Capitaliser les processus d'analyses de traces d'apprentissage indépendamment des plates-formes d'analyses de traces

Alexis Lebis[1,2], Marie Lefevre[1], Nathalie Guin[1], Vanda Luengo[2]

[1]Université Claude Bernard Lyon 1
LIRIS - Équipe TWEAK
[2]Université de Grenoble 1
LIG - Équipe MeTAH

Table of Contents

Abstract. L'étude des traces d'apprentissage issues de dispositifs d'e-learning représente un enjeu majeur puisqu'elle permet l'extraction et la production de connaissances aidant les différents acteurs impliqués dans lesdits dispositifs, sur le plan de l'enseignement et de l'apprentissage. Les processus d'analyses des traces d'interactions, permettant cette étude, reposent sur une représentation des données et sur l'application d'opérations de transformations sur ces données qui sont différentes en fonction des plates-formes d'analyses. Cette divergence rend caduque la plupart des interopérabilités envisageables entre les plates-formes, entraînant le besoin de développer des processus spécifiques à chacune. Nous proposons dans ce document une approche pour prévenir ces dissensions et ce, grâce à une démarche unificatrice qui sous-tend une analyse générique de traces. Analyse qui pourra, en plus d'être capitalisable, être instanciée dans diverses plates-formes d'analyses.

Abstract. The study of learning traces from e-learning systems represents an important challenge. This analysis allows to produce relevant knowledge, helping the users of the e-learning systems. Analysis processes are a logical sequence of operations, made by operators, fed with data. These elements are platform dependent due to their specificities. These specificities imply the fact that no interoperability at all is possible between strong dissimilar platform. In this paper, we propose an approach that unifiy the description of analysis processes, giving capitalizable and shareable analysis in addition to making the instanciation possible in different platforms.

1 Introduction

Une trace, au sens où nous l'employons dans ce document, est l'enregistrement d'interactions potentiellement intéressantes et pertinentes entre un acteur, comme un apprenant ou un enseignant, et un environnement informatique. Ces traces contenant un aperçu des interactions survenues, il est possible de les manipuler pour essayer, entre autres choses, d'inférer de nouvelles données, d'en déduire le comportement de l'acteur ainsi que l'état de ses connaissances. Cette extraction de connaissances à partir des données s'effectue par des processus d'analyses de traces (fouilles de données, apprentissage automatique...) que les analystes créent.

Dans le domaine de l'e-learning, l'exploitation des traces permet de produire des données pertinentes qui pourront être utilisées par différents acteurs comme les apprenants, les enseignants ou encore les chercheurs pour améliorer l'apprentissage, adopter de nouvelles stratégies pédagogiques, produire de nouvelles connaissances, etc. Dans le terme de *données pertinentes* on trouve, en plus des modèles mathématiques, les indicateurs qui sont des données véhiculant une information forte en sens, servant de retour à un acteur pour alimenter ses décisions [1].

L'extraction de connaissances à partir de processus d'analyses s'effectue en deux temps pour un analyste : une démarche de réflexion intellectuelle et une démarche applicative, revenant à utiliser le plus souvent un outil informatique. Bien que ces deux démarches peuvent sembler détachées, elles sont communément liées, ceci parce que l'analyste considère généralement, dans sa réflexion, les contraintes (comme la représentation des données) qui sont propres à l'outil qu'il va utiliser. De fait, cela occasionne une impossibilité de réutiliser le processus d'analyse obtenu pour d'autres plates-formes [2].

Le problème que nous tenterons donc de résoudre au long de ce rapport est comment éviter, autant que possible, l'ingérence des nécessités applicatives dans la réflexion d'un processus d'analyse pour permettre de l'exprimer génériquement.

Pour ce faire, nous définirons, dans la section 2, le contexte dans lequel se déroule ce stage de master, à savoir le projet HUBBLE. Ensuite, la section 3 reviendra sur la problématique et l'étoffera de quelques scénarios d'usage. Dans la section 4, sera dressé un état de l'art du domaine et nous préciserons où se

situe notre apport sur le plan scientifique. La section 5 décrit la solution théorique apportée, immédiatement concrétisée par la section 6 décrivant l'implémentation de l'approche théorique. La section 7 présente l'expérimentation et les résultats obtenus. Quant aux deux dernières sections, la section 8 s'attarde sur les limites de la proposition, la section 9 sur les perspectives offertes par le travail effectué.

2 Contexte

2.1 HUBBLE

Le projet HUBBLE [3] (HUman oBservatory Based on anaLysis of e-lEarning traces) est un projet national sur 42 mois financé par l'Agence Nationale de la Recherche. Impliquant les huit partenaires que sont les laboratoires scientifiques LIG, LIRIS, LIUM, LINA, IFé-S2HEP, LabSTICC et STEF et la société OpenClassrooms, ce projet propose de réunir des équipes de recherches en sciences humaines et en informatique dans l'étude d'environnements d'e-learning massif, tel que des MOOCs[1] ou de grandes promotions d'étudiants ayant recours à des outils informatiques.

L'objectif du projet HUBBLE est de créer à l'échelle nationale un observatoire pour ces traces d'e-learning, mais aussi pour leurs processus d'analyses, de telle sorte que ces processus puissent être consultables, partageables et réutilisables. Ceci permettrait l'étude, l'analyse et la compréhension des phénomènes d'apprentissage, voire même d'enseignement, susceptibles d'émerger dans les environnements d'e-learning. Les enjeux d'un tel projet sont multiples : permettre la réutilisation et la reproductibilité des analyses sur des corpus de données différents, aider à la recherche de nouvelles méthodes d'analyses pour les chercheurs, tenter de nouvelles approches au sein des environnements d'e-learning (personnalisation du contenu en fonction du profil de l'apprenant, pédagogie CSCL[2], etc...) tout en analysant leur impact sur l'apprentissage...

2.2 Traces

La source historique de la notion de trace peut être assimilée aux fichiers de consignations d'événements (log file en anglais). Les *log files* sont, encore aujourd'hui, utilisés principalement en informatique à des fins de débogage des systèmes et de sécurité [4] [5]. En analysant un tel fichier, on espère trouver le contexte et les événements incriminés : on considère donc que ces fichiers contiennent de la connaissance sur le problème.

Partant de ce présupposé, les techniques de fouille, d'apprentissage et de détection se sont progressivement vues appliquées avec succès aux traces d'interactions des utilisateurs ainsi qu'à leurs activités au sein de systèmes informatisés, permettant d'obtenir des résultats pertinents [6] [7] .

[1] Massive Open Online Course
[2] Computer-Supported Collaborative Learning

Ainsi, la trace a pris une place capitale dans le domaine des EIAH[3], et de nombreux efforts ont été déployés pour tenter d'exprimer et de formaliser cette notion pour en simplifier son exploitation. On peut noter les travaux du projet TRAILS du réseau européen Kaléidoscope dont l'apport a été de proposer une première taxonomie des traces en fonction d'objets pédagogiques (OP) liés entre eux conceptuellement et temporellement [8]. On peut aussi noter l'approche par les modèles qui consiste à structurer une trace en respectant un modèle de traces, lui donnant ainsi l'avantage d'avoir une sémantique plus importante qu'à l'accoutumée [9]. Les systèmes à base de traces sont une approche proposée pour faciliter l'exploitation de ces traces tout en permettant un raisonnement à partir de traces [10]. Enfin, UTL[4] exploite les travaux du Kaléidoscope sur les traces et les indicateurs, dont nous parlerons dans la section 4.

Cependant, malgré les travaux évoqués, la notion de trace reste hétérogène. On note ainsi une hétérogénéité des dispositifs dont les traces sont issues, de la manière de les représenter, ainsi que des données qu'elles véhiculent. Cela rend leur traitement et leur partage complexes malgré des travaux comme ceux de PSLC Datashop [11] ou de MULCE [12] visant à proposer des dépôts de partage et de réutilisation des traces.

Malgré cela, l'exploitation des traces permet d'apporter des éléments de réponse sur des problématiques survenant dans plusieurs domaines, dont celui de l'e-learning, comme l'attrition dans les MOOCs, la personnalisation du parcours d'un apprenant, le suivi de la progression, ce qui contribue à une évolution efficace des solutions mises à disposition des divers acteurs à l'heure actuelle.

2.3 Processus d'analyse de traces

Un processus d'analyse de traces est un ensemble d'actions -appliquées dans un ordre logique- qui vise à manipuler un corpus de données afin de produire des résultats pertinents et utilisables, comme peuvent l'être les modèles mathématiques, les indicateurs, les vues... Une action constituant le processus est matérialisée par un opérateur qui prend en entrée des données et produit un résultat en sortie, ce qui la rend donc assimilable à une fonction : $f(x_i...x_n) = Y$.

La découverte des connaissances dans les données s'est rapidement heurtée à l'explosion de la quantité de données à traiter (le *data deluge*) [13]. Il n'y a qu'à regarder le domaine de l'astronomie qui générait en 2011 0.5 Pétaoctets de données pour s'en convaincre. Le KDD[5] répond à cette problématique en tentant de faire plus que de la simple fouille de données dans une base : développer tout un cycle permettant de transformer des données de bas niveau vers un niveau d'expressivité plus élevé [14].

Pour en revenir à notre sujet, l'EDM[6] et le LAK[7] sont deux disciplines qui ont dans leurs objectifs communs la production et l'analyse de données des

[3] Environnements Informatiques pour l'Apprentissage Humain
[4] Usage Tracking Language
[5] Knowledge Discovery in Database
[6] Educational Data Mining
[7] Learning Analytics and Knowledge

traces d'apprentissage, principalement pour extraire de la connaissance et aider l'apprenant dans son évolution pédagogique. Mais elles effectuent cette production et cette analyse de manière différente : l'EDM s'axe sur une automatisation des traitements tandis que le LAK fait intervenir un analyste dans son processus [15], par exemple pour définir le contexte, faire un choix sur les données jugées pertinentes, etc. L'intervention humaine entraîne une plus grande flexibilité, tout comme un apport créatif plus important, mais moins d'autonomie.

Cependant, bien que les objectifs soient sensiblement identiques, ces deux disciplines ne partagent pas la même communauté, ni les données et les opérateurs utilisés. Ce qui rend la collaboration par l'échange difficile, diminuant de fait la qualité et les résultats des études.

3 Problématique

Lors de la création d'un processus d'analyse, l'analyste doit prendre en compte bien plus que la manière théorique de l'obtenir. En effet, des considérations techniques entrent en jeu, comme la disponibilité et la représentation des données utilisées, les spécificités de la plate-forme d'analyse utilisée, les limites de cette dernière, etc. Pour ces multiples raisons, le processus d'analyse dépend fortement de l'outil qui l'accueille. Généralement, utiliser un processus issu d'une autre plate-forme implique son entière réécriture afin de l'adapter, comme par exemple pour gérer une représentation différente des données.

Face à ce constat d'extrême dépendance et de redondance, on est en droit de se poser la question suivante : "Comment définir et décrire un processus d'analyse de traces indépendamment des plates-formes, pour le rendre capitalisable ?".

Nous défendons l'idée que fournir une réponse à cette question aidera l'analyste en de nombreux points, comme :

- le libérer des spécificités des plates-formes, lui permettant de se focaliser uniquement sur la définition théorique du processus ;
- lui permettre d'exprimer facilement ce qu'il cherche à obtenir, comme un indicateur ;
- lui permettre de partager efficacement son processus d'analyse entre différents utilisateurs et pour différentes plates-formes ;
- lui éviter de recréer un processus d'analyse de traces déjà existant.

Pour illustrer ces attentes, nous présentons quelques cas d'usage illustrant les difficultés rencontrées actuellement par les analystes, et nous expliquons comment notre approche peut fournir une solution, avant de présenter les questions de recherche issues de notre approche.

3.1 Cas d'usages

3.1.1 L'enrichissement des possibilités d'une plate-forme. Considérons dans ce cas d'usage un analyste travaillant avec la plate-forme d'analyse de traces UnderTracks [16], fournie par le *LIG*. Imaginons qu'il ai commencé à décrire

son processus d'analyse répondant à un besoin précis, comme par exemple la classification d'apprenants en fonction de leur activité journalière.

Le problème qu'il rencontre est l'absence d'un algorithme spécifique de classification dont il a besoin, supposons ici le *Multiclass alternating decision tree* [17]. Il lui est donc impossible de continuer la description de son processus d'analyse dans UnderTracks et il se retrouve bloqué.

L'idée, plutôt que de consulter une par une les plates-formes pour savoir si elles proposent un tel algorithme, est de passer par un outil permettant de décrire, de manière générique, la ou les opérations recherchées par l'analyste. Une fois cette description terminée, l'outil sera alors capable de lui indiquer quelle plate-forme est capable d'effectuer cette ou ces actions, et de quelles manières. Pour rejoindre notre exemple, l'outil pourrait très bien suggérer d'utiliser Weka [18], une plate-forme de *data mining*, qui implémente l'algorithme recherché sous la dénomination LADTree.

Ainsi, l'analyste est conseillé par le système dans son choix technique et est assisté dans la réalisation de sa tâche, favorisant son efficacité.

3.1.2 L'aide au choix des plates-formes pertinentes. Dans ce cas d'usage, nous considérons un analyste qui possède une pleine représentation de la manière de résoudre le problème dont il a la charge, autrement dit qu'il possède la démarche intellectuelle, mais qu'il n'arrive pas à la décrire dans une plate-forme, ou bien qu'il ne sait tout simplement pas laquelle utiliser.

L'idée ici est de proposer à l'analyste un outil lui permettant de décrire le processus d'analyse qu'il imagine. De le décrire tel quel, étape par étape, sans tenir compte de quelconques prérequis comme la représentation des données dans les traces, la manière d'utiliser une opération... La description terminée , l'outil lui indique quelle(s) plate(s)-forme(s) est capable de gérer en entier le processus d'analyse décrit, et de quelle manière procéder étape par étape, en fonction de chaque plate-forme.

Un tel outil va permettre à l'analyste de s'affranchir de la nécessité de penser son processus en fonction de plusieurs contraintes liées aux plates-formes (comme la représentation des données), améliorant ainsi la pertinence et la qualité de ses productions.

3.1.3 Le partage d'un processus d'analyse. Considérons ici une approche légèrement différente. L'intention principale n'étant plus la résolution, mais l'expression et le partage d'un processus d'analyse souvent utilisé et qui exploite toujours les mêmes catégories de données. Par exemple, l'activité d'un apprenant dans un MOOC.

L'échange de processus entre plates-formes impliquant une réécriture, il est intéressant de proposer à notre analyste un outil lui permettant d'exprimer son processus de manière indépendante des plates-formes d'analyse de traces disponibles. En effet, l'exprimer génériquement permet de l'échanger avec d'autres personnes sans se soucier de sa compatibilité.

Dans les deux cas d'usages précédents, nous avons cerné l'avantage de décrire un processus d'analyse de manière indépendante des plates-formes. Nous voyons ici qu'il est aussi nécessaire de permettre une instanciation d'un tel processus générique, sur les plates-formes compatibles et capables de l'accueillir.

Un tel outil va permettre de limiter la redondance en proposant un réservoir de processus d'analyse réutilisables et instanciables sur une diversité de plates-formes, couvrant de ce fait les besoins de réutilisation et de partage, ainsi que d'amélioration de la qualité des processus d'analyses.

3.2 Questions de recherches

Afin de fournir un outil apportant une solution aux problèmes décrits dans les cas d'usages , il est important tout d'abord d'identifier les verrous scientifiques, pour ensuite évoquer les questions scientifiques émanant de ces verrous.

L'objectif de ce stage étant d'obtenir un formalisme décrivant un processus d'analyse de manière indépendante des plates-formes, la question à se poser est alors : "Comment décrire un processus d'analyse sans aucun lien avec les plates-formes d'analyse de traces ?".

Comme évoqué précédemment, un processus d'analyse est une suite d'opérations matérialisées par ce que nous appelons des "opérateurs". De fait, si l'on souhaite obtenir un processus d'analyse indépendant, il faut que ces opérateurs qui le constituent soit eux aussi indépendants des plates-formes d'analyses. Ainsi, comment décrire ces opérateurs de manière indépendante ? Et donc comment identifier les équivalences et divergences entre opérateurs d'une même plate-forme mais aussi de plates-formes différentes ?

Un indicateur étant le résultat, auquel est attaché des données supplémentaires, de la manipulation de traces effectuées par un processus d'analyse, comment réussir exprimer un indicateur de manière indépendante des plates-formes d'analyse de traces ?

La finalité étant de pouvoir utiliser concrètement un processus d'analyse décrit avec notre formalisme, comment alors procéder à son instanciation dans les différentes plates-formes disponibles ? Dans cette optique d'instanciation, comment parvenir à lier les données, les opérateurs et le processus d'analyse aux plates-formes pour que ce dernier soit utilisable ?

4 État de l'art

Nous proposons dans cette section de nous intéresser aux différentes plates-formes impliquées dans le projet HUBBLE ainsi qu'à quelques autres, et à la notion d'indicateur, ceci principalement afin de montrer les dissensions entre les plates-formes et pour bien comprendre le concept d'indicateur. Puis nous nous positionnerons sur le plan scientifique par rapport à l'existant.

4.1 Plates-formes impliquées dans HUBBLE

4.1.1 Système à base de traces modélisées. Suite à l'approche proposée par le projet MUSETTE [9], la notion de système à base de traces a été définie. Le principe d'un système à base de traces est de permettre et faciliter l'exploitation des traces d'interactions . Un système est considéré comme tel à partir du moment où il permet la collecte, le traitement des traces collectées (notamment par l'intermédiaire de fonctions de transformations) ainsi qu'un moyen de visualisation sur le système de gestion d'une base de traces [19].

Afin de consolider le sens véhiculé par une trace pour permettre sa manipulation comme une entité à fort sens sémantique, cette notion de système à base de traces est liée avec celle de modélisation de traces. Ainsi dans un système à base de traces modélisées, les interactions d'un acteur sur un système qui sont enregistrées sont considérées comme une suite d'éléments observés, appelé obsels, et régies par un modèle. Le modèle est donc assimilable au squelette sémantique de la trace -concrétisation de ce qui est implicitement attendu- et les obsels, qui sont des données temporellement situées et structurées par l'intermédiaire des modèles, apportent l'information. Une trace modélisée est, par là même, une représentation structurée d'obsels véhiculant un sens sémantique, permettant *a posteriori* son traitement. Settouti [10] et Laflaquière [19] ont formalisé ces systèmes à base de traces modélisées.

4.1.2 kTBS : kernel for Trace Based System. Le kTBS est un framework qui implémente de manière générique la notion de système à base de traces modélisées, évoquée ci-dessus. Il permet donc à l'utilisateur de manipuler les traces avec une abstraction plus élevée qu'à l'accoutumée (contrairement à des log files) par une représentation interne en *RDF*, favorisant de fait la manipulation sémantique des données. Ce framework est apporté par l'équipe *TWEAK* du *LIRIS*.

Déployé en tant que serveur et accessible par protocole HTTP, le kTBS permet la création et la gestion de bases de traces ainsi qu'un système de visualisation sur le contenu de la base.

Ce système de visualisation permet d'afficher le contenu sous plusieurs représentations, comme une représentation XML, textuelle ou bien encore HTML, offrant l'avantage de la rendre parcourable par navigateur *via* des hyperliens.

De plus, le kTBS implémente un système de requêtes qui va permettre à l'utilisateur de manipuler ces traces, par l'intermédiaire de méthodes de transformations présentes dans le système de transformation. Les traces peuvent être exprimées en différents langages comme Turtle ou JSON. On y retrouve des méthodes classiques de manipulations de données comme le filtre temporel des obsels d'une trace ou la fusion des données de deux traces, mais également des requêtes *SPARQL* permettant de faire des manipulations sur les traces beaucoup plus personnalisables.

Le kTBS peut aussi prendre en charge des outils externes pour manipuler des données, comme c'est le cas avec *DMT4SP* [20], qui permet de faire de la

détection de motifs à partir des obsels d'une trace. De plus, fort de sa position de serveur, il peut être interrogé et utilisé par des programmes externes comme c'est le cas pour Transmute, outil graphique de manipulation du kTBS, tout comme SamoTrace [21].

4.1.3 UnderTracks. UnderTracks [16] est une instanciation du paradigme d'analyse de données DOP8, tous deux proposés par le LIG. DOP8, pour Data, Operators, Processes (littéralement données, opérateurs, processus [d'analyse]), représente le cycle de création d'un processus d'analyse par un analyste au travers de huit actions, représentées par des verbes. Ces actions sont séparées selon qu'elles concernent le cycle des données ou celui des opérateur. L'étape cohésive entre ces deux cycles est l'analyse des résultats obtenus.

UnderTracks, qui implémente donc DOP8, est constitué de deux parties indépendantes qui interopèrent : UTP[8] et UTA[9]. UTP s'axe sur la collecte, la mise en forme et le partage des données à analyser dans UnderTracks, ainsi que le partage et la création de nouveaux opérateurs pour manipuler ces données. La partie UTP contribue de cette manière à une politique de partage forte. UTA quant à elle s'axe sur la phase d'analyse. Autrement dit, comment manipuler les données pour en extraire des pertinentes. La phase d'analyse est épaulée par une vue ergonomique issue d'Orange [22].

Cette plate-forme, à la différence du kTBS susmentionné, représente ses données sous formes de tables d'une base de données : clefs primaire, colonnes, lignes, etc. La *study* représente toutes les données de la trace. Cette terminologie propre à UnderTracks propose de structurer la trace suivant quatre tables.

- *Event log* : il s'agit d'événements temporalisés, impliquant généralement un utilisateur -le système enregistré pouvant lui même être un utilisateur- effectuant une action ;
- *User* : la table *User* liste tous les utilisateurs impliqués dans l'étude ;
- *Actions* : la table *Actions* liste toutes les actions distinctes qui sont survenues ainsi que leurs paramètres éventuels au cours de l'étude. ;
- *Context* : la table *Context* défini le contexte de l'action donnée.

Pour conclure, la manipulation des données contenues dans les tables se fait par application d'algorithmes, appelés opérateurs. Un opérateur prend en entrée des données, et produit une ou des sorties. Il ne s'agit cependant pas toujours de données exploitables par d'autres opérateurs, comme c'est le cas avec les opérateurs de visualisation capable de créer une visualisation pertinente pour l'analyste.

4.1.4 UTL : Usage Tracking Language. UTL est un méta-langage pour la représentation des traces et l'instrumentation de l'analyse, basé sur le modèle sémantique DGU[10], développé par le LIUM [23].

[8] UnderTracks Production
[9] UnderTracks Analysis
[10] Defining, Getting, Using

Le modèle sémantique DGU, qui implique développeurs, analystes et concepteurs, s'inscrit dans une intention d'abstraction des données issues des traces. Ceci afin de rompre leurs dépendances avec leur représentation initiale. Le but de ce modèle est de proposer un modèle générique de représentation des données d'observations, se situant de fait dans le cadre de la réingénierie pédagogique. Ce modèle générique est décomposé en trois parties, permettant de modéliser le besoin et le moyen d'observation, et de préciser l'utilité de la donnée d'observation.

Le méta-langage UTL a plusieurs objectifs que sont la description des différentes données utiles à l'analyse des traces dans une forme indépendante des traces d'origine, la capitalisation du savoir faire en analyse de traces ainsi qu'une structuration des données d'observations. Pour ce faire, il est séparé en trois parties distinctes : UTL Patron, UTL Trace et UTL Scénario [24].

UTL Patron a pour objectif de définir les données d'observations qui seront récoltées dans les traces et la manière de les représenter, leur objectifs, etc. Il définit aussi les *derived datum*, issues de la manipulation des données d'observations, étant soit des indicateurs soit des données intermédiaires aux calculs, ces dernières ne présentant pas grand intérêt pédagogiquement parlant. Pour représenter toutes ces données, UTL Patron se base sur le modèle DGU.

UTL Trace se charge de faire la correspondance entre les traces directement collectées du système d'apprentissage et les données d'observations attendues, définies dans le modèle DGU. Cette partie permet de couvrir un vaste panel de traces venant de clivages différents.

UTL Scénario, quant à lui, permet de lier la description des indicateurs obtenus avec le scénario pédagogique du système d'apprentissage.

DCL4UTL permet d'automatiser les calculs d'indicateurs décrits dans la partie UTL Patron, en accord avec le modèle DGU, par l'intermédiaire d'un langage de programmation proche du XQuery, ce qui en fait un outil plus destiné au développeur qu'à l'analyste. Les données d'observations des systèmes d'apprentissage étant ramenées sous un formalisme identique, on peut penser que les calculs décris *via* DLC4UTL seront réutilisables même si l'environnement d'apprentissage est différent.

4.2 Introduction à d'autres plates-formes

Il existe de nombreuses plates-formes hors du projet HUBBLE, comme c'est le cas de PSLC Datashop [11] par exemple. Il s'agit d'un projet international visant à améliorer la qualité des recherches dans le domaine de l'EDM en proposant un dépôt de traces d'interactions standardisées entre des apprenants et des systèmes informatiques pour l'apprentissage multi-domaines, tentant de cette manière de répondre à l'une des problématiques majeures qu'est la diversité des représentations des traces. Pour ce faire, Datashop oblige les données à respecter trois principes qui sont la précision sur le plan sémantique, la longitudinalité -autrement dit s'étendant sur une période assez longue- et la vastitude des données, principalement pour alimenter correctement les algorithmes. Datashop propose aussi des outils intégrés de data mining et de visualisation pour manipuler ces données dans le cadre de l'EDM.

Cependant, PSLC Datashop est une solution en lien très étroit avec l'EDM. Or, en dehors de la considération des traces et des EIAH, il existe une grande variété de plates-formes destinées à la fouille de données utilisées par les analystes, comme c'est le cas de RapidMiner (anciennement YALE) [25] ou encore Weka [18]. Elles proposent des algorithmes performants qui peuvent être pertinents pour certaines analyses de données, par exemple la classification, même si elles ne répondent pas forcément aux besoins des disciplines liées à l'apprentissage médié par l'informatique.

4.3 Indicateur

Une des productions possibles des plates-formes d'analyses que nous venons de voir au travers de processus d'analyse est ce qu'on appelle un indicateur. Dans le cadre de ce stage, nous nous sommes restreints à cette notion sans prendre en compte d'autres productions comme les modèles par exemple.

Un indicateur est "*une variable au sens mathématique à laquelle est attribuée une série de caractéristiques*" [26] pour la rendre signifiante à son destinataire, afin de lui apporter de l'information sur une situation donnée. La notion de variables mathématiques définie dans la citation implique que ces variables prennent une valeur dans un domaine de définition supporté par la chaîne d'analyse (valeur numérique, *timestamp*, chaîne de caractères...). La valeur en question possède de plus un statut qui se veut soit brut (sans unité), calibré (qui existe dans une échelle) ou interprété (l'information est alors issue du jugement humain) [1].

Un indicateur peut être généré en même temps que la situation à analyser se déroule ou bien *a posteriori*, en fonction du besoin. Dans tous les cas, il est destiné à apporter une information claire, dans un contexte précis. Autrement dit, son objectif se doit d'être clairement défini. De plus, un indicateur est parfois dépendant du domaine, par exemple parce qu'il n'existe qu'en chimie. La problématique de la représentation même d'un indicateur se fait ressentir. Le projet international ICALT [1] tente de définir un modèle type d'indicateur ainsi qu'une typologie (cognitif, social ou affectif) pour répondre à la nécessité d'avoir des informations pertinentes et une définition correcte de leur objectif.

Cependant, ces travaux pour modéliser les indicateurs ne se sont que très peu attardés sur les méthodes d'analyses pour les obtenir. Si bien que ces indicateurs sont au final dépendant des outils d'apprentissages analysés, des données manipulées, de la manière de procéder de l'analyste et de la plate-forme d'analyse utilisée. Ce qui a pour conséquence de drastiquement limiter leur réutilisation (intra et extra outils d'apprentissage).

La nécessité de capitalisation se faisant de plus en plus pesante, du fait de la multiplication des outils d'apprentissages et notamment ceux massifs, on trouve dans la littérature des tentatives pour rendre générique la méthode d'analyse d'un indicateur, comme dans [27] qui propose, uniquement pour le kTBS, de définir un indicateur comme un ensemble de règles successives à respecter. Il devient alors possible de partager cette méthode d'obtention avec d'autres utilisateurs du kTBS pour les mêmes besoins. Les travaux de Diagne [28] vont un plus loin. Elle définit un patron d'indicateur découpé en trois *phases* que sont la

phase de sélection des données, la phase d'analyse et la phase de visualisation, le tout utilisant le langage de programmation JAVA pour leur représentation.

4.4 Positionnement

Ce que nous cherchons à accomplir dans le cadre de ce stage est d'apporter un élément de réponse à cette problématique du partage des processus d'analyses de traces, conséquence directe de la grande diversité des plates-formes d'analyses, des approches et des besoins. S. Iksal fait remarquer que : *"Actuellement, la communauté dispose de nombreux outils [...], malheureusement sans aucune fédération des projets. Même si ces derniers n'ont pas de liens directs, il pourrait être utile de les recenser dans un premier temps, de les rendre accessibles au travers de services web par exemple"* [24].

Bien qu'il existe des travaux allant dans se sens, ils ne sont que très peu voire jamais adaptés aux analystes, comme c'est le cas de DCL4UTL, permettant d'automatiser les calculs dans UTL , dont l'utilisation est très proche d'un langage de programmation.

De plus, l'hétérogénéité des représentations des traces crée le problème du partage. Les outils issus des travaux de recherches présentés jusqu'ici, pour résoudre cette difficulté, importent les traces dans une nouvelle représentation afin de les manipuler. Cependant, cela génère une perte d'efficacité importante puisque l'utilisation de plates-formes spécialisées ne peut plus être effectuée. Par exemple utiliser des algorithmes de Datashop dans UTL ne semble pas envisageable.

Nous cherchons donc à créer un outil fédérateur des différentes plates-formes d'analyses existantes utilisées par les analystes en les plaçant au premier plan lors de la création d'un processus, comparativement aux différents travaux de recherche examinés. Nous cherchons aussi à exprimer de manière indépendante des plates-formes d'analyse de traces un processus d'analyse, débouchant potentiellement sur un indicateur, et être capable d'assister l'analyste dans un choix pertinent des plates-formes au regard dudit processus. Ainsi, cet outil agira comme une surcouche unificatrice, ne cherchant pas à remplacer les plates-formes d'analyse mais plutôt à abstraire et combiner leurs différentes possibilités.

5 Proposition théorique

Après l'explication de notre approche dans la section 5.1, nous présenterons, via les sections 5.3 et 5.4, les fondements théoriques de notre outil. Mais avant cela, section 5.2, nous traiterons de la démarche d'obtention du formalisme générique des différents opérateurs. Enfin pour finir, la section 5.5 décrit la manière théorique d'instancier un processus générique d'analyse.

5.1 Approche

Nous avons pris le parti d'adopter une approche différente des travaux précités, approche que l'on peut imager en *top-bottom* partant de la description générique

d'un processus d'analyse pour, après, l'instancier directement dans les plates-formes de gestion de traces existantes. Cela implique donc que notre solution n'effectue pas elle même l'analyse des traces mais se repose sur les plates-formes d'analyse de traces existantes pour faire les calculs : elle fait office de "cahier de brouillon" où l'analyste doit pouvoir s'exprimer librement sur la création de processus d'analyse et d'indicateurs.

Pour matérialiser cette indépendance, il est impératif que l'analyste ne soit pas confronté à des spécificités issues des plates-formes. Pour ce faire, notre solution propose sa propre modélisation des outils d'analyses utilisables, ainsi qu'un moyen de représenter les types des données qui seront utilisées dans un processus d'analyse. C'est donc à notre solution de gérer les spécificités des plates-formes disponibles quand l'analyste va faire le lien entre ce qu'il a exprimé pour décrire son processus d'analyse et l'instanciation dans une plate-forme d'analyse de traces disponible.

On peut diviser notre approche en deux parties distinctes, comme le montre la Fig.1. La partie supérieure, qualifiée d'indépendante, montre comment un analyste va procéder pour obtenir un processus d'analyse générique à partir d'opérateurs génériques. Elle montre aussi qu'on attache le processus d'analyse à un indicateur possédant des méta-données. La partie inférieure, qualifiée de dépendante, montre comment un utilisateur -il ne s'agit plus d'une tache réservée aux analystes- va faire le lien avec les plates-formes d'analyses existantes.

L'enjeu de la partie indépendante est de proposer un moyen de définir et de décrire les processus d'analyse de traces de manière indépendante des processus d'analyse, de telle sorte qu'ils puissent être compréhensibles, facilement partageables et réutilisables. Pour ce faire, nous proposons des représentations spécifiques décrites dans la section 5.3. De plus, nous avons identifié trois étapes jugées primordiales dans notre approche, qui sont la définition des types de données à utiliser, les opérations à appliquer sur ces types de données pour créer le processus et l'utilisation du processus d'analyse résultant pour la création d'indicateurs.

Cependant, l'expression du processus d'analyse n'implique pas sa résolution. Pour le résoudre, il faut l'importer dans une plate-forme d'analyse de traces capable de l'accueillir. C'est à la partie dépendante qu'incombe la tâche d'importation. Elle demande à l'utilisateur, qui n'est plus forcément un analyste, de faire le lien entre les types des données décrits et ceux de la plate-forme d'analyse visée. La section 5.4 traite ce sujet.

5.2 Démarche d'évaluation de l'existant

Pour obtenir les représentations évoquées dans la section 5.3, il a fallu préalablement s'intéresser et analyser les possibilités des plates-formes impliquées dans le stage, à savoir UnderTracks, kTBS et UTL. En effet, définir les possibilités d'une plate-forme, ses limites et faire la correspondance, si possible, des opérations avec les autres n'est pas trivial. Nous avons donc utilisé une démarche empirique et incrémentale.

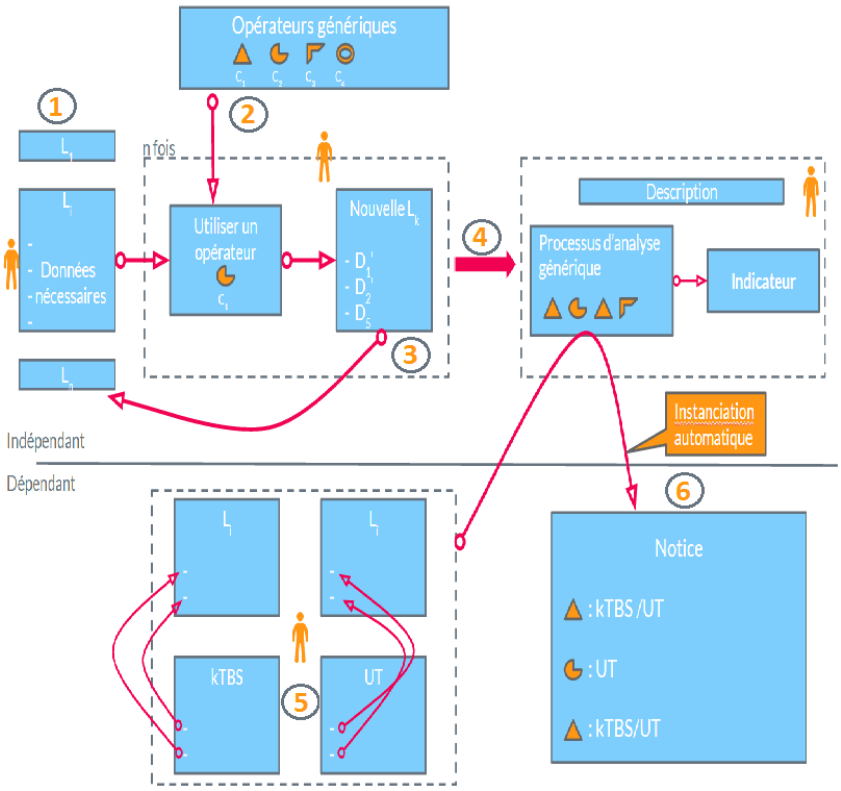


Fig. 1. Schématisation de l'approche proposée. La partie supérieure représente l'expression d'un processus d'analyse de manière indépendante des plates-formes. La partie inférieure schématise la manière d'instancier ledit processus.

Nous avons tout d'abord étudié le contexte d'utilisation de ces différentes plates-formes pour cerner les différents objectifs qu'elles tentaient respectivement de résoudre. Une fois ces objectifs définis, une première phase d'expérimentation a été réalisée afin de comprendre le comportement des plates-formes (sauf pour UTL qui ne possède pas encore de prototype utilisable).

Suite à cela, nous avons identifié une première fois, de manière non exhaustive, les différents opérateurs existant dans les plates-formes. Pour ce faire, une fiche expérimentale a été créée pour tenter de formaliser un opérateur (voir annexe F).

Ensuite, dans une tentative de formalisation générale, nous avons analysé chaque opérateur de chacune des plates-formes pour essayer de les regrouper. Cette association s'est faite en se basant sur leur objectif, leurs critères de données et ce qu'ils généraient. Nous avons réussi à obtenir un formalisme satisfaisant, que nous avons enrichi ensuite *via* une nouvelle itération de la démarche.

De fait, les opérateurs qui sont proposés dans l'implémentation, ainsi que le méta-modèle qu'ils respectent, sont issus de cette phase d'analyse et de recherche qui nous a permis de déboucher sur une identification des dissensions et un regroupement pertinent. L'annexe E présente l'une de ces fiches et la section suivante présente le méta-modèle résultant.

5.3 Modèle

Dans cette section, nous allons décrire chacun des éléments ayant un rôle dans la définition d'un processus d'analyse générique.

5.3.1 Donnée nécessaire. L'analyste doit posséder des données sur lesquelles travailler. Cependant, dans notre démarche nous ne voulons pas importer les traces d'une plate-forme d'analyse de traces puisque cela impliquerait de devoir définir une représentation convenant à toutes les plates-formes existante, aussi bien intra qu'extra HUBBLE, ce qui n'est pas, dans le cadre du stage, envisageable. De plus, importer les traces d'une plate-forme obligerait à les traduire pour toutes les autres, ce qui n'est pas, à l'heure actuelle, envisageable.

Nous proposons de représenter non pas les données contenues dans les traces, mais l'archétype de ces données (le "temps", un "score") et qui vont être impliqués dans notre processus. Cette notion d'archétype jungien est appelée *données nécessaires* : nécessaires dans le sens que, sans au moins l'une d'elles, le processus d'analyse ne peut être effectué entièrement. On peut ainsi voir cette représentation comme un contrat d'utilisation, indiquant les prérequis du processus d'analyse, ce qui renforce l'aspect capitalisable recherché. En effet, cela à l'avantage d'être parlant à l'analyste, par exemple *"pour utiliser ce processus d'analyse, vous devez avoir le nom des apprenants et leur score"*.

Pour appuyer cette représentation, nous avons défini un méta-modèle qui oblige les données à posséder un nom et une référence à une liste, de telle sorte qu'elles soient identifiables et interprétables par l'analyste. En plus de posséder un type.

5.3.2 Liste. Bien qu'un processus d'analyse manipule les données contenues dans une (ou plusieurs) trace(s), la trace en elle-même est un élément important tant sur un plan sémantique que sur un plan plus structurel.

Nous avons voulu garder cette idée de conteneur de données à travers la notion de *listes*. La liste est donc un moyen de représenter ce que doit contenir impérativement la trace, dans la plate-forme d'analyse cible. Cependant, nous n'avons pas conservé le terme de trace. Cela pouvait potentiellement être une source de confusion pour l'analyste du fait que ce ne sont plus des données à granularité fine dont il est question, mais d'archétypes de données (appelées données nécessaires).

Ainsi une liste est un élément relativement simple, qui possède un **IDBloc** lui permettant d'être identifié par le système ainsi qu'une liste non nulle de données nécessaires qu'elle possède. Impliquant de fait que ces données nécessaires pointent vers ladite liste.

5.3.3 Opérateur. Un *opérateur* matérialise une opération à appliquer sur des données. Comme précisé, nous ne recherchons pas à implémenter des méthodes de calcul directement dans notre outil, mais nous cherchons à illustrer l'utilisation d'une opération sur des données nécessaires, ce qui est censé représenter la réflexion de l'analyste.

Comme on l'a vu, un opérateur prend des données en entrée et peut générer des données en sortie. Nous proposons donc une représentation pour décrire l'opérateur dans son ensemble et répondre à la problématique d'indépendance, sans pour autant le spécialiser. Ainsi, pour ajouter un opérateur à notre outil, il faut respecter le méta-modèle suivant de la Fig.2.

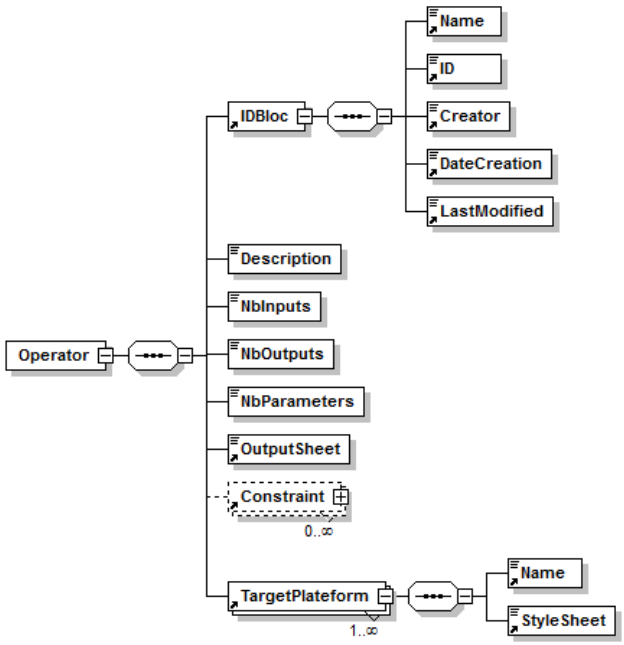


Fig. 2. Méta-modèle, en XML Schéma, d'un opérateur tel qu'exprimé dans notre approche

L'**IDBloc** contient des informations permettant de référencer l'opérateur dans notre outil, dont son nom. La **Description** permet d'apporter de l'information supplémentaire comme l'objectif ou le fonctionnement. **NbInputs** indique le nombre de données nécessaires attendu par l'opérateur. De la même manière, **NbParameters** indique le nombre de paramètres qui peuvent être réglés (la notion de paramètre est elle aussi un concept propre à notre approche) et **NbOutputs**, le nombre de listes que l'opération va produire. Les **Constraints** représentent des contraintes que doit respecter l'opérateur pour

être considéré comme correctement configuré (mais qui ne sont pas implémentées dans le cadre du stage).

Pour définir comment va se comporter un opérateur, en sachant que celui-ci ne manipule que des données abstraites contenues dans des listes, on a recourt à une **OutputSheet**. Cette feuille de sortie (cf. annexe A) , couvre dans le cadre théorique tous les besoins des opérateurs étudiés jusqu'à présent. Elle permet de créer de nouveaux archétypes de données, d'en supprimer, de copier tout ceux d'une liste, etc. Par exemple pour un opérateur de fusion entre deux listes, on sait que le résultat attendu va être une nouvelle liste contenant les données nécessaires de la première et celles de la deuxième.

Le dernier point important est **TargetPlatform**. Il permet à notre outil de savoir comment instancier l'opérateur dans la plate-forme **Name** considérée en fonction d'une **QuerySheet**. Cette feuille de requête est un langage "enrobant", qui se greffe sur la manière standard de procéder dans un système, pour lui donner de la variabilité en fonction des données nécessaires impliquées.

5.3.4 Processus d'analyse générique. Un processus d'analyse est l'application logique d'une suite d'opérations, pour répondre à un problème donné. Ainsi, un processus d'analyse est constitué d'opérateurs appliqués dans un ordre précis. Cependant, pour qu'un processus réponde au critère d'indépendance aux plates-formes, il faut que tous les opérateurs qui le constituent le soient également. Nous proposons donc le méta-modèle référencé par la Fig.3, qui permet de répondre au critère d'indépendance d'un processus, en plus de permettre sa capitalisation au sein de notre outil.

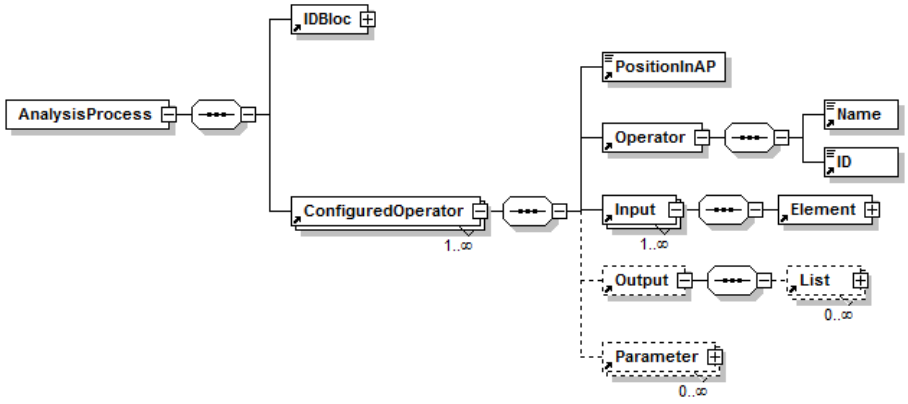


Fig. 3. Méta-modèle d'un processus d'analyse, en XML Schéma, tel qu'exprimé dans notre approche (vue simplifiée).

À l'instar d'un opérateur, un processus d'analyse possède un **IDBloc** permettant son identification, principalement à des fins de stockage. Il est aussi

constitué d'un ou plusieurs **ConfiguredOperator**. Un **ConfiguredOperator (CO)** peut être assimilé à une étape dans un processus d'analyse. Afin de savoir comment agencer les différentes étapes, nous avons recours à **PositionAP**, un entier naturel, qui indique la position du **CO** dans le processus. Le champ **Operator** indique quel opérateur utiliser à cette étape.

Ensuite, l'opérateur choisi par l'analyste va être "spécialisé". En effet, **Input** représente les données nécessaires fournies en entrée à l'opérateur. Le nombre de données dépend de **NBInputs** de l'opérateur. **Parameter** représente les paramètres qui vont être appliqués à l'opérateur en question. Pour terminer, **Outputs** représente la production en sortie de l'opérateur. La quantité de listes produites est liée à **NbOutputs**. Ces listes produites contiennent des données nécessaires issues des entrées (basées , rappelons le, sur l'**OutputSheet**).

Bien entendu, du fait de l'abstraction recherchée et de la nécessité d'interprétation de l'analyste sur les données et les opérateurs, il n'est pas exclu de créer des situations aberrantes (un filtre temporel sur un score par exemple). Cependant, vérifier la cohérence n'est pas la vocation de ce méta-modèle qui définit un processus d'analyse pour le rendre capitalisable. Nous considérons que la capacité d'interprétation de l'analyste des archétypes de données et des opérateurs lui permet de prendre des décisions cohérentes.

5.3.5 Indicateur. L'indicateur est un des résultats possibles d'un processus d'analyse de traces d'apprentissage. Dans le cadre du stage, nous nous concentrerons sur les indicateurs. Nous avons donc proposé là encore une représentation de l'indicateur pour permettre de le partager et de le réutiliser. La Fig.4 présente comment nous envisageons ce concept.

Nous proposons donc de définir un indicateur comme possédant un lien vers un ou plusieurs processus d'analyse, ce qui permet par exemple de comparer la performance de différents processus d'analyse, et tout un ensemble de métadonnées qui permettent de le définir. Cette définition reprend les résultats des projets kaleidoscope et UTL permettant de décrire un indicateur. Ainsi, on considère la valeur qui sera produite une fois que le processus d'analyse sera instancié et réalisé. On enrichit cette valeur *via* trois blocs. L'**IDBloc**, une fois encore, à des fins d'identification contenant le nom ou encore la date de création. Le bloc **contextuel** définit le contexte de l'indicateur, dont les acteurs ciblés, son objectif pédagogique, etc. Et enfin le bloc **descriptif** permet d'apporter de l'information supplémentaire, avec la possibilité d'adopter une dimension sociale en proposant un système d'avis et de notation de qualité de l'indicateur, pour apporter un peu plus de pertinence.

Cependant, cette modélisation est amenée à évoluer. En effet, suite à la première assemblée générale du projet HUBBLE, il a été noté que les besoins attendus en terme de description d'indicateurs résultent principalement d'autres lots du projet. Il faut donc attendre les résultats de leurs travaux pour modifier en conséquence cette représentation. Par exemple, l'avancement des travaux dans la dimension législative va impliquer l'incorporation d'une dimension juridique plus poussée, point qui n'est pas considéré dans ICALT ou UTL par exemple.

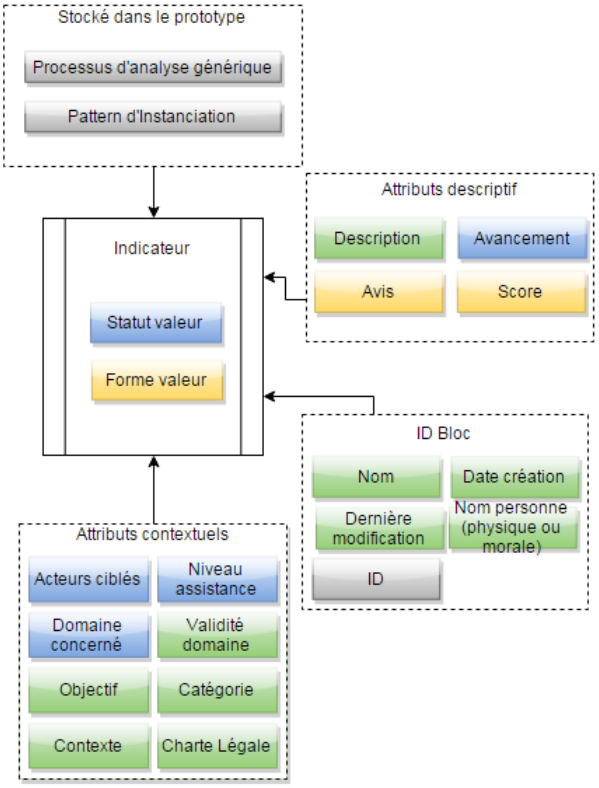


Fig. 4. Méta-modèle d'un indicateur dans notre approche. Représentation "UML" pour identifier facilement les différents états des méta-données. Les blocs gris sont les méta-données systèmes. Les blocs verts les méta-données qui seront librement remplies par l'utilisateur. Les bleus sont issus d'un choix parmi un ensemble de valeurs. Les blocs jaunes sont facultatifs.

5.4 Fonctionnement théorique

Cette section va permettre d'obtenir une vue d'ensemble du fonctionnement théorique de la partie indépendante en suivant l'exemple du processus d'analyse donnant la valeur de l'indicateur *pourcentage de visionnage d'une vidéo d'une promotion d'étudiant du 10/4 au 17/4*. La Fig.6 représente l'implémentation concrète d'un tel processus d'analyse.

Tout d'abord, l'analyste doit isoler le type des données qui vont être capitales pour réaliser le processus. Dans notre exemple, il s'agit du temps et d'une donnée indiquant si oui ou non l'apprenant a vu la vidéo, qu'il va placer dans une même liste. Dans la Fig.1, cela correspond à l'identifiant *1*.

Ensuite, l'analyste doit réfléchir à comment procéder pour obtenir la valeur recherchée, et donc comment et sur quelles données nécessaires utiliser les opéra-

teurs génériques. En reprenant notre exemple du pourcentage de visionnage d'une vidéo, les étapes du processus d'analyse seraient :

- Isoler premièrement la semaine recherchée ;
- Filtrer les apprenants ayant vu la vidéo ;
- Compter les apprenants ayant vu la vidéo ;
- Compter les apprenants totaux ;
- Effectuer finalement une division du compte d'apprenants ayant vu la vidéo par le compte d'apprenants totaux.

Sur le schéma Fig.1, pour une étape, l'analyste choisit un opérateur, identifié par *2*, qu'il va appliquer sur une ou plusieurs données, définies dans *1*. L'opérateur va produire de nouvelles listes utilisables par la suite, c'est ce que montre l'identifiant *3*. Puis en effectuant toutes les opérations, on définit notre processus d'analyse générique, comme le montre *4*.

L'analyste possède donc un processus d'analyse générique qui explique la démarche pour obtenir la valeur de l'indicateur et qu'il peut d'ores et déjà partager. Cependant, s'il désire partager l'indicateur, alors il doit définir toutes ses métadonnées et lui associer le processus d'analyse. Il devient donc possible de partager directement l'indicateur.

5.5 Processus d'instanciation

Afin de pouvoir utiliser le processus d'analyse décrit de manière indépendante -puisque, rappelons-le, notre outil n'a pas vocation à calculer lui même-, il faut procéder à son instanciation dans la ou les plates-formes cible(s), c'est-à-dire le décrire selon le formalisme de la plate-forme et savoir quelles données utiliser. Pour ce faire, nous avons proposé un processus d'instanciation, divisé en deux temps, que nous allons décrire ci-après.

5.5.1 La résolution des liens. Comme vu dans la section 5.3, nous représentons des archétypes de données et pas directement des données, en plus d'utiliser la sémantique d'un opérateur pour représenter son opération. La transcription de notre outil vers une plate-forme est ainsi non triviale puisqu'il est impossible de faire une correspondance directe et automatisée, en l'état, des données impliquées et à utiliser. Il faut donc faire intervenir un utilisateur qui aura la charge de lier les données nécessaires avec leur emplacement dans la ou les plates-formes désirées, comme le montre la Fig.1 à l'identifiant *5*.

La Fig.5 représente le méta-modèle d'instanciation pour un processus d'analyse utilisé dans notre outil, pour répondre à cette nécessité. Une **Instanciation**, représentée par un nom et un **ID** pour l'identification, concerne un seul processus d'analyse *via* **IDAnalysisProcess**. Cependant, un même processus d'analyse peut être instancié de différentes manières (plusieurs **Instanciation** référençant le même processus). On le conçoit bien puisque dans l'optique d'une capitalisation, les données impliquées seront amenées à être différentes en fonction des expérimentations et des plates-formes de stockage.

InstanciedEntities va définir comment représenter la localisation des données et des listes impliquées, respectivement par **UserMappedElement** et **UserMappedList**. Ces deux nœuds identifient les éléments concernés grâce à un **ID** et au nom de la plate-forme concernée (**PlateformName**).

Ensuite, pour chaque donnée nécessaire ou liste, en fonction de chacune des plates-formes désirées, est sélectionné le **MAPPING** correspondant. Ce **MAPPING** contient toutes les informations nécessaires pour que le système puisse opérer dans les plates-formes. Ce sont ces informations qui sont fournies par l'utilisateur et qui définissent les liens. Par exemple, l'utilisateur va indiquer pour le kTBS l'*URI* de la donnée nécessaire "temps" se trouvant dans le modèle de la trace et pour UnderTracks la colonne qui contient le temps.

Nous avons conçu ce principe de telle sorte qu'il soit facilement extensible à de nouvelles plates-formes d'analyses, répondant à l'intention d'HUBBLE d'impliquer un maximum de partenaires. En effet, il suffit d'indiquer dans les données et les listes **MAPPING** les nouveaux éléments permettant de les localiser, et de définir le comportement dans la partie logicielle de l'opérateur, notamment avec la **QuerySheet**, pour que notre outil puisse gérer de nouvelles plates-formes.

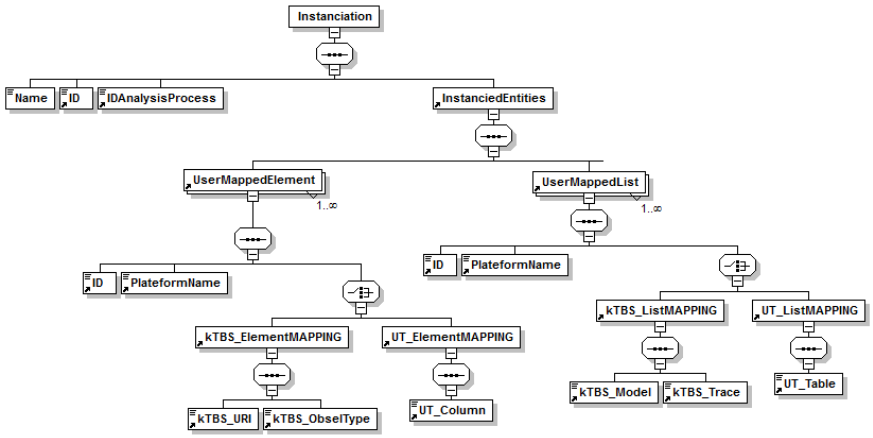


Fig. 5. Méta-modèle de l'instanciation, en XML Schéma, tel qu'exprimé dans notre approche.

5.5.2 Transcription automatique. À partir des informations obtenues dans la résolution des liens, le système est capable théoriquement d'automatiser la transcription du processus dans la ou les plates-formes désirées. Cette transcription sera sous la forme d'une "notice" d'instanciation que l'analyste devra suivre pour recréer le processus d'analyse dans la ou les plates-formes sélectionnées, la

notice indiquant, pour chaque opération, quel opérateur utiliser sur quelle(s) catégorie(s) de données. Sur la Fig.1, l'identifiant *6* schématise ladite notice.

Notre approche se base sur une logique applicative dépendante de la plate-forme concernée, assimilable à un *plug-in* utilisant les informations à sa disposition que sont les **MAPPING**, l'**OutputSheet** et la **QuerySheet**. Ce point de vue de *plug-in* permet d'étendre facilement l'instanciation à de nouvelles plates-formes, rejoignant l'aspect extensible évoqué précédemment.

Pour chaque **ConfiguredOperator** ordonné en suivant **PositionInAP**, notre outil va procéder à la résolution des liens des entrées et calculer la nouvelle représentation des types de données en sortie, adaptée à la plate-forme. Le résultat du calcul dépend de la **QuerySheet** (cf. annexe B) ainsi que de l'**OutputSheet**. Chaque résultat peut bien entendu être utilisé en entrée des étapes suivantes, en accord avec le modèle du **CO**.

6 Implémentation

Dans le cadre de ce stage, nous avons réalisé une implémentation de l'approche proposée dans la section 5.1 ainsi que des différents méta-modèles impliqués, présentés dans la section 5.3. Au vu des contraintes temporelles, nous avons cependant dû faire des choix de développement pour notre prototype. Nous avons décidé de répondre à la problématique d'expression et de capitalisation d'un processus d'analyse et de laisser en suspens la phase d'instanciation, puisque cela n'entrave pas l'analyste dans l'expression d'un processus. De plus, s'axer sur cette partie permet des expérimentations avec les analystes, en plus de préparer tout de même l'instanciation, grâce notamment aux travaux concernant les opérateurs indépendants.

L'outil développé est pensé pour être une plate-forme web basée sur une architecture client-serveur. La base de données est basée sur un système de gestion de bases de données relationnelles MySQL. Le serveur, programmé en PHP avec une couche DAO[11], est en charge de gérer toutes les requêtes du client, comme l'enregistrement de nouveaux opérateurs ou de nouveaux processus d'analyses, mise à jour des données, etc. Cependant, ces deux parties n'étant pas indispensable pour répondre à la problématique de notre stage, nous avons décidé de les laisser en suspens pour se concentrer pleinement à la partie client qui vise à solutionner notre problématique.

Côté client donc, nous avons développé l'outil pour qu'il soit utilisable avec un navigateur internet et qu'il déporte le plus de calculs possibles du serveur. Nous utilisons donc les technologies HTML 5 et CSS 3, ainsi que Javascript et JQuery pour programmer le comportement de l'outil. Nous nous sommes concentrés sur la partie description générique d'un processus d'analyse puisque c'est cette partie qui répond à la problématique du stage, la partie d'instanciation étant, comme on l'a vu, planifiée pour un futur proche. La Fig.6 présente l'interface de notre outil et montre le résultat d'une définition d'un processus d'analyse. Notre

[11] Data Access Object

prototype permet donc de créer un processus d'analyse, et apporte un début de notice grâce à la liste des plates-formes que l'outil tient à jour.

La première zone qui contient **Listes Initiales**, **Listes Calculées** et **Opérateurs** est la zone de ressources de l'analyste. **Liste Initiales** est une fenêtre qui permet à l'analyste de décrire les listes et ses données nécessaires pour son processus d'analyse. **Listes Calculées** répond à la problématique de réutilisation des sorties d'un opérateur. Cette fenêtre fait le listing de toutes les listes générées aux étapes antérieures en fonction de l'étape courante. On peut de fait récupérer une liste altérée par un filtre temporel sur sa donnée nécessaire *temps*. Enfin, **Opérateurs** est une fenêtre qui contient tous les opérateurs, décrits dans notre outil, que l'analyste peut sélectionner. Avec cette zone, l'analyste est donc en mesure de configurer les étapes de son processus.

La deuxième zone, qui se situe en dessous à gauche, est la zone d'étapes. Elle permet de créer de nouvelles étapes pour le processus d'analyse, ainsi que d'en supprimer. Pour une étape sélectionnée, on a accès aux entrées, à son opérateur et à ses sorties. La possibilité est offerte à l'analyste de modifier le nom des listes de sorties pour plus de commodité. Mais surtout, cette zone permet de configurer l'opérateur grâce aux paramètres accessibles en cliquant sur le bouton le représentant. Juste en dessous, l'outil nous indique quelle(s) plate(s)-forme(s) d'analyse de traces est capable de supporter l'entièreté du processus d'analyse décrit, en se basant sur les informations contenues dans les modèles d'opérateurs régis par le méta-modèle d'opérateur de la Fig.2.

Ensuite, la dernière zone est une représentation graphique du processus d'analyse, indiquant les dépendances de chaque étape. Cette zone permet de voir quelles plates-formes peut supporter les différentes étapes, avec la possibilité de filtrer en fonction d'une plate-forme spécifique.

Pour finir, l'analyste, afin de valider son processus d'analyse, va sélectionner les étapes à conserver au moyen d'une liste cliquable, le système sélectionnant toutes les dépendances d'étapes requises pour une instanciation correcte. Puis, il décrit l'indicateur résultant de ce processus d'analyse.

7 Expérimentation

La phase d'expérimentation va permettre de valider s'il est possible d'exprimer convenablement des processus d'analyse avec l'outil, s'il convient aux scénarios d'usages qui vont être créés dans HUBBLE et surtout permettre de diagnostiquer si l'outil est pleinement adapté aux analystes. Cependant, s'inscrivant dans un projet à long terme impliquant un grand nombre de partenaires et devant prendre en compte l'implémentation du prototype de test, les premières phases d'expérimentations avec des analystes ne débutent qu'à partir du 16 Juin avec le LIG.

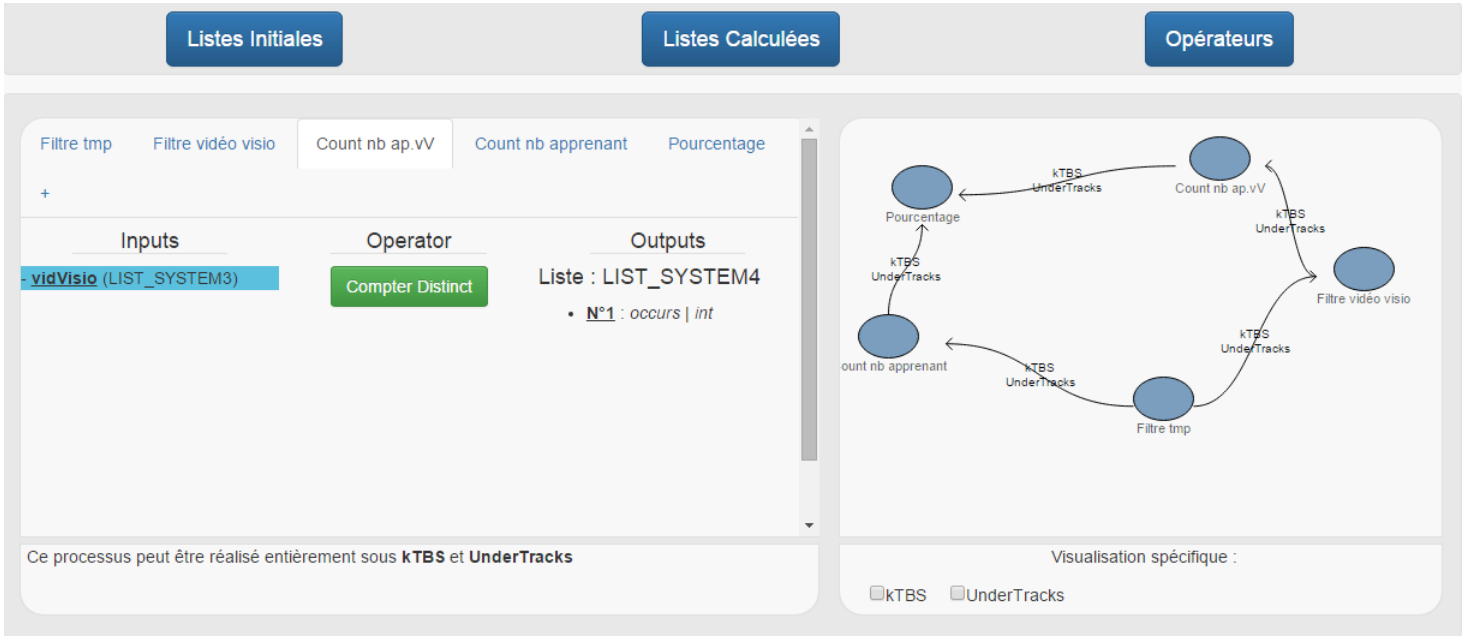


Fig. 6. Capture d'écran de l'outil développé implémentant l'approche théorique. Elle présente le processus d'analyse du calcul du pourcentage de visionnage d'une vidéo.

Les différents retours vont permettre d'améliorer l'aspect théorique et/ou concret pour proposer, courant Septembre, une nouvelle phase d'expérimentation avec de nouveaux analystes, cette fois-ci de l'IFé[12]. Le protocole prévu pour le 16 Juin est partiellement disponible en annexe G. Concrètement, nous procédons à la validation de notre approche en trois phase : une phase théorique, une phase d'expérimentation en interne et enfin une phase de validation technique.

La phase d'évaluation théorique avait pour objectif de mettre à l'épreuve les propositions présentées dans la section 5 et de valider leur pertinence, montrant qu'il est possible d'exprimer des processus d'analyses de manière indépendante des plates-formes . Pour ce faire, nous avons vérifié si l'expression des trois processus d'indicateurs que sont le pourcentage de visionnage d'une vidéo, la pertinence d'une question de QCM et la classification des groupes d'apprenants impliqués dans un environnement d'e-learning était réalisable avec les mécanismes et méta-modèles proposées. Le résultat fut positif : il est possible d'exprimer des processus d'analyse -simples- avec notre approche.

La phase d'expérimentation en interne va permettre de tester concrètement notre approche en la confrontant à des situations réelles, notamment avec des traces d'une promotion de Licence 1 Informatique de Lyon. Nous avons commencé à exprimer des analyses qu'il faut tester dans le prototype proposé. Comme cela, nous serons en mesure de voir si notre approche est capable de s'adapter à des contextes différents de ceux théoriques.

Enfin, la phase technique va nous permettre de valider la faisabilité de l'approche en ayant recours à des analystes de métier et de savoir s'ils arrivent à manipuler les concepts proposés pour s'exprimer. Pour faire cette validation, nous allons procéder en deux étapes. La première étape qui aura donc lieu le 16 Juin à Grenoble permettra de vérifier si les analystes interrogés arriveront à comprendre les concepts et créer des processus d'analyses simples, principalement pour s'assurer que l'aspect informatique a été écarté de la réflexion et que l'outil sied aux analystes. Puis, la deuxième partie prévue pour Septembre, permettra de proposer une nouvelle version en ayant pris en compte les précédentes remarques. Son objectif sera principalement d'essayer de diagnostiquer les avantages et les limites de l'approche pour les analystes qui se serviront de l'outil.

8 Limites et discussion

Ce travail de recherche a permis d'apporter des résultats théoriques sur le plan de la capitalisation des processus d'analyse de traces d'apprentissage, ainsi qu'un prototype. Néanmoins, il subsiste certaines questions et certains points qui peuvent constituer des limites et des difficultés à notre démarche.

Tout d'abord, on peut s'interroger sur le choix de l'approche *"top-bottom"* et la sensation d'impalpabilité des données et des calculs. Est-ce que cela ne représente pas une vraie limite à notre approche ? Suite à plusieurs discussions, il s'est avéré en effet que les analystes, dans leurs phases de développement, sont

[12] http://s2hep.univ-lyon1.fr/

amenés à avoir énormément recours aux feedbacks de leurs actions, par exemple pour vérifier si un classifieur est bien configuré, une expression bien écrite, etc. Or, notre plate-forme, ne permettant pas de calculer directement les résultats, ne propose pas ce genre de feedback. Du coup, l'obtention d'un processus d'analyse indépendant grâce à notre plate-forme devient potentiellement une démarche itérative où l'analyste décrit, instancie, teste et modifie.

Directement liée à cette approche qui vise à obtenir une très forte couverture des plates-formes d'analyses, la représentation même des données -données nécessaires et listes- peut être un élément limitant. Nonobstant l'utilité d'une telle démarche pour simplifier la manipulation des données et l'expressivité du processus, il faut tout de même se demander si cette perte de granularité n'entraîne pas une perte d'informations capitales pour certaines opérations qui n'ont pas encore été étudiées, ou si elle rend tout simplement impossible l'expression de certains processus.

Nous avons aussi la question des opérateurs. Comme expliqué, les opérateurs génériques proposés à l'analyste dans notre outil sont issus d'une étude empirique sur les possibilités des plates-formes d'analyses examinés. Cependant, comment s'assurer que la démarche expérimentale appliquée est conforme et qu'elle a su faire ressortir correctement les analogies et les altérités qui nous ont permis de regrouper les différents opérateurs des plates-formes. Et donc, sont-ils tous identiques ? De plus, comment gérer le cas des opérateurs acceptant des entrées variables -s'ils existent- ?

La deuxième difficulté concernant les opérateurs serait de savoir si l'on peut les considérer uniquement comme des fonctions, ou bien comme des multifonctions. Autrement dit, est-ce que lors de l'instanciation d'un opérateur quelconque, et ce quelle que soit la plate-forme cible, il va générer à chaque fois un seul et unique opérateur ou faudra-t-il utiliser plusieurs opérateurs pour le représenter ? Cette question concerne directement la procédure d'instanciation, qui peut impliquer de revoir le modèle actuellement défini.

Une autre limite de notre approche peut être l'étape de renseignement des liens. Cette étape est cruciale pour l'instanciation et doit donc être faite avec le plus de rigueur possible. Cependant, on peut identifier trois problèmes majeurs qui peuvent apparaître dans cette étape. Tout d'abord, on imagine bien qu'elle sera extrêmement fastidieuse si elle implique un très grand nombre de données nécessaires, et si elle est appliquée à beaucoup de plates-formes cibles. De plus, faire intervenir un utilisateur à ce niveau entraîne un risque d'erreur. Une telle erreur peut faire échouer l'instanciation sans pour autant que le système ne puisse le détecter -du fait du niveau d'abstraction adopté- ou altérer la qualité d'un processus... Et la dernière question, non des moindres, est de se demander que faire si jamais il n'est pas possible d'obtenir des archétypes de données pour une plate-forme ? Faut-il l'écarter d'emblée ? Ce sont des questions pertinentes qui sont directement en lien avec la qualité de l'instanciation.

Une dernière limite concerne l'outil que nous proposons et plus spécifiquement son interface utilisateur. Il faut réussir à proposer à l'analyste une interface qui n'influencera pas son raisonnement. Par exemple, l'interface que nous avons

adoptée, pour des raisons évidentes de simplicité et de manque de temps, est susceptible de biaiser l'approche d'un analyste. Le fait de séparer aussi rigidement les étapes peut bloquer l'utilisateur dans son raisonnement. Par exemple, si ce dernier ne sait pas qu'il est possible réutiliser les données produites dans toutes les étapes antérieures à celle où il se trouve, cela va réduire grandement la qualité de sa production. Une interface ergonomique de l'acabit d'Orange [22] serait d'après nous un bon moyen de palier à ce problème. Dans Orange, les opérations sont toutes placées sur une même "feuille", et l'utilisateur peut les manipuler et les lier directement et explicitement.

9 Conclusion et perspectives

Dans le cadre de ce stage au sein du projet de recherche HUBBLE, nous avons présenté une approche, sous l'étendard du projet HUBBLE, visant à capitaliser les processus d'analyse de traces dans le domaine du e-learning, ce qui représente aujourd'hui une problématique majeure. Cette capitalisation s'effectue au travers d'une approche différente de ce qu'il nous a été donné d'étudié : faire fi des dépendances en ne cherchant pas l'importation des données ni leur calcul, mais la représentation de la démarche intellectuelle de l'analyste.

Les propositions faites dans ce document n'ont pas la prétention d'être la solution unique à la problématique, comme l'a montré la section 8, mais plutôt un début de piste, servant ainsi de travail préparatoire à une thèse qui prendra le relais. En effet, les méta-modèles doivent encore être confrontés au monde de l'analyse et il reste à savoir s'il n'existe pas de limite dans l'expressivité de notre démarche. Malgré cela, les résultats obtenus sont encourageants et permettent d'espérer des perspectives intéressantes.

On peut être en droit d'espérer une gestion d'un nombre de plates-formes plus important par notre outil, du fait que l'approche proposée se veut extensible. Actuellement, seules les plateformes kTBS et UnderTracks sont prises en charge dans le prototype implémenté, mais les plateformes UTL [24] et Smoople [29]qui ont été étudiées, sont aussi prévues. De plus, s'étendre à des plates-formes extra projet, comme Weka [18], représenterait une vraie valeur ajoutée en terme de possibilité et d'impact dans la communauté.

De surcroît, notre approche ne vise pour le moment que la production des indicateurs. Il serait intéressant d'étendre cette couverture à d'autres éléments, comme les modèles ou encore la visualisation. Cela apporterait une grande diversité à l'outil et concernerait encore plus de scénario d'analyses.

Une évolution intéressante serait aussi de remplacer cette notion de notice par un mécanisme d'instanciation automatique de telle sorte que le système puisse configurer directement la plate-forme d'analyse de traces afin d'éviter à l'analyste de recopier ou faire lui-même les actions indiquées . Cela aurait un double effet : lui faire gagner du temps et éliminer le facteur d'erreur lors de la recopie.

Enfin, une perspective qui pourrait être extrêmement intéressante est d'offrir la possibilité d'importer un processus d'analyse de traces exprimé dans une plate-

forme d'analyse vers notre outil, ayant pour effet direct de le traduire en processus indépendant de la plate-forme. Une telle possibilité serait remarquable sur le plan de la capitalisation, puisqu'il deviendrait possible de partager n'importe quel processus d'analyse dépendant d'une plate-forme donnée, tout en bénéficiant du feedback de ladite plate-forme, palliant ainsi à cette limitation de notre outil. De plus, cela permettrait d'obtenir très rapidement un réservoir de processus d'analyse de traces conséquent et pertinent.

En conclusion, notre approche, consistant en la capitalisation des processus d'analyse, est une première étape vers une solution fédératrice des différentes plates-formes d'analyses, qui ne demande qu'à être enrichie pour offrir aux analystes une nouvelle manière, plus intuitive et efficace, de penser leurs démarches.

References

1. A. Dimitrakopoulou, A. Petrou, A. Martinez, J. A. Marcos, V. Kollias, P. Jermann, A. Harrer, Y. Dimitriadis, and L. Bollen, "State of the art of interaction analysis for metacognitive support & diagnosis," 2006.
2. R. R. Springmeyer, M. M. Blattner, and N. L. Max, "A characterization of the scientific data analysis process," in *Proceedings of the 3rd conference on Visualization'92*. IEEE Computer Society Press, 1992, pp. 235–242.
3. HUBBLE. (2015) Hubble web site. [Online]. Available: http://hubblelearn.imag.fr/
4. J. H. Andrews, "Testing using log file analysis: tools, methods, and issues," in *Automated Software Engineering, 1998. Proceedings. 13th IEEE International Conference on*. IEEE, 1998, pp. 157–166.
5. J. H. Andrews and Y. Zhang, "General test result checking with log file analysis," *Software Engineering, IEEE Transactions on*, vol. 29, no. 7, pp. 634–648, 2003.
6. W. Van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 9, pp. 1128–1142, 2004.
7. M. Sachan, D. Contractor, T. A. Faruquie, and L. V. Subramaniam, "Using content and interactions for discovering communities in social networks," in *Proceedings of the 21st international conference on World Wide Web*. ACM, 2012, pp. 331–340.
8. J. Schoonenboom, K. Keenoy, L. Montandon, Y. Goita, D. Faure, J.-P. David, A. Lejeune, C. Blake, Z. Pluhár, P. Kaszás *et al.*, "Trails of digital and non-digital los," 2004.
9. P.-A. Champin, Y. Prié, and A. Mille, "Musette: a framework for knowledge from experience," in *Extraction et gestion des connaissances (EGC'2004)(article court)*, 2004, pp. 129–134.
10. L. S. Settouti, Y. Prié, A. Mille, and J.-C. Marty, "Systèmes à base de trace pour l'apprentissage humain," in *Colloque international TICE*, 2006.
11. K. R. Koedinger, R. S. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper, "A data repository for the edm community: The pslc datashop," *Handbook of educational data mining*, vol. 43, 2010.
12. C. Reffay, M.-L. Betbeder, and T. Chanier, "Multimodal learning and teaching corpora exchange: lessons learned in five years by the mulce project," *International Journal of Technology Enhanced Learning*, vol. 4, no. 1, pp. 11–30, 2012.
13. A. J. Hey and A. E. Trefethen, "The data deluge: An e-science perspective," 2003.
14. U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, p. 37, 1996.

15. R. S. Baker, A. T. Corbett, K. R. Koedinger, and A. Z. Wagner, "Off-task behavior in the cognitive tutor classroom: when students game the system," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2004, pp. 383–390.
16. N. Mandran, M. Ortega, V. Luengo, and D. Bouhineau, "Dop8: merging both data and analysis operators life cycles for technology enhanced learning," in *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*. ACM, 2015, pp. 213–217.
17. G. Holmes, B. Pfahringer, R. Kirkby, E. Frank, and M. Hall, "Multiclass alternating decision trees," in *Machine learning: ECML 2002*. Springer, 2002, pp. 161–172.
18. S. R. Garner *et al.*, "Weka: The waikato environment for knowledge analysis," in *Proceedings of the New Zealand computer science research students conference*. Citeseer, 1995, pp. 57–64.
19. J. Laflaquière, "Conception de système à base de traces numériques pour les environnements informatiques documentaires," Ph.D. dissertation, Université de Technologie de Troyes, 2009.
20. B. Fuchs and A. Belin, "A trace-based approach for managing users experience," *Luc Lamontagne and Juan A. Recio-Garcıa (Editors)*, p. 163, 2012.
21. A. Cordier, F. Derbel, and A. Mille, "Observing a web based learning activity: a knowledge oriented approach," Ph.D. dissertation, LIRIS UMR CNRS 5205, 2015.
22. U. of Ljubljana. (2015) Orange web site. [Online]. Available: http://orange.biolab.si/
23. C. Choquet and S. Iksal, "Modélisation et construction de traces d'utilisation d'une activité d'apprentissage: une approche langage pour la réingénierie d'un eiah," *Revue des Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation (STICEF)*, vol. 14, pp. 24–pages, 2007.
24. S. Iksal, "Ingénierie de l'observation basée sur la prescription en eiah," Ph.D. dissertation, Université du Maine, 2012.
25. I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, "Yale: Rapid prototyping for complex data mining tasks," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 935–940.
26. A. Dimitracopoulou and E. Bruillard, "Enrichir les interfaces de forums par la visualisation d'analyses automatiques des interactions et du contenu," *Revue des Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation (STICEF)*, vol. 13, pp. 40–pages, 2006.
27. T. Djouad, A. Mille, C. Reffay, and M. Benmohamed, "Ingénierie des indicateurs d'activités à partir de traces modélisées pour un environnement informatique d apprentissage humain," *Revue des Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation (STICEF)*, vol. 16, pp. 28–pages, 2009.
28. F. Diagne, "Instrumentation de la supervision par la réutilisation d'indicateurs: Modèles et architecture," Ph.D. dissertation, Université Joseph-Fourier-Grenoble I, 2009.
29. J. Gilliot, S. Garlatti, I. Rebai, and C. Pham Nguyen, "A mobile learning scenario improvement for hst inquiry based learning," in *EWFE workshop proceedings*. Citeseer, 2012.

A OutputSheet - Grammaire

```
CREATE TOK_INTEGER List.
LISTS !

LISTS : LIST$TOK_INTEGER : INSTRUCS
       | LIST$TOK_INTEGER : INSTRUCS LISTS

INSTRUCS : INSTRUC.
          | INSTRUC. INSTRUCS

INSTRUC        : MAINTAIN {$TOK_ENTITY, $TOK_LIST}
         | DELETE {$TOK_ENTITY}
         | CREATE {$TOK_ENTITY, (TOK_NAME, TOK_TYPE)}

TOK_INTEGER : xs:integer
TOK_NAME    : xs:string
TOK_TYPE    : xs:strng
TOK_ENTITY  : eTOK_INTEGER
TOK_LIST    : eTOK_INTEGER.list
```

Fig. 7. Gammaire du langage utilisé dans les OutputSheets pour définir le comportement des opérateurs dans la zone indépendante.

B QuerySheet - Exemples

```
@prefix : <http://liris.cnrs.fr/silex/2009/ktbs#> .

<> :contains <condFilter$x/> .

<condFilter$x/>
    a :ComputedTrace ;
    :hasMethod :sparql ;
    :hasSource $e1.list ;
    :hasParameter """sparql=
PREFIX : <http://liris.cnrs.fr/silex/2009/ktbs#>
PREFIX m:  $1.model

CONSTRUCT {
    [ a m:$c1.obselType ;
    m:$c1.URI ?val ;
    :hasTrace <%(__destination__)s> ;
    :hasSubject ?val ;
    :hasBegin ?begin ;
    :hasEnd ?end ;
    :hasSourceObsel ?o1, ?o2 ;
    ] .
} WHERE {
            SELECT ?val
            {
                 ?o1 $e1 ?val.
                 FILTER ( ?val $p1 $p2)
            }
}""" .
```

Fig. 8. Feuille de requête (QuerySheet) utilisée lors de l'instanciation de l'opérateur *filtre conditionnel* dans la plate-forme d'analyse de traces kTBS.

```
Pour la table $e1.list, utiliser SELECT DATA.
Sur $e1, effectuer le test conditionnel avec $p1,
tel que $e1 soit $p1 à $p2.
Lier SELECT DATA à une nouvelle table intitulée filterTable$x.
```

Fig. 9. Feuille de requête (QuerySheet) utilisée lors de l'instanciation de l'opérateur *filtre conditionnel* dans la plate-forme d'analyse UnderTracks.

C Procédé théorique de la résolution d'une QuerySheet

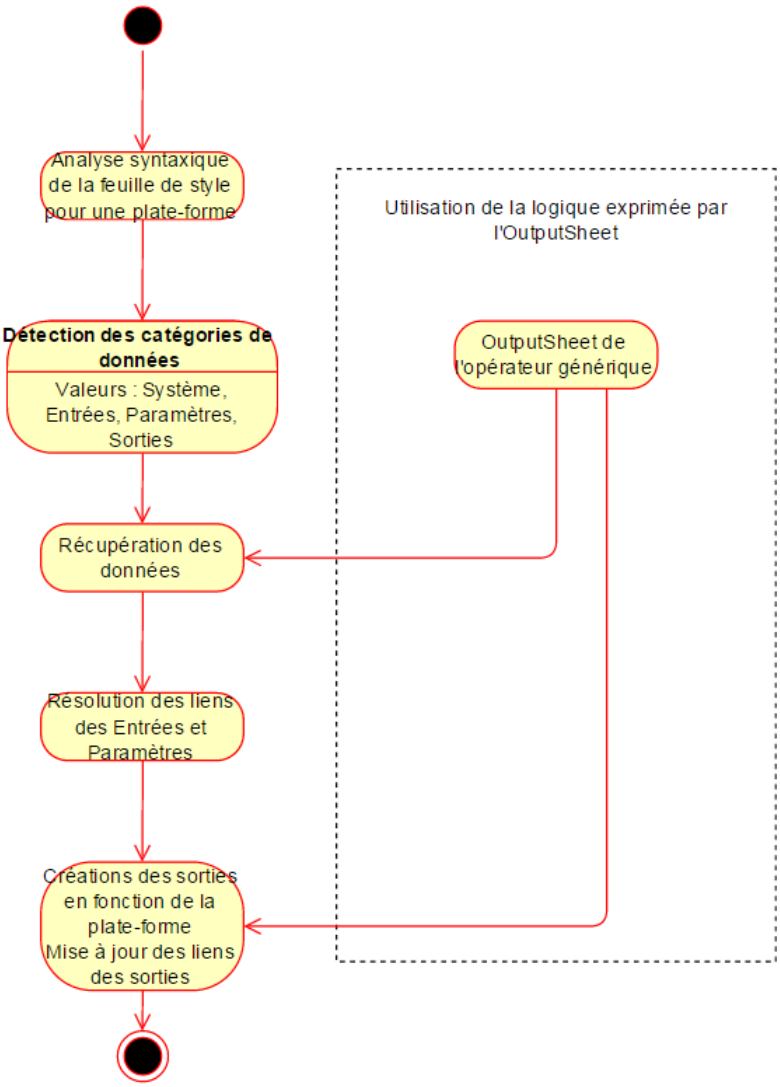


Fig. 10. Schéma de la méthode de résolution d'une QuerySheet lors de l'instanciation d'une étape, pour une plate-forme. L'utilisation de l'OutputSheet est nécessaire pour accéder aux données manipulées dans la zone indépendante.

D Méta-modèles

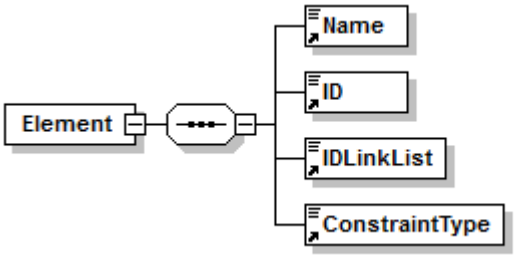


Fig. 11. Méta-modèle d'une donnée nécessaire. Le nom ainsi que son Id permettent l'identification, tandis que l'IDLinkList permet de faire référence à une liste. ConstraintType représente le type de valeur dont il s'agit.

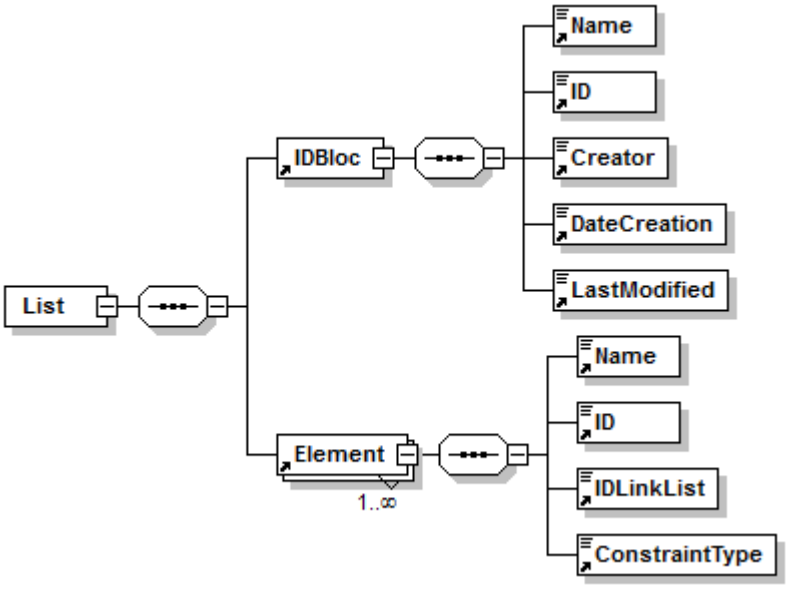


Fig. 12. Méta-modèle d'une liste. On retrouve le bloc d'identification IDBloc, en plus d'une liste de données nécessaires représentant l'idée de conteneur structurel.

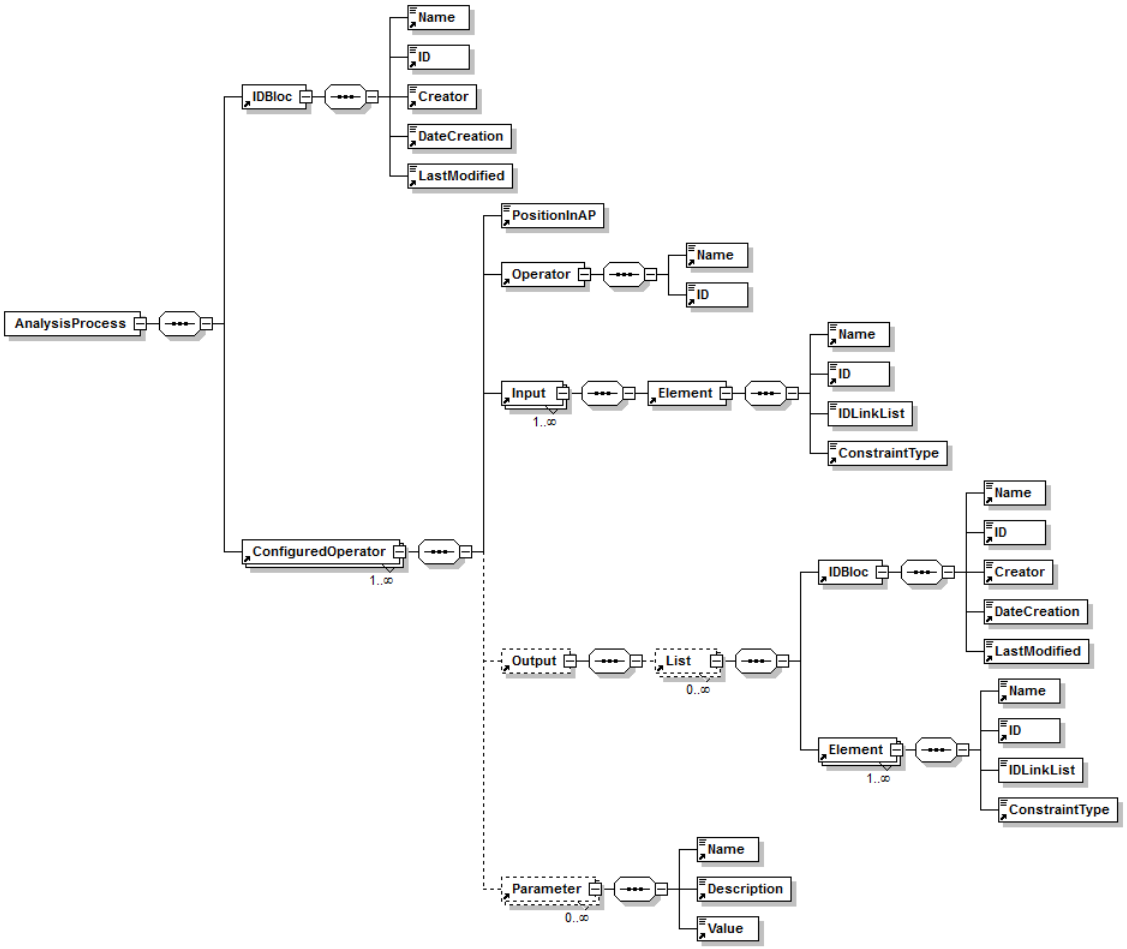


Fig. 13. Méta-modèle d'un processus d'analyse de traces, dans son ensemble.

E Fiche d'un opérateur indépendant

Filtre Conditionnel

Formalisme

$$F_{val} = (\ categorie_{elem}, val, operande)$$

Objectif

Filtrer les valeurs en fonction d'une condition.

Description

Filtre Valeur applique un filtre conditionnel sur les éléments considérés, et laisse passer tout ceux qui le respectent. La valeur peut être numérique ou textuelle, et l'opérande choisie parmi une liste contenant : <, ≤,≥,>,!=,==.

Entrée

$categorie_{elem}$, toute la catégorie des éléments à analyser,
val, la valeur à respecter,
$operande$, l'opérande de la condition à choisir parmi <, ≤,≥,>,!=,==.

Sortie

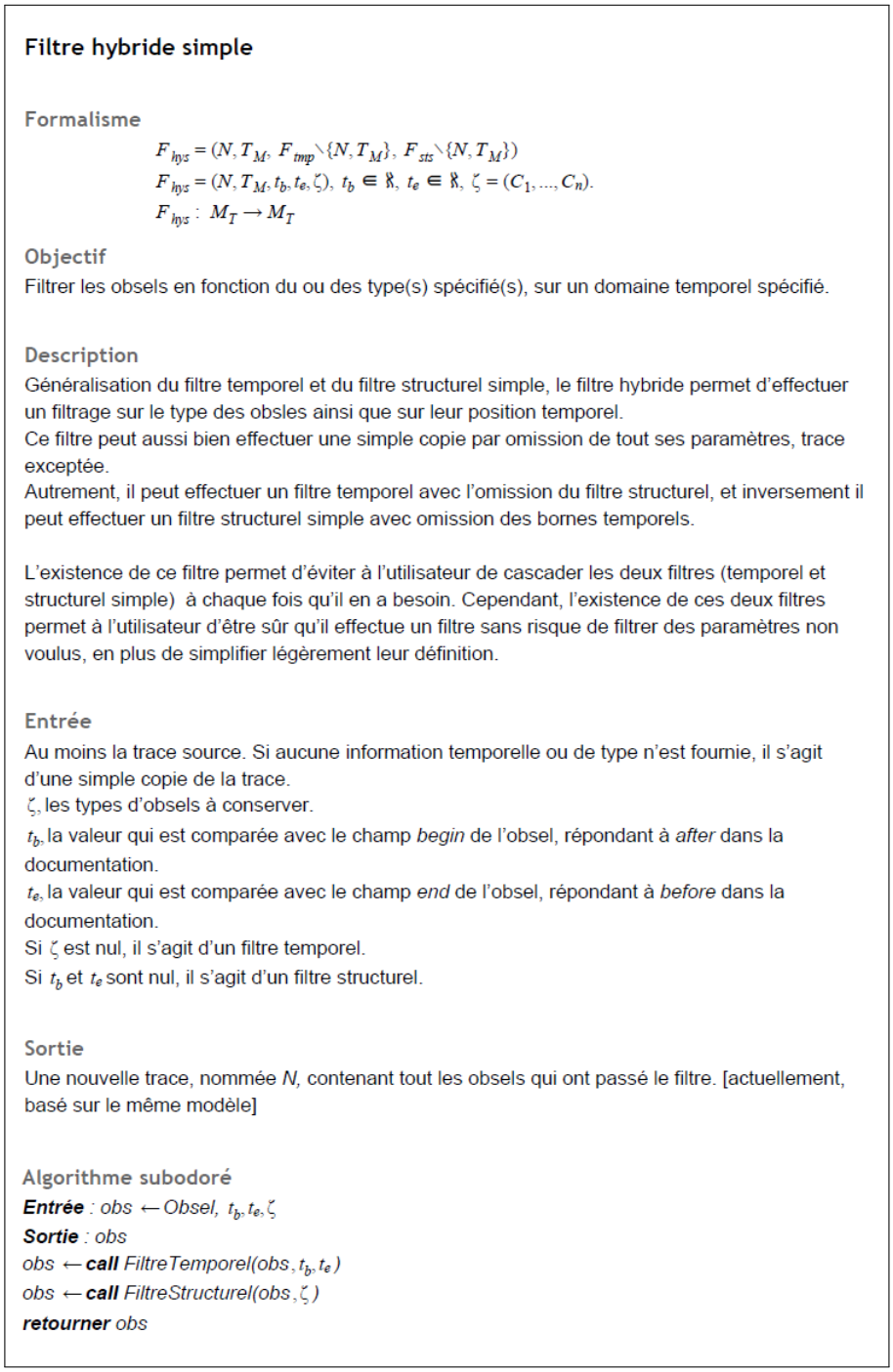
Une nouvelle trace basée sur la trace contenant $categorie_{elem}$, avec les éléments qui ont passé le filtre.

Processus

- $F_{approx\_kTBS} = (N, T_{M\_element}, D_{element}, val, operande)$
- $F_{approx\_UT} = (\tau_{element}, \Psi_{element}, val, operande)$

Fig. 14. Fiche montrant la démarche adoptée pour représenter un opérateur, et comment l'instancier. À noter que la partie processus fait référence aux opérateurs dépendant des plates-formes, que nous avons formalisés.

F Fiche d'un opérateur dépendant d'une plate-forme



Filtre hybride simple

Formalisme

$$F_{hys} = (N, T_M, F_{tmp} \setminus \{N, T_M\}, F_{sts} \setminus \{N, T_M\})$$
$$F_{hys} = (N, T_M, t_b, t_e, \zeta),\ t_b \in \aleph,\ t_e \in \aleph,\ \zeta = (C_1, ..., C_n).$$
$$F_{hys} :\ M_T \rightarrow M_T$$

Objectif

Filtrer les obsels en fonction du ou des type(s) spécifié(s), sur un domaine temporel spécifié.

Description

Généralisation du filtre temporel et du filtre structurel simple, le filtre hybride permet d'effectuer un filtrage sur le type des obsles ainsi que sur leur position temporel.
Ce filtre peut aussi bien effectuer une simple copie par omission de tout ses paramètres, trace exceptée.
Autrement, il peut effectuer un filtre temporel avec l'omission du filtre structurel, et inversement il peut effectuer un filtre structurel simple avec omission des bornes temporels.

L'existence de ce filtre permet d'éviter à l'utilisateur de cascader les deux filtres (temporel et structurel simple) à chaque fois qu'il en a besoin. Cependant, l'existence de ces deux filtres permet à l'utilisateur d'être sûr qu'il effectue un filtre sans risque de filtrer des paramètres non voulus, en plus de simplifier légèrement leur définition.

Entrée

Au moins la trace source. Si aucune information temporelle ou de type n'est fournie, il s'agit d'une simple copie de la trace.
ζ, les types d'obsels à conserver.
t_b, la valeur qui est comparée avec le champ *begin* de l'obsel, répondant à *after* dans la documentation.
t_e, la valeur qui est comparée avec le champ *end* de l'obsel, répondant à *before* dans la documentation.
Si ζ est nul, il s'agit d'un filtre temporel.
Si t_b et t_e sont nul, il s'agit d'un filtre structurel.

Sortie

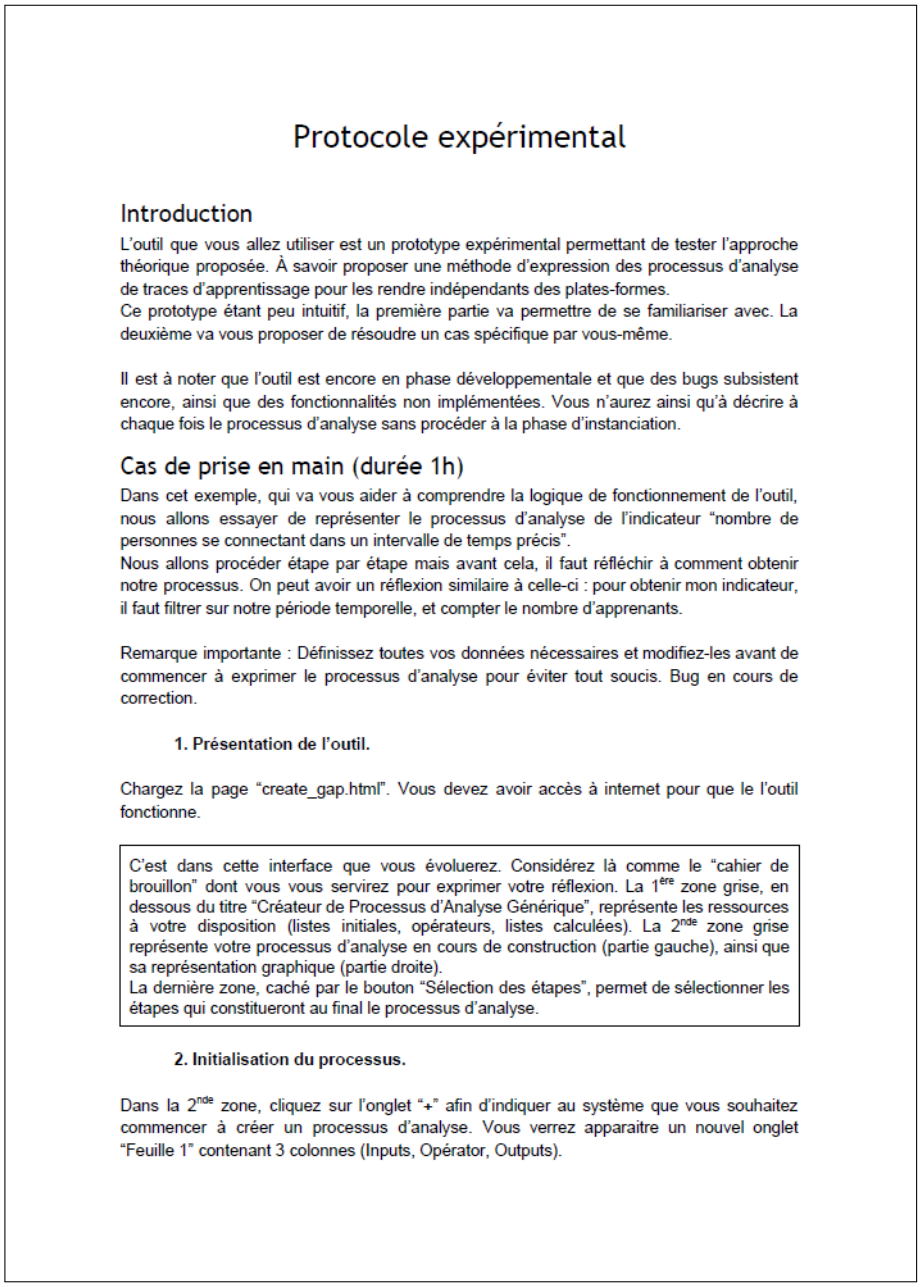
Une nouvelle trace, nommée *N*, contenant tout les obsels qui ont passé le filtre. [actuellement, basé sur le même modèle]

Algorithme subodoré

Entrée *: obs* $\leftarrow$ *Obsel,* t_b, t_e, ζ
Sortie *: obs*
obs $\leftarrow$ ***call*** *FiltreTemporel(obs*, t_b, t_e*)*
obs $\leftarrow$ ***call*** *FiltreStructurel(obs*, ζ*)*
retourner *obs*

Fig. 15. Fiche montrant la démarche adoptée pour *formaliser* un opérateur dépendant d'une plate-forme.

G Protocole expérimental - Aperçu



Protocole expérimental

Introduction

L'outil que vous allez utiliser est un prototype expérimental permettant de tester l'approche théorique proposée. À savoir proposer une méthode d'expression des processus d'analyse de traces d'apprentissage pour les rendre indépendants des plates-formes.
Ce prototype étant peu intuitif, la première partie va permettre de se familiariser avec. La deuxième va vous proposer de résoudre un cas spécifique par vous-même.

Il est à noter que l'outil est encore en phase développementale et que des bugs subsistent encore, ainsi que des fonctionnalités non implémentées. Vous n'aurez ainsi qu'à décrire à chaque fois le processus d'analyse sans procéder à la phase d'instanciation.

Cas de prise en main (durée 1h)

Dans cet exemple, qui va vous aider à comprendre la logique de fonctionnement de l'outil, nous allons essayer de représenter le processus d'analyse de l'indicateur "nombre de personnes se connectant dans un intervalle de temps précis".
Nous allons procéder étape par étape mais avant cela, il faut réfléchir à comment obtenir notre processus. On peut avoir un réflexion similaire à celle-ci : pour obtenir mon indicateur, il faut filtrer sur notre période temporelle, et compter le nombre d'apprenants.

Remarque importante : Définissez toutes vos données nécessaires et modifiez-les avant de commencer à exprimer le processus d'analyse pour éviter tout soucis. Bug en cours de correction.

1. Présentation de l'outil.

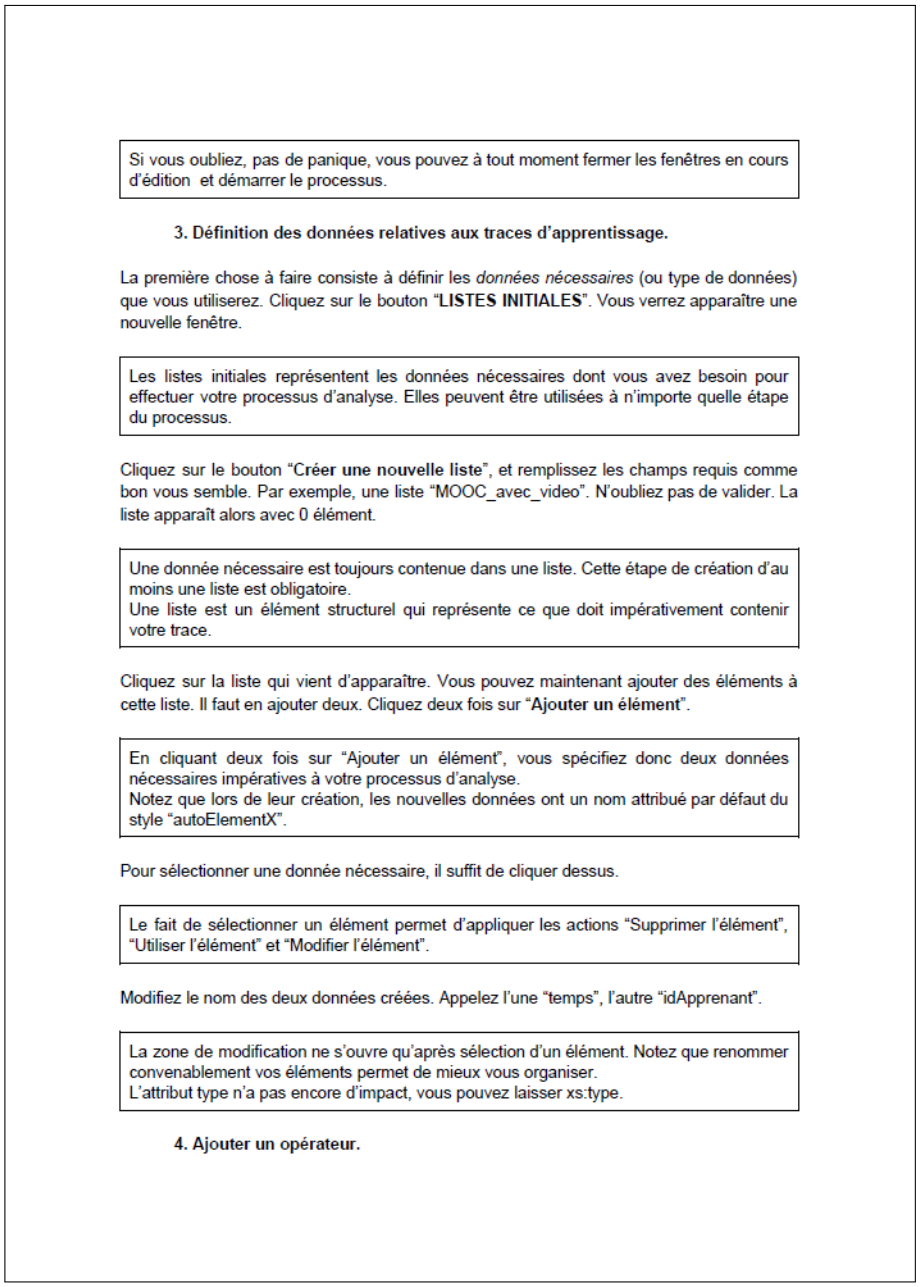
Chargez la page "create_gap.html". Vous devez avoir accès à internet pour que le l'outil fonctionne.

C'est dans cette interface que vous évoluerez. Considérez là comme le "cahier de brouillon" dont vous vous servirez pour exprimer votre réflexion. La 1ère zone grise, en dessous du titre "Créateur de Processus d'Analyse Générique", représente les ressources à votre disposition (listes initiales, opérateurs, listes calculées). La 2nde zone grise représente votre processus d'analyse en cours de construction (partie gauche), ainsi que sa représentation graphique (partie droite).
La dernière zone, caché par le bouton "Sélection des étapes", permet de sélectionner les étapes qui constitueront au final le processus d'analyse.

2. Initialisation du processus.

Dans la 2nde zone, cliquez sur l'onglet "+" afin d'indiquer au système que vous souhaitez commencer à créer un processus d'analyse. Vous verrez apparaitre un nouvel onglet "Feuille 1" contenant 3 colonnes (Inputs, Opérator, Outputs).

Fig. 16. Première page de notre protocole expérimental. Le premier cas, à la différence des deux autres qui ne sont pas présentés ici, décrit étape par étape les actions à effectuer pour prendre en main le prototype proposé.



Si vous oubliez, pas de panique, vous pouvez à tout moment fermer les fenêtres en cours d'édition et démarrer le processus.

3. Définition des données relatives aux traces d'apprentissage.

La première chose à faire consiste à définir les *données nécessaires* (ou type de données) que vous utiliserez. Cliquez sur le bouton "**LISTES INITIALES**". Vous verrez apparaître une nouvelle fenêtre.

Les listes initiales représentent les données nécessaires dont vous avez besoin pour effectuer votre processus d'analyse. Elles peuvent être utilisées à n'importe quelle étape du processus.

Cliquez sur le bouton "**Créer une nouvelle liste**", et remplissez les champs requis comme bon vous semble. Par exemple, une liste "MOOC_avec_video". N'oubliez pas de valider. La liste apparaît alors avec 0 élément.

Une donnée nécessaire est toujours contenue dans une liste. Cette étape de création d'au moins une liste est obligatoire.
Une liste est un élément structurel qui représente ce que doit impérativement contenir votre trace.

Cliquez sur la liste qui vient d'apparaître. Vous pouvez maintenant ajouter des éléments à cette liste. Il faut en ajouter deux. Cliquez deux fois sur "**Ajouter un élément**".

En cliquant deux fois sur "Ajouter un élément", vous spécifiez donc deux données nécessaires impératives à votre processus d'analyse.
Notez que lors de leur création, les nouvelles données ont un nom attribué par défaut du style "autoElementX".

Pour sélectionner une donnée nécessaire, il suffit de cliquer dessus.

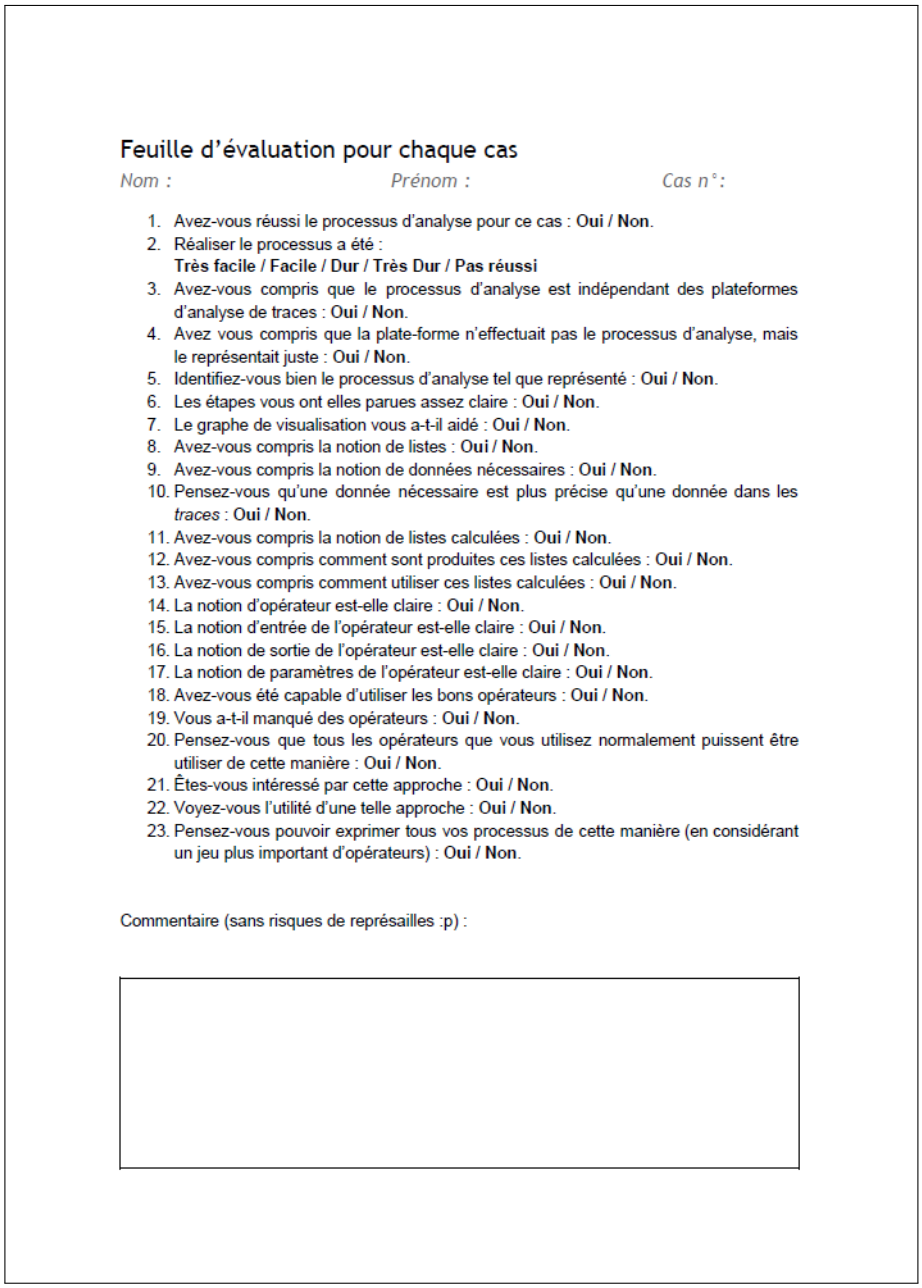
Le fait de sélectionner un élément permet d'appliquer les actions "Supprimer l'élément", "Utiliser l'élément" et "Modifier l'élément".

Modifiez le nom des deux données créées. Appelez l'une "temps", l'autre "idApprenant".

La zone de modification ne s'ouvre qu'après sélection d'un élément. Notez que renommer convenablement vos éléments permet de mieux vous organiser.
L'attribut type n'a pas encore d'impact, vous pouvez laisser xs:type.

4. Ajouter un opérateur.

Fig. 17. Seconde page du protocole expérimental. Cette page concerne la définition des données nécessaires au processus d'analyse.



Feuille d'évaluation pour chaque cas

Nom : *Prénom :* *Cas n° :*

1. Avez-vous réussi le processus d'analyse pour ce cas : **Oui / Non**.
2. Réaliser le processus a été :
   Très facile / Facile / Dur / Très Dur / Pas réussi
3. Avez-vous compris que le processus d'analyse est indépendant des plateformes d'analyse de traces : **Oui / Non**.
4. Avez vous compris que la plate-forme n'effectuait pas le processus d'analyse, mais le représentait juste : **Oui / Non**.
5. Identifiez-vous bien le processus d'analyse tel que représenté : **Oui / Non**.
6. Les étapes vous ont elles parues assez claire : **Oui / Non**.
7. Le graphe de visualisation vous a-t-il aidé : **Oui / Non**.
8. Avez-vous compris la notion de listes : **Oui / Non**.
9. Avez-vous compris la notion de données nécessaires : **Oui / Non**.
10. Pensez-vous qu'une donnée nécessaire est plus précise qu'une donnée dans les *traces* : **Oui / Non**.
11. Avez-vous compris la notion de listes calculées : **Oui / Non**.
12. Avez-vous compris comment sont produites ces listes calculées : **Oui / Non**.
13. Avez-vous compris comment utiliser ces listes calculées : **Oui / Non**.
14. La notion d'opérateur est-elle claire : **Oui / Non**.
15. La notion d'entrée de l'opérateur est-elle claire : **Oui / Non**.
16. La notion de sortie de l'opérateur est-elle claire : **Oui / Non**.
17. La notion de paramètres de l'opérateur est-elle claire : **Oui / Non**.
18. Avez-vous été capable d'utiliser les bons opérateurs : **Oui / Non**.
19. Vous a-t-il manqué des opérateurs : **Oui / Non**.
20. Pensez-vous que tous les opérateurs que vous utilisez normalement puissent être utiliser de cette manière : **Oui / Non**.
21. Êtes-vous intéressé par cette approche : **Oui / Non**.
22. Voyez-vous l'utilité d'une telle approche : **Oui / Non**.
23. Pensez-vous pouvoir exprimer tous vos processus de cette manière (en considérant un jeu plus important d'opérateurs) : **Oui / Non**.

Commentaire (sans risques de représailles :p) :

Fig. 18. Page questionnaire du protocole expérimental. Cette page permet d'interroger l'utilisateur et de savoir si les concepts ont été correctement compris et assimilés, en plus de récupérer l'avis objectif d'un analyste sur le prototype.