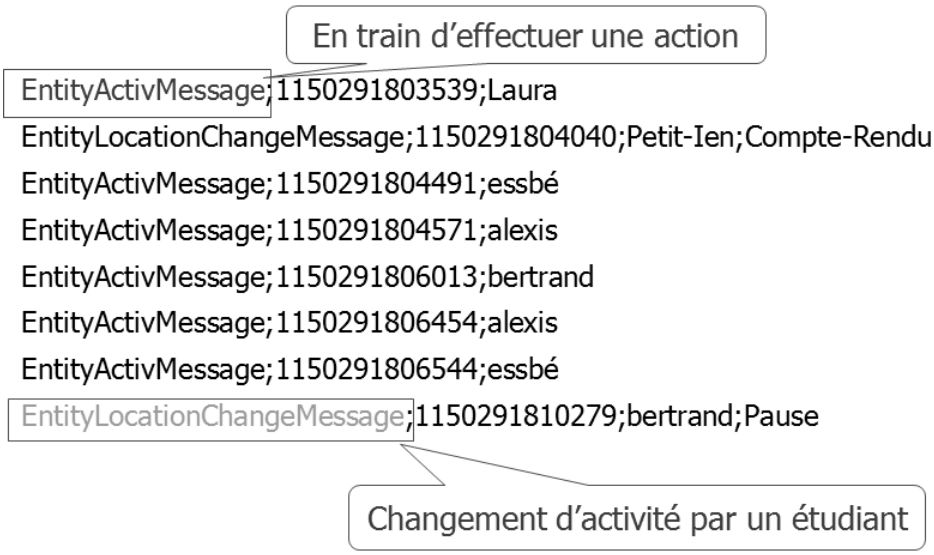

Figure 5-4 Les traces générées par l'environnement réalisé au laboratoire SysCom

Donc, pour le calcul de l'indicateur *"le parcours par apprenant"* nommé *Syscom-I-Parcours*, nous ne nous intéressons qu'au deuxième type de traces. Chaque trace de ce type se compose de quatre champs : *un mot-clé*, *la date de début de réalisation de l'activité par l'étudiant, l'identifiant* de *l'étudiant concerné et le nom de l'activité*. Cette trace est transformée en une donnée brute UTL nommée *Syscom-RD-DebSess* sous la forme comme suivante :

```
<rawDatum type="Syscom-RD-DebSess">
      <date>string</date>
      <etudiant>string</etudiant>
      <activite>string</activite>
</rawDatum>
```

Tableau 5-36 Le format de la donnée brute *Syscom-RD-DebSess*

Pour réutiliser la méthode de calcul de l'indicateur *VAPS-I-Parcours* dans la modélisation de l'indicateur *Syscom-I-Parcours*, une donnée intermédiaire doit être créée. Cette donnée nommée *Syscom-ID-Activite* a pour but de transformer la donnée brute *Syscom-RD-DebSess* en une donnée dont le format est similaire au format de la donnée brute *VAPS-RD-Action*. La description de cette donnée intermédiaire est proposée ci-dessus. Nous ne nous intéressons qu'aux champs nécessaires pour le calcul de l'indicateur *Syscom-I-Parcours.*

```
<intermediateDatum type="Syscom-ID-Activite">
          <patient>string</patient>
          <action>string</action>
          <pointDeControle>time</pointDeControle>
</intermediateDatum>
```

Tableau 5-37 Le format de la donnée intermédiaire *Syscom-ID-Activite*

En considérant cette description de la donnée intermédiaire, la méthode de calcul associée est décrite dans le tableau 5-38. Cette donnée est produite avec l'aide des données brutes de type *Syscom-RD-DebSess*.

```
cal {
    for each $i in R.date do {
       $Rtemp=R.* as filter(R.date==i);
       ID.patient= Rtemp.etudiant ;
       ID.action= Rtemp.activite ;
       ID.pointDeControle= i;
    }
}
Where
    ID = Syscom-ID-Activite.using.data.intermediate,
    R=   Syscom-RD-DebSess.using.data.rawDatum
```

Tableau 5-38 La méthode de calcul de la donnée intermédiaire *Syscom-ID-Activite*

La partie correspondant à la méthode de calcul du tableau 5-38 transforme les données de l'élément *etudiant* en données de l'élément *patient* (*ID.patient= Rtemp.etudiant*), les données de l'élément *activite* en données de l'élément *action* (*ID.action= Rtemp.activite*) et les données de l'élément *date* en données de l'élément *pointDeControle* (*ID.pointDeControle= i*).

La ligne *$Rtemp=R.* as filter(R.date==i)* sélectionne tous les éléments d'une donnée brute pour laquelle la date est *i.*

En ce qui concerne la méthode de calcul de l'indicateur *Syscom-I-Parcours* (cf. tableau 5-39), ce dernier possède le même format que celui de *VAPS-I-Parcours*, sauf que le nom d'indicateur est différent.

```
<indicator type="Syscom-I-Parcours">
     <parcours identifiant="string"> string </parcours>
</indicator>
```

Tableau 5-39 Le format de l'indicateur *Syscom-I-Parcours*

Sa méthode de calcul est décrite dans le tableau 5-40. Dans cette description, l'ensemble des combinaisons *C* (cf. définition (5.5)) est réutilisé tandis que l'ensemble des données *D* a été changé. La donnée brute *VAPS-RD-Action* est remplacée par la donnée intermédiaire *Syscom-ID-Activite*. La donnée additionnelle *VAPS-AD-Utilisateur* est remplacée par *Syscom-AD-Utilisateur*. Ces deux données additionnelles sont différentes, mais elles possèdent le même format de donnée. Cet indicateur *Syscom-I-Parcours* est alors modélisé en réutilisant la même méthode de calcul que l'indicateur *VAPS-I-Parcours* mais sur un ensemble de données initiales différentes.

L'ensemble des combinaisons C

```
cal { for each $u in AD.utilisateur do  {
                $parcours =""  ;
                for each  $a in R.action as
                                filter(R.patient==u) do
                  {
                     concat(parcours, a) ;
                     concat(parcours, "-->") ;
                  }
               I.parcours[identifiant=u]= parcours ;
        }
     }

where                                          L'ensemble des données D
    I = Syscom-I-Parcours.using.data.indicator,
    R=  Syscom-ID-Activite.using.data.intermediateDatum,
    AD= Syscom-AD-Utilisateur.using.data.additionalDatum
```

Tableau 5-40 La méthode de calcul de l'indicateur *Syscom-I-Parcours*

Cet exemple présente le même indicateur dans deux EIAH différents. L'objectif d'observation n'est pas le même entre deux EIAH. Dans le système VAPS, l'objectif d'observation est d'évaluation. Cet indicateur permet d'évaluer la connaissance sémantique liée à l'action du patient. Dans l'environnement réalisé au laboratoire Syscom, l'objectif d'observation est la régulation et la réingénierie. Cet indicateur permet d'étudier l'évolution d'un étudiant dans la liste des activités préconisées pour le scénario pédagogique.

5.5.2 Donnée intermédiaire paramétrée

Comme nous l'avons mentionné dans le chapitre 2, les données intermédiaires UTL sont calculées à partir des traces et sont nécessaires pour l'élaboration d'indicateurs. Pour faciliter la modélisation de la méthode de calcul d'indicateurs utilisant les données intermédiaires, DCL4UTL permet de créer des données intermédiaires paramétrées (Parameterized Intermediate Data - PID). Une PID est proche d'un template UML [OMG (2007)].

Dans les modèles UML, les templates sont des éléments de modèle comportant des paramètres formels. Ceci permet de définir une famille de classes par liaison des paramètres formels à des valeurs actuelles (appelé le *bind*). Une classe ne devient donc effective que lorsque cette liaison (*bind*) est faite. Le principe du paramétrage peut aussi être utilisé pour paramétrer d'autres éléments tels que les packages, les opérations et même les expressions. Les éléments paramétrables sont appelés *TemplateableElement*.

Exemple : un template définit une liste d'éléments de type générique avec les paramètres formels *T* du type générique. Par la liaison, on peut construire une classe *ListeAdresses* (la figure 5-5).

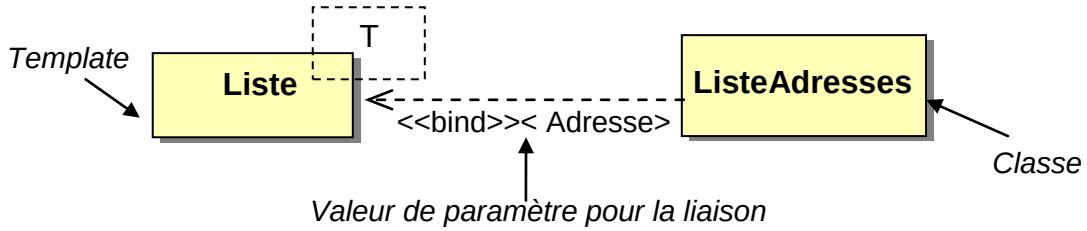

Figure 5-5 La déclaration et liaison d'un paramètre en UML

Une PID est une donnée intermédiaire UTL calculée selon la valeur des paramètres (le paramètre est l'un des quatre types opérandes de DCL4UTL). Elle est en quelque sorte un patron de donnée, mais elle peut jouer le rôle d'une fonction/méthode des langages de programmation. La différence avec la fonction/méthode traditionnelle est que la PID permet de capitaliser la méthode d'acquisition des données et leurs valeurs ensemble. L'objectif principal de la PID est de faciliter la modélisation des méthodes de calcul d'indicateurs similaires dans un même contexte ainsi que la réutilisation des données intermédiaires dans la modélisation d'indicateurs. Ces PID doivent être utilisées dans les indicateurs, elles ne peuvent pas être calculées de façon indépendante.

Dans la spécification de la grammaire du langage DCL4UTL présentée dans la section 5.3, la règle de production <*FormalParamsList*> décrit la création de paramètres d'une PID, tandis que la règle de production <*RealParameterList*> décrit l'utilisation de ces paramètres dans les indicateurs. Par conséquent, les règles pour <*FormalParamsList*> sont déclarées dans les données intermédiaires UTL et les règles pour <*RealParameterList*> sont utilisées dans les indicateurs. Pour créer un PID, sa méthode de calcul doit contenir une partie ‘‘*parameter”*. Dans cette partie, les paramètres sont décrits sous la forme d'une liste *“variable = parameter”*. La *variable* est aussi l'un des quatre types opérandes de DCL4UTL. Elle joue le rôle d'un argument formel. Ces paramètres prennent les valeurs réelles à partir

des indicateurs. Ces PID sont ensuite déclarées dans la partie *"where"* des indicateurs avec les valeurs réelles des paramètres. Lors du calcul d'un indicateur, ces PID sont d'abord évaluées. Leurs résultats sont ensuite utilisés pour élaborer l'indicateur. La figure 5-6 présente ce mécanisme entre un indicateur et une PID. Nous présentons un exemple d'utilisation de la donnée PID dans le chapitre 7.

Les trois différences entre une PID et un template UML sont les suivantes :

i) Les templates UTL ne sont pas calculés, tandis que les PID sont calculés.
ii) Après le passage de paramètres entre un indicateur et une PID, cette dernière est évaluée pour produire ses valeurs. Ces valeurs sont ensuite utilisées par l'indicateur pour produire ses propres valeurs. En revanche, après la liaison entre un template UML et un élément de modèle, cet élément n'utilise aucun élément du template.
iii) La valeur de l'indicateur va changer selon les valeurs de la PID, par contre l'élément de modèle participant à la liaison ne change pas.

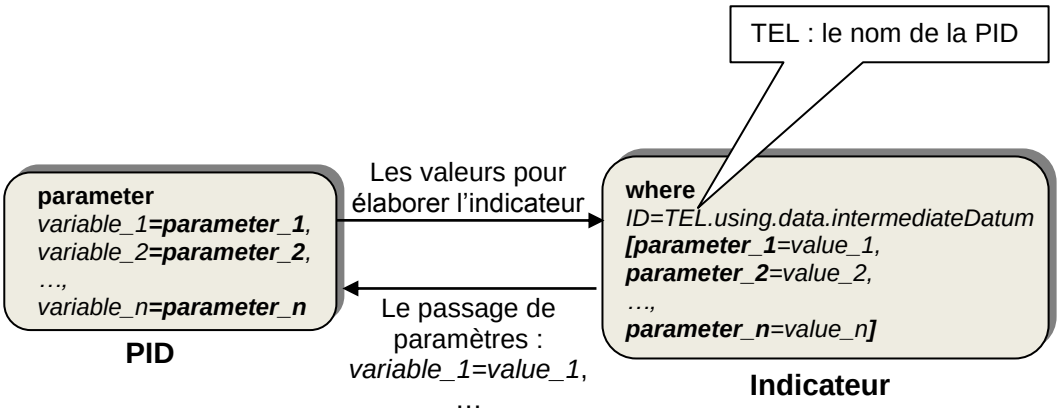

Figure 5-6 Le passage de paramètres et valeurs entre un indicateur et une PID

5.6 Conclusion

Les travaux présentés dans ce chapitre portent sur la proposition du langage de combinaison DCL4UTL. Il s'agit d'un langage de description des calculs permettant la production automatisée de ces indicateurs. Ce langage est utilisé pour créer des patrons de données exécutables et capitalisables. Ce langage est générique et intégré dans UTL. Il a été conçu pour être dans la lignée d'UTL, indépendant du format de représentation des traces, du langage de scénarisation, mais aussi de l'architecture et du type de base de données. C'est un langage qui permet l'intégration de fonctions externes et de faciliter la

génération d'outils d'analyse. DCL4UTL dispose également d'une possibilité de créer des données intermédiaires paramétrées ce qui est proche des fonctions traditionnelles des langages de programmation et des templates UML. La PID permet de capitaliser la méthode d'acquisition de la donnée et ses valeurs. Elle facilite aussi la modélisation de la méthode de calcul d'indicateurs. UTL permet la création des patrons de données réutilisables, mais ces patrons se limitent à la description. DCL4UTL est ajouté pour que ces patrons soient exécutables et afin que leurs méthodes de calcul soient aussi réutilisables dans différents contextes d'apprentissage. Le chapitre suivant présente un outil d'analyse de traces permettant d'interpréter ce langage pour produire les indicateurs.

6 Conception et implémentation d'un outil d'analyse

Sommaire

Ce chapitre présente d'abord l'architecture générale de l'outil d'analyse qui exécute le langage DCL4UTL. Cet outil est conçu pour faciliter l'activité d'observation de l'enseignant pendant la session. Puis, nous détaillons l'architecture du composant de calcul des données qui est une partie et aussi le noyau de l'outil d'analyse. Ensuite, nous présentons la façon d'intégrer des fonctions externes dans cet outil d'analyse. Finalement, deux applications de cet outil pour calculer des indicateurs après la session sont décrites avec les traces générées par le système VAPS et l'environnement réalisé au laboratoire SysCom décrits dans le chapitre 5.

6.1 Introduction

Dans le chapitre précédent, nous avons présenté le langage DCL4UTL, une extension d'UTL. Ce langage décrit formellement des moyens d'observation dans la modélisation d'indicateurs UTL sous forme de patron. Notre objectif était aussi de générer automatiquement des outils d'analyse pour exécuter ces patrons. Le langage DCL4UTL permet de décrire la méthode de calcul des données dérivées UTL en combinant d'autres données dans une forme indépendante du format de traces et de l'architecture de la base de données. Pour mettre à l'épreuve nos propositions, nous avons développé un outil d'analyse pour ce langage. Nos objectifs sont de :

- exécuter automatiquement des indicateurs,
- fournir à l'enseignant une collection d'indicateurs,
- offrir la possibilité d'intégrer des fonctions externes à notre outil.

En exécutant des indicateurs définis à partir des besoins d'observation de l'enseignant, l'outil aide ce dernier à pouvoir évaluer ce qu'il veut observer dans une session. Il permet également de capitaliser la méthode de calcul des indicateurs et leur résultat sous la forme de patrons de données. Cela permet aux enseignants/concepteurs de puiser des indicateurs dans une banque d'indicateurs afin d'étudier une session d'apprentissage, par exemple pour faire émerger les différences entre le scénario d'apprentissage prescrit et les scénarios qui ont été observés lors de la session. Lorsque plusieurs indicateurs sont modélisés et capitalisés, les savoir-faire et expériences peuvent alors être partagés et réutilisés entre les enseignants. Cela aide l'enseignant/concepteur dans la définition de ses besoins d'observation.

Dans les sections suivantes, nous présentons l'architecture générale de l'outil. Puis, nous détaillons l'architecture du composant de calcul des données qui est le noyau de notre outil d'analyse. Avant de détailler une application de l'outil sur les traces générées par le système VAPS et l'environnement réalisé au laboratoire SysCom, nous présentons la méthode d'intégration de fonctions externes dans notre outil.

6.2 Architecture générale de l'outil

Cette partie aborde l'architecture générale de l'outil d'analyse qui permet d'exécuter le langage DCL4UTL et d'évaluer des indicateurs. Elle décrit les fonctionnalités de chaque composant constituant l'outil d'analyse. Une version simple de cette architecture est présentée dans [Pham-Thi et al. (2009a)]. La figure 6-1 présente cette architecture. Elle se

compose de sept composants. Les composants encadrés par une ligne pointillée constituent le noyau de notre outil.

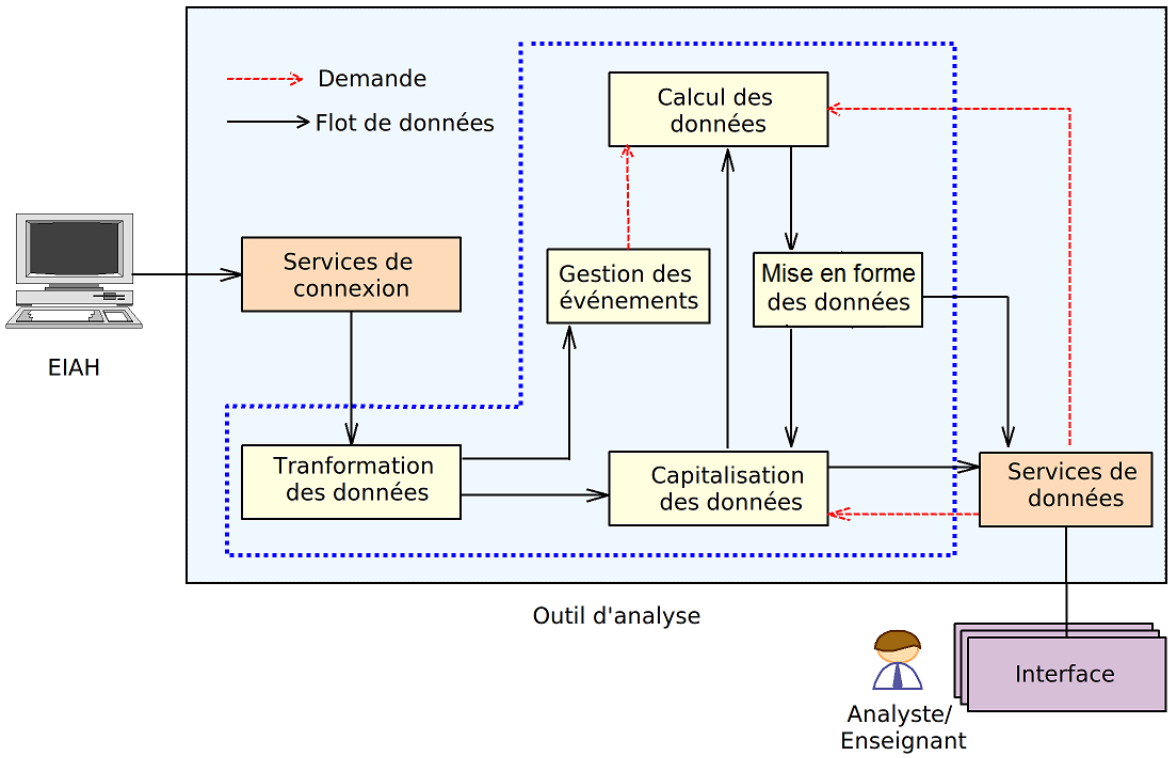

Figure 6-1 L'architecture générale de l'outil d'analyse des traces

Cette architecture est conçue en se basant le processus d'analyse présenté dans le chapitre 4. Chaque processus (à l'exception du processus de modélisation de données) du processus d'analyse correspond à un composant de cette architecture. Les autres composants sont ajoutés pour produire l'outil d'analyse. Ce dernier nécessite de communiquer avec des EIAH pour récupérer des traces. C'est pourquoi le composant ‘‘*Service de connexion”* est présent dans l'architecture. Il se connecte aux EIAH pour fournir des traces au composant ‘‘*Transformation des données”*. Ce dernier transforme les traces en données brutes et le composant ‘‘*Calcul des données”* produit les valeurs des données dérivées. Cependant, pour le calcul d'indicateurs en temps réel, l'outil d'analyse nécessite un composant pour déclencher automatiquement ce calcul. Comme nous l'avons présenté dans le chapitre précédent, ce calcul peut être déclenché quand un événement spécifique est généré par le dispositif d'apprentissage. Le composant ‘‘*Gestion des événements”* est donc ajouté dans ce but. Il se connecte au composant ‘‘*Transformation des données”* pour vérifier si des événements spécifiques sont générés dans les traces.

Les données produites par le composant ‘‘*Calcul des données”* doivent être formatées selon le champ *using.format* avant d'être capitalisées dans la base de données UTL. Le

composant ‘‘*Mise en forme des données”* réalise ce formatage. Il se connecte donc aux composants ‘‘*Calcul des données”* et ‘‘*Capitalisation des données”*.

Le composant ‘‘*Services de données”* joue un rôle intermédiaire entre l’utilisateur et l’outil d’analyse.

6.2.1 Le composant ‘‘Services de données’’

Le composant ‘‘*Services de données*’’ fournit des services qui permettent d'envoyer des demandes de calcul au composant ‘‘*Calcul des données”* et de recevoir les résultats de calcul de ce dernier. Il fournit également des fonctionnalités nécessaires pour ajouter de nouveaux indicateurs et/ou de nouvelles données intermédiaires modélisées dans la base de données UTL via le composant ‘‘*Capitalisation des données”*. Il comprend également des services qui permettent d'envoyer des requêtes simples (par exemple, la consultation d’une donnée UTL) au composant ‘‘*Capitalisation des données”* pour interroger la base de données UTL.

6.2.2 Le composant ‘‘Interface’’

Le composant ‘‘*Interface”* se connecte à celui de ‘‘*Services de données”* afin de fournir des services aux utilisateurs (l’analyste et/ou l’enseignant). Il aide l'analyste à recharger ou ajouter les données modélisées nécessaires pour calculer des indicateurs. Ce composant se compose également de fonctionnalités qui permettent notamment à l'enseignant de voir ou revoir le résultat des indicateurs, de choisir des indicateurs à exécuter ou à ré-exécuter après la session, ainsi que de consulter les données capitalisées. Ce composant ‘‘*Interface”* peut être un composant d'autres outils qui se connectent au composant ‘‘*Services de données”* pour utiliser des indicateurs.

6.2.3 Le composant ‘‘Service de connexion’’

Ce composant est conçu dans le but d'échanger des données entre notre outil d'analyse et un EIAH. Il collecte les traces générées par l'EIAH et les envoie au composant ‘‘*Transformation des données”*.

6.2.4 Le composant ‘‘Transformation des données’’

Les traces générées par des environnements d'apprentissage sont variées (fichiers textuels, XML, base de données, etc.). Ce composant a donc pour le but de transformer les traces recueillies par un EIAH via le composant ‘‘*Service de connexion”* en données UTL. Cette

transformation est réalisée en se basant sur les données primaires modélisées dans le processus de modélisation de données (cf. chapitre 4).

6.2.5 Le composant ‘‘Calcul des données’’

La méthode de calcul des indicateurs est décrite par une combinaison de données UTL via le langage DCL4UTL (l'élément *Getting.method.tool.description.DCL4UTL-formula*). Ce composant exécute cette combinaison pour produire une nouvelle donnée (indicateur ou donnée intermédiaire). C'est le cœur de notre outil. Il est détaillé dans la section 6.3.

6.2.6 Le composant ‘‘Mise en forme des données’’

Les données sorties du composant ‘‘*Calcul des données”* (c'est-à-dire le résultat des indicateurs ou des données intermédiaires) sont utilisées comme les entrées du composant ‘‘*Mise en forme des données*’’. Ce dernier structure ces données conformément à la définition des données dérivées (*using.format*). Ces données structurées sont capitalisées par le composant ‘‘*Capitalisation des données”* pour une utilisation prochaine. Ces données peuvent également être envoyées au composant ‘‘*Service de données”* qui les affiche à l'utilisateur via le composant ‘‘*Interface”*.

6.2.7 Le composant ‘‘Capitalisation des données’’

Toutes les données modélisées avec UTL et possédant éventuellement des méthodes de calcul sont capitalisées par ce composant. Il joue le rôle d'un répertoire de données. Il fournit des données nécessaires au composant ‘‘*Calcul de données”* et capitalise les données produites par la combinaison des données et formatées par le composant ‘‘*Mise en forme des données*’’.

6.2.8 Le composant ‘‘Gestion des événements’’

Ce composant est particulièrement conçu pour le calcul d'indicateurs en temps réel. Il se compose de services permettant de gérer des événements qui déclenchent le calcul des indicateurs en temps réel. Ce calcul peut être lancé lors de la réception d'une donnée particulière, après un intervalle de temps donné ou selon une demande de l'utilisateur. Comme nous l'avons présenté dans le chapitre 5, UTL dispose d'un champ qui permet de gérer le déclenchement du calcul d'un indicateur en temps réel. Ce composant travaille en lien avec ce champ. Il vérifie la valeur de ce champ pour tous les indicateurs à calculer dans une session. Il regroupe ces indicateurs en plusieurs groupes selon les événements de calcul (c'est-à-dire une donnée particulière, un intervalle de temps donné ou une demande

de l'utilisateur). Le calcul des indicateurs dans le même groupe est lancé quand ce composant reçoit l'événement correspondant.

6.3 Architecture du composant ‘‘Calcul des données’’

Le composant principal de notre outil d'analyse (cf. figure 6-1) est celui du *Calcul des données*. Il se compose de quatre composants : ‘‘*Analyse des formules de calcul*’’, ‘‘*Combinaison des données*’’, ‘‘*Gestion des fonctions externes*’’ et ‘‘*Services d'adaptation*’’. La figure 6-2 illustre son architecture.

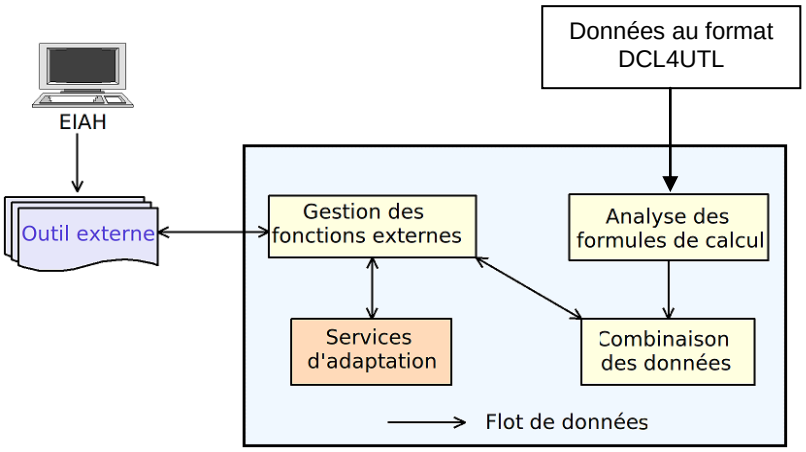

Figure 6-2 L'architecture de composant ‘‘*Calcul des données*’’

Le composant ‘‘*Analyse des formules de calcul*’’ lit la méthode de calcul des données dérivées écrite en DCL4UTL et vérifie la syntaxe de la méthode de calcul. Ces méthodes proviennent de données fournies par le composant ‘‘*Capitalisation des données*’’ (cf. figure 6-1).

Ensuite, si la syntaxe DCL4UTL utilisée pour décrire la méthode de calcul est correcte, le composant ‘‘*Combinaison des données*’’ exécute l'expression de ce calcul et produit les résultats correspondants. Ces résultats sont envoyés au composant ‘‘*Mise en forme des données*’’ (cf. figure 6-1).

Le calcul d'indicateurs peut utiliser les opérateurs/fonctions internes (c'est-à-dire les fonctions prédéfinies du langage DCL4UTL) et/ou les opérateurs/fonctions externes. Le composant ‘‘*Gestion des fonctions externes*’’ gère, exécute des fonctions externes et retourne leur résultat au composant ‘‘*Combinaison des données*’’.

Nous distinguons deux types d'opérateurs externes :

- Les opérateurs du premier type retournent les résultats dont le type de donnée est numérique ou alphanumérique. Nous les classons dans les *fonctions externes*

simples. Le résultat de ces opérateurs/fonctions est donc directement utilisé par le composant ‘‘*Combinaison des données”* pour le calcul de l'indicateur.

- Les opérateurs du deuxième type retournent les résultats dont le type de données est complexe (par exemple, un objet, un tableau, une liste, etc.). Nous les classons dans les *fonctions externes complexes*. Le résultat de ces opérateurs/fonctions doit être traité par le composant ‘‘*Services d'adaptation”* avant d’être utilisé par le composant ‘‘*Combinaison des données”*. Le composant ‘‘*Services d'adaptation”* est abordé dans la section suivante.

6.4 Intégration des outils externes

Comme nous l’avons présenté dans le chapitre 5, l'intégration des outils d'analyse externes correspond à l'utilisation des fonctions/opérateurs de ces outils comme un élément du langage DCL4UTL pour spécifier le calcul des indicateurs. L'objectif de l'intégration des outils d'analyse externes est de pouvoir réutiliser des méthodes d'analyse de données, des fonctions disponibles, etc. avec peu de modifications. Cette intégration permet notamment d’utiliser des fonctions trop complexes pour être déclarées. Le composant ‘‘*Gestion des fonctions externes”* a pour but de gérer et d’exécuter ces fonctions. Cependant, le composant ‘‘*Combinaison des données”* (cf. figure 6-2) ne traite pas directement des données sorties des *opérateurs/fonctions externes complexes*. Donc, le composant ‘‘*Services d'adaptation”* est conçu pour fournir des services permettant de produire les données compatibles pour que le composant ‘‘*Combinaison des données”* puisse opérer le calcul de l'indicateur. Une donnée compatible est une donnée que le langage DCL4UTL peut utiliser (c’est-à-dire une valeur numérique, alphanumérique ou sous la forme d’une donnée UTL). C’est un des opérandes du langage. Les règles de reconnaissance de ces données sont définies dans la grammaire de DCL4UTL.

L'objectif principal du composant ‘‘*Services d'adaptation”* est de faire des fonctions externes complètement transparentes par rapport au système interne. C'est-à-dire que plusieurs fonctions externes peuvent être utilisées et intégrées sans modifier le composant ‘‘*Combinaison des données”*. Le composant ‘‘*Services d'adaptation”* lit la spécification de la donnée sortie des *fonctions externes complexes* utilisées afin de générer automatiquement des données compatibles sous la forme d’une donnée UTL (c’est-à-dire la donnée intermédiaire). Le composant ‘‘*Combinaison des données”* peut alors traiter ces données comme des opérandes de DCL4UTL. L'utilisation des *fonctions externes complexes* nécessite de les spécifier lors de la modélisation des indicateurs. Cette spécification est faite par l’analyste. Le tableau 6-1 présente cette spécification en XML.

```
<fonctions>
  <fonction name="string" location= "string">
    <returnType>
           <type name="string" dataType="string"
                          array="boolean" dimension="int"/>
           <field name="string" type="string"
                         object="boolean" parent="string"/>
    </returnType>
  </fonction>
</fonctions>
```

Tableau 6-1 La spécification du type de retour des fonctions externes

Cette spécification décrit le prototype de chaque fonction comprenant :

- Le nom de la fonction externe (élément *fonction*, attribut *name*),
- La localisation (élément *fonction*, attribut *location*) de laquelle la fonction externe provient (par exemple d'un fichier java, d'un fichier jar).
- Le détail de type du retour de la fonction (*returnType*) comme son nom de type (élément *type*, attribut *name*), ses attributs (*field*).
- Au cas où le type de retour serait un tableau (une suite d'éléments de même type dont les adresses mémoires sont consécutives), la valeur de l'attribut *array* est vraie (*true*) et l'attribut *dimension* donne le nombre de dimensions du tableau. Si le type de retour est une liste ou un vecteur, la valeur de l'attribut *dataType* décrit le type de donnée des éléments d'une liste (ou d'un vecteur).
- Pour chaque attribut, son nom et son type de donnée sont présents. Si l'attribut est un objet (la valeur de l'attribut *object* est vraie), il est nécessaire d'associer son nom (*parent*) à chacun de ses attributs. Cette règle est appliquée pour tous les sous-attributs qui sont des objets.

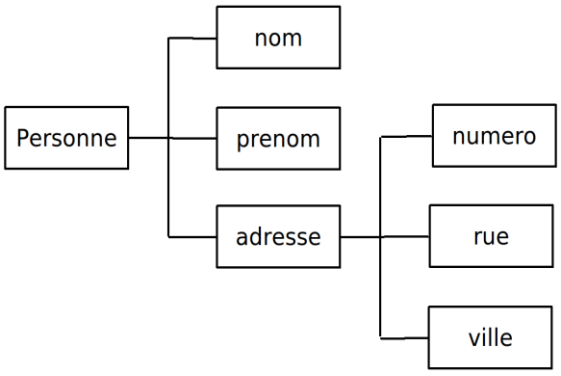

Figure 6-3 Exemple de type de retour d'une fonction externe

La figure 6-3 donne comme exemple, la sortie d'une *fonction externe complexe getPersonne*. Cette fonction provient d'un fichier, par exemple *Fonctions.java*. Le tableau 6-2 décrit sa

spécification. Le tableau 6-3 décrit la spécification d'une fonction externe *getListePersonne* qui retourne une liste de *Personne*. Le tableau 6-4 décrit la spécification d'une fonction externe *getTableau* qui retourne un tableau unidimensionnel de nombre entier.

```
<fonctions>
   <fonction name="getPersonne" location="Fonctions.java" >
     <returnType>
            <type name="Personne" dataType="" array="false"
                    dimension="0"/>
            <field name="nom" type="string" object="false"
                                parent=""/>
            <field name="prenom" type="string" object="false"
                                parent=""/>
            <field name="adresse" type="Adresse" object="true"
                                parent=""/>
            <field name="numero" type="string" object="false"
                                parent="adresse"/>
            <field name="rue" type="string" object="false"
                                parent="adresse"/>
            <field name="ville" type="string" object="false"
                                parent="adresse"/>
      </returnType>
   </fonction>
</fonctions>
```

Tableau 6-2 Exemple de spécification d'un type de retour d'une fonction externe pour un objet

```
<fonctions>
   <fonction name="getListePersonne"
                              location="Fonctions.java" >
     <returnType>
            <type name="List" array="false"
                    dataType="Personne"    dimension="0"/>
            <field name="nom" type="string" object="false"
                                parent=""/>
            <field name="prenom" type="string" object="false"
                                parent=""/>
            <field name="adresse" type="Adresse" object="true"
                                parent=""/>
            <field name="numero" type="string" object="false"
                                parent="adresse"/>
            <field name="rue" type="string" object="false"
                                parent="adresse"/>
            <field name="ville" type="string" object="false"
                                parent="adresse"/>
      </returnType>
   </fonction>
</fonctions>
```

Tableau 6-3 Exemple de spécification d'un type de retour d'une fonction externe pour une liste

```
<fonctions>
   <fonction name="getTableau" location="Fonctions.java" >
     <returnType>
            <type name="Int" dataType="" array="true"
                    dimension="1"/>
     </returnType>
   </fonction>
</fonctions>
```

Tableau 6-4 Exemple de spécification d'un type de retour d'une fonction externe pour un tableau

Le protocole de communication entre l'outil d'analyse et des fonctions externes

Comme nous l'avons présenté ci-dessus, le composant "*Gestion des fonctions externes"* fait appel et exécute des fonctions externes. Avant l'exécution, ce composant vérifie la liste des *fonctions externes complexes* utilisées. La liste est produite en récupérant tous les attributs *name* et *location* de l'élément *function* (`<fonction name="string" location= "string">`) (cf. tableau 6-1). Si la fonction externe utilisée est complexe (cf. section 6.3, page 135), le résultat de cette fonction est transformé en une donnée UTL par le composant "*Services d'adaptation"*. Ce dernier lit le fichier comprenant la spécification de la *fonction externe complexe* utilisée. Il utilise le résultat de la fonction et cette spécification pour créer une donnée UTL. Cette donnée UTL possède la même structure (ou format) que le type de retour de la fonction externe utilisée. Les tableaux 6-5, 6-6 et 6-7 présentent des données UTL correspondant à la spécification dans les tableaux 6-2, 6-3 et 6-4. Si la fonction *getListePersonne* retourne une liste comprenant plusieurs éléments *Personne*, l'élément *Personne* dans le tableau 6-6 est répété plusieurs fois. Pour un type de retour de type *tableau* (cf. tableau 6-4), chaque élément du tableau correspond à un élément de la donnée UTL (le tableau 6-7).

```
<getPersonne>
   <Personne>
     <nom>String</nom>
     <prenom>String</prenom>
     <adresse>
        <numero>String</numero>
        <rue>String</rue>
        <ville>String</ville>
     </adresse>
   </Personne>
</getPersonne>
```

Tableau 6-5 Exemple d'une donnée UTL correspondant à une spécification du tableau 6-2

```
<getListePersonne>
   <Personne>
      <nom>String</nom>
     <prenom>String</prenom>
     <adresse>
         <numero>String</numero>
         <rue>String</rue>
         <ville>String</ville>
     </adresse>
   </Personne>
</getListePersonne>
```

Tableau 6-6 Exemple d'une donnée UTL correspondant à une spécification du tableau 6-3

```
<getTableau>
     <_1>int</_1>
     <_2>int</_2>
     <_3>int</_3>
     . . .
</getTableau>
```

Tableau 6-7 Exemple d'une donnée UTL correspondant à une spécification dans le tableau 6-4

Les exemples ci-dessous illustrent l'utilisation d'un résultat de retour de fonctions externes complexes avec DCL4UTL.

L'utilisation de l'élément *nom* du type *Personne* :

```
$p1=externe Fonctions getPersonne() ; //l'appel d'une fonction
$utilisateur1=p1.getPersonne.Personne.nom ;
```

L'utilisation du champ *nom* du deuxième élément dans la liste *Personne* :

```
$p2=externe Fonctions getListePersonne() ;//l'appel d'une fonction
//retourne le deuxième élément dans la liste
$temp=index(getListePersonne.Personne,2) ;
$utilisateur2=temp.Personne.nom ;
```

L'utilisation du cinquième élément du tableau :

```
$t=externe Fonction getTableau() ;l'appel d'une fonction
$i=getTableau._5 ;
```

Pour que l'outil d'analyse UTL et que les fonctions externes puissent communiquer ensemble, ils doivent être placés dans la même position. La figure 6-4 présente ces éléments. Les fonctions externes peuvent provenir d'un fichier jar (par exemple *outilExtern.jar*) ou être définies dans un fichier java (par exemple *FonctionsExternes.java*). Les spécifications des fonctions externes complexes sont décrites dans un seul fichier modifiable (par exemple *FonctionsExternes.xml*) qui est situé dans un répertoire "*Config*". Si

les fonctions externes sont utilisées dans la méthode de calcul des indicateurs, elles seront chargées dynamiquement lors de l'exécution des indicateurs.

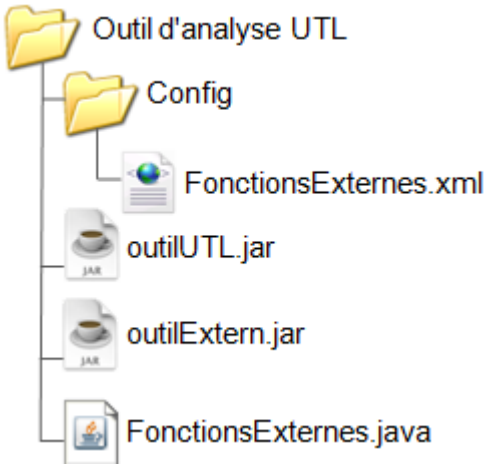

Figure 6-4 L'arborescence de l'outil d'analyse

6.5 L'interpréteur DCL4UTL

Le composant "*Calcul des données"* (cf. figure 6-1) est implémenté comme un interpréteur. Il est développé en utilisant JavaCC (Java Compiler Compiler) qui est le plus utilisé des générateurs de parseurs pour Java. JavaCC lit la spécification de la grammaire DCL4UTL écrite selon la spécification JavaCC. Puis, il la convertit en un programme Java. Des traitements sont ajoutés pour créer l'interpréteur DCL4UTL. Son architecture [Pham-Thi et al. (2010b)] est illustrée dans la figure 6-5.

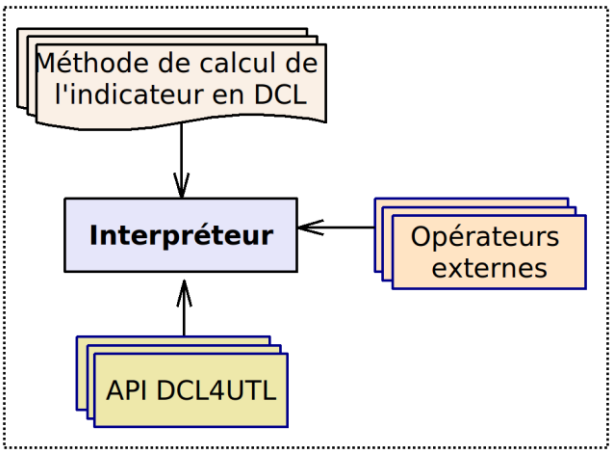

Figure 6-5 L'architecture de l'interpréteur de DCL4UTL

L'interpréteur lit la formule de calcul des données dérivées écrite en DCL4UTL ligne par ligne, la traduit et l'exécute s'il n'y a pas d'erreurs de syntaxe. Lors de l'exécution, l'interpréteur peut utiliser des opérateurs internes (API – ‘‘Application Programming Interface’’ DCL4UTL) ou des opérateurs externes pour établir une nouvelle donnée. Ces derniers opérateurs sont indiqués par l'analyste lors de la modélisation des indicateurs avec leurs spécifications du type de retour.

6.6 Editeur UTL

Pour faciliter la création et la modélisation des indicateurs, nous avons travaillé en collaboration avec un ingénieur de recherche au LIUM pour développer un éditeur UTL basé sur le web (cf. figure 6-6).

Cet éditeur assiste l'analyste dans sa tâche. Il permet de décrire n'importe quelle donnée UTL, et notamment la méthode de calcul de l’indicateur avec DCL4UTL et selon les trois facettes du modèle DGU. Il intègre également un parseur DCL4UTL ce qui facilite la formalisation et l'expression des méthodes de calcul. La mise à jour des données UTL peut également être réalisée à travers cet éditeur. La section suivante présente un exemple d'utilisation de cet interpréteur.

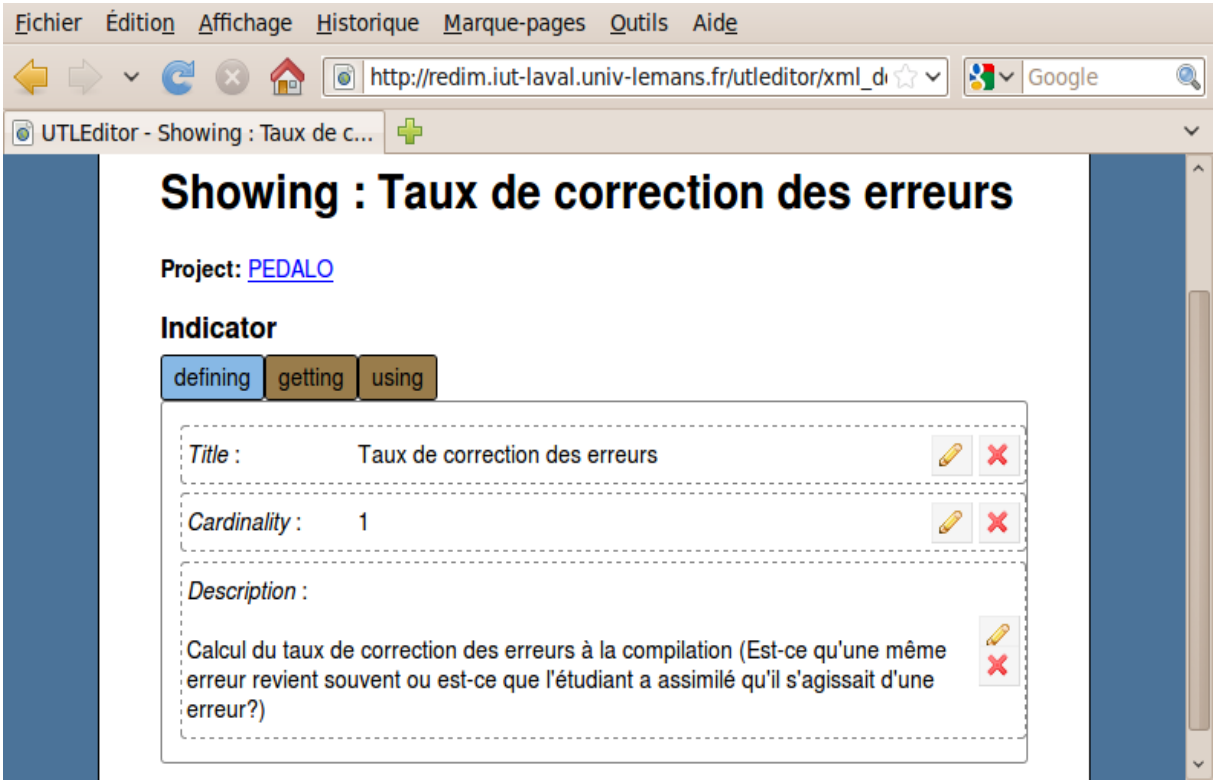

Figure 6-6 L’éditeur UTL

6.7 Application sur les traces collectées au laboratoire SysCom

6.7.1 Introduction

Comme nous l'avons présenté, nous avons utilisé les indicateurs et les traces générées par un environnement réalisé au laboratoire SysCom [Iksal et al. (2007)] pour construire le langage DCL4UTL. Dans cette application, nous utilisons ces traces pour calculer l'indicateur non quantitatif ''*le parcours par étudiant*''. Cet indicateur a été modélisé lors de la construction du langage DCL4UTL. Il est calculé après la session par notre outil d'analyse.

6.7.2 Calcul de l'indicateur ''le parcours par étudiant''

L'indicateur ''*Le parcours par étudiant''* (*Syscom-I-ParcoursEtu)* permet d'étudier et de comparer l'évolution des étudiants dans la liste des activités préconisées pour la session. Comme nous l'avons présenté dans le chapitre 5, cet environnement récolte toutes les activités des étudiants dans un seul fichier de log. Il génère deux types de traces (cf. figure 5-4). Nous créons donc deux données brutes UTL correspondant à ces deux types. La donnée brute *Syscom-RD-DebSess* décrit le changement d'activité par un étudiant. La donnée brute *Syscom-RD-EnCoursSess* spécifie que l'étudiant est en train d'effectuer une action. La donnée brute nécessaire pour le calcul de cet indicateur est *Syscom-RD-DebSess*. Pour décrire la liste des étudiants, nous disposons de la donnée additionnelle *Syscom-AD-Utilisateur*. Les descriptions des données nécessaires pour le calcul de cet indicateur sont :

- la donnée brute :

```
<rawDatum type="Syscom-RD-DebSess">
          <date>string</date>
          <etudiant>string</etudiant>
          <activite>string</activite>
</rawDatum>
```

Tableau 6-8 Le format de la donnée brute *Syscom-RD-DebSess*

- la donnée additionnelle :

```
<additionalDatum type="SysCom-AD-Utilisateur">
      <utilisateur>string</utilisateur>
</additionalDatum>
```

Tableau 6-9 Le format de la donnée additionnelle *SysCom-AD-Utilisateur*

- l'indicateur :

```
<indicator type="SysCom-I-ParcoursEtu">
  <etudiant nom="string" >
         <parcours>string</parcours>
  </etudiant>
</indicator>
```

Tableau 6-10 Le format de l'indicateur *SysCom-I-ParcoursEtu*

Cet indicateur calcule le parcours de tous les étudiants. Il s'agit de regrouper les données *Syscom-RD-DebSess* par étudiant et d'en combiner les activités. La figure 6-7 extrait une partie de l'indicateur ‘‘*Le parcours par étudiant*’’ qui décrit sa méthode de calcul. Dans cette méthode, une fonction ‘‘*concat*’’ prédéfinie de DCL4UTL est utilisée pour concaténer les activités d'un étudiant. Le résultat de cet indicateur est présenté dans la figure 6-9.

```
<description>
      <text>Calcul de la parcours par etudiant</text>
      <DCL4UTL-formula>
          cal { for each $u in AD.utilisateur do  {
                  $parcours =""  ;
                  for each  $a in R.activite as filter(R.etudiant==u) do
                  {
                        concat(parcours, a) ;
                        concat(parcours, "-->") ;
                  }
                  ID.etudiant[nom=u].parcours = parcours ;
                }
          }
         where     ID = SysCom-I-ParcoursEtu.using.data.indicator,
                   R = Syscom-RD-DebSess.using.data.rawDatum,
                   AD=Syscom-AD-Utilisateur.using.data.additionalDatum
      </DCL4UTL-formula>
</description>
```

Figure 6-7 Extrait de la description d'un indicateur pour l'environnement de SysCom (méthode de calcul)

6.7.3 Le prototype

Cette section présente une application de l'architecture proposée. Cette application a été réalisée après la session d'apprentissage. C'est-à-dire que les indicateurs sont calculés après la séance. La figure 6-8 décrit l'architecture de cette application. Il s'agit d'un modèle simple de l'architecture générale de la figure 6-1. Puisque les indicateurs sont calculés post session, le composant ‘‘*Gestion des événements*’’ de l'architecture générale n'est donc pas présent dans la figure 6-8.

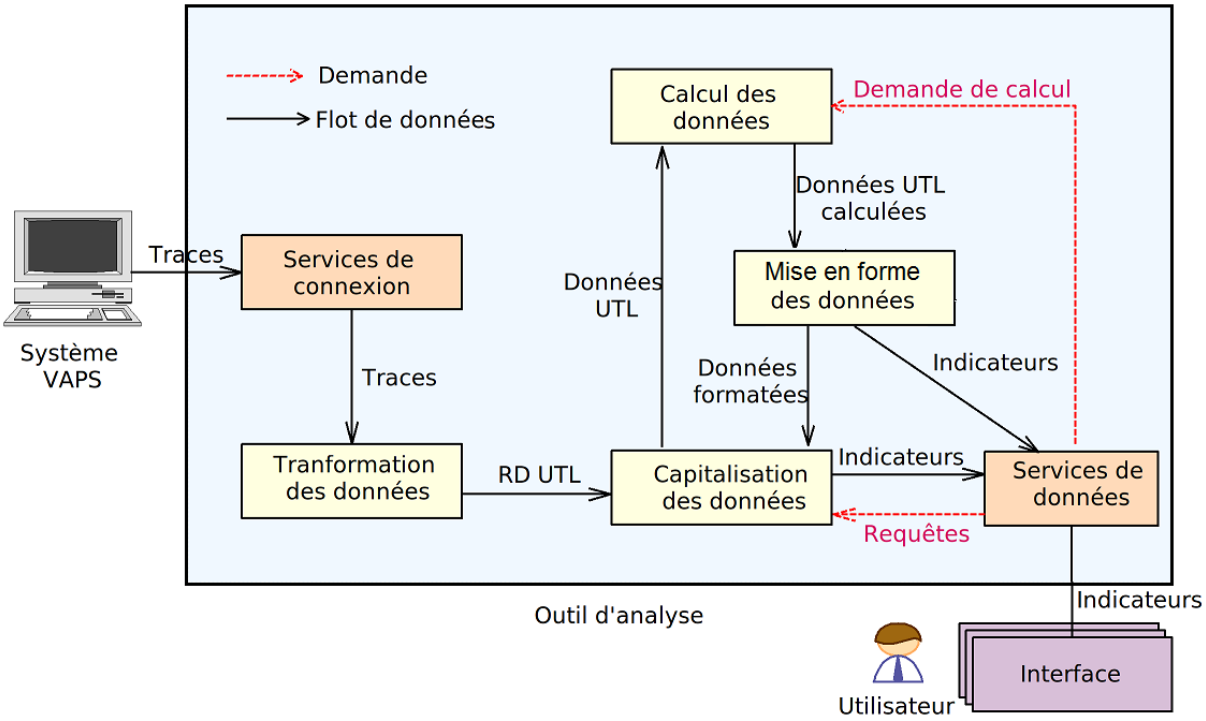

Figure 6-8 Le modèle d'application pour le calcul d'indicateurs après la session

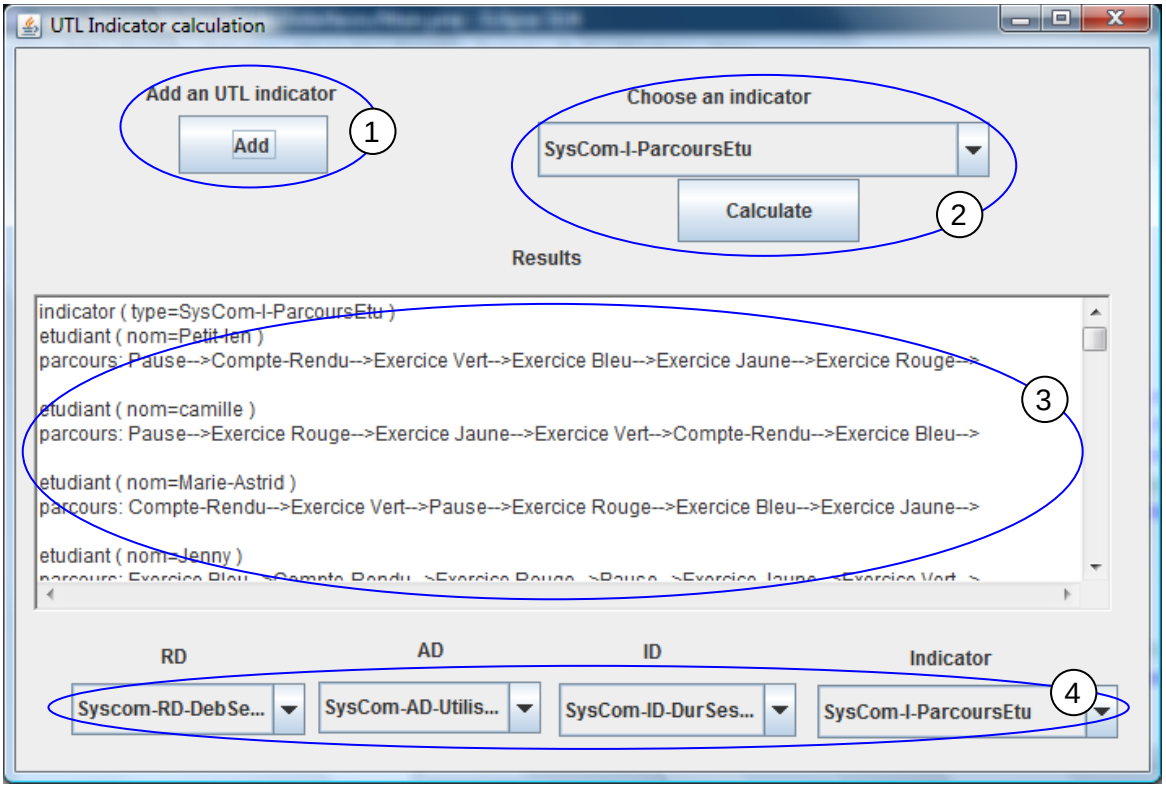

Figure 6-9 Le résultat de l'indicateur "*le parcours par étudiant*"

La figure 6-9 présente une interface simple de l'application. Cette interface se compose de quatre zones. À travers cette interface, les indicateurs modélisés sont ajoutés (la zone ①) pour être disponibles au calcul. L'utilisateur peut également choisir un indicateur disponible pour le calculer (la zone ②) et voir le résultat (la zone ③). Une fois qu'un indicateur est calculé (la zone ②), il est ajouté à la liste des indicateurs (la zone ④). L'ajout des indicateurs peut être réalisé à tout moment. Toutes les données UTL comprenant des données brutes, des données intermédiaires et des indicateurs sont capitalisées. L'utilisateur peut donc revoir le résultat des indicateurs (la liste déroulante ''*Indicator''* dans la zone ④) ainsi que consulter les autres données existantes dans la base de données UTL (les autres listes déroulantes dans la zone ④). La zone ③ est utilisée pour afficher le résultat des calculs ou le contenu des données UTL dans la base. Au niveau de la visualisation des indicateurs, nous les affichons actuellement sous une forme textuelle simple (XML, texte).

6.8 Application sur les données du système VAPS (Virtual Action Planing Supermarket)

Les indicateurs et les traces du système VAPS ont également été utilisés pour concevoir le langage DCL4UTL. Pour créer notre langage et aussi tester notre outil dans un EIAH particulier dédié au ré-apprentissage, nous avons travaillé en collaboration avec l'Equipe Lavalloise Handicaps et Innovations Technologiques (ELHIT) pour modéliser et calculer des indicateurs après une session avec des traces générées par le système VAPS (Virtual Action Planning Supermarket) [Klinger (2006)]. Cette partie présente une application de l'outil d'analyse et du langage de combinaison DCL4UTL sur les traces produites par ce système [Pham-Thi et al. (2009a)].

6.8.1 Les traces

Les traces générées par le système VAPS sont enregistrées dans des fichiers textuels. Il y a trois types de fichiers.

- Le premier décrit les actions du patient dans le VAPS. Ces actions se composent de la prise des produits et du paiement. Chaque trace du fichier a le format suivant :

  <action> <l'instant de réalisation de l'action > <évaluation d'action>

  Exemple :

  Prise du tee-shirt 02m 10s 924ms bonne

  Où :

- *Prise du tee-shirt* correspond à l'action du patient.
- *02m 10s 924ms* est *l'instant de réalisation* de l'action par le patient.
- *bonne* correspond à l'évaluation de l'action, cet élément est optionnel.

La figure 6-10 est un extrait d'un fichier de ce type.

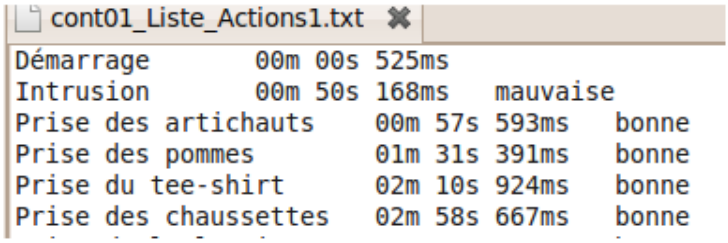

Figure 6-10 Exemple de traces VAPS concernant des actions d'un patient

- Le deuxième enregistre des points de temps correspondant aux arrêts du patient dans le VAPS. Le format de la trace est le suivant :

  <position de temps> *<durée>*

  Exemple : *00m 00s 525ms* *00min 06s*

Où :

- *00m 00s 525ms* est l'instant où le patient s'arrête dans le VAPS
- *00min 06s* donne la durée d'un arrêt.

La figure 6-11 présente un extrait d'un fichier de ce type.

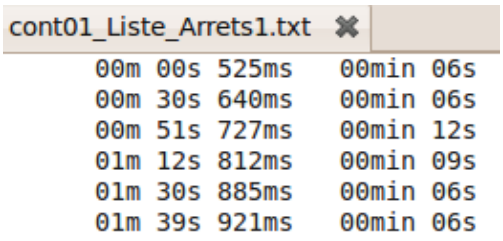

Figure 6-11 Exemple de traces VAPS concernant des temps d'arrêts

- Le dernier enregistre des positions correspondant aux points d'arrêt d'un patient dans le VAPS. Le format de la trace est le suivant :

  <coordonnée> *<position de temps>* *<distance>* *<action>*

Exemple *:*

22.1453,1,2.13595 *02m 58s 240ms* *61.5929* *Prise des chaussettes*

Où :

- *22.1453,1,2.13595* est la coordonnée spatiale à 3 dimensions d'un point d'arrêt.
- *02m 58s 240ms* donne l'instant où le patient s'arrête dans le VAPS.
- *61.5929* indique la distance parcourue en mètre
- *Prise des chaussettes* correspond à l'action du patient

La figure 6-12 présente un extrait d'un fichier de ce type.

cont01_Liste_Positions1.txt

```
-28.1486,1,-25.2772     00m 51s 727ms   21.0826
-28.1486,1,-25.2772     00m 54s 740ms   21.0826
-28.1486,1,-25.2772     00m 57s 753ms   21.0826 Prise des artichauts
-28.1486,1,-25.2772     01m 00s 765ms   21.0826
-28.1486,0.999587,-24.4067      01m 03s 777ms   21.953
-28.1486,0.999583,-22.8715      01m 06s 788ms   23.4883
-27.3453,1,-22.4056     01m 09s 800ms   24.417
-26.8843,1,-23.0535     01m 12s 812ms   25.2122
```

Figure 6-12 Exemple de traces VAPS concernant des positions d'arrêts

6.8.2 Les indicateurs

Plusieurs indicateurs sont proposés dans le VAPS. Nous modélisons et calculons certains d'entre eux avec UTL et notre outil. Nous ne proposons pas ces indicateurs. Ces derniers sont divisés en deux groupes : la connaissance sémantique liée à la tâche et la vitesse de traitement de l'information. Les indicateurs appartenant au premier groupe se composent :

- Du nombre de bonnes actions : les bonnes actions concernent la réalisation des tâches conformément à la consigne. C'est-à-dire des actions cohérentes et rationnelles par rapport à un objectif que le patient a réalisé dans le supermarché virtuel (par exemple, la prise de produits, le dépôt et la reprise de produits sur le tapis roulant, le paiement).
- Du nombre de mauvaises actions : ces mauvaises actions correspondent à la prise de produits erronés, ou la prise multiple de produits de la liste, mais aussi toutes les erreurs lors du passage aux caisses et lors du paiement.
- Du taux de réalisation : le taux de réalisation est donc le pourcentage de bonnes actions réalisées.
- Du taux d'erreur : le pourcentage de mauvaises actions parmi toutes les actions réalisées durant la séance.

Le deuxième groupe d'indicateurs a pour objectif d'évaluer la capacité à planifier des actions et la capacité à formuler un but. Les indicateurs de ce groupe sont :

- Le temps de mise en mouvement : le temps qui s'écoule entre l'apparition de la position de départ dans le supermarché et le moment où le sujet commence à se déplacer.
- Le temps de la première action : le temps qui s'écoule depuis le démarrage jusqu'au premier achat.
- La vitesse de réalisation de la tâche : le rapport entre la distance parcourue et la durée de la séance.

6.8.3 Modélisation des données

Cette partie présente les données nécessaires pour le calcul des indicateurs décrits ci-dessus. En se basant sur les traces générées par le VAPS, trois types de données brutes UTL ont été créés correspondant à trois types de fichiers de traces. Le tableau 6-11 présente ces données brutes.

Dans le cas du VAPS, les données de production ne sont pas collectées, mais une donnée additionnelle *VAPS-AD-Patient* a été créée pour décrire la liste des patients participant à la séance. Cette donnée est nécessaire pour le calcul de tous les indicateurs. La figure 6-13 est un extrait de cette donnée.

Traces	Données brutes UTL
Action	VAPS-RD-Action
Arrêt	VAPS-RD-Arret
Position	VAPS-RD-Position

Tableau 6-11 Les données brutes UTL correspondant aux traces générées par le VAPS

Nous créons également deux données intermédiaires *VAPS-ID-DisParcours* et *VAPS-ID-Duree* :

- *VAPS-ID-DisParcours* produit la distance parcourue par le patient. Il s'agit de regrouper les données *VAPS-RD-Position* par patient et d'en calculer la distance parcourue.
- *VAPS-ID-Duree* produit la durée de la séance par patient. Il s'agit aussi de regrouper les données *VAPS-RD-Position* par patient et d'en calculer la durée de la séance.

Finalement, sept indicateurs sont modélisés selon les trois facettes *defining*, *getting* et *using* avec leurs méthodes d'acquisition de données en DCL4UTL. Le tableau 6-12 présente ces indicateurs.

```
<utl>
    <additionalDatum>
        <defining>
            <title>VAPS-AD-Utilisateur</title>
            <type>ad-hoc data</type>
            <description>Fichier listant tous les patients</description>
        </defining>
        <getting>
            <location>Data_Subject.txt</location>
        </getting>
        <using>
            <data>
                <additionalDatum type="VAPS-AD-Utilisateur">
                    <utilisateur>cont01</utilisateur>
                    <utilisateur>cont02</utilisateur>
                    <utilisateur>pat04</utilisateur>
                    <utilisateur>pat05</utilisateur>
                    <utilisateur>pat06</utilisateur>
                </additionalDatum>
            </data>
            <format>
                <additionalDatum type="VAPS-AD-Utilisateur">
                    <utilisateur> string</utilisateur>
                </additionalDatum>
            </format>
            <usedBy/>
        </using>
    </additionalDatum>
</utl>
```

Figure 6-13 Exemple de donnée additionnelle du VAPS

Indicateur	Description
VAPS-I-BA	Nombre de bonnes actions
VAPS-I-MA	Nombre de mauvaises actions
VAPS-I-Realisation	Taux de réalisation
VAPS-I-Taux-Erreur	Taux d'erreur
VAPS-I-Premier-Arret	Temps de mise en mouvement
VAPS-I-Premier-Action	Temps de la première action
VAPS-I-Vitesse	Vitesse de réalisation de la tâche

Tableau 6-12 La liste de noms d'indicateurs du VAPS

La figure 6-14 présente un diagramme de relation de toutes les données UTL nécessaires pour le calcul de ces indicateurs.

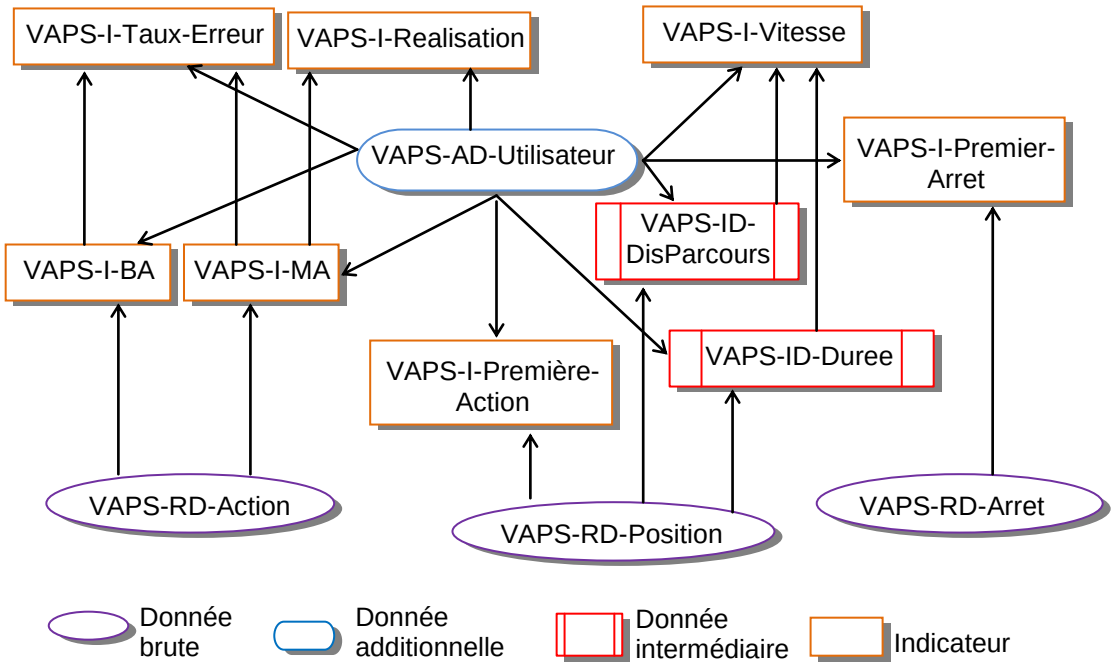

Figure 6-14 La cartographie des données du VAPS

La figure 6-15 présente un extrait d'une partie de l'indicateur *Taux d'erreur* qui décrit sa méthode de calcul. Cet indicateur est calculé en réutilisant deux indicateurs *Nombre de bonnes actions (ELHIT-I-BA)* et *Nombre de mauvaises actions (ELHIT-I-MA).* Il s'agit de récupérer le nombre de bonnes actions (`I2.nombreBonnesActions[patient==i]`) ainsi que le nombre de mauvaises actions (`I1.nombreMauvaisesActions[patient==i])` et de calculer le taux d'erreur (`I.tauxErreur[patient=i]`) par patient selon l'expression : *taux d'erreur = (MA*100/(BA+MA)).*

```
<description>
    <text>Il s'agit du pourcentage de mauvaises actions parmi
          toutes les actions réalisées lors de la séance
          (MA*100/(BA+MA))</text>
    <DCL4UTL-formula>
      cal{
            for each $i in AD.utilisateur do {
                I.tauxErreur[patient=i]=
                  (I1.nombreMauvaisesActions[patient==i]*100)/
                     (I2.nombreBonnesActions[patient==i]  +
                      I1.nombreMauvaisesActions[patient==i])
            }
        }
      where I = ELHIT-I-Taux-Erreur.using.data.indicator,
           I1= ELHIT-I-MA.using.data.indicator,
           AD= ELHIT-AD-Utilisateur.using.data.additionalDatum,
            I2= ELHIT-I-BA.using.data.indicator
   </DCL4UTL-formula>
</description>
```

Figure 6-15 Extrait de la description d'un indicateur du VAPS (méthode de calcul)

Nous avons exécuté les indicateurs décrits ci-dessus avec notre outil. L'interpréteur lit la méthode de calcul d'un indicateur comme donnée d'entrée (par exemple, voir dans la figure 6-15 l'élément *DCL4UTL-formula*). Il produit les résultats indiqués dans la figure 6-16. Ces résultats sont affichés directement à l'utilisateur. Lors de l'exécution de l'outil, un nouvel indicateur peut être ajouté et calculé sans modifier l'interpréteur

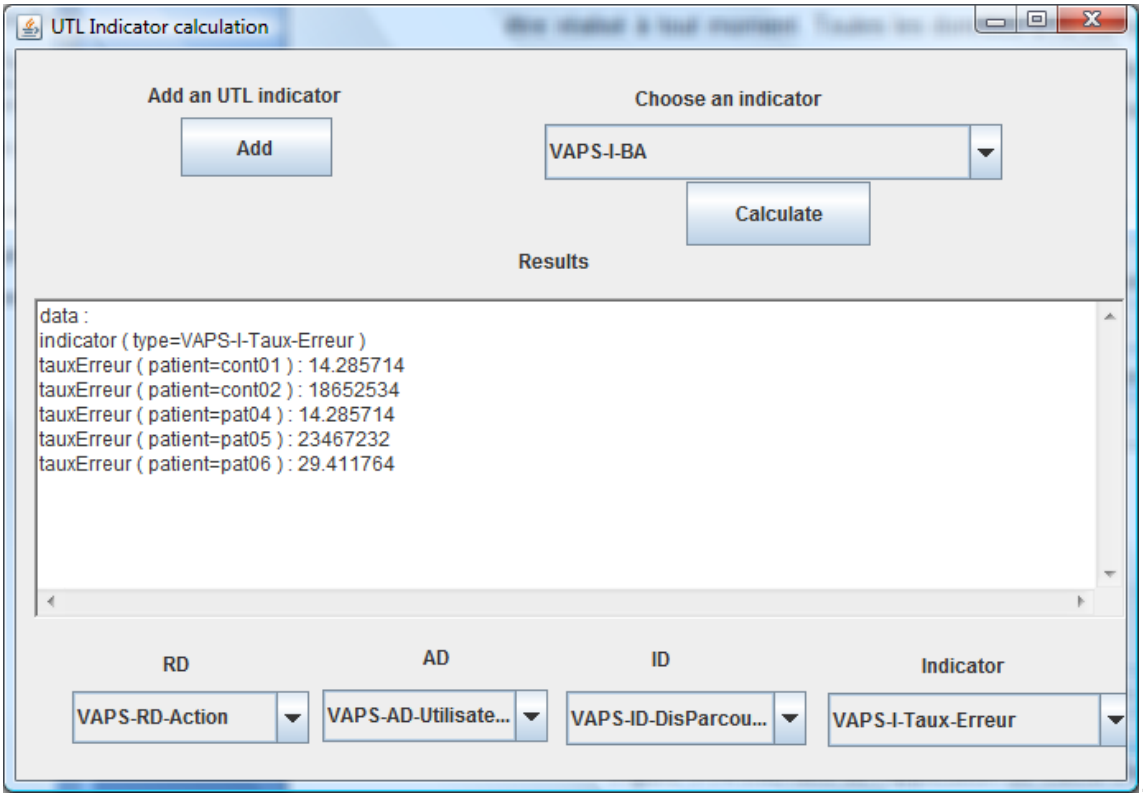

Figure 6-16 L'interface de l'application de calcul d'indicateurs du VAPS

6.8 Conclusion

Les travaux que nous avons présentés portent sur la conception et l'implémentation d'un outil d'analyse pour valider notre proposition. Nous avons proposé une architecture générale pour notre outil d'analyse qui intègre l'interpréteur DCL4UTL. Cet outil permet de transformer les traces générées par des EIAH en données brutes UTL et de calculer les indicateurs à partir de données UTL. Il permet également de faire appel à des fonctions externes et de les exécuter. Nous avons testé notre outil sur les traces générées par deux systèmes dans deux domaines différents. Avec les traces du VAPS, certains indicateurs simples et quantitatifs sont calculés. Avec les traces de l'environnement réalisé au laboratoire SysCom, nous avons spécifié la méthode de calcul et le résultat d'un indicateur non quantitatif. Dans le chapitre suivant, nous allons présenter une mise à l'épreuve de l'outil avec des indicateurs

complexes. La modélisation des indicateurs dans ce chapitre n'utilise pas les fonctions externes. Les fonctions externes et les données intermédiaires paramétrées sont utilisées dans le chapitre suivant.

Actuellement, le composant ‘‘*Transformation des données”* du noyau de notre outil d'analyse est développé de manière *ad hoc*. Il ne peut donc pas transformer automatiquement toutes les traces des EIAH. Ce composant doit être modifié pour s'adapter à chaque EIAH considéré. En fait, ce composant peut être implémenté de manière générique pour permettre de transformer différentes traces générées par différents EIAH en utilisant la description de l'élément ‘‘*track*’’ [Choquet et Iksal (2007a)] de la donnée brute UTL. Cet élément permet de décrire la méthode d'obtention de la donnée brute à partir de plusieurs types de trace (par exemple base de données, XML, fichier de log). En se basant sur cette description (par exemple une requête SQL, une expression XPATH ou des positions de la donnée dans un fichier de log), l'implémentation du composant ‘‘*Transformation des données”* peut être faite de manière automatique. Toutefois, dans un premier temps, nous nous sommes focalisés sur l'analyse des traces stockées dans des fichiers textes (par exemple les traces du VAPS et les traces de SysCom) et nous avons décidé de mettre en œuvre l'interpréteur DCL4UTL. C'est pourquoi nous n'avons pas encore pris en compte l'implémentation générique de ce composant.

Au niveau des fonctions externes, même si le langage DCL4UTL peut être utilisé pour faire appel aux fonctions externes à distance (par exemple, via un service web, via RPC), notre outil d'analyse permet actuellement d'exécuter des fonctions externes locales développées en Java, c'est-à-dire que ces fonctions doivent être placées au même endroit que notre outil d'analyse dans une machine. De plus, ces fonctions externes se limitent aux arguments dont le type de donnée est un primitif (numérique, alphanumérique, booléen), un tableau ou un objet.

7 Mise à l'épreuve

Sommaire

Ce chapitre aborde une expérimentation de mise en œuvre de notre approche instrumentée dans un contexte d'apprentissage des bases de la programmation orientée objet et événementielle. Dans cette expérimentation, les indicateurs sont calculés en temps réel à partir des traces générées par la plateforme Hop3x avec des étudiants en première année de DUT (Diplôme Universitaire de Technologie). Les résultats de ces indicateurs sont utilisés pour aider le tuteur à réguler l'activité de l'apprenant, mais aussi afin de réguler sa propre activité et pour améliorer le scénario d'apprentissage.

7. 1 Description de l'expérimentation

7. 1.1 Contexte de l'expérimentation

Cette expérimentation a été réalisée au département SRC (Services et Réseaux de Communication) au sein de l'IUT de Laval. Elle a impliqué 90 étudiants en première année de DUT, sur une séance de travaux pratiques (TP). Ces étudiants ont été divisés en six groupes comprenant chacun 15 étudiants. L'expérimentation s'est déroulée en mai et juin 2010. L'objectif pédagogique de la séance concerne la programmation orientée objet en Java dans le cadre du module d'apprentissage des *Bases de la programmation orientée objet et événementielle*. Pendant une durée de trois heures, l'apprenant doit répondre à douze questions au travers du support d'un environnement de programmation nommé Hop3x [Lekira (2010)], développé au LIUM.

7.1.2 Objectifs de l'expérimentation

Nous avons travaillé en collaboration avec d'autres membres du laboratoire pour la réalisation de cette expérimentation. Pendant la session, les indicateurs calculés en temps réel sont fournis à l'enseignant/tuteur. Nous n'avons pas élaboré ces indicateurs. Ils ont été définis par un enseignant. Durant la session les tuteurs utilisent ces indicateurs pour réguler les activités d'apprentissage des étudiants ainsi que leur propre activité tutorale. Les objectifs de l'expérimentation étaient de :

- Valider la capacité des langages UTL et DCL4UTL dans la réponse à la description formelle des besoins d'observation de l'enseignant.
- Valider la capacité de calcul d'indicateurs en temps réel de notre outil d'analyse.

D'autres chercheurs du laboratoire ont également cherché à :

- Identifier les éléments du scénario d'apprentissage à améliorer pour la session suivante.
- Mesurer l'impact des actes tutoraux sur l'amélioration de l'apprentissage des étudiants avec l'appui d'indicateurs.

7.1.3 Description de la plateforme de formation

Le dispositif d'apprentissage utilisé pour cette expérimentation est le système Hop3x. Il s'agit d'une plate-forme qui aide les étudiants à apprendre les techniques de programmation orientée objet avec des langages comme Java ou Ruby. Il est composé de trois parties :

Hop3x-Etudiant (la figure 7-1) permet aux apprenants d'éditer du code, de le compiler et de l'exécuter. Il leur permet également de communiquer avec l'enseignant/tuteur via un outil de communication (le menu *Assistance* dans la figure 7-1) afin de solliciter l'aide de l'enseignant/tuteur.

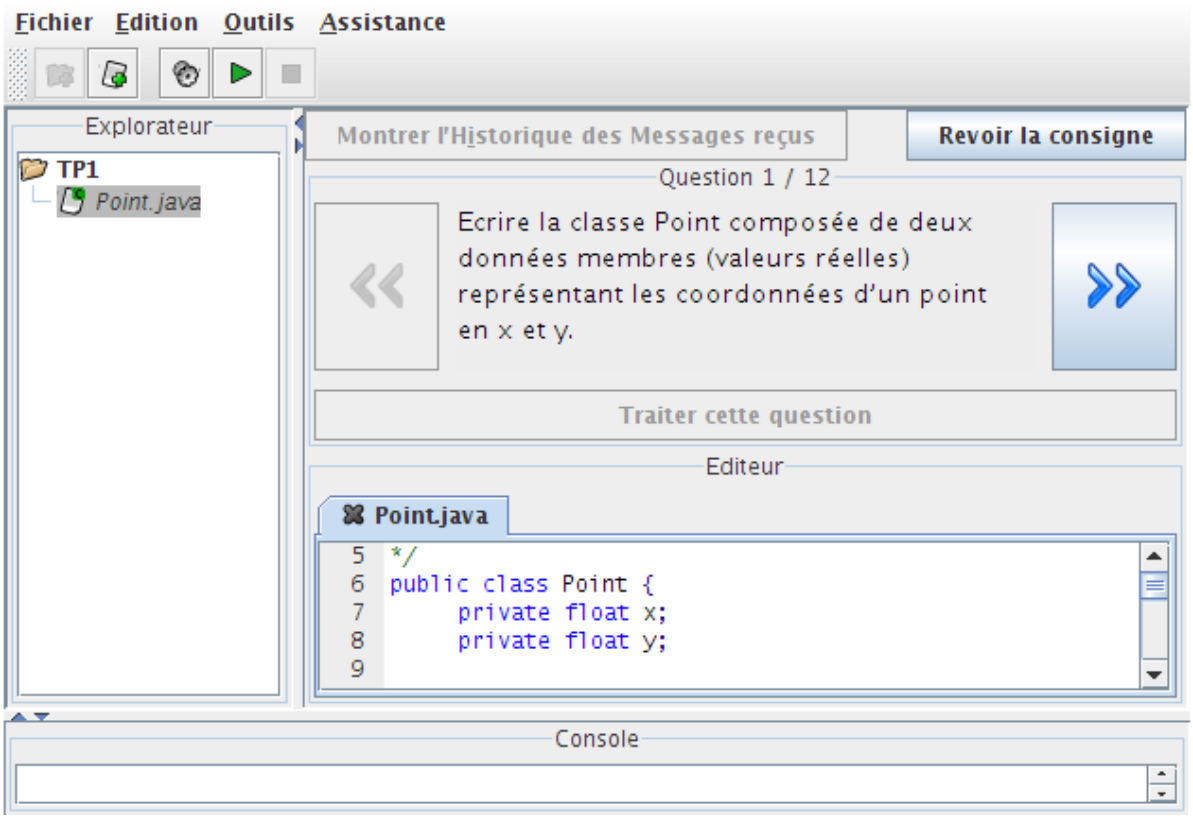

Figure 7-1 L'interface d'étudiant Hop3x-Etudiant

Hop3x-Enseignant (cf. figure 7-2) est un outil de supervision destiné aux tuteurs et leur permettant de gérer des apprenants, de suivre en temps réel ce qu'ils sont en train de faire ou d'annoter une partie du code d'un apprenant. L'outil dispose également de fonctionnalités dédiées à la communication entre le tuteur et l'apprenant (par exemple : envoyer un message à un apprenant ou lancer une conversation).

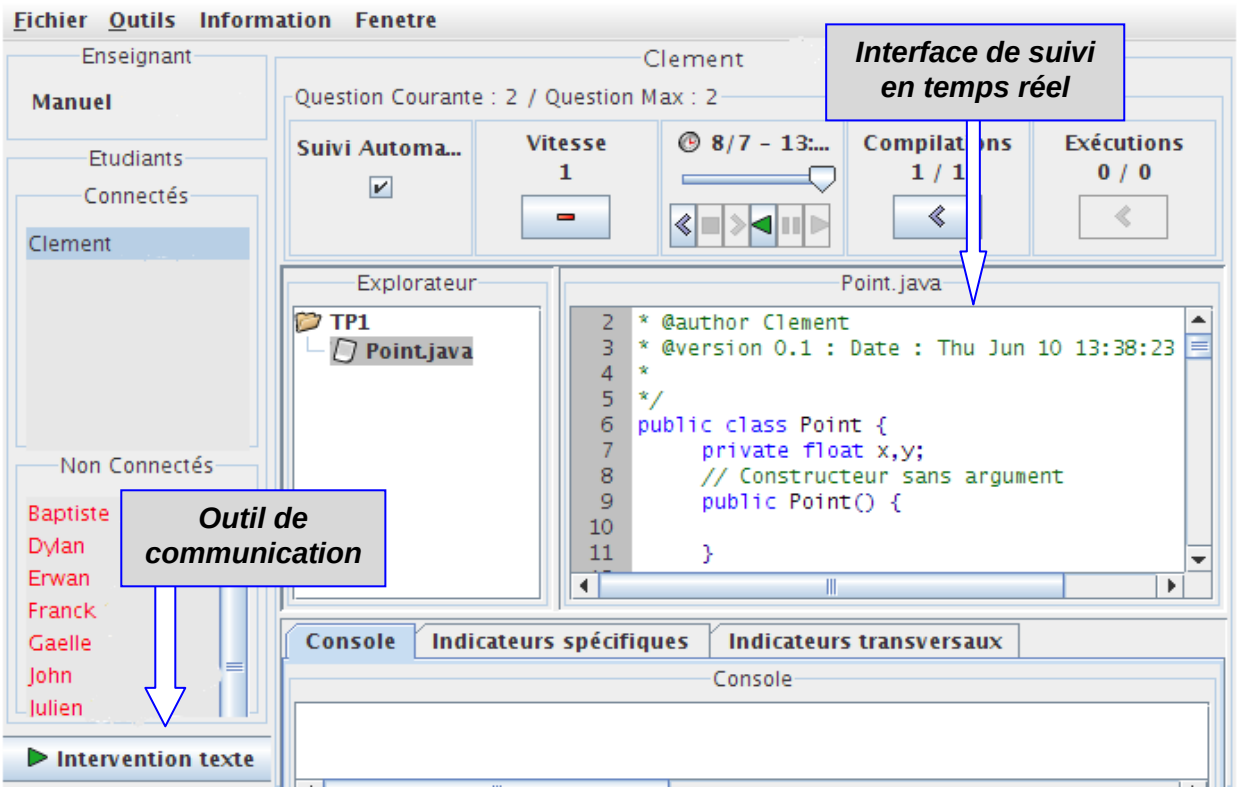

Figure 7-2 L'interface d'enseignant Hop3x-Enseignant

Hop3x-Serveur collecte les traces d'interactions de l'apprenant sous forme d'événements (cf. figure 7-3, par exemple : l'insertion d'un texte, la suppression d'un texte, l'insertion d'un projet, la suppression d'un projet, le choix d'une question, la compilation manuelle, la compilation automatique, l'exécution d'un programme, etc.)

```
<?xml version="1.0" encoding="UTF-8"?>
<TRACE>
<E K="SQ" T="1276514837499" H="13:27:17, 499"><NQ>1</NQ></E>
<E K="AP" T="1276514843829" H="13:27:23, 829"><P>TP1</P><TP>Java</TP></E>
<E K="AF" T="1276514850674" H="13:27:30, 674"><TF>Java</TF><F>Point.java</F><P>TP1</P></E>
<E K="IT" T="1276514850678" H="13:27:30, 678"><T></T><F>Point.java</F><P>TP1</P><N>0</N></E>
<E K="IT" T="1276514850704" H="13:27:30, 704"><T>/**\n* @author Pierre \n* @version 0.1 :
Date : Mon Jun 14 13:27:30 CEST 2010\n*\n*/\npublic class Point {\n\t\n\t// Constructeur sans
argument\n\tpublic Point() {\n\t\t\n\t}\n\t\n}</T><F>Point.java</F><P>TP1</P><N>0</N></E>
<E K="IT" T="1276514861355" H="13:27:41, 355"><T>p</T><F>Point.java</F><P>TP1</P><N>109</N></E>
<E K="IT" T="1276514861927" H="13:27:41, 927"><T>u</T><F>Point.java</F><P>TP1</P><N>110</N></E>
<E K="ST" T="1276514862343" H="13:27:42, 343"><D>u</D><F>Point.java</F><P>TP1</P><A>110</A>
<Z>111</Z></E>
<E K="IT" T="1276514863265" H="13:27:43, 265"><T>r</T><F>Point.java</F><P>TP1</P><N>110</N></E>
<E K="IT" T="1276514863529" H="13:27:43, 529"><T>i</T><F>Point.java</F><P>TP1</P><N>111</N></E>
<E K="IT" T="1276514863801" H="13:27:43, 801"><T>v</T><F>Point.java</F><P>TP1</P><N>112</N></E>
<E K="IT" T="1276514864040" H="13:27:44, 040"><T>a</T><F>Point.java</F><P>TP1</P><N>113</N></E>
<E K="IT" T="1276514864255" H="13:27:44, 255"><T>t</T><F>Point.java</F><P>TP1</P><N>114</N></E>
<E K="IT" T="1276514864518" H="13:27:44, 518"><T>e</T><F>Point.java</F><P>TP1</P><N>115</N></E>
<E K="IT" T="1276514865142" H="13:27:45, 142"><T> </T><F>Point.java</F><P>TP1</P><N>116</N></E>
<E K="IT" T="1276514866112" H="13:27:46, 112"><T>f</T><F>Point.java</F><P>TP1</P><N>117</N></E>
<E K="IT" T="1276514866433" H="13:27:46, 433"><T>l</T><F>Point.java</F><P>TP1</P><N>118</N></E>
<E K="IT" T="1276514866660" H="13:27:46, 660"><T>o</T><F>Point.java</F><P>TP1</P><N>119</N></E>
<E K="IT" T="1276514866864" H="13:27:46, 864"><T>a</T><F>Point.java</F><P>TP1</P><N>120</N></E>
<E K="IT" T="1276514867061" H="13:27:47, 061"><T>t</T><F>Point.java</F><P>TP1</P><N>121</N></E>
```

Figure 7-3 Traces collectées par Hop3x-serveur

Les traces

Comme nous l'avons présenté ci-dessus, les activités des apprenants pendant la session sont enregistrées par le Hop3x-serveur sous la forme d'événements dans des fichiers XML (un fichier par un apprenant). La figure 7-3 présente une instance de ces fichiers. Le tableau 7-1 présente les éléments et les attributs des balises de ces traces. Les types d'événement collectés par Hop3x sont résumés dans l'annexe C. Hop3x génère vingt-trois événements différents au total.

Elément/Attribut	Description
E	Description d'un événement
T (attribut)	Temps
H	Heure équivalente T(temps)
K	Type d'événement
T (élément)	Texte
F	Nom de fichier
P	Nom de projet
N	Une position du T(Texte) dans un fichier
TP	Type de projet
NQ	Numéro de question

Tableau 7-1 Les éléments et attributs des traces Hop3x

7.1.4 Les rôles participant à l'expérimentation

Comme nous l'avons présenté dans le chapitre 4, le processus d'analyse de traces implique trois rôles : l'enseignant, l'analyste et le développeur. En compagnie des apprenants et de l'enseignant qui a défini le scénario, trois autres acteurs participent également à l'expérimentation. Ils sont :

- Deux enseignants qui travaillent à distance (dans leur bureau) comme tuteurs. Ils observent, régulent l'activité de l'apprenant et font des interventions textuelles ou audio via l'interface d'enseignant Hop3x (cf. figure 7-2).
- Un analyste (chercheur) qui a en même temps le rôle d'un développeur, définit l'ensemble des données primaires UTL à partir des traces. Il transforme également les besoins d'observation proposés par l'enseignant (concepteur pédagogique) en indicateurs au sens d'UTL. Une négociation entre le développeur et l'enseignant a été réalisée pour obtenir un consensus sur le format des indicateurs qui répondra aux attentes de ce dernier. Comme nous l'avons expliqué dans la section 4.2, le

développeur définit également les fonctions externes nécessaires pour le calcul d'indicateurs.

7.1.5 La méthodologie de l'expérimentation

Contrairement aux exemples présentés dans le chapitre précédent, cette expérimentation prend en compte le calcul d'indicateurs en temps réel. La méthodologie de l'expérimentation est la suivante.

- Avant la session, lors de la conception du scénario d'apprentissage, l'enseignant est sollicité afin de lister l'ensemble de ses besoins d'observation qui deviendront des indicateurs.
- La définition des données primaires UTL à partir de traces recueillies par le dispositif d'apprentissage Hop3x et la modélisation des indicateurs à partir de besoins d'observations sont également réalisées avant la session.
- Le développeur écrit les fonctions externes nécessaires pour la modélisation des indicateurs.
- Durant la session, les étudiants effectuent leur TP de manière habituelle, une personne par machine.
- Les traces d'interaction entre l'étudiant et la plate-forme d'apprentissage sont collectées en temps réel par le dispositif.
- En se basant sur ces traces, les indicateurs sont également calculés en temps réel. Ces indicateurs sont adressés aux tuteurs (cf. figure 7-2) pour qu'ils puissent observer l'activité des apprenants et éventuellement intervenir.
- Durant la session, l'interaction entre les étudiants et les tuteurs est réalisée via l'outil Hop3x. Les tuteurs peuvent, en temps réel, suivre les activités (les programmes et les codes Java) des apprenants. Si l'apprenant a des problèmes dans la résolution des questions, il peut solliciter l'aide de l'enseignant/tuteur via l'interface d'étudiant Hop3x (cf. figure 7-1). Le tuteur peut communiquer avec ses étudiants (via les outils de communication) en sélectionnant un ensemble d'indicateurs et en choisissant le type d'intervention (en envoyant des messages textuels ou en lançant une conversation audio (cf. figure 7-2)).

7.1.6 Description du scénario d'apprentissage

L'apprenant est débutant en Java, mais il a suivi des cours concernant les concepts de base de la programmation orientée objet en Java comme : l'encapsulation, la classe, l'objet, les instances, le polymorphisme, etc. Dans une séance de TP de trois heures, l'apprenant doit

écrire des petits programmes pour appliquer ce qu'il a appris en répondant à un ensemble de questions concernant les notions de classe et de méthode, sans héritage. Avant de commencer l'activité, l'apprenant est invité à lire l'énoncé du TP. Il peut discuter avec le tuteur via l'outil et ce dernier peut expliquer les questions à l'apprenant. Cependant, durant la session, les apprenants ne peuvent pas communiquer entre eux via l'outil. L'apprenant doit utiliser l'environnement de programmation Hop3x afin d'effectuer ses activités d'apprentissage. Pendant la session, l'apprenant peut réaliser les questions dans l'ordre de son choix, cependant il est encouragé à les faire dans l'ordre préconisé par l'enseignant. L'étudiant a douze questions à réaliser dans un TP qui se compose de l'énoncé suivant :

1. Ecrire la classe *Point* composée de deux champs (valeurs réelles) représentant les coordonnées d'un point en x et y.
2. Redéfinir la méthode *equals* pour la classe *Point*.
3. Ajouter un comportement, que l'on appellera *distance*, permettant à un objet de la classe *Point* de calculer la distance qui le sépare d'un autre point.
4. Un triangle étant défini par trois points, écrire la classe *Triangle*.
5. Ajouter un comportement, que l'on appellera *perimetre*, permettant à un objet de la classe *Triangle* de calculer son périmètre.
6. Ajouter un comportement, que l'on appellera *surface*, permettant de calculer la surface d'un triangle.
7. Ecrire la classe *Droite*, une droite étant définie par deux réels (a et b) représentant les coefficients de l'équation de droite y=ax+b.
8. Faire le nécessaire pour qu'il soit possible de créer une droite à partir de deux points (c'est-à-dire la droite qui passe par ces deux points).
9. Ajouter, à la classe *Point*, un comportement, que l'on appellera *estSurDroite*, permettant à un point de tester s'il se trouve sur une droite.
10. Ajouter, à la classe *Droite*, un comportement, que l'on appellera *estParallele*, permettant à une droite de tester si elle est parallèle à une autre.
11. Ajouter, à la classe *Droite*, un comportement, que l'on appellera *intersection*, permettant à une droite de calculer son intersection avec une autre droite.
12. Faire le nécessaire pour qu'il soit possible de créer un *Triangle* à partir de trois droites.

7.1.7 Description des indicateurs définis par l'enseignant

Nous souhaitons mettre l'accent sur le fait que l'enseignant choisit et définit des indicateurs décrits dans cette partie. Ce dernier spécifie ses besoins d'observation. Ces besoins ont été définis a priori, sur la base de l'expérience de l'enseignant/concepteur et visent principalement à la programmation orientée objet, en même temps que la définition du scénario d'apprentissage. L'enseignant doit indiquer les éléments qu'il souhaite observer et les résultats qu'il veut avoir sous la forme d'une valeur dans un format particulier. Dans notre cas, ces besoins se sont groupés en deux types d'indicateurs : les indicateurs spécifiques et les indicateurs transversaux [Lekira (2010)].

- Les indicateurs transversaux ne sont pas liés à une question du sujet de TP. Ils peuvent être utilisés pour tous les TP. Par exemple, les indicateurs sur l'état des connaissances de l'apprenant par rapport au reste du groupe, l'utilisation d'un style de programmation approprié (c'est-à-dire le respect des règles de javaStyle), l'écriture de commentaires (en particulier les commentaires JavaDoc) sur chaque bloc de code du programme, etc.
- Les indicateurs spécifiques sont liés à chaque question du sujet de TP, par exemple, des indicateurs sur la maîtrise de la notion d'encapsulation, sur la connaissance de la différence entre overriding et overloading, etc. Ce sont ces indicateurs qui aident l'enseignant/tuteur à surveiller le déroulement de la situation d'apprentissage.

7.1.7.1 Indicateurs spécifiques

Les indicateurs spécifiques sont proposés pour chaque question du scénario. Nous décrivons ici les indicateurs pour les questions 1 et 2. Les autres sont listées dans l'annexe B.

Les indicateurs pour la question 1 :

1. La détection de l'existence d'une classe s'appelant ‘‘Point’’ : cet indicateur a pour but de vérifier si l'apprenant respecte la consigne et s'il sait comment créer une classe. L'existence d'une classe Point est la valeur attendue de cet indicateur.
2. Quelle est la visibilité de la classe ‘‘Point’’?: ‘‘public’’ est la valeur que l'enseignant veut avoir.
3. Le nombre de champs de la classe ‘‘Point’’ : la classe ‘‘Point’’ est composée de deux données représentant les coordonnées d'un point en x et y. Donc, la valeur attendue de cet indicateur doit être 2.

4. Quels sont les types de donnée de ces champs ? : les valeurs des coordonnées d'un point en x et y doivent être réelles. Donc, la valeur attendue de cet indicateur doit être ‘‘float’’ ou ‘‘double’’.

5. Quelle est la visibilité de ces champs ? : ‘‘private’’ est la valeur que l'enseignant veut avoir pour cet indicateur.

6. La détection de l'existence d'un constructeur spécifique à 2 arguments ? un constructeur spécifique à 2 arguments est la valeur que l'enseignant veut avoir pour cet indicateur.

7. Quels sont les types de donnée des arguments de ce constructeur ? : ‘‘float’’ ou ‘‘double’’ est la valeur attendue de cet indicateur.

8. La détection de l'existence des méthodes ‘‘getter''. Dans la programmation orientée objet en Java, une méthode *get* s'emploie pour lire un attribut de la classe (une variable d'instance privée). La classe ‘‘Point’’ est composée de deux données représentant les coordonnées d'un point en x et y. Donc, la valeur attendue de cet indicateur doit être une méthode *get* pour x et une méthode *get* pour y.

9. La détection de l'existence des méthodes ‘‘setter''. De façon similaire, une méthode *set* s'emploie pour écrire une valeur dans un attribut de la classe (une variable d'instance privée). La classe ‘‘Point’’ est composée de deux données représentant les coordonnées d'un point en x et y. Donc, la valeur attendue de cet indicateur doit être une méthode *set* pour x et une méthode *set* pour y.

Les indicateurs pour la question 2 :

1. La détection de l'existence d'une méthode appelée ‘‘equals’’ et non ‘‘Equals’’ dans la classe ‘‘Point’’ : une méthode appelée ‘‘equals’’ est la valeur attendue de cet indicateur.

2. Quelle est la visibilité de cette méthode ? : ‘‘public’’ est la valeur que l'enseignant veut avoir.

3. Quel est le type de retour de cette méthode ? Le type de ‘‘booléen*’’* est le résultat attendu de l'indicateur.

4. Le nombre d'arguments de la méthode. La valeur attendue de cet indicateur doit être 1.

5. Quel est le type de donnée de l'argument de cette méthode ? La valeur attendue de cet indicateur est le type de ‘‘java.lang.Object’’

7.1.7.2 Indicateurs transversaux

Les indicateurs transversaux pour ce TP sont les suivants :

1. La fréquence de compilation manuelle (par minute) : cet indicateur a pour but de vérifier si l'apprenant compile régulièrement son programme.
2. La fréquence d'exécution (par minute) : cet indicateur a pour but de vérifier si l'apprenant exécute régulièrement son programme.
3. Le taux de correction des erreurs à la compilation : cet indicateur a pour but de vérifier si une même erreur revient souvent ou si l'étudiant a assimilé qu'il s'agissait d'une erreur.
4. Le pourcentage de noms de variable significatifs (on se basera sur la longueur des noms) : cet indicateur aide l'enseignant/tuteur à savoir si l'étudiant respecte les règles de programmation. Le calcul de cet indicateur prenant en compte la longueur des noms des variables n'est pas de notre choix. Il s'agit d'une solution préconisée par l'enseignant.
5. De la même façon, le pourcentage de noms de méthode significatifs (on se basera aussi sur la longueur des noms).
6. Le pourcentage de variables d'instances privées : cet indicateur aide l'enseignant/tuteur à obtenir le pourcentage de variables d'instances privées par rapport à d'autres variables (variables d'instances publiques et variables de classe).
7. La détection de l'utilisation d'un membre d'instance (variable ou méthode) dans une méthode d'instance (s'il n'y a pas de membre d'instance, la méthode devrait être une méthode de classe).
8. Quel est le temps moyen passé par question pour chaque étudiant ? : cet indicateur aide l'enseignant à modifier le nombre de questions dans une session de TP.
9. Le nombre de compilations par question et par étudiant : cet indicateur a pour but de vérifier si l'apprenant compile régulièrement son programme pour chaque question.

7.1.8 Les données UTL

Cette partie décrit les données primaires et dérivées au sens d'UTL nécessaires pour élaborer les indicateurs.

7.1.8.1 Les données brutes

Comme nous l'avons expliqué dans la section 7.1.3 (la page 156), les traces collectées par Hop3x-serveur sont représentées par des événements. En ce qui concerne les données brutes UTL, chaque événement correspond à une donnée brute. Ainsi, nous avons 23 données brutes à modéliser. Ces données brutes sont nécessaires pour calculer notamment les indicateurs transversaux. Les autres indicateurs sont basés sur une donnée de production correspondant à une description du fichier JAVA. La figure 7-4 présente une donnée brute utilisée dans cette expérimentation.

```
<rawDatum>
    <defining>
        <title>PEDALO-RD-SelectionQuestion</title>
        <cardinality>1</cardinality>
        <description>Selection d'une question</description>
    </defining>
    <getting>
        <acquisitionTime>During-Session</acquisitionTime>
        <collectionType>
            <automaticCollection>
                <recordType>Log file</recordType>
                <recordTool>
                    <title>Hop3X</title>
                    <description>Un EIAH pour apprendre la programmation Java</description>
                </recordTool>
                <track>
                    <content>
                        <category>Value</category>
                        <title>Session identifier</title>
                        <type>XML</type>
                        <path>/TRACE/session</path>
                    </content>
                    ...
    </getting>
    <using>
       <data/>
        <format>
            <rawDatum type="PEDALO-RD-SelectionQuestion">
                <session>String</session>
                <user>String</user>
                <E K="SQ" T="String">
                    <NQ>String</NQ>
                </E>
            </rawDatum>
        </format>
        <usedBy>usedBy</usedBy>
    </using>
</rawDatum>
```

Figure 7-4 Donnée brute ‘‘Sélection d'une question’’

7.1.8.2 Les données de production

En ce qui concerne les indicateurs spécifiques, leurs calculs doivent prendre en compte le contenu des fichiers Java créés par l'étudiant pendant la session. Il s'agit d'une donnée de

production au sens d'UTL. Avec l'aide de l'enseignant, nous en avons créé une pour ce type de donnée. Le contenu de cette donnée est une cartographie d'un projet Java qui se compose des informations comme : le nom de projet, le nom de classe, le nombre de constructeurs, le nom des champs, etc. La figure 7-5 présente un exemple de cette cartographie.

```
<datum time="1274273786747" heure="14:56:26, 747" event="CM" diagnostic="compilation_OK">
        <projet nom="Pr1">
            <classe nom="Point" visibilite="public" isInterface="false" nomSuperClasse="java.lang.Object"
                    nbInterfaces="0" nbConstructeurs="1" statut="fichier_COMPILE">
                <interfaces/>
                <constructeurs>
                    <constructeur nom="Point" nbParametres="2" nbExceptions="0" body="{&#xA;}">
                        <parametres>
                            <parametre>double</parametre>
                            <parametre>double</parametre>
                        </parametres>
                        <exceptions/>
                    </constructeur>
                </constructeurs>
                <champs nb="2">
                    <champ nom="y" type="double" visibilite="private" isStatic="false" isFinal="false"/>
                    <champ nom="x" type="double" visibilite="private" isStatic="false" isFinal="false"/>
                </champs>
                <methodes nb="0"/>
                <features>
                    <feature nom="nbCommentairesJavaDoc" valeur="1"/>
                </features>
            </classe>
        </projet>
</datum>
```

Figure 7-5 Exemple de cartographie d'un projet Java

7.1.8.3 Les données additionnelles

Dans cette expérimentation, nous avons défini deux données additionnelles qui concernent le scénario et les groupes d'étudiants. La première décrit la liste des questions que l'étudiant doit réaliser. La deuxième décrit la liste des étudiants de chaque groupe. Ces données sont aussi créées avec l'aide de l'enseignant avant la session.

7.1.8.4 Les données intermédiaires

Dans cette expérimentation, nous trouvons plusieurs indicateurs similaires. Par exemple, les trois indicateurs suivants sont quasiment identiques, ils varient sur le nom de la classe :

1. La détection de l'existence d'une classe s'appelant ‘‘Point’’,
2. La détection de l'existence d'une classe s'appelant ‘‘Triangle’’,
3. La détection de l'existence d'une classe s'appelant ‘‘Droite’’.

L'objectif principal de ces indicateurs est de vérifier si les apprenants respectent les instructions données dans les questions et s'ils savent comment créer une classe Java. L'existence des classes "Point", "Droite" et "Triangle" sont les valeurs attendues de ces

indicateurs. Puisque ces indicateurs sont similaires, et plutôt que de modéliser trois indicateurs différents sans réutiliser la méthode de calcul, nous avons utilisé une donnée intermédiaire paramétrée (PID, cf. chapitre 5) pour les modéliser en réutilisant la méthode de calcul. Ceci est réalisé en deux étapes :

1. Créer une PID selon les trois facettes du modèle DGU dont la méthode de calcul comprend un paramètre formel (*className,* cf. figure 7-6) en utilisant la caractéristique "*parameter"* du langage DCL4UTL. Ce paramètre représente le nom de la classe, tandis que la PID correspond au calcul de "*la détection de l'existence d'une classe*". La figure 7-6 montre la méthode de calcul de cette PID. Dans cette figure, la PID (nommée *PEDALO-ID-Class*) est calculée à partir d'une donnée de production (*PEDALO-CD-ClassDetail*) et d'une donnée brute (nommée *PEDALO-RD-SelectionQuestion*). La donnée *PEDALO-CD-ClassDetail* se compose des informations concernant les classes créées par l'apprenant (cf. section 7.1.8.2). La donnée *PEDALO-RD-SelectionQuestion* concerne le choix de question de l'apprenant. Cette donnée est utilisée car pour chaque indicateur, l'enseignant veut savoir dans quelle question se situe l'apprenant.

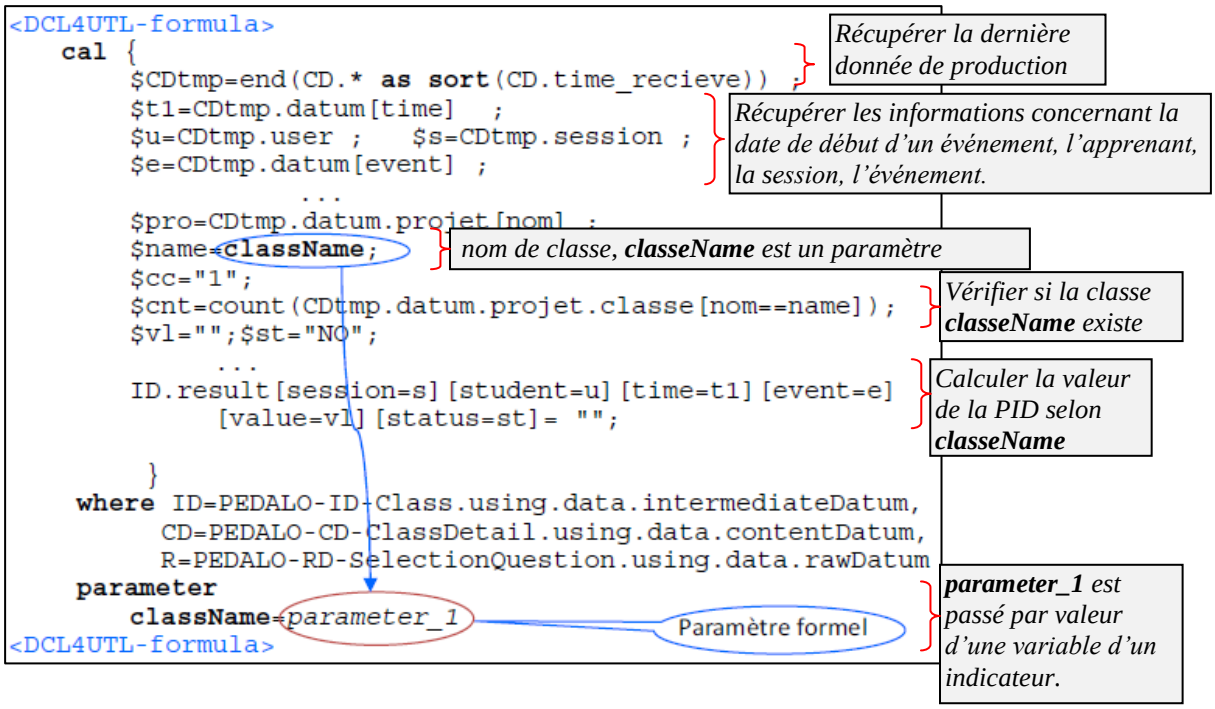

Figure 7-6 Extrait de la méthode de calcul d'une donnée intermédiaire paramétrée

2. Créer trois indicateurs selon les trois facettes du modèle DGU en réutilisant la PID créée à la première étape. Le détail de la modélisation d'un indicateur selon ces trois facettes n'est pas présenté ici parce qu'il est déjà décrit dans le chapitre 5 (cf. section

5.4.2, les tableaux 5-16, 5-17 et 5-18). La formule de calcul de chaque indicateur est modélisée en passant une valeur ("Point", "Droite" ou "Triangle") du paramètre réel au paramètre formel de la PID. La figure 7-9 montre la méthode de calcul de l'indicateur ‘‘*la détection de l'existence d'une classe s'appelant Point*". Cet indicateur est décrit dans la section suivante.

7.1.8.5 Les indicateurs

Pour la modélisation et l'élaboration des indicateurs, comme nous l'avons expliqué ci-dessus, une phase de négociation a été entreprise entre l'enseignant et le développeur afin de spécifier la méthode de calcul ainsi que le format du résultat. Les facettes *getting* et *using* ont été définies dans ce but. La section *getting* permet de définir correctement la méthode de calcul ainsi que les données primaires et/ou les données dérivées qui sont nécessaires pour le calcul. L'élément *using* des indicateurs nécessite de spécifier le format qui correspond aux attentes de l'enseignant. Après la négociation, le format du résultat de tous les indicateurs est défini selon l'exemple de la figure 7-7.

```
<format>
      <indicator type="PEDALO-I-ClassPointPublic">
            <result    session="String"
                       student="String"
                       time="String"
                       event="String"
                       value="String"
                       status="String"
                       calculable="String"
                       reference="String"
                       location="String"
                       currentQuestion="String"
                       concernedQuestion="String"
                       category="String"/>
      </indicator>
</format>
```

Figure 7-7 Le format de résultat d'un indicateur pour Hop3x

Le tableau 7-2 décrit les éléments du format. Tous ces éléments sont également demandés par l'enseignant.

Nous avons modélisé quatre-vingt-trois indicateurs correspondant aux besoins d'observation définis par l'enseignant. Tous ces indicateurs sont modélisés avant la session. La figure 7-8 présente la formule de calcul de l'indicateur *“la détection de l'existence d'une classe*

s'appelant Point" (nommé *PEDALO-I-ClassPointExist)*. Dans la modélisation de cet indicateur, nous avons utilisé une PID *PEDALO-ID-ClassExist* qui *est* décrite dans la section 7.1.8.4 (cf. figure 7-6).

Elément	Description
type	Nom d'indicateur
session	Nom de session d'apprentissage
student	Nom d'étudiant
time	L'instant où l'étudiant a lancé un événement
event	Un événement que l'étudiant a réalisé
value	La valeur calculée de l'indicateur
status	Le statut de l'indicateur, ce statut est déduit à partir de sa valeur. Il permet à l'enseignant de savoir si la valeur de l'indicateur satisfait à son attente.
calculable	Le statut de calcul de l'indicateur, ce statut permet de savoir si un indicateur peut être calculé. Cela dépend des données de production. Par exemple, l'indicateur "*détection de l'existence d'une méthode appelée equals et non Equals dans la classe Point"* ne peut pas être calculé si la classe *Point* n'existe pas.
reference	La valeur référentielle (attendue) de l'indicateur
location	Localisation d'une variable ou d'une méthode dans une classe java
currentQuestion	Question courante que l'apprenant est entrain de faire
concernedQuestion	Question possible (parmi les12 questions) pour laquelle l'indicateur est calculé.
category	Type d'indicateur, soit spécifique, soit transversal

Tableau 7-2 Les éléments du format de résultat d'un indicateur Hop3x

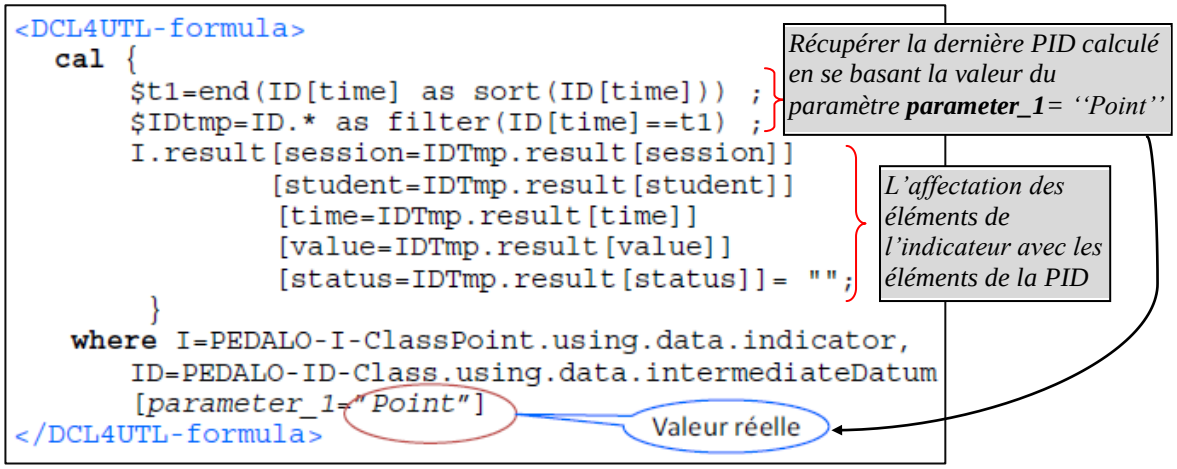

Figure 7-8 La méthode de calcul de l'indicateur *PEDALO-I-ClassPointExist*

Nous avons également utilisé certaines fonctions externes. Elles sont conçues par le développeur dans le processus de modélisation des indicateurs pour analyser les messages

d'erreur générés lors de la compilation d'un programme. Ces fonctions externes sont utilisées pour calculer l'indicateur *''Le taux de correction des erreurs à la compilation"*. La figure 7-9 extrait la méthode de calcul de cet indicateur. Dans cette figure, la fonction externe *''numberError"* est utilisée pour calculer le nombre d'erreurs de compilation à partir d'un message d'erreur généré par Hop3x. Cette fonction est écrite indépendamment de notre outil d'analyse. Lors de l'exécution, les fonctions externes doivent être placées au même endroit que notre outil d'analyse.

```
<description>
  <text>Calculer le taux de correction des erreurs à la compilation</text>
  <DCL4UTL-formula>
    cal {
      $Rtmp=end(R.* as sort(R.E[T] ));
                  $t1 = Rtmp.E[T] ;
      ...
      $numCurrent = extern  utl.external.hop3x.CompilationErrorDetection numberError(Rtmp.E.S );
      if(numCurrent == 0) {vl = "erreur";
      ....
   }
   where I = PEDALO-I-ErrorCorrectionRate.using.data.indicator,
         R = PEDALO-RD-CompilationManuelle.using.data.rawDatum,
         R1 = PEDALO-RD-SelectionQuestion.using.data.rawDatum
  </DCL4UTL-formula>
</description>
```

Fonction externe

Figure 7-9 Extrait de la méthode de calcul de l'indicateur *''Le taux de correction des erreurs à la compilation"*

7.1.9 Calcul des indicateurs en temps réel

Les indicateurs modélisés avant la session sont calculés en temps réel durant la session. Dans cette expérimentation, l'enseignant s'intéresse aux traces liées aux événements : choix d'une question (SQ), compilation manuelle (CM) et exécution d'un programme (E). Il veut avoir la valeur des indicateurs chaque fois qu'une trace correspondant à l'un de ces événements est produite. Le déclenchement du calcul des indicateurs est donc effectué chaque fois que l'étudiant réalise un des trois événements : SQ, E ou CM. A chaque événement correspond certains indicateurs transversaux et spécifiques. Par exemple, si l'étudiant *compile son programme*, les indicateurs comme *''la fréquence de compilation manuelle"*, *''le taux de correction des erreurs à la compilation"* sont évalués. Si l'étudiant passe d'*une question à une autre*, les indicateurs comme *''le temps moyen passé par question par étudiant"*, *''le nombre de compilations par question par étudiant"* sont établis. Ces indicateurs transversaux ne dépendent pas des questions traitées par l'apprenant. Donc, le nombre d'indicateurs transversaux à calculer pour chaque événement ne change pas pendant la session.

Par contre, le nombre d'indicateurs spécifiques dépend des questions réalisées par l'étudiant. Au début, si un des trois événements (SQ, E ou CM) est généré, les indicateurs spécifiques de la première question choisie par l'apprenant sont calculés. Puis chaque fois que l'étudiant réalise un des trois événements, tous les indicateurs spécifiques des questions traitées par l'étudiant sont recalculés. Donc, si l'étudiant réalise plusieurs questions, plusieurs indicateurs spécifiques sont calculés lorsqu'il passe d'une question à une autre, compile ou exécute son programme. Les indicateurs sont recalculés car lorsque l'apprenant réalise une question, il modifie son code existant. Il peut donc faire de nouvelles erreurs. Ce calcul a pour but de vérifier si de nouvelles erreurs peuvent être produites.

Comme nous l'avons présenté dans la section 5.3, le déclenchement du calcul d'une liste d'indicateurs liés à un événement est fait grâce au champ *CalculationPeriod* de la facette *getting* de chaque indicateur modélisé.

7.1.10 Description du prototype

Cette section présente une application de l'architecture proposée dans le chapitre précédent sur les traces générées par le système Hop3x. Ce modèle est un cas particulier qui comprend l'échange des données entre notre outil d'analyse et l'environnement de formation Hop3x. En effet, l'outil d'analyse reçoit les traces générées par Hop3x, mais envoie aussi des requêtes pour demander les données de production nécessaires pour le calcul d'indicateurs. Nous détaillons cet échange dans la suite. Pour permettre ce dialogue particulier en temps réel, nous avons développé un module complémentaire, nommé *Interface UTL-Hop3x* (cf. figure 7-10).

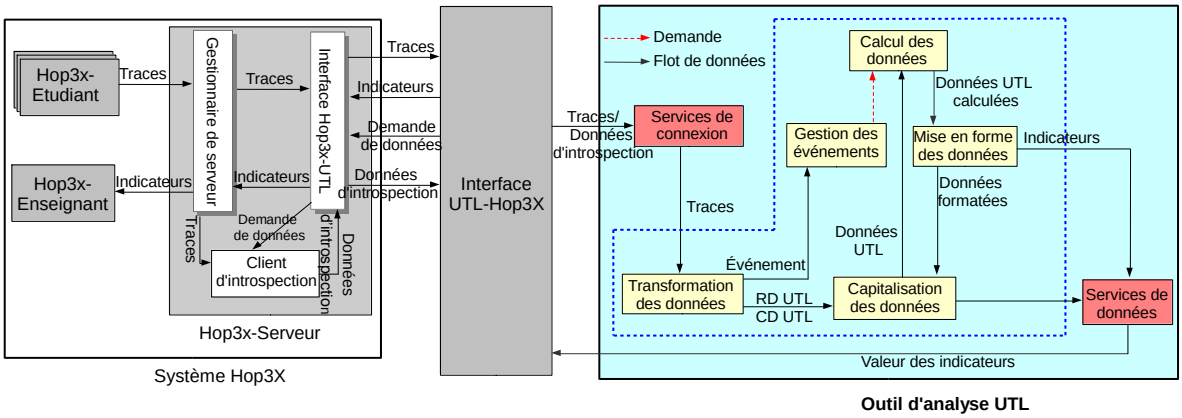

Figure 7-10 L'architecture de l'outil d'analyse en interaction avec Hop3x

L'échange des données entre notre outil et Hop3x est réalisé à deux niveaux. Une première gestion de dialogue entre le composant "*Interface Hop3x-UTL*" de Hop3x-Serveur et le

composant ‘‘*Interface UTL-Hop3x”* permet d'échanger en temps réel les traces collectées par le ‘‘*Gestionnaire de Serveur”* de Hop3x-Serveur. Une deuxième gestion de dialogue est intégrée afin d'échanger des données de production entre un ‘‘*Client d'introspection”* d'Hop3x (via le composant ‘‘*Interface Hop3x-UTL”*) et notre outil (via le composant ‘‘*Interface UTL-Hop3x”*). *Le* ‘‘*Client d'introspection”* a pour objectif d'analyser le code tapé par les étudiants et d'en retenir une cartographie d'un projet Java avec un certain nombre de caractéristiques (le nombre de méthodes, leurs prototypes, le nombre de variables de classe, d'instance, etc.).

Ce système fonctionne de la façon suivante. Le composant ‘‘*Interface UTL-Hop3x”* reçoit les traces d'Hop3x en temps réel. Il envoie ces dernières au composant ‘‘*Service de connexion”*. Ce composant vérifie également si la trace reçue appartient au type CM, SQ ou E, il sollicite le ‘‘*Client d'introspection”* afin d'obtenir les informations détaillées des fichiers Java d'un étudiant à un instant donné. Il s'agit de données de production au sens d'UTL. Il attend le résultat des requêtes du ‘‘*Client d'introspection”* pour l'envoyer au composant ‘‘*Service de connexion”*.

Le composant ‘‘*Transformation des données”* transforme en temps réel les traces et les données de production reçues par le composant ‘‘*Service de connexion”* en données brutes (RD) et données de production (CD) d'UTL qui sont capitalisées par le composant ‘‘*Capitalisation des données”*. Il sollicite ensuite le composant ‘‘*Gestion des événements”* pour que ce composant vérifie la donnée obtenue.

Le composant ‘‘*Gestion des événements”* identifie la donnée obtenue selon l'événement et déclenche le calcul d'indicateurs si la trace appartient au type CM, SQ ou E. Lorsqu'une trace est transformée en donnée brute, le composant *Gestion des événements* la identifie pour déclencher un calcul d'indicateurs transversaux. Dès qu'une donnée de production est produite, le composant *Gestion des événements* déclenche le calcul d'indicateurs spécifiques en se basant sur ces données de production.

Le composant *Calcul des données* calcule les indicateurs (spécifiques et transversaux) correspondant à chaque événement pour lequel le composant *Gestion des événements* déclenche les calculs.

Le résultat des indicateurs est capitalisé par le *composant Capitalisation* des données et est envoyé au composant *Service des données*. Ce dernier renvoie ces résultats à un composant d'Hop3x-Serveur via le composant *Interface UTL-Hop3x*.

Le résultat des indicateurs est ensuite adressé à l'interface d'enseignant Hop3x par un composant d'Hop3x-Serveur. La figure 7-11 présente le résultat des indicateurs spécifiques et la figure 7-12 présente le résultat des indicateurs transversaux.

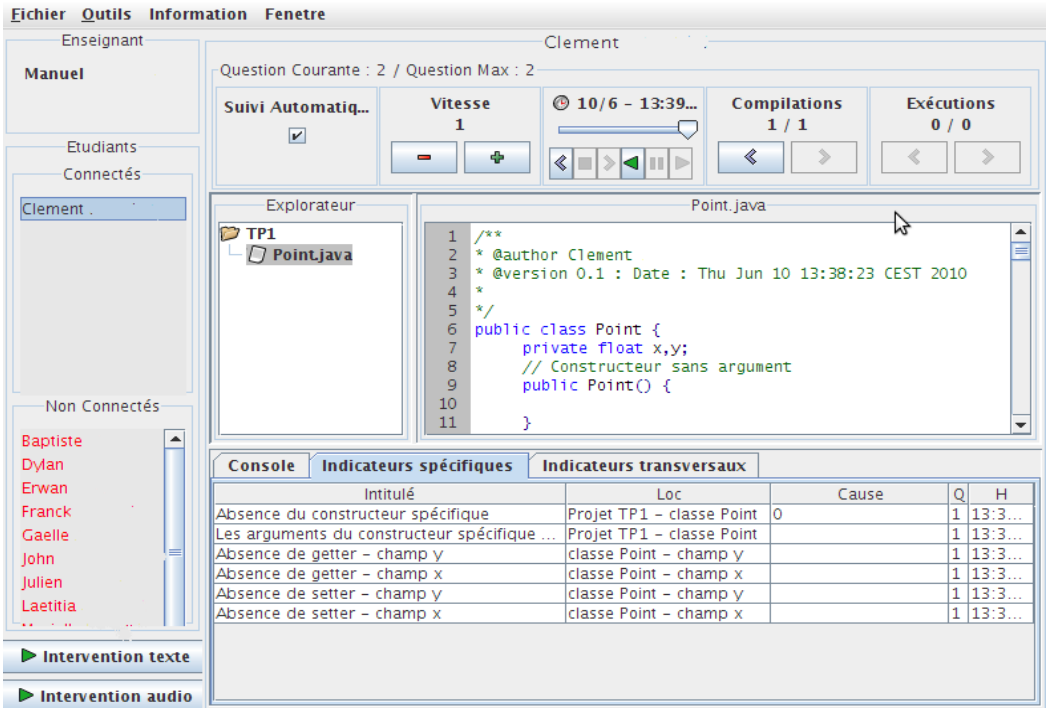

Figure 7-11 L'enseignant Hop3x avec des indicateurs spécifiques

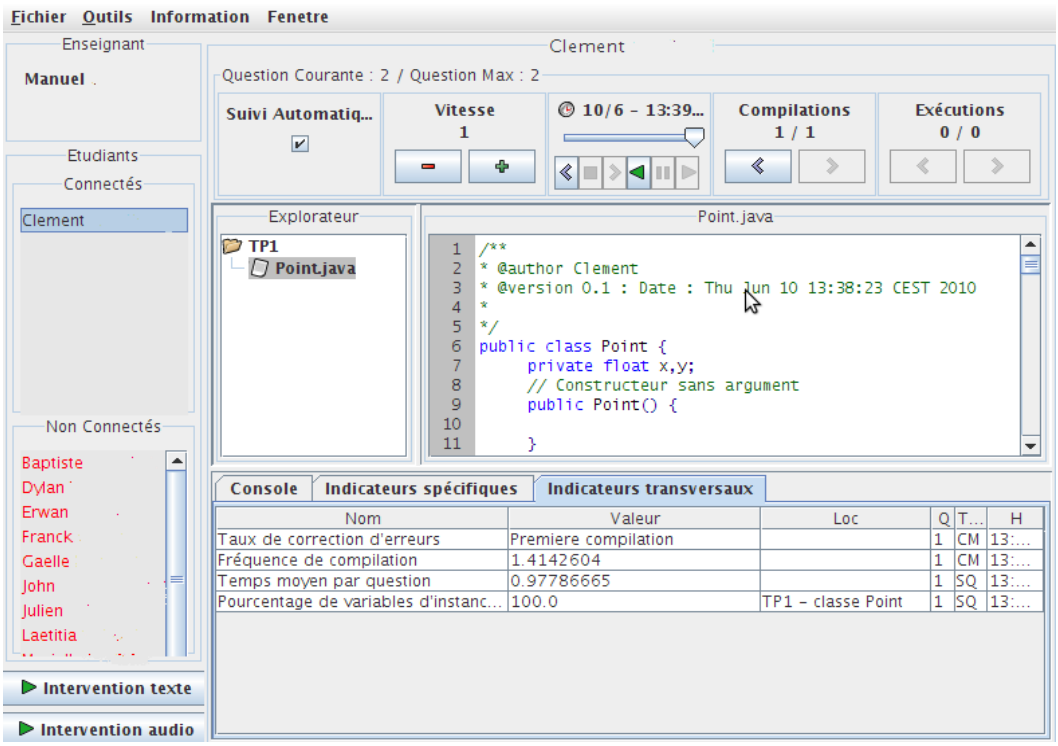

Figure 7-12 L'enseignant Hop3x avec des indicateurs transversaux

L'enseignant utilise ces résultats pour évaluer son scénario ou ses apprenants et pour proposer des actions tutorales appropriées afin d'améliorer la session d'apprentissage. Par exemple, en se basant sur l'indicateur *"est-ce qu'une même erreur revient souvent ou est-ce que l'étudiant a assimilé qu'il s'agissait d'une erreur?"*, l'enseignant peut donner des conseils ou de l'aide aux étudiants qui n'arrivent pas à assimiler leurs erreurs. Dans un autre exemple, si l'indicateur *"le temps moyen passé par question"* donne généralement une valeur qui est plus grande que la valeur de référence, l'enseignant peut modifier son scénario pour la prochaine session ou proposer des solutions afin d'aider les étudiants à résoudre leurs problèmes plus rapidement, etc. Pour réguler les activités d'apprentissage des apprenants durant la session avec l'appui des indicateurs, le tuteur peut choisir des indicateurs dans la liste des indicateurs pour réaliser des interventions textuelles ou audio (cf. les figures 7-11 et 7-12).

7.2 Analyse des résultats de l'expérimentation

Cette section présente les résultats obtenus lors de l'expérimentation ainsi que les discussions sur ces résultats concernant les objectifs de l'expérimentation.

7.2.1 Résultats obtenus

Nous avons réalisé l'expérimentation sur six groupes d'étudiants (GA, GB, GC, GD, GE, GF). Nous avons obtenu au total 218773 instances de données brutes(RD), 3943 instances de donnée de production (CD) et 135581 instances d'indicateurs (I). Plusieurs interventions audio et interventions textuelles ont été réalisées. Le tableau 7-3 résume les données de chaque groupe.

	GA	GB	GC	GD	GE	GF
RD	32713	48125	38632	27914	40003	31386
CM	245	474	629	370	495	185
E	125	128	62	55	239	22
SQ	192	194	223	174	72	192
CM+SQ+E	562	796	914	599	806	399
Nombre d'étudiants	14	15	14	11	15	14
CD	547	747	891	586	791	381
I	17798	26668	28032	21165	29156	12762
Nombre d'indicateurs moyen par étudiant	1271.29	1777.87	2002.29	1924.09	1943.73	911.57
Interventions audios	20	26	26	8	29	39
Interventions textuelles	38	14	26	38	17	21

Tableau 7-3 Les données obtenues lors de l'expérimentation Hop3x

Dans le tableau 7-3, nous pouvons voir que la somme des événements (CM+SQ+E) n'est pas en corrélation avec le nombre de données de production. Elle est toujours supérieure au dernier. En théorie, chaque fois qu'un événement SQ, CM ou E est réalisé, une donnée de production CD est générée. Toutefois, en pratique, des données de production générées au début de la session peuvent être vides lorsque l'étudiant choisit continuellement une ou plusieurs questions et qu'il ne fait rien. Donc, le nombre de données de production est différent de la somme d'événements SQ, CM et E.

7.2.2 Discussions

7.2.2.1 La capacité de calcul

Cette expérimentation nous a permis de modéliser la méthode de calcul de 83 indicateurs et de recueillir 218773 instances de données brutes. Notre outil a produit 135581 instances d'indicateurs. En moyenne, environ 3.37 données brutes sont produites par seconde et 2.09 indicateurs sont calculés par seconde. Durant tout le processus d'expérimentation, notre outil a bien répondu au calcul des indicateurs en temps réel avec une session de 15 étudiants. En pratique, il y a des moments où il n'y a pas d'indicateurs calculés, mais de temps en temps, plusieurs indicateurs sont calculés continuellement. Dans le cas où plusieurs des indicateurs sont calculés continuellement (lorsque plusieurs événements sont générés continuellement), le temps différé maximum entre un événement généré et une valeur de l'indicateur produite correspondant à cet événement est de dix secondes, parce que le calcul des indicateurs est déclenché lorsqu'un événement est généré et le nombre d'indicateurs à calculer est augmenté selon des questions effectuées. Par exemple, quand on est à question 10, à chaque événement généré, environs 70 indicateurs à recalculer et ça prend du temps si plusieurs événements sont continuellement générés. Le nombre d'instances des indicateurs (I) et celui des données brutes (RD) pour chaque groupe sont décrits dans la figure 7-13 a). Le nombre moyen d'instances des indicateurs (I) par seconde et celui des données brutes (RD) par seconde pour chaque groupe sont décrits dans la figure 7-13 b).

Comme cela est précisé dans la figure 7-13 b), la plus grande valeur moyenne des indicateurs est 2.7 et des données brutes est 4.46. C'est-à-dire que dans cette expérimentation, en moyenne, chaque étudiant a réalisé au maximum 0.3 (4.46/15 ~ 0.3) événement par seconde, et le tuteur a reçu au maximum 2.7 indicateurs en une seconde. Cependant, nous avons observé que le tuteur a parfois reçu trop d'indicateurs parce que plusieurs étudiants ont réalisé en même temps des événements qui déclenchent le calcul d'indicateurs. Dans ce cas, le tuteur ne peut pas observer tous ses étudiants. On observe alors un phénomène de surcharge cognitive.

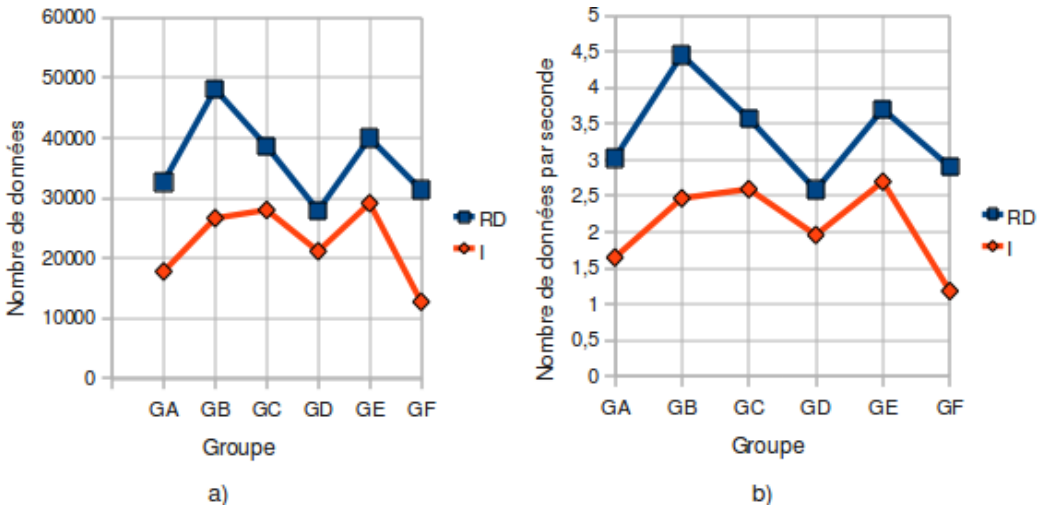

Figure 7-13 Le diagramme de données brutes et d'indicateurs

Pour estimer la capacité de calcul de notre système, nous avons réalisé une simulation sur les données brutes obtenues lors de l'expérimentation. Cette simulation est faite sur le même système que l'expérimentation. Avec un temps de simulation de 4187 secondes, notre outil a obtenu 65819 données brutes et a produit 48102 instances des indicateurs. En moyenne, environ 15,72 événements sont réalisés et 11,49 indicateurs sont calculés par seconde. Nous en déduisons que le système peut répondre au maximum avec 52 étudiants (15.72/0.3=52.4).Toutefois, cette capacité peut être supérieure ou inférieure à cette valeur. Cela dépend de plusieurs facteurs, par exemple, les matériels informatiques, le niveau de connaissance des apprenants, le nombre d'indicateurs à calculer, etc. De plus, des groupes avec peu d'étudiants sont préconisés parce qu'un tuteur ne peut pas observer en même temps trop d'étudiants.

7.2.2.2 Utilité des langages UTL et DCL4UTL

Pour analyser l'utilité des langages UTL et DCL4UTL ainsi que notre outil, nous présentons brièvement deux autres expérimentations réalisées au département informatique, l'Université du Maine [Lekira et al. (2011)]. Ces expérimentations ont utilisé le même le scénario pédagogique que celui décrit dans la section 7.1.5 et sont réalisées avec les étudiants en troisième année de Licence d'Informatique. Les deux mêmes enseignants ont participé à ces deux expérimentations. La communication entre enseignants et étudiants a été également réalisée via l'outil Hop3x.

La première expérimentation (*expérimentation A*) a eu lieu de Janvier à Mars 2009 avec 48 apprenants répartis en trois groupes (deux groupes de dix-huit et un groupe de quatorze). Les données recueillies dans les trois groupes sont similaires en termes de nombre moyen

d'événements produits par apprenant. En moyenne, chaque étudiant dans *l'expérimentation A* a réalisé environ 4050 événements. Cette expérimentation ne comportait pas d'indicateurs calculés.

La deuxième expérimentation (*expérimentation B*) est réalisée de Janvier à Mars 2010. Elle a impliqué 36 étudiants divisés en deux groupes de dix-huit. Les données recueillies entre les deux groupes sont similaires en termes de nombre moyen d'événements produits par étudiant. En moyenne, chaque étudiant dans *l'expérimentation B* a réalisé environ 3995 événements. Cette expérimentation intégrait les indicateurs. Ces derniers ont été modélisés avec les langages UTL et DCL4UTL et calculés par notre outil d'analyse.

Comme nous l'avons indiqué ci-dessus, la différence entre les deux expérimentations en termes de quantité d'événements produits par apprenant est faible (4050 événements pour *l'expérimentation A* et 3995 événements pour *l'expérimentation B*). Cependant, le nombre d'interventions entre les deux expérimentations est très différent. On a obtenu en moyenne 6 interventions par groupe durant *l'expérimentation A* et 27 interventions par groupe durant *l'expérimentation B*. Cette différence peut s'expliquer par le fait que dans *l'expérimentation A*, les enseignants ont eu très peu d'informations sur les activités des étudiants (car les indicateurs n'étaient pas inclus dans cette expérimentation). Par conséquent, ils sont intervenus principalement pour répondre aux demandes des étudiants. Pour *l'expérimentation B*, plusieurs indicateurs ont été calculés et envoyés aux enseignants. En se basant sur les valeurs de ces indicateurs, les enseignants ont pu observer et suivre en temps réel les activités des apprenants. Ils ont pu détecter les problèmes des apprenants (par exemple, il manque une méthode *set* pour une variable d'instance, une variable d'instance sans visibilité privée, etc.). Ils ont donc pu prendre des décisions pédagogiques en faisant des interventions. Grâce à ces indicateurs, les enseignants sont intervenus 4,5 fois plus que dans *l'expérimentation A*. De plus, avec ces interventions, on a estimé qu'environ 53,61% des problèmes dans lesquels les valeurs des indicateurs ne satisfaisaient pas les attentes des enseignants ont été résolus pour *l'expérimentation A* et environ 73,52% pour *l'expérimentation B* [Lekira et al. (2011)]. Les groupes de *l'expérimentation B* dans lesquels les enseignants avaient des indicateurs ont alors mieux résolu leurs problèmes que ceux de *l'expérimentation A* dans lesquels les enseignants n'avaient pas d'indicateurs.

A partir de ces deux expérimentations, nous avons pu constater que les langages UTL et DCL4UTL ainsi que notre outil sont utiles pour l'enseignant/tuteur. Ils ont fourni les indicateurs calculés à l'enseignant/tuteur. Ces indicateurs ont aidé l'enseignant/tuteur à avoir différentes informations sur le déroulement d'une session d'apprentissage. En se basant les

valeurs de ces indicateurs, l'enseignant/tuteur a pu superviser les activités des apprenants et donc détecter leurs problèmes, mais aussi réguler leurs activités. En ce qui concerne notre expérimentation au sein de l'IUT de Laval, les tuteurs ont pris des décisions tutorielles appropriées pour chaque étudiant. Concrètement, les tuteurs ont réalisé 148 interventions audio et 154 interventions textuelles. En moyenne, environ 50 interventions sont produites pour chaque groupe. Ces interventions peuvent être une aide à la résolution des problèmes de programmation, une correction des erreurs de programmation, une question pour mieux comprendre un problème, etc. Selon les valeurs des indicateurs calculés en temps réel, environ 82% des interventions produisent l'effet attendu (pour l'expérimentation au sein de l'IUT de Laval), ce taux est environ de 75% pour l'*expérimentation B*. Toutes les interventions sont enregistrées par Hop3x-serveur. La figure 7-14 présente une trace d'une intervention audio réalisée en se basant l'interprétation des indicateurs ‘‘*Le nombre d'arguments de la méthode equals”* et ‘‘*Quel est le type de donnée de l'argument de cette méthode”*. L'objectif de cette intervention est d'indiquer les erreurs dans la définition de la méthode ‘‘*equals”* à l'apprenant, par exemple, la méthode ‘‘*equals”* doit avoir un seul argument de type *Object*.

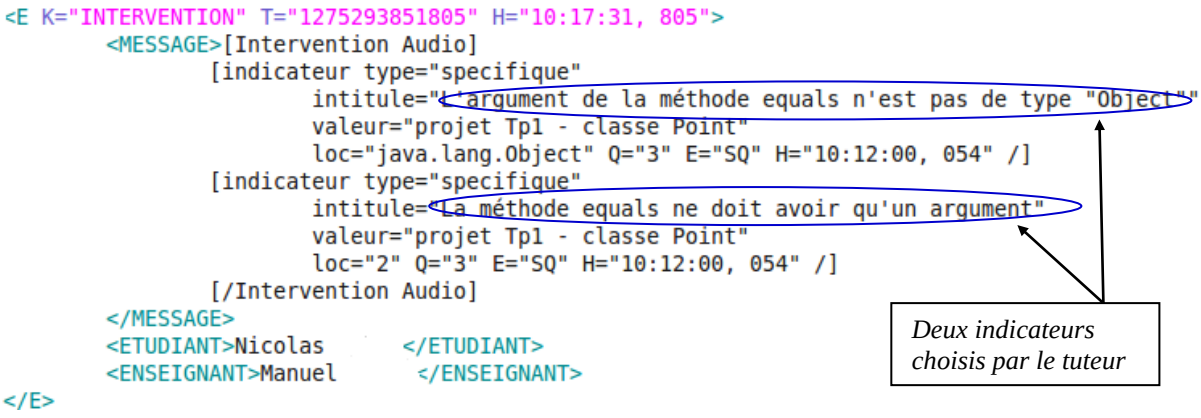

Figure 7-14 Une intervention audio retranscrite

Comme le montre la figure 7-15, une trace d'une intervention textuelle a été réalisée en se basant sur l'interprétation des indicateurs ‘‘*La détection de l'existence des méthodes “getter” ”* et ‘‘*La détection de l'existence des méthodes “setter” ”*. L'objectif de cette intervention est d'indiquer les erreurs dans la définition de la classe *Triangle* à l'apprenant, par exemple, il manque des accesseurs pour les variables privées. C'est-à-dire que l'apprenant a besoin de définir des méthodes *getter* et *setter* pour les variables privées.

Les indicateurs retenus dans deux traces (les figures 7-14 et 7-15) sont ceux choisis dans la liste des indicateurs (cf. les figures 7-11 et 7-12) par le tuteur au moment de l'intervention.

```
<E K="MESSAGE_ENSEIGNANT" T="1275299123093" H="11:45:23, 093">
        <POSDEB>206</POSDEB>
        <MESSAGE>En fait, dès que tu mets tes données en privées,
                il faut que tu définisses des accesseurs pour ces données,
                dans la classe Point, il te faut alors une méthode getX et
                une méthode getY que tu utiliseras dans Droite.
        </MESSAGE>
        <HEUREFICHIER>1275299056237</HEUREFICHIER>
        <F>Point.java</F>
        <ETUDIANT>Shengchen    </ETUDIANT>
        <INDICATEURS_SELECTIONNES>[INTERVENTION TEXTE][/INTERVENTION TEXTE]</INDICATEURS_SELECTIONNES>
        <SESSION>TP1GB</SESSION>
        <POSFIN>206</POSFIN>
        <ENSEIGNANT>Manuel          </ENSEIGNANT>
</E>
<E K="INTERVENTION" T="1275299133153" H="11:45:33, 153">
        <MESSAGE>[Intervention Texte]
                    [indicateur type="specifique"
                        intitule="Absence de setter - champ p3"
                        valeur="classe Triangle - champ p3"
                        loc="" Q="11" E="SQ" H="11:20:02, 244" /]
                ...
                [/Intervention Texte]
        </MESSAGE>
        <ETUDIANT>Shengchen    </ETUDIANT>
        <ENSEIGNANT>Manuel          </ENSEIGNANT>
</E>
```

Intervention

Indicateurs sélectionnés pour réaliser une intervention

Figure 7-15 Extrait d'une intervention textuelle

De plus, notre outil facilite également la communication ‘‘feedback’’ après la session entre les tuteurs et l'enseignant/concepteur pédagogique. Cette communication aide le concepteur à améliorer éventuellement le scénario pédagogique ainsi que dans la définition des besoins d'observation pour la prochaine session.

- En observant les valeurs des indicateurs, les tuteurs peuvent indiquer quels indicateurs donnent souvent une valeur attendue et quels indicateurs fournissent souvent une valeur inattendue. Si un indicateur a une valeur attendue, cela peut provenir de deux raisons : l'étudiant ne comprend pas la question ou l'étudiant n'a pas les connaissances requises pour résoudre la question. Pour la première raison, le concepteur devrait éventuellement considérer la modification du scénario. Par exemple, en ce qui concerne notre expérimentation, le tuteur a constaté que les résultats des indicateurs liés à la question 2 sont souvent inacceptables (inattendus). En utilisant les interventions associées à ces indicateurs, le tuteur a appris que la plupart des apprenants n'ont pas compris la question, comme par exemple, dans une session pour le groupe C, le tuteur a expliqué 13 fois l'utilisation de la méthode *equals* pour résoudre la question 2. Donc, la question 2 devra être modifiée pour que

les apprenants puissent mieux comprendre la signification de la méthode *equals* afin de l'appliquer dans la situation proposée. Dans un autre exemple, le concepteur a voulu avoir la valeur de l'indicateur ‘‘*Quel est le temps moyen passé par question par étudiant”*, par contre il n'indique pas le temps référentiel pour chaque question. S'il y a cette valeur référentielle, le tuteur peut évaluer la performance des apprenants. Donc, le concepteur devrait indiquer le temps pour réaliser chaque question dans son scénario.

- En utilisant les indicateurs associés aux interventions, le tuteur peut indiquer quels indicateurs sont nécessaires et quels indicateurs ne sont pas utiles. Cela aide le concepteur à améliorer la définition de ses besoins d'observation. Par exemple, plusieurs indicateurs ne sont pas utilisés par les tuteurs (par exemple *“le pourcentage de noms de variable significatifs”*, *“le pourcentage de noms de méthode significatifs”*). La plupart des indicateurs utilisés sont souvent spécifiques (par exemple des indicateurs concernant le type et la visibilité des variables et méthodes). C'est-à-dire que les indicateurs transversaux ne répondent pas encore complètement aux besoins des tuteurs. Ce qui signifie que ces indicateurs devraient être reconsidérés. Selon les tuteurs, certains indicateurs peuvent être rajoutés comme ‘‘*le parcours de chaque étudiant”* (c'est-à-dire la liste ordonnée des questions réalisées par l'apprenant). Cet indicateur aiderait le tuteur à évaluer la performance de l'apprenant. Un autre indicateur peut être rajouté comme ‘‘*La détection des intervalles de temps inactifs de chaque étudiant”* ou encore ‘‘*le nombre d'interventions que chaque apprenant a reçu à un moment donné”*, etc.

Nous souhaitons mettre l'accent sur le fait que la définition du scénario et la proposition des indicateurs ne font pas partie de notre travail. Notre objectif est de fournir un outil qui peut calculer des indicateurs pour assister l'enseignant/tuteur dans sa tâche (l'amélioration de scénario, la réalisation d'un tutorat, etc.). Comme nous l'avons présenté, nous travaillons en collaboration avec l'équipe ‘‘*ingénierie des EIAH*’’ au LIUM pour lui fournir les indicateurs calculés, les limites concernant le scénario et les indicateurs sont donc considérées par cette équipe.

7.3 Conclusion

Ce chapitre porte sur une expérimentation menée au département SRC au sein de l'IUT de Laval avec des étudiants de première année sur les traces générées par la plateforme de formation Hop3x. Notre langage DCL4UTL a permis de modéliser tous les indicateurs proposés par l'enseignant. Le prototype utilisé pour cette expérimentation applique

l'architecture proposée dans le chapitre précédent. Il a bien fonctionné pendant tout le processus d'expérimentation et calculé en temps réel 135581 instances de 83 indicateurs. Les résultats des indicateurs sont précis et permettent aux tuteurs de faire leurs interventions à temps.

Dans la modélisation des indicateurs, nous avons utilisé DCL4UTL et ses spécificités comme les fonctions externes, les données intermédiaires paramétrées pour décrire formellement la méthode de calcul d'indicateurs. Ces caractéristiques n'existent pas dans les autres approches existantes : l'approche de patron d'indicateur réutilisable [Diagne (2009)], l'approche SBT [Settouti (2011)], le calcul des indicateurs dirigé par les modèles de trace [Djouad et al. (2009)], l'approche de patron d'un indicateur de collaboration [Gendron (2010)], etc. Dans cette expérimentation, nous réutilisons également certains indicateurs pour calculer d'autres indicateurs, par exemple, le calcul d'indicateurs da la question 2 utilise l'indicateur *la détection de l'existence d'une classe s'appelant "Point"* de la question 1.

A travers cette expérimentation, notre outil a démontré son utilité dans l'aide aux tuteurs en observation des activités des apprenants pendant une session d'apprentissage. Grâce à l'observation et l'utilisation des résultats des indicateurs, les concepteurs comprennent l'importance des indicateurs dans l'amélioration du scénario d'apprentissage. De plus, l'outil les aide également à évaluer la pertinence (ou l'utilité) des indicateurs grâce à leur utilisation ainsi que le besoin d'ajouter de nouveaux indicateurs.

Comme nous l'avons présenté dans le chapitre 4, la définition des indicateurs dépend des compétences et de l'expérience de l'enseignant/concepteur dans la spécification ce qu'il veut observer. Nous pouvons donc constater que le partage et la réutilisation des savoir-faire entre les enseignants/concepteurs en conception et en analyse d'une session sont importants. Le projet DPULS [Choquet (2005)] avait également ce but. Cependant, dans ce projet les patrons de conception ne pouvaient pas être calculés. Notre proposition répond bien à ce besoin. UTL permet de créer des patrons de données (c'est-à-dire des indicateurs dans la signification d'UTL), de spécifier la méthode de calcul pour acquérir la donnée. Ces patrons sont ensuite exécutés et capitalisés. Ils permettent de décrire le savoir-faire en observation d'une session de l'enseignant/concepteur (par exemple, en ce qui concerne un concept du scénario, quels indicateurs sont nécessaires), mais aussi la méthode de combinaison des données brutes et d'autres données produites en session pour l'élaboration d'un indicateur. Donc plusieurs indicateurs sont modélisés et capitalisés, plusieurs savoir-faire et expériences peuvent être partagés et réutilisés entre enseignants.

8 Conclusions et perspectives

Sommaire

Dans cette thèse, nous avons proposé un processus d'analyse de traces collectées en session d'apprentissage par les EIAH. Ce processus a pour but de faciliter la modélisation et le calcul d'indicateurs. Dans ce processus, le langage UTL est utilisé comme un langage de modélisation pour les indicateurs et le langage DCL4UTL (une extension d'UTL) est utilisé pour spécifier la méthode d'acquisition des valeurs de ces indicateurs. La nouvelle version d'UTL permet de capitaliser (1) le savoir-faire des enseignants dans la définition de leurs besoins d'observation, ainsi que (2) le savoir-faire des analystes dans la spécification des méthodes de calcul d'indicateur sous une forme d'un patron réutilisable et automatisable. Nous avons développé un outil d'analyse pour exécuter le langage DCLU4UL. Ce chapitre présente un bilan de notre travail, des résultats obtenus et aussi les limites de nos propositions. Nous concluons avec quelques perspectives en termes de travaux que nous envisageons de poursuivre.

8.1 Synthèse des travaux

Le travail présenté dans ce document s'inscrit dans le cadre du projet de recherche REDiM (Réingénierie des EIAH Dirigée par les Modèles) et porte précisément sur l'étape de modélisation, de collecte et d'analyse des traces du processus de réingénierie d'un scénario pédagogique. L'objectif du travail est de proposer et de valider un cadre d'analyse des traces générées par des EIAH. Dans ce cadre, nous considérons trois rôles participant au processus d'analyse. L'enseignant définit son scénario d'apprentissage, propose les situations et activités pédagogiques, présente des ressources nécessaires et définit des besoins d'observations. La modélisation d'un indicateur et sa méthode de calcul sont des tâches complexes qui nécessitent des connaissances techniques et informatiques. L'enseignant non informaticien pourra difficilement réaliser ces tâches. Nous prenons donc en compte le rôle de l'analyste dans le processus d'analyse des traces. Il est responsable de la modélisation des indicateurs. Le rôle du développeur informatique est également considéré dans ce processus. Il crée des sondes pour la mise en place du scénario d'observation ou les modifie à travers différentes étapes : la conception, l'analyse et la réingénierie. Il développe si nécessaire des fonctions externes.

Notre travail de recherche se base sur le méta-langage UTL. Il est utilisé comme un cadre théorique pour nos propositions. Ces dernières se focalisent sur la proposition d'un langage spécifique pour UTL (appelé DCL4UTL) et d'un outil d'analyse. Le langage DCL4UTL permet de combiner des données UTL pour produire une nouvelle donnée UTL (plus précisément un indicateur ou une donnée intermédiaire dans la signification d'UTL). Ce langage est associé à UTL pour modéliser et structurer des indicateurs (dans une forme proche d'un patron de conception) ainsi que pour spécifier une méthode de calcul des indicateurs à partir de données décrites en UTL. DCL4UTL permet également d'intégrer des opérateurs externes et de créer des données intermédiaires paramétrées. Ce langage s'adresse à l'analyste ou l'expert informatique pour l'aider à décrire les méthodes de calcul d'indicateurs de manière capitalisable et réutilisable, ce qui va permettre dans un deuxième temps à l'enseignant de réutiliser des indicateurs.

Nous avons développé un interpréteur permettant d'exécuter des indicateurs. Cet interpréteur peut calculer plusieurs types d'indicateurs (quantitatifs et qualitatifs) pendant ou après une session d'apprentissage. Tous les indicateurs et toutes les données produites dans le processus de calcul d'indicateurs sont capitalisés. Un environnement autour de DCL4UTL est développé pour faciliter l'activité de l'enseignant, mais UTL et DCL4UTL sont principalement conçus pour des analystes de données. Ils ne sont pas d'un accès évident

pour les enseignants. Un éditeur a donc été développé permettant à l'analyste et à l'enseignant de modéliser des indicateurs et leurs formules de calcul.

L'analyse de traces collectées par des EIAH utilisant UTL et DCL4UTL aide à produire des indicateurs sous la forme de patrons de données exécutables et capitalisables. Les spécifications de tous les indicateurs et de leurs instances sont également stockées et capitalisées. Ceci facilite la réutilisation et le partage des compétences des enseignants sur l'analyse d'une session d'apprentissage. Les résultats de ces indicateurs sont employés par des enseignants pour évaluer et analyser une session afin de modifier éventuellement leur scénario en vue de la session suivante (réingénierie) ou de proposer des solutions appropriées afin d'aider les étudiants à résoudre leurs problèmes plus rapidement (régulation), etc.

Nous avons conçu notre langage et notre outil grâce à plusieurs tests reposant sur les indicateurs du projet ICALTS, du système VAPS et d'une séance de travaux pratiques (TP) avec le laboratoire SysCom. Nous avons également validé nos propositions par une expérimentation dans le cadre de travaux pratiques utilisant l'environnement Hop3x qui permet d'apprendre la programmation orientée objet (par exemple en Java). Des indicateurs ont été calculés en temps réel à partir des traces de Hop3x pendant la session d'apprentissage.

Actuellement, notre outil continue à être utilisé avec les étudiants de troisième année au département informatique de l'Université du Maine afin d'aider l'enseignant/tuteur à observer et réguler l'activité de l'apprenant, mais aussi à réguler sa propre activité. Grâce aux indicateurs calculés par notre outil en temps réel, l'enseignant/tuteur peut prendre des décisions pédagogiques et donner des interventions appropriées. Il peut évaluer l'efficacité de ses décisions grâce à l'évolution des valeurs des indicateurs dans le temps [Lekira et al. (2011)].

8.2 Apports de la thèse

Les apports de notre travail concernent à la fois la modélisation et le calcul d'indicateurs. L'apport principal de la thèse est de fournir un cadre général de modélisation et de calcul d'indicateurs définis par l'enseignant/concepteur sous la forme de patrons de données exécutables et réutilisables. Ces patrons sont conçus pour pouvoir être exécutés par la machine et pour être compréhensibles par l'humain. Ils permettent de capitaliser la méthode d'acquisition des données et leurs instances. Le résultat des indicateurs peut être interprété par l'enseignant.

Concrètement, le premier apport de cette thèse concerne la proposition d'un langage de combinaison de données UTL appelé DCL4UTL. Ce langage est conçu pour formaliser la méthode de calcul d'indicateurs à partir de données UTL. En fait, UTL est employé comme un langage de modélisation pour décrire et définir des indicateurs et d'autres données servant à leurs calculs. Tandis que DCL4UTL est utilisé comme un langage de formalisation de la méthode de calcul des indicateurs à partir des données décrites en UTL. Cette proposition est une étape vers la définition des patrons de conception pour des indicateurs, dans laquelle le problème concerne l'obtention d'un indicateur à partir des données collectées en session et la solution correspond à la méthode de calcul associée à cet indicateur. Ce langage est générique et intégré dans UTL pour faciliter la génération et la réutilisation d'outils d'analyse dans différents contextes. Il a été conçu comme une extension d'UTL pour être dans la lignée d'UTL, indépendant du format de représentation des traces et du langage de scénarisation, mais aussi de l'architecture et du type de base de données. C'est un langage qui permet l'intégration d'opérateurs et de fonctions externes qui proviennent des outils d'analyse existants et/ou d'opérateurs et fonctions qui sont spécialement développés pour une application donnée. Il est donc possible de développer des APIs d'exécution comme celle que nous avons développée pour la base de données XML Exist. Cette intégration permet de réutiliser des fonctions existantes. C'est alors moins coûteux que de développer une nouvelle fonction. Elle permet aussi de réduire la complexité de la déclaration des fonctions trop complexes dans la méthode de calcul d'indicateurs. Ce langage permet également de créer des données intermédiaires paramétrées qui facilitent la modélisation et la réutilisation la méthode de calcul d'indicateurs.

Le second apport concerne la proposition d'un cadre qui s'appuie sur DCL4UTL pour l'exécution des indicateurs. Le langage que nous avons présenté nous a amenés à développer un outil d'analyse. Ce dernier est développé en prenant en compte les spécificités du langage DCL4UTL, c'est-à-dire qu'il intègre les opérateurs externes et exécute les données intermédiaires paramétrées. Cet outil permet d'ajouter, modifier, calculer et recalculer des indicateurs à différents moments (avant, pendant ou après une session) d'une session d'apprentissage sans modification de l'outil d'analyse.

8.3 Perspectives des travaux

Dans cette partie, nous présentons les limites de notre proposition, les travaux en cours et les perspectives à moyen terme et à long terme sur lesquelles nous envisageons de poursuivre.

8.3.1 Les limites

L'une des limites de la thèse tient dans la visualisation des indicateurs. UTL ainsi que l'outil d'analyse développé ne fournissent aucune méthode (ou description) ni outil qui permettent une représentation graphique des valeurs des indicateurs. Le résultat des indicateurs est actuellement affiché sous une forme textuelle. Ce format n'est sans doute pas satisfaisant par rapport aux attentes des enseignants/tuteurs. Par exemple, dans le cas où l'enseignant/tuteur veut faire une comparaison entre les activités réalisées par les apprenants/groupes d'apprenants, un affichage graphique s'avère plus significatif directement que du texte.

Une autre limite concerne la modélisation dynamique en session. Comme nous l'avons présenté, un indicateur peut être calculé à partir d'autres indicateurs. Les indicateurs et les autres données nécessaires pour la modélisation d'un indicateur doivent être déterminés lors de la modélisation avant la session. La modélisation automatique en session ne peut donc pas être réalisée pour deux raisons. La première est que le langage doit être utilisé par un expert informatique pour spécifier la méthode de calcul d'indicateurs. Notre outil ne peut pas générer automatiquement cette méthode, l'enseignant ou le tuteur non plus. La deuxième concerne le problème des sondes manquantes. Comme nous l'avons présenté, la modélisation des indicateurs peut nécessiter des sondes ou des fonctions externes. En pratique, ces dernières peuvent être trop complexes à déclarer. Elles ne peuvent donc pas être faites automatiquement en session.

8.3.2 Travaux en cours sur l'outil d'analyse

Comme nous l'avons présenté dans le chapitre 6, le composant de transformation des données est développé de manière *ad hoc*. De plus, l'appel des fonctions externes est réalisé localement. Les travaux en cours portent donc sur ces deux problèmes.

En ce qui concerne la génération d'un outil générique de transformation des données, la méthode est la suivante : utiliser la spécification de la méthode d'acquisition de la donnée brute UTL à partir de traces (par exemple, des requêtes SQL pour extraire des traces stockées dans une base de données, cf. l'élément *track*, l'annexe D) pour transformer des traces en données brutes UTL.

En ce qui concerne l'appel des fonctions externes, nous développerons des modules nécessaires permettant l'appel aux fonctions externes à distance via le service web ou un protocole de communication comme RPC (Remote Procedure Call) en se basant sur la spécification décrite dans la grammaire DCL4UTL (cf. l'annexe A). Nous prendrons en

compte l'appel aux fonctions externes liées aux arguments dont le type de donnée peut être élargi à autre chose qu'un type primitif, un tableau ou un objet. Cela nécessite les étapes suivantes :

- Elargir la grammaire DCL4UTL en ajoutant des règles de production permettant de décrire d'autres types de données, par exemple une liste.

- Améliorer l'interpréteur en prenant en compte de nouvelles règles de production.

8.3.3 Les perspectives

8.3.3.1 La visualisation des indicateurs

Comme nous l'avons présenté ci-dessus, notre système ne permet pas la visualisation graphique des valeurs des indicateurs. Une première perspective est donc liée à la représentation graphique des valeurs des indicateurs. Pour cela, nous pensons au regroupement des indicateurs en trois classes (la figure 8-1).

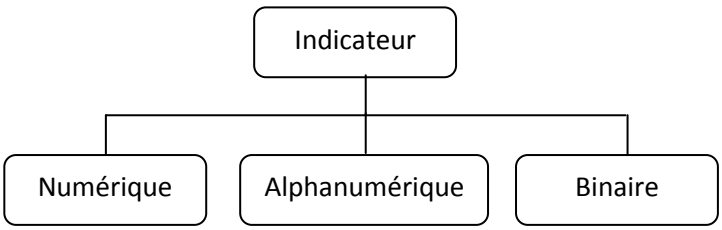

Figure 8-1 Le regroupement des indicateurs

La première contient des indicateurs qui donnent une valeur numérique. La deuxième se compose des indicateurs dont les valeurs sont binaires (par exemple, oui/non, vraie/fausse, etc.). La dernière comprend les indicateurs dont les valeurs sont textuelles (alphanumériques). Ce regroupement facilite la génération d'outil de visualisation.

En ce qui concerne la visualisation des indicateurs, les indicateurs alphanumériques sont affichés textuellement, seuls les indicateurs numériques et binaires seront représentés graphiquement. En fait, la visualisation d'un indicateur dépend de plusieurs facteurs (par exemple : le format de l'indicateur, l'élément à afficher, le nombre d'instances, etc.). En pratique, plusieurs travaux portent sur la visualisation. Dans la section 2.1, nous avons présenté différents travaux autour de la visualisation des informations. [France et al. (2006)] utilisent des figures de Chernoff, des liens et des bulles pour permettre au tuteur de voir en temps réel les activités des apprenants, les parcours effectués et les activités réalisées ou en cours de réalisation au sein de l'EIAH. [Heraud et al. (2005)] utilisent la barre d'ombre afin

d'aider l'enseignant à comprendre l'activité des apprenants grâce à la notion de clarté. [May (2008)] utilise des sphères avec différentes couleurs pour visualiser en temps réel la lecture d'un message par personne ainsi que le temps passé pour la lecture d'un message dans un forum de discussion. [Mazza et Dimitrova (2003), Mazza et Milani (2005)] utilisent différentes techniques de visualisation pour aider l'enseignant/tuteur à obtenir une compréhension des aspects sociaux, cognitifs et comportementaux des apprenants dans une session d'apprentissage à distance.

Une approche intéressante consiste à proposer plusieurs moyens de visualisation à l'enseignant/tuteur afin qu'il choisisse la solution qui lui convient le mieux. Nous développons la suite autour de quelques exemples simples de visualisation. Dans cette optique, un diagramme à bandes vertical, un diagramme à lignes brisées ou un diagramme circulaire pourront éventuellement être utilisés pour représenter les valeurs de ces indicateurs.

Pour la génération d'outil de visualisation, nous prenons en compte plusieurs types de visualisation pour chaque indicateur. Une solution possible est d'élargir UTL en ajoutant dans l'indicateur un champ ''*visualisations*'' qui se compose des éléments suivants :

```
<visualisations>
      <visualisation>
            <typeDiagramme>string</typeDiagramme>
            <typeIndicateur>string</typeIndicateur>
            <valeur>string</valeur>
            <identification>string</identification>
            <temps>string</temps>
      </visualisation>
</visualisations>
```

Tableau 8-1 Le champ de *visualisation*

- Type de diagramme : ce type peut être un diagramme à bandes verticales (BV), un diagramme à lignes brisées (LB) ou un diagramme circulaire (C).
- Type d'indicateur : ce type appartient à une de trois classes ci-dessus, c'est-à-dire numérique, binaire ou alphanumérique.
- La valeur à afficher : cette valeur correspond à un élément/attribut de l'élément *using.data*. Il s'agit de la valeur de l'indicateur, par exemple le nombre de messages envoyés, le nombre d'actions réalisées, etc.
- L'identification des valeurs : cette valeur est également un élément/attribut de l'élément *using.data*. Il s'agit de l'identifiant concernant la valeur à afficher de l'indicateur, par exemple l'apprenant, l'action, l'activité.

- Le temps : cette valeur est le moment de calcul de l'indicateur si ce dernier est calculé en temps réel. Cet élément est aussi un élément/attribut de l'élément *using.data*.

Exemple 1 : nous reprenons le format de l'indicateur "*La visibilité de la classe **Point** doit être **public***" dans le chapitre 7 :

```
<indicator type="PEDALO-I-ClassPointPublic">
      <result     session="String" student="String"
                  time="String" event="String"
                  value="String" status="String"
                  calculable="String" reference="String"
                  location="String" currentQuestion="String"
                  concernedQuestion="String" category="String"/>
</indicator>
```

Tableau 8-2 Le format de l'indicateur "*La visibilité de la classe **Point** doit être **public***"

La spécification pour la visualisation de cet indicateur est décrite ci-dessous :

```
<visualisations>
   <visualisation>
      <typeDiagramme>LB</typeDiagramme>
      <typeIndicateur>binaire</typeIndicateur>
      <valeur>using.data.indicator.result[value]</valeur>
      <identification>using.data.indicator.result[student]
                                                </identification>
      <temps>using.data.indicator.result[time]</temps>
   </visualisation>
</visualisations>
```

Tableau 8-3 Le champ "visualisation" de l'indicateur "*La visibilité de la classe **Point** doit être **public***"

- Type de diagramme : par exemple un diagramme à lignes brisées (LB).
- Type d'indicateur : binaire.
- La valeur à afficher : using.data.indicator.result[value].
- L'identification des valeurs : using.data.indicator.result[student].
- Le temps: using.data.indicator.result[time].

Exemple 2 : nous reprenons l'indicateur "*le nombre de bonnes actions par patient*" dans la section 6.8. Le format de cet indicateur est décrit dans le tableau 8-4.

```
<indicator type="VAPS-I-BA">
     <nombreBonnesActions patient="string"> integer
     </nombreBonnesActions>
</indicator>
```

Tableau 8-4 Le format de l'indicateur ''*le nombre de bonnes actions par patient''*

La spécification pour la visualisation de cet indicateur est la suivante :

```
<visualisations>
  <visualisation>
     <typeDiagramme>BV</typeDiagramme>
     <typeIndicateur>numérique</typeIndicateur>
     <valeur>using.data.indicator.nombreBonnesActions</valeur>
        <identification>using.data.indicator.
                nombreBonnesActions[patient]</identification>
     <temps/>
  </visualisation>
</visualisations>
```

Tableau 8-5 Le champ ''visualisation'' de l'indicateur ''*le nombre de bonnes d'actions par patient''*

- Type de diagramme : par exemple un diagramme à bandes verticales (BV).
- Type d'indicateur : numérique.
- La valeur à afficher : using.data.indicator.nombreBonnesActions.
- L'identification des valeurs : using.data.indicator.nombreBonnesActions[patient].

Ce champ ''*visualisations*'' se compose des informations nécessaires permettant de générer la visualisation graphique des valeurs des indicateurs. Nous souhaitons donc étudier la création des diagrammes en utilisant cette spécification (champ ''*visualisations*'') et des techniques de visualisation.

8.3.3.2 La capitalisation des connaissances d'un problème à résoudre

Actuellement, chaque indicateur UTL permet de capitaliser le besoin d'observation de l'enseignant, la méthode d'acquisition de la donnée et ses données. Dans le contexte des actions tutorales (cf. chapitre 7), en se basant sur les valeurs des indicateurs, l'enseignant/tuteur peut détecter des problèmes, des erreurs, des incompréhensions, etc. de l'apprenant et réalise donc des interventions pour proposer des solutions. Nous pensons que ces connaissances (erreurs, solutions, etc.) sont importantes et nécessaires pour l'enseignant/tuteur et l'apprenant. En ce qui concerne l'enseignant/tuteur notamment le

tuteur novice, il peut réutiliser ces connaissances pour résoudre une situation similaire ou pour prendre facilement une décision pédagogique (par exemple, le projet TeTraKap [Quénu-Joiron et Condamines (2009)], cf. section 2.3.5). Pour l'apprenant, il peut consulter des erreurs communes et donc éviter des problèmes généraux ou il peut auto-corriger ses erreurs en utilisant les solutions préconisées par l'enseignant/tuteur. Cette idée est proche du travail de [Hadj M'tir et al. (2008)] (cf. section 2.3.5) qui s'intéressent à la capitalisation et à la réutilisation des expériences d'apprentissage des apprenants. Actuellement, ces connaissances ne sont pas capitalisées dans UTL.

Nous souhaitons à moyen terme travailler sur la capitalisation et la réutilisation de ces connaissances pour l'enseignant/tuteur et l'apprenant. Cette capitalisation et cette réutilisation consistent en un certain nombre d'étapes :

- Une extension d'UTL peut être réalisée en ajoutant des champs (pour les problèmes et pour les solutions) afin de permettre la capitalisation de ces connaissances.
- Extraire et organiser ces connaissances pour les stocker dans ces champs.
- Développer les outils nécessaires à la mise en œuvre de la réutilisation de ces connaissances chez l'enseignant et l'apprenant au travers de l'outil d'analyse.

Notre objectif est donc faire évoluer l'outil d'analyse en facilitant la réutilisation du savoir-faire des enseignants ainsi que des apprenants.

8.3.3.3 L'expérience utilisateur

Actuellement, l'outil d'analyse que nous avons développé permet de calculer des indicateurs (pendant ou après une session). Nous avons réalisé une expérimentation pour valider nos propositions. Cependant, nous ne faisons pas encore d'expérimentations dans la communauté des enseignants pour ouvrir une enquête recueillant la perception des enseignants sur notre système. Nous souhaitons donc à plus long terme travailler sur l'expérience utilisateur pour améliorer notre outil d'analyse afin de satisfaire au mieux les besoins des utilisateurs notamment les enseignants/tuteurs.

Le concept d'expérience utilisateur a été largement diffusé et rapidement accepté dans la communauté de l'interaction homme-machine [Law et al. (2009)]. Plusieurs définitions sur l'expérience utilisateur sont publiées. ISO 9241-210 [ISO 9241-210:2010 (2010)] définit l'expérience utilisateur comme ‘‘*les perceptions et les réponses d'une personne qui résultent*

de l'utilisation ou de l'utilisation prévue d'un produit, d'un système ou d'un service''[19]. [Hassenzahl et Tractinsky (2006)] considèrent l'expérience utilisateur comme ''*une conséquence de l'état interne d'un utilisateur (des prédispositions, des attentes, des besoins, des motivations, un humeur, etc.), les caractéristiques du système conçu (par exemple la complexité du dispositif, les buts des tâches, l'utilisabilité, la fonctionnalité, etc.) et le contexte ou l'environnement dans lequel l'interaction se produit (par exemple l'établissement organisationnel/social, le sens de l'activité, l'intégration dans les pratiques sociales et culturelles, etc.)*'' [20]. Selon la plupart des chercheurs, l'expérience de l'utilisateur est une construction dynamique, subjective, dépendante du contexte et produite dans les interactions humaines, technologiques et sociales [Law et al. (2009)].

Travailler sur l'expérience utilisateur consiste donc en un certain nombre d'étapes :

- Etudier et proposer une métrique pour évaluer l'expérience utilisateur sur notre outil. Cette métrique est construite selon les attentes, les besoins des utilisateurs, l'utilisabilité, la fonctionnalité de l'outil, etc. Par exemple :
    - L'outil permet à des enseignants/tuteurs d'atteindre le résultat prévu ? A quel niveau ? (par exemple, bien, satisfaisant, insatisfaisant, etc.)
    - Les enseignants/tuteurs peuvent-ils facilement et rapidement accomplir des tâches prévues ? A quelle vitesse ?
    - La vitesse de calcul d'indicateurs satisfait des attentes des enseignants/tuteurs ?
    - Est-ce que le calcul d'indicateurs aide les enseignants/tuteurs ?
    - Quelles sont les fonctionnalités nécessaires, inutiles, à ajouter, etc. ?
    - Etc.
- Faire des expérimentations dans la communauté des enseignants/tuteurs pour leur permettre d'utiliser notre système et pour recueillir leurs évaluations, leurs points de vue en utilisant la métrique définie. Cette évaluation peut ensuite être réalisée en ligne ou hors ligne.

[19] En anglais *''A person's perceptions and responses that result from the use or anticipated use of a product, system or service''*

[20] En anglais ''*a consequence of a user's internal state (predispositions, expectations, needs, motivation, mood, etc.), the characteristics of the designed system (e.g. complexity, purpose, usability, functionality, etc.) and the context (or the environment) within which the interaction occurs (e.g. organisational/social setting, meaningfulness of the activity, voluntariness of use, etc.).''*

- Améliorer notre système selon les évaluations des utilisateurs (enseignants/tuteurs) en se basant sur les critères définis ou les propositions appropriées des utilisateurs. Notre objectif est de satisfaire des besoins, des attentes de la communauté des enseignants/tuteurs dans l'analyse, l'observation, l'évaluation et régulation d'une session.

Pour le moment, notre outil est un prototype qui nécessite des évolutions de façon à prendre en compte des expérimentations à plus grande échelle ou pour travailler sur l'expérience utilisateur. Actuellement, le SGBD (système de gestion de base de données) XML (eXist-db) que nous utilisons pour stocker les données UTL ne permet pas des accès parallèles aux données. Si la base de données est grande, la vitesse de traitement des requêtes de ce SGBD devient lourde. Le calcul d'indicateurs prend donc du temps. Dans un premier temps, nous avons décidé de choisir ce SGBD parce qu'il s'agit d'un logiciel libre et facile à utiliser. L'amélioration porte donc sur la vitesse de calcul. Pour cela, nous nous intéressons à l'amélioration des index, des requêtes, ainsi qu'au choix d'un système de gestion de base de données XML plus approprié. Cette amélioration doit aussi prendre en compte les travaux en cours décrits dans la section 8.3.2 et les perspectives concernant la visualisation et la capitalisation des connaissances d'un problème à résoudre.

Abréviations

Les abréviations présentées ci-dessous sont celles qui sont utilisées dans cette thèse.

AD	Additional Datum
API	Application Programming Interface
ATL	ATLAS Transformation Language
BNF	Backus-Naur Form
CA	Collaborative Action function indicator
CAViCoLA	Computer based Analysis and Visualization of Collaborative Learning Activities
CD	Content Datum
CIM	Common Information Model
ColAT	Collaboration Analysis Toolkit
CLF	Common Log Format
CLIPS	C Language Integrated Production System
CQL	CIM Query Language
DCL4UTL	Data Combination Language for UTL
DD	Derived Datum
DGU	Defining-Getting-Using
DIAS	Discussion Interaction Analysis System
DPULS	Design Patterns for collecting and analysing Usage of Learning Systems
DTD	Document Type Definition
DUT	Diplôme Universitaire de Technologie
ECA	Entreprise Collaboration Architecture
ELHIT	Equipe Lavalloise Handicaps et Innovations Technologiques
EIAH	Environnements Informatiques pour l'Apprentissage Humain
EM-AGIIR	Environnement Multi-Agent de supervIsion à base d'Indicateurs Réutilisés.
FOAD	Formation Ouverte et À Distance
I	Indicator
IA	Interaction Analysis

ICALTS	Interaction & Collaboration Analysis supporting Teachers & Students' Self-regulation
ID	Intermediate Datum
IMS-LD	IMS Learning Design
ITS	Intelligent Tutoring Systems
JavaCC	Java Compiler Compiler
LDL	Learning Design Language
LINA	Laboratoire Informatique de Nantes Atlantique
LIRIS	Laboratoire d'InfoRmatique en Image et Systèmes d'information
LISTEN	Literacy Innovation that Speech Technology Enables
LIUM	Laboratoire d'Informatique de l'Université du Maine
LMS	Learning Management Systems
LTEE	Learning Technology and Educational Engineering
OMG	Object Management Group
MDA	Model Driven Architecture
PD	Primary Datum
PID	Parameterized Intermediate Data
PIR	Patron d'Indicateur Réutilisable
PLAIT	Pattern Language for Architectures of Intelligent Tutors
PPP	Pedagogical Patterns Project
RD	Raw Datum
REDiM	Réingénierie des EIAH Dirigée par les Modèles
RPC	Remote Procedure Call
SBT	Système à Base de Traces
SeRéCom	Services et Réseaux de Communication
SGBD	Système de Gestion de Base de Données
SRC	Services et Réseaux de Communication
SI	Systèmes d'Informations
SPARQL	Query Language for RDF
SQL	Structured Query Language
SysCom	Laboratoire Systèmes Communicants de l'Université de Savoie
TADA-Ed	Tool for Advanced Data Analysis in Education
TeTraKap	TEacher TRAIning by Knowledge cAPitalization
TP	Travaux Pratiques
UML	Unified Modeling Language
UTL	Usage Tracking Laguage

VAPS	Virtual Action Planning Supermarket
W3C	World Wide Web Consortium
WBEM	Web Based Entreprise Management
XML	Extensible Markup Language
Xpath	XML Path Language
XSLT	Extensible Stylesheet Language Transformations
Xquery	XML Query Language

Table des figures

Table des tableaux

Références bibliographiques

[Alcalá-Fdez et al. (2008)] J. Alcalá-Fdez, L. Sánchez, S. García, M. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, Rivas, J. C. Fernández, et F. Herrera. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Comput.*, 13: 307–318, October 2008. ISSN 1432-7643.

[Alexander et al. (1977)] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, et S. Angel. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1977.

[Anjewierden et al. (2007)] A. Anjewierden, B. Kollöffel, et C. Hulshof. Towards educational data mining: Using data mining methods for automated chat analysis and support inquiry learning processes. In *The International Workshop on Applying Data Mining in e-Learning (ADML 2007)*, Crete, Greece, 2007.

[Avgeriou et al. (2003)] P. Avgeriou, A. Papasalouros, S. Retalis, et M. Skordalakis. Towards a Pattern Language for Learning Management Systems. *Educational Technology & Society*, 6: 11–24, 2003. ISSN 1436-4522.

[Avouris et al. (2003)] N. Avouris, A. Dimitracopoulou, V.Komis, et M. Margaritis. Participatory analysis of synchronous collaborative problem solving using the OCAF methodology and tools. In *CSCL2003 Community events*, pages 232–234, 2003.

[Avouris et al. (2005)] N. Avouris, V.Komis, G.Fiotakis, M. Margaritis, et E. Voyiatzaki. Logging of fingertip actions is not enough for analysis of learning activities. In *The AIED05 Workshop on Usage Analysis in Learning Systems*, Amsterdam, Pays-bas, 2005.

[Beitzel et al. (2004)] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, et O. Frieder. Hourly analysis of a very large topically categorized web query log. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 321–328, New York, NY, USA, 2004. ACM. ISBN 1-58113-881-4.

[Bergin (2000)] J. Bergin. Fourteen Pedagogical Patterns. Accessible in http://csis.pace.edu/ bergin/PedPat1.3.html (Consulté en mars 2011), 2000.

[Berry et Linoff (1997)] M. J. Berry et G. S. Linoff. *Data Mining Techniques: For Marketing, Sales, and Customer Support*. John Wiley & Sons Inc, 1997.

[Bissay et al. (2008)] A. Bissay, A. Lefebvre, P. Pernelle, et A. Bouras. Approche de capitalisation des connaissances au sein des systèmes PLM. In *1er Colloque International sur les Systèmes Industriels et Logistiques (SIL'08)*, 2008.

[Bousbia et Labat (2007)] N. Bousbia et J.-M. Labat. Perception de l'activité de l'apprenant dans un environnement de formation Sémantique du parcours de l'apprenant. In *Actes des Environnements Informatiques pour l'Apprentissage Humain (EIAH2007)*, pages 233–238, Lausanne, Suisse, juin 2007.

[Bousbia et al. (2009a)] N. Bousbia, J.-M. Labat, I. Rebai, et A. Balla. How to determine the learners' learning styles in e-learning situation? In *The 16th International Conference on Computers in Education, ICCE'08*, pages 185–186, Taipei, Taiwan, 2009.

[Bousbia et al. (2009b)] N. Bousbia, J.-M. Labat, I. Rebai, et A. Balla. Indicators for deducting the learners' learning styles: Case of the navigation typology indicator. In *The 9th IEEE International Conference on Advanced Learning Technologies (ICALT2009)*, Riga, Latvia, 2009.

[Bratitsis et Dimitracopoulou (2005)] T. Bratitsis et A. Dimitracopoulou. Data recording and usage interaction analysis in asynchronous discussions: The D.I.A.S. system. In *The AIED05 Workshop on Usage Analysis in Learning Systems*, Amsterdam, Pays-bas, 2005.

[Bratitsis et Dimitracopoulou (2008)] T. Bratitsis et A. Dimitracopoulou. Interpretation Issues in Monitoring and Analyzing Group Interactions in Asynchronous Discussions. *International Journal of e-Collaboration*, Vol. 4(1) : 20–40, 2008.

[Broisin et Vidal (2007)] J. Broisin et P. Vidal. Une approche conduite par les modèles pour le traçage des activités. *Revue STICEF numéro spécial : Analyses des traces d'utilisation dans les EIAH*, Vol. 14, 2007.

[Buschmann et al. (1996)] F. Buschmann, R. M. H. Rohnert, et P. Sommerlad. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. John Wiley & Sons, 1996.

[Després et Coffinet (2004)] C. Després et T. Coffinet. Reflet, un miroir sur la formation. In *Actes de TICE 2004*, pages 19–24, Compiègne (France), 20-22 octobre 20 2004.

[Carron et al. (2006)] T. Carron, J.-C. Marty, J.-M. Heraud, et L. France. Helping the teacher to re-organize tasks in a collaborative learning activity: An agent-based approach. In *Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies*, ICALT '06, pages 552–554. IEEE Computer Society, 2006. ISBN 0-7695-2632-2.

[Castro et al. (2007)] F. Castro, A. Vellido, A. Nebot, et F. Mugica. *Applying Data Mining Techniques to e-Learning Problems*, pages 183–221. Springer, 2007.

[CAViCoLA (2006)] CAViCoLA. Computer-based analysis and visualization of collaborative learning activities. Accessible in: http://www.noe-kaleidoscope.org, 2006.

[Champin et al. (2003)] P.-A. Champin, Y. Prié, et A. Mille. MUSETTE: Modeling uses and tasks for tracing experience. In *Proceedings of the workshop From Structured Cases to Unstructured Problem Solving Episodes For Experience-Based Assistance, ICCBR'03*, 2003.

[Chan et al. (2004)] A. Y. Chan, P. K. Chow, et K. S. Cheung. Student participation index: Student assessment in online courses. In *Proceedings of the third International Conference on Advances in Web-Based Learning on Technology Enhanced Learning (ICWL2004). Lecture Notes in Computer Science. Springer*, volume 3143/2004, pages 103–118, 2004.

[Chau et al. (2005)] M. Chau, X. Fang, et O. R. L. Sheng. Analysis of the Query Logs of a Web Site Search Engine. *Journal of the American Society for Information Science and Technology*, 56(13): 1363-1376, 2005.

[Chikofsky et Cross (1990)] J.-E. Chikofsky et J.-H. Cross. Reverse engineering and design recovery: A taxonomy. *Software IEEE*, 7: 13–17, 1990.

[Choquet (2005)] C. Choquet. DPULS Project - Design Patterns for recording and analysing Usage of Learning Systems. Technical report, Action of The European Network of Excellence Kaleidoscope, Site Accessible à: http://www.noe-kaleidoscope.org, 2005.

[Choquet (2007)] C. Choquet. *Ingénierie et réingénierie des EIAH: l'approche REDiM*. Habilitation à diriger des recherches en informatique, Université du Maine, 2007.

[Choquet et Iksal (2007a)] C. Choquet et S. Iksal. Modeling Tracks for the Model Driven Reengineering of a TEL System. *The Journal of Interactive Learning Research (JILR)*, Vol. 18(2): 161–184, 2007.

[Choquet et Iksal (2007b)] C. Choquet et S. Iksal. Modélisation et construction de traces d'utilisation d'une activité d'apprentissage : une approche langage pour la réingénierie d'un eiah. *Revue STICEF*, Vol. 14, 2007.

[Corbière (2006)] A. Corbière. *Analyses des apports du méta-standard, ODP-RM à la communauté EIAH, Instances sur un système de formation*. Doctorat, Université du Maine, 2006.

[Daele et al. (2002)] A. Daele, C. Drassard, L. Esnault, M. O'donoghue, E. Uyttebrouck, et R. Zeiliger. Conception, mise en œuvre, analyse et évaluation de scénarios pédagogiques recourant à l'usage des Technologies de l'Information et de la Communication. Technical report, Réseau des Centres de Ressources pour l'Enseignement Supérieur, Tecfa, Suisse, 2002.

[De Ketele et Roegiers (2009)] J.-M. De Ketele et X. Roegiers. *Méthodologie du recueil d'informations, fondements des méthodes d'observation de questionnaires d'interviews et d'étude de documents*. 2009.

[Delozanne et al. (2005)] E. Delozanne, J.-M. Labat, F. L. Calvez, et A. Merceron. DPULS-D32.6.1, a Structured Set of Design Patterns for the Usage Analysis. Technical report, Action of The European Network of Excellence Kaleidoscope, Accessible in: http://www.noe-kaleidoscope.org, 2005.

[Delozanne et al. (2007)] E. Delozanne, F. L. Calvez, J.-M. Labat, et A. Merceron. Design patterns en EIAH : vers un langage de patterns pour l'évaluation des apprenants. *Revue STICEF*, Vol. 14, 2007.

[Devedzic et Harrer (2005)] V. Devedzic et A. Harrer. Software Patterns in ITS Architectures. *International Journal of Artificial Intelligence in Education (IJAIED)*, 15(2) (2): 63–94, 2005.

[Diagne (2008)] F. Diagne. EM-AGIIR : un Environnement Multi-AGent ouvert pour la supervision à partir d'Indicateurs Réutilisés. In *Actes des Rencontres Jeunes Chercheurs en EIAH (RJC2008)*, pages 65–70, Lille, France, 2008.

[Diagne (2009)]F. Diagne. *Instrumentation de la supervision de l'apprentissage par la réutilisation d'indicateurs : Modèles et Architecture*. Doctorat Université Joseph Fourier, 2009.

[Dimitracopoulou (2004a)] A. Dimitracopoulou. D26.1.1: State of the art on interaction and collaboration analysis. Technical report, Kaleidoscope Deliverables, 2004.

[Dimitracopoulou (2004b)] A. Dimitracopoulou. D26.1.1 state of the art on interaction and collaboration analysis. Technical report, Action of The European Network of Excellence Kaleidoscope, Accessible in: http://www.noe-kaleidoscope.org, 2004.

[Dimitracopoulou (2005)] A. Dimitracopoulou. D31.1.1 State of the art of interaction analysis for Metacognitive Support & Diagnosis. Technical report, Action of The European Network of Excellence Kaleidoscope, Accessible in: http://www.noe-kaleidoscope.org, 2005.

[Djouad (2008)] T. Djouad. Analyser l'activité d'apprentissage collaboratif : Une approche par transformations spécialisées de traces d'interactions. In *Actes des Rencontres Jeunes Chercheurs en EIAH (RJC2008)*, pages 93–98, Lille, France, 2008.

[Djouad et al. (2009)] T. Djouad, A. Mille, C. Reffay, et M.Benmohamed. Ingénierie des indicateurs d'activités à partir de traces modélisées pour un environnement informatique d'apprentissage humain. *Revue STICEF*, Vol. 16, 2009.

[DMTF (2005)] DMTF. Common Information Model (CIM) Infrastructure Specification. Technical report, Distributed Management Task Force, Inc., http://www.dmtf.org/sites/default/files/standards/documents/DSP0004V2.3_final.pdf, 2005.

[Dyke et al. (2010)] G. Dyke, K. Lund, et J. Girardot. Tatiana, un environnement d'aide à l'analyse de traces d'interactions humaines. *Revue des sciences et technologies de l'information*, Vol. 4(1): 1179–1205, 2010.

[E-LEN (2003)] E-LEN. E-LEN: A network of e-learning centres. Accessible in http://www2.tisip.no/E-LEN/patterns_info.php (Consulté en mars 2011), 2003.

[Eckstein (2000)] J. Eckstein. Learning to Teach and Learning to Learn Running a Course. Accessible in http://www.pedagogicalpatterns.org/examples/LearningAndTeaching.pdf (Consulté en mars 2011), 2000.

[El-Kechaï (2008)] H. El-Kechaï. *Conception collective de scénarios pédagogiques dans un contexte de réingénierie. Une approche par la méta-modélisation située*. Doctorat, Université du Maine, 2008.

[Etzioni (1996)] O. Etzioni. The world-wide web: quagmire or gold mine? *Commun. ACM*, 39: 65–68, November 1996. ISSN 0001-0782.

[Ferraris et al. (2007)]C. Ferraris, C. Martel, et L. Vignollet. *Handbook of visual languages for instructional design: Theories and practices.*, chapter LDL for collaborative activities, pages 226–253. Hershey, PA : IDEA Group, 2007.

[France et al. (2006)] L. France, J.-M. Heraud, J.-C. Marty, T. Carron, et J. Heili. Monitoring virtual classroom: Visualization techniques to observe student activities in an e-learning system. In *The Sixth IEEE International Conference on Advanced Learning Technologies (ICALT2006)*, pages 716–720, 2006.

[Fricke et Voelter (2000)] A. Fricke et M. Voelter. SEMINARS - A Pedagogical Pattern Language about teaching seminars effectively. Accessible in http://www.voelter.de/data/pub/tp/html/index.htm (Consulté en mars 2011), 2000.

[Gamma et al. (1996)] E. Gamma, R. Helm, R. Johnson, et J. Vlissides. *Design Patterns : Catalogue de modèles de conception réutilisables*. International Thomson Publishing France, 1996.

[Gaudioso et Talavera (2006)] E. Gaudioso et L. Talavera. Data mining to support tutoring in virtual learning communities: Experiences and challenges. *Data mining in e-learning*, pages 207–226, 2006.

[Gendron (2010)] E. Gendron. *Cadre conceptuel pour l'élaboration d'indicateurs de collaboration à partir des traces d'activité*. Doctorat université de Claude Bernard Lyon 1, 2010.

[Guéraud et al. (2004)] V. Guéraud, J.-M. Adam, J.-P. Pernin, G. Calvary, et J.-P. David. L'exploitation d'Objets Pédagogiques Interactifs à distance : le projet FORMID. *Revue STICEF*, Vol. 11, 2004.

[Hadj M'tir et al. (2008)] R. Hadj M'tir, L. Jeribi, et B. Rumpler. Learners Experiences Reuse to Improve Personalized E-Learning. In *IEEE ICTTA'08*, Apr. 2008.

[Hamalainen et al. (2004)] W. Hamalainen, J. Suhonen, E. Sutinen, et H. Toivonen. Data mining in personalizing distance education course. In *World conference on open learning and distance education*, Hong Kong, 2004.

[Hassenzahl et Tractinsky (2006)] M. Hassenzahl et N. Tractinsky. User experience – a research agenda. *Behaviour & Information Technology*, 25 (2): 91–97, 2006.

[Heiner et al. (2004)] C. Heiner, J. Beck, et J. Mostow. Lessons on using its data to answer educational research questions. In *Workshop Proceedings of ITS-2004*, pages 1–9, 2004.

[Heraud et al. (2005)] J.-M. Heraud, J.-C. Marty, L. France, et T. Carron. Helping the interpretation of web logs: Application to learning scenario improvement. In *The AIED05 Workshop on Usage Analysis in Learning Systems*, Amsterdam, Pays-bas, 2005.

[Herlocker et al. (2004)] J. L. Herlocker, J. A. Konstan, L. G. Terveen, et J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22: 5–53, January 2004. ISSN 1046-8188.

[Hijon et Velazquez (2006)] R. Hijon et A. Velazquez. E-learning platforms analysis and development of students tracking functionality. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2006*, 2006.

[Hohpe et Woolf (2003)] G. Hohpe et B. Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. AddisonWesley, 2003.

[Iksal et al. (2004)] S. Iksal, C. C. V. Barré, et A. Corbière. Comparing prescribed and observed for the re-engineering of e-learning systems. In *Proceedings of the IEEE Sixth International Symposium on Multimedia Software Engineering (ISMSE'04)*, Kerkrade, The Netherlands, 2004.

[Iksal et al. (2007)] S. Iksal, T. Carron, et J.-C. Marty. Modéliser les expérimentations basées sur les traces : Une étude de cas avec le langage UTL. Technical report, Université du Maine, 2007.

[Iksal et al. (2010)] S. Iksal, C. Choquet, et N.-D. Pham-Thi. A generic modeling of indicator with UTL: The collaborative action function example. In *The 2nd International Conference on Computer Supported Education (CSEDU2010)*, pages 114–119, Valencia, Spain, 2010.

[IMS-LD (2003)] IMS-LD. *IMS Learning Design v1.0 Final Specification*. IMS Global Learning Consortium, Accessible in http://www.imsglobal.org/learningdesign/index.html, 2003.

[ISO 9241-210:2010 (2010)] ISO 9241-210:2010. *Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems*. International Organization for Standardization (ISO), Switzerland, 2010.

[Jansen et Pooch (2001)] B. J. Jansen et U. Pooch. A review of web searching studies and a framework for future research. *J. Am. Soc. Inf. Sci. Technol.*, 52: 235–246, February 2001. ISSN 1532-2882.

[Jansen et al. (1998)] B. J. Jansen, A. Spink, J. Bateman, et T. Saracevic. Real life information retrieval: a study of user queries on the web. *SIGIR Forum*, 32: 5–17, April 1998. ISSN 0163-5840.

[Jansen et al. (2009)] B. J. Jansen, I. Taksa, et A. Spink. *Handbook of Research on Web Log Analysis*, chapter Research and Methodological Foundations of Transaction Log Analysis, pages 226–253. IGI Global, 2009.

[Mostow et Beck (2006)] J. Mostow et J. Beck. Some useful tactics to modify, map and mine data from intelligent tutors. *Natural Language Engineering*, 12 (2): 195–208, 2006. ISSN 1351-3249.

[Jones et al. (1998)] S. Jones, S. J. Cunningham, et R. McNab. Usage analysis of a digital library. In *Proceedings of the third ACM conference on Digital libraries*, DL '98, pages 293–294, New York, NY, USA, 1998. ACM. ISBN 0-89791-965-3.

[Jones et al. (2000)] S. Jones, S. J. Cunningham, R. Mcnab, et S. Boddie. A transaction log analysis of a digital library. *International Journal on Digital Libraries*, 3: 152–169, 2000.

[Juan et al. (2009)] A. Juan, T. Daradoumis, J. Faulin, et F. Xhafa. A data analysis model based on control charts to monitor online learning processes. *Journal International Journal of Business Intelligence and Data Mining*, 4: 159–174, July 2009. ISSN 1743-8195.

[Klinger (2006)] E. Klinger. *Apports de la réalité virtuelle à la prise en charge de troubles cognitifs et comportementaux*. Doctorat ENST Paris, 2006.

[Law et al. (2009)] E. L.-C. Law, V. Roto, M. Hassenzahl, A. P. Vermeeren, et J. Kort. Understanding, scoping and defining user experience: a survey approach. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 719–728, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7.

[Lekira (2010)] A. Lekira. Les indicateurs à la base de l'aide à l'auto-régulation tutorale. In *Actes des Rencontres Jeunes Chercheurs en EIAH (RJC2010)*, Lyon, France, 2010.

[Lekira et al. (2011)] A. Lekira, C.Després, P. Jacoboni, C. Choquet, S. Iksal, D. Py, et N.-D. Pham-Thi. Using indicators during synchronous tutoring of practical work. In *The 11th IEEE*

International Conference on Advanced Learning Technologies (ICALT2011), Athens - Gerogia(USA), 2011.

[LISTEN (2011)] LISTEN. LISTEN project. Accessible in http://www.cs.cmu.edu/ listen/(Consulté en mars 2011), 2011.

[May (2008)] M. May. Une approche pour tracer finement les communications médiatisées en situation d'apprentissage. In *Actes des Rencontres Jeunes Chercheurs en EIAH (RJC2008)*, pages 71–76, Lille, France, 2008.

[May (2009)] M. May. *Utilisation des traces comme outils réflexifs pour les apprenants et les enseignants à distance : Application aux communications médiatisées*. Doctorat INSA Lyon, 2009.

[Mazza et Dimitrova (2003)] R. Mazza et V. Dimitrova. Coursevis: Externalising student information to facilitate instructors in distance learning. In *The International conference in Artificial Intelligence in Education*, Sydney, 2003.

[Mazza et Milani (2005)] R. Mazza et C. Milani. Exploring usage analysis in learning systems: Gaining insights from visualisations. In *Workshop on usage analysis in learning systems at 12th international conference on artificial intelligence in education*, 2005.

[Meister et Sullivan (1967)] D. Meister et D. Sullivan. Evaluation of user reactions to a prototype on-line information retrieval system. Technical report, Bunker-Ramo Corporation, Canoga Park, CA. Systems Effect Lab., 1967.

[Merceron (2009)] A. Merceron. *Personnalisation des Environnements Informatiques pour l'Apprentissage Humain*, chapter Fouiller les traces d'activité d'apprentissage, pages 96–113. Hermès, 2009.

[Merceron et Yacef (2003)] A. Merceron et K. Yacef. A web-based tutoring tool with mining facilities to improve learning and teaching. In *The International conference in Artificial Intelligence in Education*, Sydney, 2003.

[Merceron et Yacef (2004a)] A. Merceron et K. Yacef. Mining student data captured from a web-based tutoring tool: Initial exploration and results. *Journal of Interactive Learning Research, Special Issue on Computational Intelligence in Web-Based Education*, Vol. 15: 319–346, 2004.

[Merceron et Yacef (2004b)] A. Merceron et K. Yacef. Train, store, analyse for more adaptative teaching. In *Colloque International en Technologies de l'Information et de la Communication dans l'Enseignement Supérieur et l'Entreprise(TICE2004)*, pages 52–59, Compiegne, France, 2004.

[Merceron et Yacef (2005)] A. Merceron et K. Yacef. TADA-Ed for educational data mining. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 7 (1), 2005.

[Miller et Mukerji (2003)] J. Miller et J. Mukerji. *MDA Guide Version 1.0*. Object Management Group, Accessible in http://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf, 2003.

[Minaei-Bidgoli et Punch (2003)] B. Minaei-Bidgoli et W. F. Punch. Using genetic algorithms for data mining optimization in an educational web-based system. In *Proceeding of Genetic and Evolutionary Computation Conference*, pages 2252–2263, 2003.

[Mostow et al. (2005)]J. Mostow, J. Beck, H. Cen, E. Gouvea, et C. Heiner. Interactive demonstration of a generic tool to browse tutor-student interactions. In *The AIED05 Workshop on Usage Analysis in Learning Systems*, Amsterdam, Pays-bas, 2005.

[Nilakant et Mitrovic (2005)] K. Nilakant et A. Mitrovic. Application of data mining in constraint-based intelligent tutoring systems. In *Proceedings of the artificial intelligence in education, AIED*, pages 896–898, 2005.

[O'Connor et Herlocker (1999)] M. O'Connor et J. Herlocker. Clustering Items for Collaborative Filtering. In *Proceedings of the Workshop on Recommender Systems:Algorithms and Evaluation*. ACM, 1999.

[OMG (2004)] OMG. Enterprise collaboration architecture (ECA) specification, version 1.0. Technical report, 2004.

[OMG (2007)] OMG. *OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2*. Object Management Group, Accessible in http://www.omg.org/spec/UML/2.1.2/Superstructure/pdf, 2007.

[Ouraiba et al. (2010)] E.-A. Ouraiba, C. Choquet, P. Cottier, C. Després, et P.Jacoboni. Engineering of open learning scenarios : The case of hop3x learning scenarios. In *The 10th IEEE International Conference on Advanced Learning Technologies (ICALT2010)*, Sousse, Tunisia, 2010.

[Parsons et al. (2004)] J. Parsons, P. Ralph, et K. Gallagher. Using Viewing Time To Infer User Preference in Recommender Systems. In *Proceedings of the Workshop in Semantic Web Personalization*. AAAI, 2004.

[Pendergast (2006)] M. Pendergast. An analysis tool for the assessment of student participation and implementation dynamics in online discussion forums. *SIGITE Newsletter*, 3 (2): 10–17, 2006.

[Pernin et Lejeune (2004)] J.-P. Pernin et A. Lejeune. Dispositifs d'apprentissage instrumentés par les technologies : vers une ingénierie centrée sur les scénarios. In *Colloque TICE2004*, Compiègne, France, 2004.

[Pham-Thi (2010)] N.-D. Pham-Thi. Calcul d'indicateur pour la ré-ingénierie de scénario pédagogique en utilisant UTL et DCL4UTL. In *Actes des Rencontres Jeunes Chercheurs en EIAH (RJC2010)*, Lyon, France, mai 2010.

[Pham-Thi et al. (2009a)] N.-D. Pham-Thi, S. Iksal, C. Choquet, et E. Klinger. Dcl4utl : Une proposition de langage de calcul déclaratif pour le processus d'analyse de traces d'apprentissage. In *Actes des Environnements Informatiques pour l'Apprentissage Humain (EIAH2009)*, pages 29–36, Le Mans, France, juin 2009.

[Pham-Thi et al. (2009b)] N.-D. Pham-Thi, S. Iksal, C. Choquet, et E. Klinger. UTL-CL : A declarative calculation language proposal for a learning tracks analysis process. In *The 9th IEEE International Conference on Advanced Learning Technologies (ICALT2009)*, Riga, Latvia, 2009.

[Pham-Thi et al. (2010a)] N.-D. Pham-Thi, S. Iksal, et C. Choquet. Learning tracks' analysis with DCL4UTL: An Instantiation with the Division of Labor Indicator. In AACE, editor, *Global Learn Asia Pacific 2010*, pages 3334–3343, Penang, Malaysia, 2010.

[Pham-Thi et al. (2010b)] N.-D. Pham-Thi, S. Iksal, et C. Choquet. Re-engineering of pedagogical scenarios using the data combination language and usage tracking language. In *The 10th IEEE International Conference on Advanced Learning Technologies (ICALT2010)*, Sousse, Tunisia, July 2010.

[Postic et De Ketele (1988)] M. Postic et J.-M. De Ketele. *Observer les situations éducatives*. Presses Universitaires de France, 1988.

[Pozzi (2005)] F. Pozzi. DPULS-D32.4.1, The set of recurrent problems and description of solutions. Technical report, Action of The European Network of Excellence Kaleidoscope, Site Accessible à: http://www.noe-kaleidoscope.org, 2005.

[PPP (2000)] PPP. Pedagogical Patterns Project. Accessible in http://www.pedagogicalpatterns.org/ (Consulté en mars 2011), 2000.

[Quénu-Joiron et Condamines (2009)] C. Quénu-Joiron et T. Condamines. Utiliser le RaPC pour favoriser le transfert de savoir-faire entre enseignants : le projet TeTraKap. In *Actes des Environnements Informatiques pour l'Apprentissage Humain (EIAH2009)*, pages 141–148, Le Mans, France, juin 2009.

[Randriamalaka et Iksal (2006)] N. Randriamalaka et S. Iksal. Patterns approach in the re-engineering process of learning scenario. In *The 6th IEEE International Conference on Advanced Learning Technologies (ICALT2006)*, Kerkrade, The Netherlands, 2006.

[Reffay et Lancieri (2006)] C. Reffay et L. Lancieri. Quand l'analyse quantitative fait parler les forums de discussion. *Revue STICEF*, Vol. 13, 2006.

[Richards et al. (2001)] G. Richards, V. Rayward-Smith, P. Sönksen, S. Carey, et C. Weng. Data mining for indicators of early mortality in a database of clinical records. *Artificial Intelligence in Medicine*, 22(3): 215–231, 2001.

[Riesbeck et Schank (1989)] C. K. Riesbeck et R. C. Schank. *Inside Case-Based Reasoning*. Psychology Press, 1989.

[Romero et Ventura (2007)] C. Romero et S. Ventura. Educational data mining: A survey from 1995 to 2005. *Expert Systems with application*, Vol. 33: 135–146, 2007.

[Romero et al. (2004)] C. Romero, S. Ventura, et P. De Bra. Knowledge discovery with genetic programming for providing feedback to courseware authors. *User Modeling and User-Adapted Interaction*, 14(5): 105–129, 2004.

[Romero et al. (2007)] C. Romero, S.Ventura, et E. Garía. Data mining in course management systems: Moodle case study and tutorial. *Computers & Education*, 2007.

[Ruet (2002)] M. Ruet. *Capitalisation et réutilisation d'expériences dans un contexte multiacteur*. Doctorat, L'Institut National Polytechnique De Toulouse, 2002.

[Schmidt et al. (2000)] D. Schmidt, M. Stal, H. Rohnert, et F. Buschmann. *Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects*. John Wiley & Sons, 2000.

[Settouti (2011)] L. S. Settouti. *Systèmes à Base de traces modélisées - Modèles et langages pour l'exploitation des traces d'Interactions*. Thèse de doctorat en informatique, Université Claude Bernard Lyon 1, Jan. 2011. URL http://liris.cnrs.fr/publis/?id=4984.

[Settouti et al. (2006)] L. S. Settouti, Y. Prié, A. Mille, et J.-C. Marty. Système à base de traces pour l'apprentissage humain. In *TICE Colloque International en Technologies de l'Information et de la Communication dans l'Enseignement Supérieur et l'Entreprise*, Toulouse, France, 2006.

[Settouti et al. (2007)] L. S. Settouti, Y. Prié, J.-C. Marty, et A. Mille. Vers des Systèmes à Base de Traces modélisées pour les EIAH. Technical Report RR-LIRIS-2007-016, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon, May 2007. URL http://liris.cnrs.fr/publis/?id=2882.

[Settouti et al. (2009)] L. S. Settouti, Y. Prié, P.-A. Champin, J.-C. Marty, et A. Mille. A Trace-Based Systems Framework: Models, Languages and Semantics. Research report, 2009. Work in progress.

[Silverstein et al. (1999)] C. Silverstein, H. Marais, M. Henzinger, et M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33: 6–12, September 1999. ISSN 0163-5840.

[Srivastava et al. (1998)] J. Srivastava, R. Cooley, M. Deshpande, et P. Tan. Cluster analysis and display of genome-wide expresssion patterns. *Proc. Nat. Acad. Sci. USA*, 95: 14863–14868, 1998.

[Srivastava et al. (2000)] J. Srivastava, R. Cooley, M. Deshpande, et P. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2): 12–23, 2000.

[Takács et al. (2009)] G. Takács, I. Pilászy, B. Németh, et D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *J. Mach. Learn. Res.*, 10: 623–656, June 2009. ISSN 1532-4435.

[Talavera et Gaudioso (2004)] L. Talavera et E. Gaudioso. Mining student data to characterize similar behavior groups in unstructured collaboration spaces. In *Workshop on artificial intelligence in CSCL. 16th European conference on artificial intelligence*, pages 17–23, 2004.

[Tang et McCalla (2005)] T. Tang et G. McCalla. Smart recommendation for an evolving e-learning system: Architecture and experiment. *International Journal on E-Learning*, 4: 105–129, 2005.

[Tchounikine (2002)] P. Tchounikine. Pour une ingénierie des EIAH. *Revue I3 Information-Interaction-Intelligence*, Volume 2, n°1: 59–95, 2002.

[Tchounikine (2009)] P. Tchounikine. *Précis de recherche en ingénierie des EIAH*. En ligne sur le web : http://hal.archives-ouvertes.fr/docs/00/42/98/59/PDF/PrecisV1.pdf, 2009.

[Tchounikine et al. (2004)] P. Tchounikine, M. Baker, N. Balacheff, M. Baron, A. Derycke, D. Guin, J.-F. Nicaud, et P. Rabardel. Platon-1 : quelques dimensions pour l'analyse des travaux de recherche en conception d'EIAH. Technical report, D'épartement STIC du CNRS, 2004.

[Teutsch et al. (2004)] P. Teutsch, J.-F. Bourdet, et O. Gueye. Perception de la situation d'apprentissage par le tuteur en ligne. In *Actes de la conférence Technologies de l'Information et de la Communication pour l'Education, TICE2004*, 2004.

[Tichkiewitch (2008)] S. Tichkiewitch. Capitalization and reuse of forging knowledge in integrated design. In A. Bernard and S. Tichkiewitch, editors, *Methods and Tools for Effective Knowledge Life-Cycle-Management*, pages 479–485. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-78431-9.

[Verdejo et al. (2005)]M. Verdejo, C. Celorrio, et Contributeurs. DPULS-D32.5.1, The design pattern language. Technical report, Action of The European Network of Excellence Kaleidoscope, Site Accessible à: http://www.noe-kaleidoscope.org, 2005.

[Völter (2004)]M. Völter. Hope, Belief et Wizardry. Three different perspectives on project management. Accessible in http://www.voelter.de/data/pub/hbw.pdf (Consulté en mars 2011), 2004.

[Witten et Frank (2005)] I. H. Witten et E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann, 2005.

[Yacef (2005)] K. Yacef. The Logic-ITA in the classroom: A medium scale experiment. *International Journal of Artificial Intelligence in Education*, 15 (1): 41–62, 2005. ISSN 1560-4292.

[Zaïane et Luo (2001)] O. Zaïane et J. Luo. Application of data mining in constraint-based intelligent tutoring systems. In *Proceedings of conference on advanced technology for education*, pages 60–64, 2001.

[Zendagui (2010)] B. Zendagui. *Support 'a la spécification des besoins d'observation dans un contexte de ré-ingénierie des scénarios pédagogiques : une approche dirigée par les modèles*. Doctorat, Université du Maine, 2010.

[Zendagui et al. (2008)] B. Zendagui, V. Barré, et P. Laforcade. Support to the specification of observation needs. In *The 8th IEEE International Conference on Advanced Learning Technologies (ICALT2008)*, Santander, Cantabria, Spain, 2008.

[Zendagui et al. (2009a)] B. Zendagui, P. Laforcade, et V. Barré. Aide à la spécification des besoins d'observation une approche dirigée par les modèles. In *Actes des Environnements Informatiques pour l'Apprentissage Humain (EIAH2009)*, pages 231–238, Le Mans, France, juin 2009.

[Zendagui et al. (2009b)] B. Zendagui, P. Laforcade, et C. Choquet. Modélisation de l'observation dans un eiah une approche structurante pour contextualiser un besoin d'observation. In *Actes des Environnements Informatiques pour l'Apprentissage Humain (EIAH2009)*, pages 239–246, Le Mans, France, juin 2009.

[Zinn et Scheuer (2006)] C. Zinn et O. Scheuer. Getting to know your learner in distance learning contexts. innovative approaches innovative approaches for learning and knowledge sharing. In *The First European Conference on Technology Enhanced Learning (EC-TEL 2006). Lecture Notes in Computer Science (LNCS 4227). Springer*, pages 437–451, 2006.

[Zorrilla et al. (2010)] M. Zorrilla, D. García, et E. Álvarez. An approach to measure student activity in learning management systems. In *Proceedings of the 2nd International Conference on Computer Supported Education (CSEDU2010)*, pages 21–28, 2010.

Annexe A Grammaire DCL4UTL

Cette partie détaille la grammaire DCL4UTL proposée dans le chapitre 5 pour le développement de l'interpréteur décrit dans le chapitre 6. Cette grammaire utilise une notation BNF modifiée (Extended Backus-Naur Form), qui consiste en un ensemble de productions composées de noms terminaux et non-terminaux. Un nom terminal représente un ou plusieurs caractères Unicode. Chaque nom non-terminal est défini par une ou plusieurs productions. Dans une production, les noms non-terminaux sont encadrés par < et >, alors que les noms terminaux sont représentés entre " et ". Le texte sans mise en forme particulière et entouré des symboles que sont les crochets pointus représente des noms terminaux informels (par exemple, "< tous les caractères Unicode >"). Chaque grammaire débute par le nom non-terminal Début, dans ce cas, c'est le nom DCL4UTL.

Un ensemble de productions commence par un nom non-terminal, suivi de deux signes deux-points (::) et d'un signe égal (=). La partie droite contient une production terminale ou non-terminale. Une production non-terminale peut présenter plusieurs productions séparées par le symbole barre verticale (|). Les éléments encadrés par des crochets ([]) sont optionnels. Le symbole plus (+) à la suite d'un élément indique que ce dernier peut avoir une ou plusieurs occurrences. Les éléments encadrés par { et } indique que ces éléments peuvent avoir zéro ou plusieurs occurrences. Un ˆ au début d'une classe de caractères entre crochets ([ˆ] signifie l'ensemble des caractères qui ne sont pas listés dans la classe. Des sauts de ligne et des retraits peuvent être ajoutés afin d'améliorer la lisibilité et ne font pas partie de la production.

Les productions principales

```
<DCL4UTL> ::= <CompilationUnit>
<CompilationUnit> ::= "cal" <Statements> ["as" <ConditionList>]
                "where" <AliasList>
                ["parameter" <FormalParamsList>]
```

Les productions pour les données nécessaires pour le calcul d'indicateurs

```
<AliasList> ::= <Alias> { "," <Alias> } [ <RealParameterList> ]
<Alias> ::= <Id> "=" <UTLElement>
```

La production pour un élément UTL sans attribut

```
<UTLElement> ::= (<Id> ".” "*”) | (( <Id> | "*” ) (“.” <Id> )+)
```

Les productions pour la création des données intermédiaires paramétrées

```
<RealParameterList> ::="[”<RealParameter>{“,”<RealParameter>}"]”
<RealParameter> ::= <Id> “=” <Expression>

<FormalParamsList> ::= <FormalParameter> {“,”<FormalParameter>}
<FormalParameter> ::= <Id> “=” <Id>
```

Les productions pour les commandes

```
<Statements> ::= “{” ( <Statement> )+ “}”
<Statement> ::= <ForStatement>
                | <AssignmentStatements>
                | <IfStatement>
```

La production pour la déclaration d’un variable

```
<VariableDeclaration> ::= “$” <Id>
```

Les productions pour la boucle “for”

```
<ForStatement> ::= “for” “each” [“all”] <VariableDeclaration>
                     “in”( <Expression> | <ValueSet> ) “do”
                     [ “as” <ConditionList> ] <Statements>

<ValueSet> ::= “[” <Expression> “ ;” <Expression> “]”
```

Les productions pour la structure “if”

```
<IfStatement> ::= “if” “(” <Expression> “)” <Statements>
                 [“else” <Statements> ]

<ConditionList> ::= (<Filter> [ <Sort> ] ) | <Sort>
```

Les productions pour le filtrage et le tri

```
<Filter> ::= “filter” “(” <Expression> “)”

<Sort> ::= “sort” “(” <SortList> “)”
<SortList> ::= <SortCondition> { “,” <SortCondition> }
<SortCondition> ::= <UTLElement> [ “[” <Id> “]” ] [“asc” | “desc” ]
```

Les productions pour l’affectation

```
<AssignmentStatements> ::= <AssignmentStatement> “ ;”
<AssignmentStatement> ::= <ElementAssignment>
                          | <VariableAssignment>
                          | <ArrayAssignment>
                          | <StructureAssignment>
```

```
<Declaration> ::= <VariableDeclaration> | <Id>
```

L'affectation d'une valeur dans une variable

```
<VariableAssignment> ::= <Declaration> "=" <Expression>
```

L'affectation d'un tableau dans une variable

```
<ArrayAssignment> ::= <Declaration> "=" (<Array>|
<TwoDimensionsArray>)
<Array> ::= "[" <Expression> {"," <Expression>} "]"
<TwoDimensionsArray> ::= "[" <Array> {"," <Array>} "]"
```

L'affectation d'un objet dans une variable

```
<StructureAssignment> ::= <Declaration> "=" <Structure>
<Structure> = "[" <StructureName>  <Field> {"," <Field> } "]"
<StructureName> ::= <Id> {"." <Id> }
<Field> = <Id> ":" (<Expression> | <Structure>)
```

L'affectation d'une valeur dans un élément UTL

```
<ElementAssignment> :: = <ElementDeclaration> "=" <Expression>

<ElementDeclaration> ::= <UTLElement> [ <AttributeList> ]

<AttributeList> ::= ("[" <AttributeAssignment> "]" )+
                      { "." <AttributeElementList> }
```

L'affectation d'une valeur dans un attribut d'un élément UTL

```
<AttributeAssignment> ::= <Term> "=" <Expression>

<AttributeElementList> ::= <Id> {"[" <AttributeAssignment> "]" }
```

Les productions pour l'expression arithmétique et relationnelle

```
<Expression> ::= <ConditionalOrExpression>

<ConditionalOrExpression> ::= <ConditionalAndExpression>
                                {"or" <ConditionalAndExpression> }

<ConditionalAndExpression> ::= <EqualityExpression>
                                { "and" <EqualityExpression>}

<EqualityExpression> ::= <RelationalExpression>
                           { "==" <RelationalExpression>
                           | " !=" <RelationalExpression> }

<RelationalExpression> ::= <AdditiveExpression>
```

```
                { "<" <AdditiveExpression>
                | ">" <AdditiveExpression>
                | "<=" <AdditiveExpression>
                | ">=" <AdditiveExpression> }

<AdditiveExpression> ::= <MultiplicativeExpression>
                { "+"<MultiplicativeExpression>
                | "-" <MultiplicativeExpression>}

<MultiplicativeExpression> ::= <UnaryExpression>
                      { "*" <UnaryExpression>
                      | "/" <UnaryExpression> }

<UnaryExpression> ::= "-" <UnaryExpression> | <Term>

<Term> ::= <Literal>
           | "(" <Expression> ")"
           | <AttributeComparison>
           | <Element>
           | <Functions>
           | <Id>
<Literal> ::= <IntLiteral>
                | <FloatLiteral>
                | <StringLiteral>
                | <BooleanLiteral>
```

La production pour un élément UTL avec les attributs et le filtrage

```
<Element> ::= <UTLElement> [ "[" <Id> "]" ] [ "as" <ConditionList> ]
```

Les productions pour comparer un attribut d'un élément UTL avec une valeur

```
<AttributeComparison> ::= <UTLElement> <ComparisonList>

<ComparisonList> ::= ("[" <Comparison> "]" )+
                      {"." <AttributeElement> }[ "[" <Id> "]" ]

<Comparison> ::= <Id> "==" <Expression>

<AttributeElement> ::= <Id> {"[" <Comparison> "]" }
```

Les productions pour les fonctions prédéfinies

```
<Functions> ::= <First>
                | <End>
                | <Index>
                | <Max>
                | <Min>
                | <Count>
                | <CountDis>
                | <Sum>
                | <SumDis>
```

```
                | <Concat>
                | <Contains>
                | <Compare>
                | <CallExterne>

<First> ::= "first" "(" <Expression> ")"

<End> ::= "end" "("<Expression> ")"

<Index> ::= "index" "("<Expression> "," <Expression> ")"

<Max> ::= "max" "(" <Expression> ")"

<Min> ::= "min" "(" <Expression> ")"

<Count> ::= "count" "(" <Expression> ")"

<CountDis> ::= "countDis" "(" <Expression> ")"

<Sum> ::= "sum" "(" <Expression> ")"

<SumDis> ::= "sumDis" "(" <Expression> ")"

<Concat> ::= "concat" "(" <Expression> "," <Expression> ")"

<Contains> ::= "contains" "(" <Expression>"," <Expression> ")"

<Compare> ::= "compare" "(" <Expression> "," <Expression> ","
                    <Expression> ")"
```

Les productions pour l'appel des fonctions externes

```
<CallExterne> ::= "extern" [<Remote>] <Prototype>
<Prototype> ::= [<Class>] <ExFunctionName> <Arguments>
<Remote>  ::= ("ws" |"rpc") <Host>
<Host> ::= <Expression>

<Class> ::= <Id> {"." <Id> }
<ExFunctionName> ::= <Id>

<Arguments> ::= "(" [<ArgumentList>] ")"

<ArgumentList> ::= <Expression> { "," <Expression> }
```

Les productions pour les constantes

```
<IntLiteral> :: = ( <Digit> )+

<FloatLiteral> ::= "." ( <Digit> )+ | (<Digit>)+ "." { <Digit> }

<StringLiteral> ::= <DoubleQuoteChar> [<StringCharacter>]
                          <DoubleQuoteChar>
```

```
<DoubleQuoteChar> ::= “ " ”

<StringCharacter> ::= { <Character> }

<Character> ::= [^"]

<BoolLiteral> ::= “true” | “false”
```

Les productions pour l’identifiant

```
<Id> ::= (<Letter> | “_” ) { <Letter> | <Digit> | “-” | “_” }

<Digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<Letter> ::= (“a”-“z” | “A”-“Z”)
```

Annexe B Liste des indicateurs

Cette annexe explicite la liste des indicateurs utilisés dans l'expérimentation (cf. chapitre 7).

Les indicateurs pour question 1

1. Une classe s'appelant "Point" doit être créée.
2. La visibilité doit être "public".
3. La classe doit avoir deux champs.
4. Les champs doivent être réels (float ou double).
5. La visibilité des champs doit être "private".
6. La classe doit avoir un constructeur spécifique à deux arguments.
7. Les arguments de ce constructeur spécifique doivent être réels (double ou float).
8. Les champs doivent avoir un ''getter''.
9. Les champs doivent avoir un ''setter''.

Les indicateurs pour question 2

1. Une méthode appelée ''equals'' doit exister dans la classe ''Point''.
2. La méthode doit s'appeler ''equals'' et non ''Equals''.
3. La méthode ''equals'' doit avoir une visibilité "public".
4. La méthode ''equals'' doit avoir un type de retour booléen.
5. La méthode ''equals'' doit avoir 1 paramètre.
6. Le paramètre de la méthode ''equals'' doit être de type *java.lang.Object.*

Les indicateurs pour question 3

1. Une méthode appelée distance doit exister dans la classe ‘‘Point’’.
2. La méthode doit s’appeler ‘‘distance’’ et non ‘‘Distance’’.
3. La méthode ‘‘distance’’ doit avoir une visibilité “public”.
4. La méthode ‘‘distance’’ doit avoir un type de retour réel (float ou double).
5. La méthode ‘‘distance’’ doit avoir un paramètre.
6. Le paramètre de la méthode ‘‘distance’’ doit être de type ‘‘Point’’.

Les indicateurs pour question 4

1. Un classe s’appelant “Triangle” doit être créée.
2. Sa visibilité doit être “public”.
3. La classe doit avoir trois champs.
4. Les champs doivent être de type ‘‘Point’’.
5. La visibilité des champs doit être “private”.
6. La classe doit avoir un constructeur spécifique à trois arguments.
7. Les arguments de ce constructeur spécifique doivent être de type ‘‘Point’’.
8. Les champs doivent avoir un ‘‘getter’’.
9. Les champs doivent avoir un ‘‘setter’’.
10. La classe ‘‘Triangle’’ ne doit pas hériter de la classe ‘‘Point’’.

Les indicateurs pour question 5

1. Une méthode appelée ‘‘perimetre’’ doit exister dans la classe ‘‘Triangle’’.
2. La méthode doit s’appeler ‘‘perimetre’’ et non ‘‘Perimetre’’.
3. La méthode ‘‘perimetre’’ doit avoir une visibilité “public”.

4. La méthode ‘‘perimetre’’ doit avoir un type de retour réel (float ou double).

5. La méthode ‘‘perimetre’’ doit avoir zéro paramètre.

6. La méthode ‘‘perimetre’’ doit utiliser la méthode ‘‘distance’’.

Les indicateurs pour question 6

1. Une méthode appelée ‘‘surface’’ doit exister dans la classe ‘‘Triangle’’.

2. La méthode doit s’appeler ‘‘surface’’ et non ‘‘Surface’’.

3. La méthode ‘‘surface’’ doit avoir une visibilité “public”.

4. La méthode ‘‘surface’’ doit avoir un type de retour réel (float ou double).

5. La méthode ‘‘surface’’ doit avoir zéro paramètre.

6. La méthode ‘‘surface’’ doit utiliser la méthode distance.

Les indicateurs pour question 7

1. Un classe s’appelant “Droite” doit être créée.

2. Sa visibilité doit être “public”.

3. La classe doit avoir deux champs.

4. Les champs doivent être réels.

5. La visibilité des champs doit être “private”.

6. La classe doit avoir un constructeur spécifique à deux arguments.

7. Les arguments de ce constructeur spécifique doivent être réels.

8. Les champs doivent avoir un ‘‘getter’’.

9. Les champs doivent avoir un ‘‘setter’’.

Les indicateurs pour question 8

Un constructeur avec deux arguments de type ‘‘Point’’ doit exister dans la classe ‘‘Droite’’.

Les indicateurs pour question 9

1. Une méthode appelée ‘‘estSurDroite’’ doit exister dans la classe ‘‘Point’’.

2. La méthode doit s’appeler ‘‘estSurDroite’’ et non ‘‘EstSurDroite’’.

3. La méthode ‘‘estSurDroite’’ doit avoir une visibilité “public”.

4. La méthode ‘‘estSurDroite’’ doit avoir un type de retour booléen.

5. La méthode ‘‘estSurDroite’’ doit avoir un paramètre.

6. Le paramètre de la méthode ‘‘estSurDroite’’ doit être de type ‘‘Droite’’.

Les indicateurs pour question 10

1. Une méthode appelée ‘‘estParallele’’ doit exister dans la classe ‘‘Droite’’.

2. La méthode doit s’appeler ‘‘estParallele’’ et non ‘‘EstParallele’’.

3. La méthode ‘‘estParallele’’ doit avoir une visibilité “public”.

4. La méthode ‘‘estParallele’’ doit avoir un type de retour booléen.

5. La méthode ‘‘estParallele’’ doit avoir un paramètre.

6. Le paramètre de la méthode ‘‘estParallele’’ doit être de type ‘‘Droite’’.

Les indicateurs pour question 11

1. Une méthode appelée ‘‘intersection’’ doit exister dans la classe ‘‘Droite’’.

2. La méthode doit s’appeler ‘‘intersection’’ et non ‘‘Intersection’’.

3. La méthode ‘‘intersection’’ doit avoir une visibilité “public”.

4. La méthode ‘‘intersection’’ doit avoir un type de retour ‘‘Point’’.

5. La méthode ‘‘intersection’’ doit avoir un paramètre.

6. Le paramètre de la méthode ‘‘intersection’’ doit être de type ‘‘Droite’’.

7. La méthode ‘‘intersection’’ doit faire appel à la méthode ‘‘estParallele’’ (écrite à la question 10).

Les indicateurs pour question 12

1. Un constructeur avec trois arguments de type ‘‘Droite’’ doit exister dans la classe ‘‘Triangle’’.

2. Ce constructeur doit utiliser la méthode ‘‘estParallele’’ (écrite à la question 10).

Les indicateurs transversaux

1. La fréquence de compilation manuelle (par minute).

2. La fréquence d’exécution (par minute).

3. Le taux de correction des erreurs à la compilation (Est-ce qu’une même erreur revient souvent ou est-ce que l’étudiant a assimilé qu’il s’agissait d’une erreur ?).

4. Le pourcentage de noms de variable significatifs (On se basera sur la longueur des noms).

5. Le pourcentage de noms de méthode significatifs (On se basera sur la longueur des noms).

6. Le pourcentage de variables d’instances privées.

7. La détection de l’utilisation d’un membre d’instance (variable ou méthode) dans une méthode d’instance (s’il n’y a pas de membre d’instance, la méthode devrait être une méthode de classe).

8. Quel est le temps moyen passé par question par étudiant.

9. Le nombre de compilation par question par étudiant.

Annexe C Evénement Hop3x

Cette partie présente les événements collectés par le système Hop3x. Ces événements sont générés lorsque l'étudiant réalise une action (l'ajout d'un projet, la compilation, la suppression d'un texte, etc.).

Evénement	Description
AF	Ajout d'un fichier
AP	Ajout d'un projet
ANNOTATION	Enseignant annote une partie du code d'un apprenant
HELP	Appel au secours
CM	Compilation manuelle
CA	Compilation automatique
EF	Enregistrement d'un fichier
EP	Enregistrement d'un projet
MESSAGE_ENSEIGNANT	Enseignant envoie un message vers un étudiant
E	Exécution d'un programme
FF	Fermeture d'un fichier
FIN	Fin d'une session
INITIERCAUSERIE	Début d'une causerie
IT	Insertion d'un texte
NQ	Navigation d'une question
OF	Ouverture d'un fichier
SE	Choix d'une erreur
SQ	Choix d'une question
MESSAGE_SERVEUR	Serveur envoie un message vers un étudiant
SF	Suppression d'un fichier
SP	Suppression d'un projet
ST	Suppression d'un texte
TERMINERCAUSERIE	Fin d'une causerie

Tableau C-1 Evénements Hop3x

Annexe D Le modèle d'information d'une donnée brute UTL

Cette partie présente le modèle d'information d'une donnée brute UTL. Elle est extraite de [Choquet et Iksal (2007b)]. La figure D-1 présente les trois facettes d'une donnée brute.

La facette "définition" (*defining*) de la donnée brute est décrite par trois éléments : le titre (*title*) de la donnée doit être concis et sans ambiguïtés pour exprimer la sémantique de la donnée ; la cardinalité (*cardinality*) de la donnée représente le nombre d'instances possibles de la donnée ; une description (*description*) plus détaillée peut être ajoutée pour décrire de manière informelle la nature de la donnée.

La facette "obtention" (*getting*) se concentre sur la description des moyens d'observation, c'est-à-dire de la méthode d'analyse permettant d'établir la donnée. Elle est composée de trois éléments :

- la période d'acquisition (*acquisitionTime*) prend ses valeurs dans la liste fermée : avant la session (*Before-session*), en cours de session (*During-session*) et après la session (*After-session*).
- le type de collection (*collectionType*) est (cf. figure D-2) :
    - un recueil (*humanCollection*) opéré par au moins un rôle (*role*) (par exemple, un observateur de la session) en utilisant un moyen de collecte (*collectionVector*) (par exemple une caméra, un papier et un crayon).
    - un recueil automatique (*automaticCollection*), caractérisé par :
        - le type d'enregistrement (*recordType*) qui prend ses valeurs dans une liste ouverte (par exemple, fichier de log, chat, mail),
        - un outil de collecte (*recordTool*) dont la localisation (*location*) est connue s'il est déjà développé et utilisable dans l'environnement de formation. Sinon, il est nécessaire d'en donner une description

(*description*) et/ou quelques exemples (*example*) afin d'aider au développement de l'outil.

- la description de la trace (*track*) (cf. figure D-3).

- la localisation (*location*) de la donnée, il s'agit souvent de l'URL du fichier qui la contient.

La facette "utilisation" (*using*) comporte trois éléments : utilisé-par (*usedBy*) est proposé afin de faciliter la navigation dans un graphe de dépendances des données ; contenu (*data*) permet de stocker la donnée après son extraction de la trace source ; *format* décrit comment la donnée est représentée, une fois obtenue (son schéma XML).

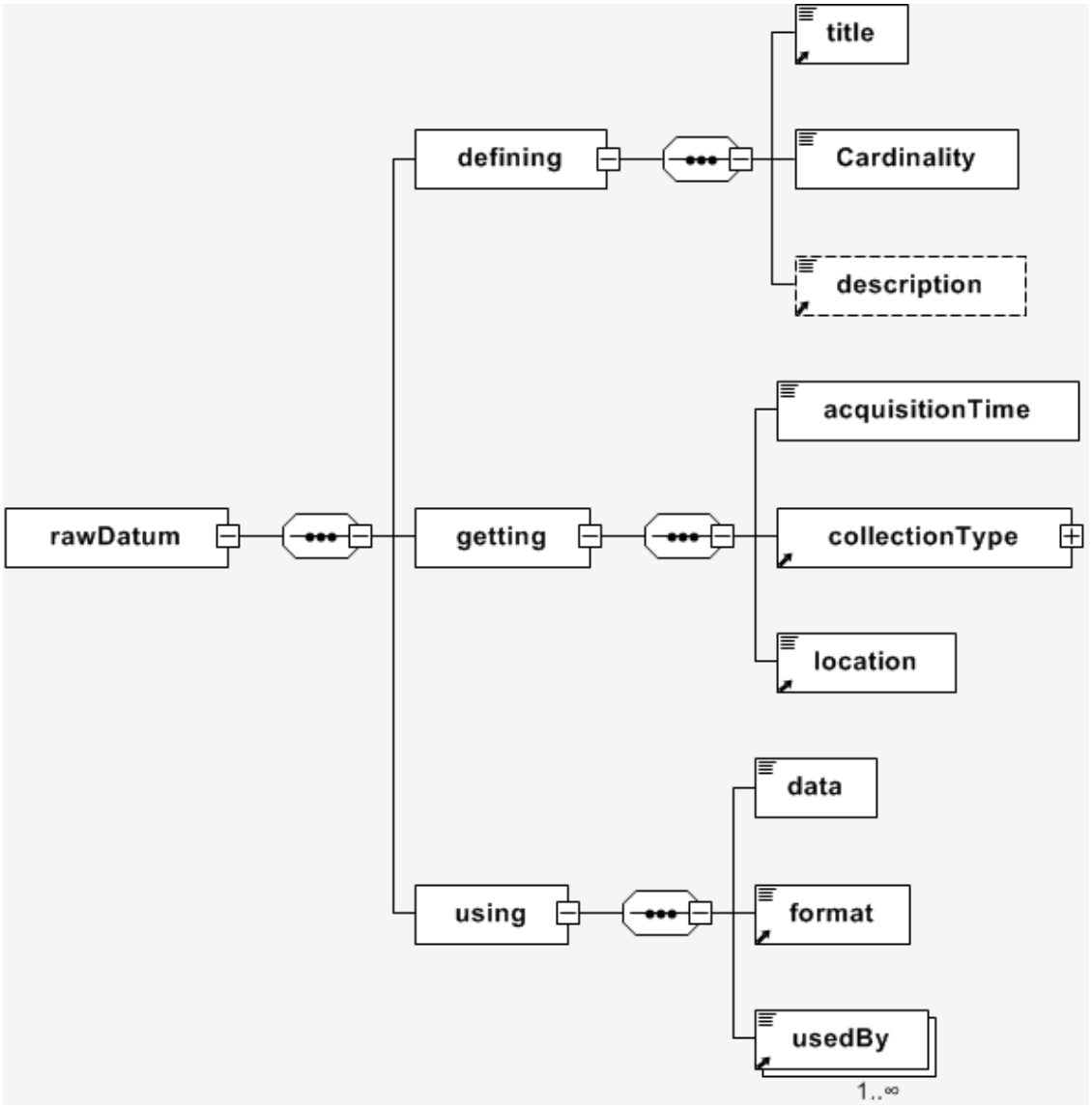

Figure D-1 Le modèle d'information d'une donnée brute

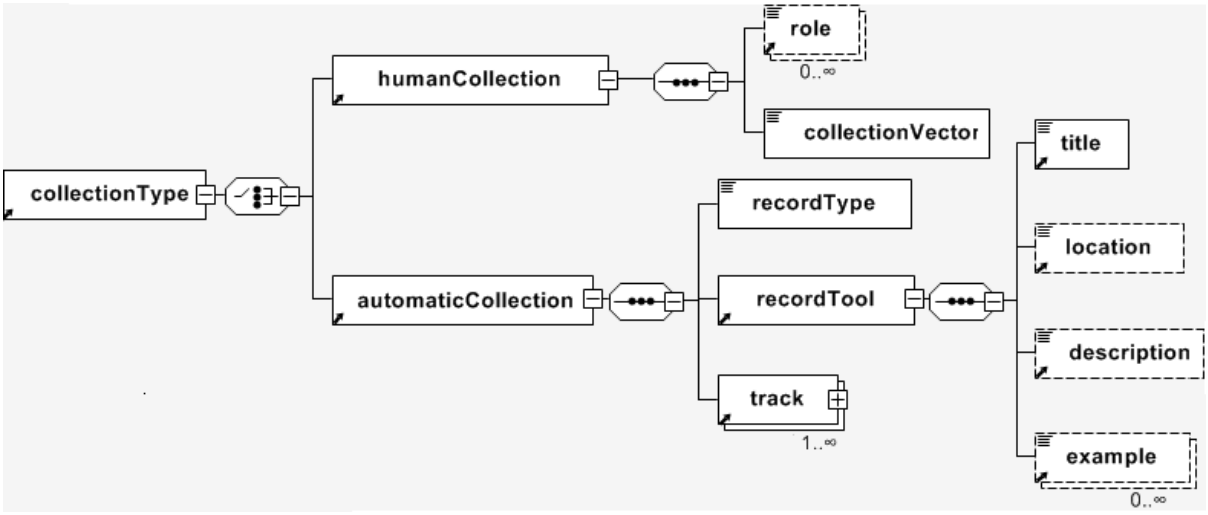

Figure D-2 Le modèle d'information d'un type de collection d'une donnée brute

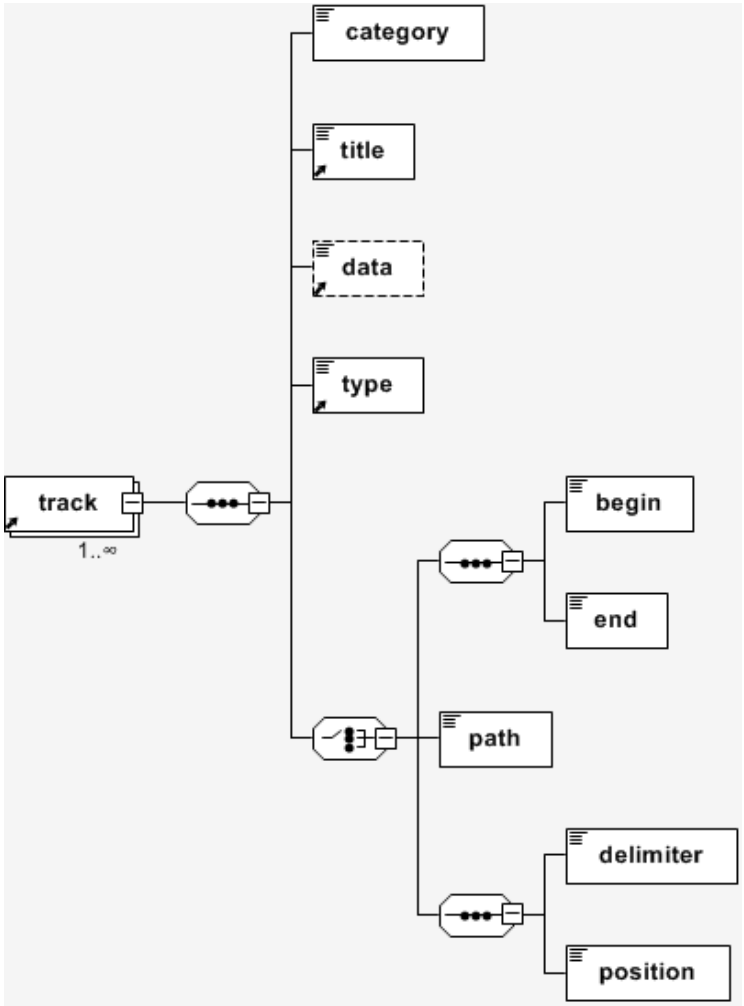

Figure D-3 Le modèle d'information d'une trace

Une trace est généralement composée de plusieurs informations dont seules certaines sont riches en signification pour le concepteur. Le modèle d'information d'une trace (cf. figure D-3) de la donnée brute UTL est générique et compatible avec la majorité des formats de traces collectées de manière automatique.

La description d'une trace (*track*) se décompose en objets de deux catégories (*category*) : les mots clés (*keyword*) et les valeurs (*value*). Ces concepts génériques permettent la description de nombreux formats de traces depuis des fichiers textes structurés jusqu'aux bases de données et aux vidéos (la sous-section suivante donne un exemple pour un fichier de "logs" de type texte).

Le titre (*title*) d'une trace désigne le contenu et lui associe un sens (par exemple : "date de début d'activité", "réponse au QCM").

Le champ donnée (*data*) est utilisé pour stocker la valeur ou le mot clé.

Le type peut prendre pour valeur "Text", "XML" ou "Database". Cette liste est ouverte.

Pour une trace de type "Database" ou "XML", le chemin (*path*) contient le chemin d'accès à la donnée, soit une requête XPath ou SQL par exemple.

En ce qui concerne les fichiers de "logs" sous forme de fichiers texte structurés, UTL propose un ensemble de champs permettant d'indiquer la localisation de la donnée dans une chaîne de caractères, soit par la position des caractères, soit en utilisant des marqueurs.

La donnée peut être un mot clé, c'est-à-dire un mot ou une phrase toujours présent au même endroit dans la trace, ce qui permet de la retrouver et de l'identifier, ou une valeur, c'est-à-dire une donnée collectée automatiquement par le dispositif d'apprentissage sur l'activité d'un acteur de la session d'apprentissage, comme le temps passé à lire une page, ou le nom d'une page lue, ou encore le résultat à un exercice. Dans les deux cas, la localisation du contenu permet de spécifier la position de la valeur ou du mot clé dans la trace. Des attributs spécifiques ont été prévus :

- Début (*begin*) donne la position du premier caractère du contenu.
- Fin (*end*) donne la position du dernier caractère (-1 pour la fin de la ligne).
- Délimiteur (*delimiter*) correspond au délimiteur utilisé pour séparer la chaîne de caractères en différentes sections.
- Position donne la *position* de la section de chaîne de caractères à conserver.