Opérateurs et fonctions de transformation de profils d'apprenant sous forme de services web

Hoang Phuoc, Stéphanie Jean-Daubias

▶ **To cite this version:**

Hoang Phuoc, Stéphanie Jean-Daubias. Opérateurs et fonctions de transformation de profils d'apprenant sous forme de services web. RR-LIRIS-2010-020, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon. 2010. hal-04371680

HAL Id: hal-04371680
https://hal.science/hal-04371680

Submitted on 3 Jan 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Opérateurs et fonctions de transformation de profils d'apprenant sous forme de services web

Hoang Phuoc Truong, Stéphanie Jean-Daubias

Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France
Stephanie.Jean-Daubias@liris.univ-lyon1.fr

Juillet 2010

Résumé. Ce rapport traite des opérateurs sur des profils d'apprenant dans le cadre du projet PERLEA. Les opérateurs manipulant ce type des profils sont proposés et décrits. Un modèle permettant aux utilisateurs de créer des fonctions de composition d'opérateurs est aussi présenté. Une approche de l'implémentation des opérateurs sous forme de services web est mentionnée comme une solution pour l'usage par plusieurs applications.

Mots clés : Profil d'apprenant, opérateur, fonction, service web.

Abstract. In this report, we present the operators on the learner profiles in the project PERLEA. The operators manipulating this kind of profiles are proposed and described. A model that allows the users to create the functions by combining the operators is also proposed. Then, an approach of the implementation of the operators in the form of the web services is mentioned as a solution for the use by multiple applications.

Keywords: Learner profile, operators, function, web service.

Table des matières

1 Introduction

Dans le domaine de l'éducation, le profil d'apprenant joue un rôle très important. Il fait l'objet d'attentions particulières, à la fois des praticiens, des chercheurs et des institutions. Concernant des praticiens, les enseignants du primaire et du secondaire sont incités à individualiser toujours plus l'apprentissage. Concernant les apprenants eux-mêmes, des recherches ont démontré l'intérêt de leur présenter des informations concernant l'état de leurs connaissances en vue de les aider à développer des compétences réflexives et de renforcer leur motivation et leur responsabilisation face à leur apprentissage. Concernant les institutions, les initiatives menées autour des référentiels de compétences, les travaux de standardisation visant à favoriser l'échange de documents pédagogiques [1].
Le projet PERLEA (Profils d'Elèves Réutilisés pour L'Enseignant et l'Apprenant) vise à concevoir un système permettant aux enseignants de gérer des profils existants (aussi bien papier - crayon qu'issus de logiciels, de tous niveaux, scolaires et supérieurs, dans tous les domaines) en en proposant une visualisation riche (pour un élève ou pour une classe) facilitant le suivi de l'apprenant, notamment par une étude de l'évolution des profils dans le temps.
L'environnement EPROFILEA (Exploitation de PRofils par L'Enseignant et l'Apprenant) est un environnement informatique développé dans le cadre du projet PERLEA. Il propose des exploitations riches des profils d'apprenant : des activités interactives sur les profils et proposition des activités pédagogiques personnalisées en fonction du profil.
Ce travail consiste à étudier les opérateurs qui peuvent être effectués sur des modèles d'utilisateurs, plus précisément sur des profils d'apprenants et puis à construire un module qui permet de réaliser des traitements sur ces profils. Le module construit nommé GROUPE va être totalement intégré à l'environnement EPROFILEA. Grâce à ce module, l'utilisateur peut utiliser des opérateurs ainsi que des fonctions de composition pour effectuer nombre de traitements sur des profils d'apprenant.
Par la suite, nous présentons l'état de l'art sur des modèles de l'utilisateur dans la deuxième partie, la méthodologie dans la troisième partie, l'implémentation du module GROUPE dans la quatrième partie et en fin la conclusion.

2 État de l'art

Dans cette partie, nous présentons et analysons la notion de modèle de l'utilisateur. Ces approches nous permettent de connaitre les différentes manières présentées et exploitées des informations sur l'utilisateur. Puis, nous précisons cet état de l'art en présentant les représentations des profils d'apprenant.

2.1 Le modèle de l'utilisateur défini par l'ontologie

Une ontologie est l'ensemble structuré des termes et concepts représentant le sens d'un champ d'information, que ce soit par les métadonnées d'un espace de nom, ou les éléments d'un domaine de connaissance. Grâce à l'ontologie, on peut créer de façon riche de représentation des informations de l'utilisateur.
Le modèle SUMO (Suggested Upper Merged Ontology) [2] est créé par la fusion de nombre des ontologies de haut niveau existant en constituant un grand ensemble de l'ontologie. Ces ontologies contiennent les contenus créés par Sowa [8], Guarino et ses collègues [9], Allen [10] et Smith [11] ainsi que les ontologies plus concrètes de Standford KSL et ITBM-CNR. Actuellement, SUMO se compose de 11 sections : l'ontologie structurelle qui contient les concepts des relations pour la définition de l'ontologie propre, l'ontologie de base se compose des notions ontologiques très fondamentales, la section numérique définit des notions arithmétiques, la section temporelle, la section de l'ontologie Mereotopology [12] fournit des axiomatisations, la section de la théorie des

graphes fournit les notions théoriques de graphe, la section de l'unité de mesure, et les reste section fournit les sous-hiérarchies, les axiomes correspondant aux types des objets.
Le modèle *GUMO* (*the Generalized User Model Ontology*) [3] propose une ontologie générale pour les modèles de l'utilisateur qui sont utilisés par des systèmes adaptés l'utilisateur. Cette ontologie ménage des différences structurelles et syntaxiques parmi des ontologies existantes en collectionnant les dimensions de l'utilisateur qui sont déjà modélisés par des systèmes adaptés l'utilisateur. Par la réutilisation de l'idée principale de l'approche *UserML* [4], *GUMO* utilise les « *Situational Statement* » [13] pour représenter les situations. Il dévide les dimensions de l'utilisateur en trois parties : *auxiliaire*, *prédicat*, et *rangée*. Par exemple, si on a une phrase « *something about the user's interested in football* », donc on a *auxiliaire* = « *hasInterest* », *prédicat* = « *football* » et *rangée* = « *low-high-medium »*. Environs 1000 GROUPEs des auxiliaires, prédicats et rangées sont insérés dans l'ontologie. Mais le problème actuel est que tous les situations peuvent devenir un *prédicat* de l'auxiliaire *hasInterest* ou *hasKnowledge* et cela peut causer le problème de modularité. Donc, la solution proposée est d'identifier les dimensions de base de l'utilisateur d'une part, de laisser la connaissance plus générale du monde ouvert pour les ontologies existant d'autre part.

2.2 La modélisation de l'utilisateur – les différentes approches pour le calcul ubiquitaire

La modélisation de l'utilisateur consiste en général en des traitements sur des informations sur l'utilisateur pour connaître tous les aspects de l'utilisateur mais surtout ses comportements dans un contexte spécifique.
La modélisation ubiquitaire de l'utilisateur (*Ubiquitous User Modeling - UUM*) [4] est un nouveau concept défini par Wahlster. Elle indique la capacité d'un système à décrire la modélisation au courant et à exploiter des comportements de l'utilisateur venant de plusieurs autres systèmes qui partagent ses modèles de l'utilisateur. En proposant le modèle « *Situational Statement* », les situations complexes sont décomposées et décrites par les phrases uniformes et structurées. Sa structure s'organise en couches. La couche « *MainPart* » est entourée par les autres couches. La couche « *MainPart* » (ou « *MainPart Box* ») contient les informations de situation partielle et elle se compose de cinq attributs dont trois obligatoires : *sujet, prédicat, objet* et deux optionnels : *auxiliaire* et *rangée*. « *Situation Box* » contient les informations temporelles et spatiales de la situation entière. « *Explanation Box »* contient les méta-attributs explicatifs qui sont utilisés pour la résolution des conflits. « *Privacy Box »* contient les méta-attributs qui décident le partage des informations sensitives dans un environnement distribué. Et finalement, « *Administration Box »* est utilisé pour améliorer la vitesse des requêtes sur des phrases situationnelle.
Le service de modélisation de l'utilisateur (*User Modeling Service – UM Service*) est proposé dans [6] pour modéliser l'utilisateur à travers de plusieurs applications. Ce service n'est pas limité par un domaine spécifique, donc il peut être intégré dans n'importe quelle application web. *UM Service* est un service centralisé qui est implémenté dans le « *Personal Reader Framework »*, un *framework* permettant le développement rapide des applications web interopérables et personnalisés. Par rapport au stockage des données, le « *Ressource Description Framework (RDF)»* est choisi comme un modèle d'annotation des informations. Par rapport aux vocabulaires, le modèle de *« Generalized User Modeling Ontology (GUMO) »* est choisi pour définir ses concepts d'ontologie appelée *UMO. UMO* contient quatre segments appelés « *Main Segment », « Explanation Segment », « Validity Segment »,* et *« Administration Segment ».* « *Main Segment »* contient des informations de base de l'utilisateur. « *Explanation Segment »* contient les informations sur le créateur de la phrase. « *Validity Segment »* contient le période (prenant en compte depuis le début) où la phrase sera validée. Et « *Administration Segment »* contient beaucoup de métadonnées des notes, des replacements, ou des suppressions de la phrase. Par rapport au contrôle d'accès, les auteurs ont proposé un *endpoint SESAME*[1] amélioré. Avec *SeRQL* (un langage de requête *RDF* qui est similaire à *SPARQL*), des services peuvent accéder et exploiter les données *RDF* brutes qui sont stockées dans l'*UM Service*.

[1] www.openrdf.org/

2.3 Les profils d'apprenant et les modèles d'apprenant

Un arbre de connaissances est une représentation imagée et structurée de la somme des richesses que chaque membre apporte à une communauté, selon diverses réalités (connaissances, compétences, opinions, événements, projets, besoins etc.) vécues par un GROUPE de personnes [21]. Le concept d'arbre de connaissances proposé par Pierre Levy et Michel Authier [14] indique qu'un arbre de connaissances est utile particulièrement dans trois situations. Premièrement, pour les personnes qui n'entrent dans « aucune des configurations de métier, d'expérience ou de savoirs traditionnelle requis pour occuper les emplois existants » et qui n'ont donc d'autre solution que de mettre en perspective les morceaux de parcours qu'ils ont pu entraîner et qui sont révélateurs d'une capacité d'agir. Deuxièmement, dans les institutions éducatives où se déroule l'apprentissage, dans lequel « se nouent des relations réussites ou manquées entre le faire et le savoir ». Troisièmement, dans les contextes en forte évolution qui ne permettent pas de prescrire le travail, mais supposent de la part de l'individu une capacité d'exploration, de création de sens et de mobilisation de ressources [7]. Selon [15], un arbre de connaissances rend visible la multiplicité de la richesse organisée, les connaissances et compétences apportées par une communauté sans la moyenne écrase l'individualité des personnes.

Portfolio de l'apprenant est un ensemble des informations de l'apprenant qui permettent de représenter certaines informations sur l'apprentissage d'un individu en termes de production [7]. D'après la littérature, un portfolio contient un ensemble des informations sur des travaux d'un apprenant sélectionné pour connaître son progrès et ses réalisations dans un ou plusieurs disciplines [16] [17]. D'après les institutions, un portfolio consiste en des efforts, des progrès et des réalisations d'un apprenant dans un ou plusieurs domaines [18]. Un portfolio se compose de deux parties : la partie structure qui contient une table de matières indiquant le type de document présenté et la partie de contenu contenant plusieurs types d'information sur l'apprenant comme ses évaluations, ses compétences, des commentaires [19].

Le modèle d'apprenant utilisé par ViSMod [24] (un système permettant de visualiser et d'exploiter des modèles d'apprenant) peut intégrer des informations venant d'EIAH possédant un modèle de l'apprenant représentées sous forme de réseaux Bayésiens [25].

DynMap+ [28] utilise aussi un modèle d'apprenant de type réseau bayésien constitué par d'autres systèmes informatiques. Les données sont traduites par DynMap+ par un traducteur pour chaque EIAH source, dans un formalisme XML interne à DynMap+. Ce formalisme se compose de deux parties : la définition des données du domaine et la définition des données d'apprentissage de l'apprenant [1].

Le modèle REPro [1] est proposé comme un modèle de processus de gestion de profils. Il nous donne les moyens de comprendre le processus de gestion et d'exploitation de profils en modélisant la chaîne des traitements nécessaires pour aller de la situation d'apprentissage initiale jusqu'à l'exploitation des profils par les différents acteurs. Ce modèle s'organise en quatre étapes : *l'étape de constitution de profils*, *l'étape d'harmonisation de profils*, *l'étape de transformation de profils*, et l'*étape d'exploitation de profils*. L'*étape de constitution de profils* nous permet de constituer des profils d'apprenant initiaux en utilisant des logiciels ou par des enseignants. L'*étape d'harmonisation de profils* nous permet d'harmoniser des profils d'apprenant qui peuvent être sous forme papier-crayon ou numérique. L'*étape de transformation de profils* est responsable de fournir des opérateurs sur profils d'apprenant aux utilisateurs pour faire des opérations sur la structure et la donnée de profils. Et l'*étape d'exploitation de profils* permet aux différents acteurs de la situation d'apprentissage d'exploiter les informations du profil d'apprenant en fonction de leurs rôles respectifs. Pour *l'étape d'harmonisation de profils,* un langage de modélisation de profils appelé PMDL [7] est utilisé pour modéliser des profils d'apprenant venant de plusieurs sources. PMDL (Profils MoDeling Language) est un langage qui permet aux concepteurs pédagogiques de réécrire les structures des profils selon un même formalisme et d'être ensuite en mesure de les traiter au sein d'un système d'informatique unique [7]. PMDL utilise la notation de Backus Naur (BNF : Backus Naur Form) [22] et la notation utilisée dans les spécifications d'IMS [23] pour décrire des profils

d'apprenant. Un profil d'apprenant décrit par PMDL est constitué d'une par d'une partie structure et d'autre part d'une partie données propres à l'apprenant (cf. *Figure 1)*.

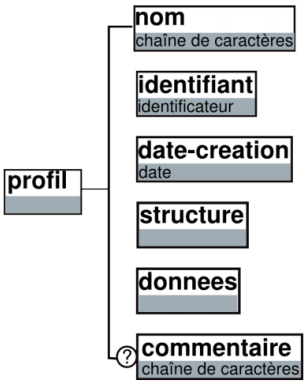


Figure 1: Schéma du profil d'apprenant défini par PMDL

2.4 Bilan de l'état de l'art

Les profils d'apprenant sont des éléments très importants qui contribuent au succès de tous les systèmes d'apprentissage éducatifs. Ces profils doivent être capables de contenir des informations de l'apprenant afin de bien présenter ou analyser ses compétences ainsi que ses démarches pendant tout son apprentissage. Les profils d'apprenant qui sont une spécialisation des profils de l'utilisateur doivent être bien modélisés pour la capacité de coopérer à travers de plusieurs systèmes grâce aux techniques modernes comme le web sémantique, l'ontologie, le RDF, l'XML…

3 Opérateurs et fonctions sur des profils d'apprenant dans l'environnement EPROFILEA

Dans cette partie, nous présentons les opérateurs (détaillé dans Annexe 1) et les fonctions (exemple dans Annexe 2) que nous proposons comme moyens permettant aux utilisateurs, mais surtout aux enseignants de pouvoir effectuer nombre des traitements sur des profils d'apprenant de l'environnement EPROFILEA. Tout d'abord, nous présentons le modèle de profil d'apprenant utilisé par l'environnement EPROFILEA. Puis, nous présentons des opérateurs et fonctions sur des profils d'apprenant venant de l'environnement EPROFILEA que nous proposons.

3.1 Le modèle de profil d'apprenant dans l'environnement EPROFILEA

Les profils d'apprenant utilisé dans l'environnement EPROFILEA sont modélisés par un langage de modélisation de profils PMDL (Profiles MoDeling Language) [7].
Le profil d'apprenant modélisé par PMDL se divise en deux parties : la partie structurelle (l'instance structurelle de profil) et la partie de données (l'instance de données de profil). Une instance structurelle de profil peut avoir plusieurs instances de donnée de profil et une instance de données de profil doit appartenir uniquement à une instance structurelle de profil.
Une instance structurelle de profil décrit la structure d'un profil d'apprenant. Il peut avoir cinq types d'élément dans une structure de profil. Premièrement, l'élément « *Infomations_eleve »* contient tous les informations administratives de l'apprenant comme son nom, son prénom, son adresse… Deuxièmement, l'élément « *Liste_composantes »* contient les informations sur les compétences de l'apprenant qui sont présentées sous forme d'un arbre des sujets étudiés (arbre des composantes) et ses évaluations sont remplies par des instances de données de cette structure. Un élément « *Liste_composantes »* peut être pondéré. Troisièmement, l'élément « *Liste_repartition »* contient aussi les informations sur les compétences de l'apprenant mais dans ce cas, ces informations

indiquent la répartition des compétences de l'apprenant entre les composantes. Quatrièmement, l'élément « *Graphe* » contient une graphe des composantes avec les liens entre composantes qui expriment les relations entre des compétences de l'apprenant. Finalement, l'élément « *Texte* » contient les informations sous forme des textes comme des commentaires.
Une instance de données de profil d'apprenant contient toutes les évaluations de l'apprenant. Sa structure est bien la même que la structure de l'instance structurelle de profil à laquelle elle appartient. Ici, les évaluations de profil d'apprenant sont des évaluations évolutives pour pouvoir bien tracer des compétences de l'apprenant.

3.2 Opérateurs sur des profils d'apprenant

Comme nous avons montré dans la partie précédente, un profil d'apprenant dans l'environnement EPROFILEA se compose de deux parties : la partie structurelle et la partie de données. Donc, nous proposons de diviser les opérateurs qui effectuent des traitements sur ces profils aussi en deux GROUPEs : le GROUPE des opérateurs sur la structure et le GROUPE des opérateurs sur les données. Des opérateurs sont présentés dans l'image suivante.

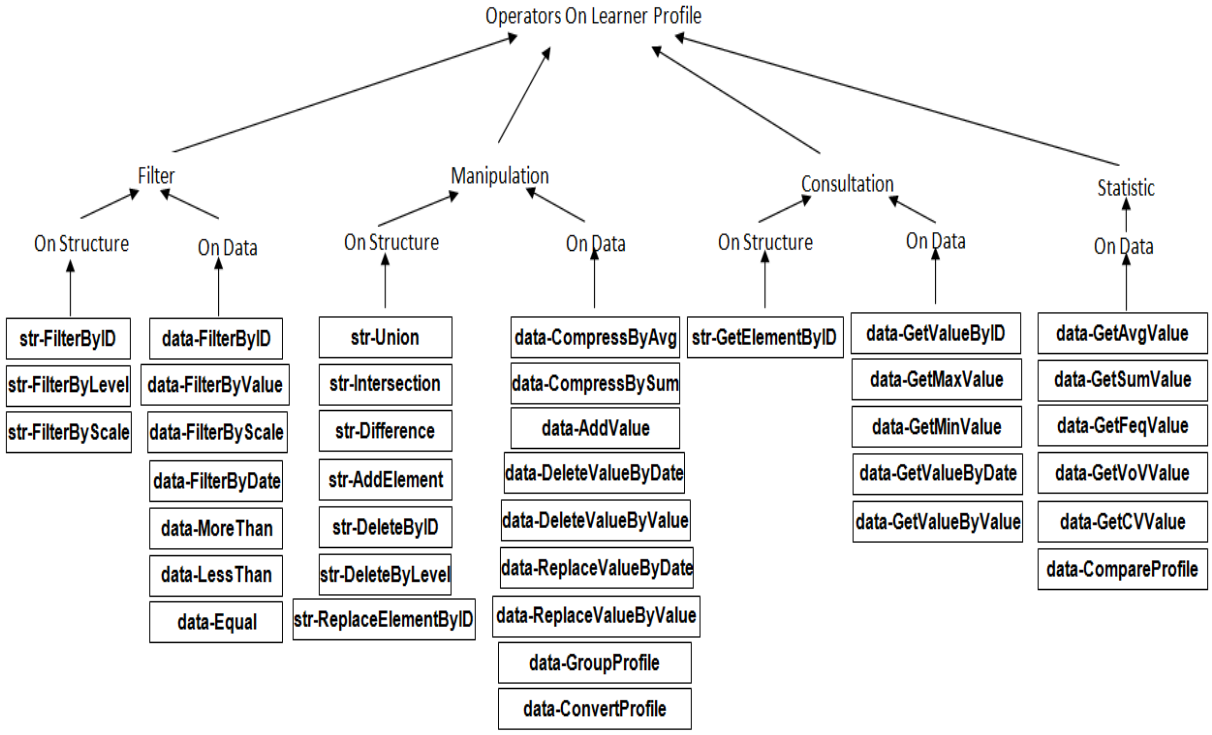


Figure 2: Des opérateurs sur des profils d'apprenant

Les opérateurs proposés s'organisent en quatre catégories : *Filtrage*, *Manipulation, Consultation*, et *Statistiques*.

3.2.1 Les opérateurs de la catégorie *Filtrage*

La catégorie *Filtrage* contient un ensemble des opérateurs qui effectuent un filtrage sur des profils d'apprenant. L'utilisateur peut faire le filtrage *sur la structure de profil* ou le filtrage *sur la donnée de profil*.

- *Le filtrage sur la structure de profil* est le traitement sur des instances structurelles de profil qui ne garde que les composantes satisfaisant la condition de filtrage. Par exemple, l'opérateur « *str_FilterByID* » filtre des composantes par identifiants. Le résultat est une instance structurelle de profil qui ne contient que les composantes dont ses identifiants se trouvent dans la liste des

identifiants entrés. Le traitement de filtrage dépend de chaque type d'élément. Par exemple dans le filtrage sur les types d'élément *Graphe* et *Texte* le traitement ne s'effectue qu'au niveau de l'élément, par contre, le filtrage sur les types d'élément *Liste_composante* et *Liste_repartition* s'effectue non seulement au niveau de l'élément mais aussi au niveau de la composante.

- *Le filtrage sur les données de profil* est le traitement sur des instances de données de profil qui ne garde que les évaluations satisfaisant la condition de filtrage. C'est-à-dire que la partie structurelle des profils n'est changée en rien. C'est une différence importante entre le *filtrage sur la structure* et le *filtrage sur les données de profil*. Par exemple, l'opérateur « *data_FilterByID »* filtre des évaluations de composantes dont ses identifiants se trouvent dans la liste des identifiants entrées. L'opérateur « *data_FilterByDate »* filtre des évaluations des composantes dont la date est comprise dans l'intervalle de date entré. Les opérateurs « *data_MoreThan », « data_LessThan », « data_Equal »* permettent de filtrer des évaluations d'un profil d'apprenant qui sont supérieures, inférieures ou égales aux évaluations d'un autre profil d'apprenant.

3.2.2 Les opérateurs de la catégorie *Manipulation*

Les opérateurs de cette catégorie permettent aux utilisateurs de réaliser nombre de traitements manipulant des profils d'apprenant. Ces traitements peuvent être des traitements d'ajoute, de suppression, de modification, de compression, de synthèse, ou de conversion de profils d'apprenant. Comme la catégorie présentée précédemment, cette catégorie se compose aussi de deux parties : *manipulation sur la structure de profil* et *manipulation sur les données de profil*.

- *La manipulation sur la structure de profil* permet aux utilisateurs de manipuler des instances structurelles de profil d'apprenant et le résultat des traitements est évidement une instance structurelle de profil. Dans ce GROUPE des opérateurs, nous proposons sept opérateurs qui sont *str_Union, str_Intersection, str_Diffenrence, str_AddElement, str_DeleteByID, str_DeleteByLevel, str_ReplaceElementByID.* Ces opérateurs permettent aux utilisateurs de faire l'union, l'intersection, la différence sur des profils d'apprenant ou l'ajout d'un élément, la suppression d'un élément par son identifiant, la suppression d'un élément par le niveau ou le replacement d'un élément par un autre. Ces sont des opérateurs très utiles pour la reconstruction des structures de profils d'apprenant.
- *La manipulation sur les données de profil* est responsable de fournir aux utilisateurs les traitements sur des instances de données des profils d'apprenant. Ce GROUPE d'opérateurs se compose de neuf opérateurs qui réalisent la compression, la synthèse, la transformation ainsi que l'ajoute, la suppression, le remplacement sur des évaluations de profils d'apprenant. Les opérateurs « *data_CompressByAvg »* et « *data_CompressBySum »* permettent aux utilisateurs de compresser des évaluations de profils d'apprenant en remplaçant les valeurs actuelles par la valeur moyenne ou la valeur totale de ces valeurs. L'opérateur « *data_AddValue »,* « *data_DeleteValueByDate », « data_DeleteValueByValue »* permettent d'ajouter ou supprimer des évaluations. L'opérateur « *data_ReplaceValueByDate », « data_ReplaceValueByValue »* permettent de remplacer des évaluations par des autres évaluations. L'opérateur « *data_GroupProfile »* permet de faire la synthèse sur deux instances de données de profil en constituant une autre instance de données de profil dont les évaluations sont les valeurs moyennes des évaluations de ces deux profils. Enfin l'opérateur « *data_ConvertProfile »* permet de convertir des évaluations d'une instance de données de profil en utilisant une fonction de conversion entrée.

3.2.3 Les opérateurs de la catégorie *Consultation*

Les opérateurs de la catégorie *Consultation* permettent aux utilisateurs de récupérer les données souhaitées selon la condition entrée. Les opérateurs de cette catégorie se divisent en deux

GROUPEs : les opérateurs de *consultation de la structure* et les opérateurs de *consultation des données* des profils d'apprenant.
Le GROUPE des opérateurs de *consultation de la structure* ne contient qu'un opérateur appelé « *str_GetElementByID »*. Cet opérateur nous permet de récupérer la structure d'un élément d'une instance structurelle de profil d'apprenant selon l'identifiant entré.
Le GROUPE des opérateurs de *consultation des données* contient des opérateurs permettant de récupérer les évaluations d'une instance de données de profil d'apprenant. On peut récupérer les évaluations par leur identifiant, leur date de création, ou par la comparaison avec une autre valeur en utilisant des opérateurs « *data_GetValueByID », « data_GetValueByDate » et « data_GetValueByValue ».* En utilisant des opérateurs « *data_GetMaxValue », « data_GetMinValue »*, on peut récupérer les évaluations les plus grandes ou les plus petites parmi les évaluations du même élément.

3.2.4 Les opérateurs de la catégorie *Statistiques*

Les opérateurs de la catégorie *Statistiques* nous fournissent les opérations statistiques sur des évaluations d'une instance de données de profil d'apprenant. À la différence des catégories précédentes, les opérateurs de cette catégorie sont dans un même GROUPE : *statistique sur les données* de profils d'apprenant. Cette catégorie se compose de six opérateurs. Les opérateurs « *data_GetAvgValue »* et « *data_GetSumValue »* permettent de calculer la valeur moyenne ou la valeur totale des évaluations d'un élément donné pour une instance de données de profil d'apprenant. Les opérateurs « *data_GetFeqValue », « data_getVoVValue » et « data_GetCVValue »* permettent de calculer la fréquence, la variance ou le coefficient de variation des évaluations d'une instance de donnée de profil d'apprenant. Et l'opérateur « *data_CompareProfile »* nous permet de faire la comparaison entre des évaluations d'une instance de données de profil d'apprenant et des évaluations d'une autre instance. On peut donc savoir si un apprenant est meilleur qu'un autre apprenant ou non pour une compétence donnée.

3.3 Fonctions de transformation de profils d'apprenant

Pour mieux assister les utilisateurs dans les traitements sur des profils d'apprenant, nous proposons des fonctions qui permettent aux utilisateurs de combiner des opérateurs sur des profils d'apprenant. Les fonctions peuvent être échangées parmi des utilisateurs pour la réutilisation des traitements sur des profils d'apprenant.
Les fonctions proposées sont écrites en utilisant un langage à balises. Avec un ensemble de six types de balises, l'utilisateur peut définir les fonctions souhaitées. Chaque fonction se compose de plusieurs opérations sur des profils d'apprenant.

3.3.1 Les balises proposées pour la définition de fonctions

La balise « *Execution »* est la racine d'une fonction. Dans chaque fonction, il n'y a qu'une balise « *Execution »* qui signale le début de la fonction avec une suite d'opérations contenues cette balise. Il y a des propriétés obligatoires dont l'utilisateur doit spécifier la valeur comme la propriété « *Name »* représentant le nom de la fonction, la propriété « *ReturnType »* représentant le type de données du résultat de la fonction. Il y a aussi des propriétés optionnelles dont la valeur peut être spécifiée si l'utilisateur la souhaite comme la propriété « *Desc* » représentant la description de la fonction ou la propriété « *SaveAs »* représentant le chemin de l'endroit où le résultat de la fonction est stocké. (cf. *Figure 3*)

Figure 3: La balise "*Execution*"

La balise « *Operation* » signale une opération à exécuter. Dans une fonction, il peut avoir une ou plusieurs balises « *Operation* ». Chaque balise « *Operation* » contient un opérateur sur profil d'apprenant. Il y a des propriétés obligatoires comme la propriété « *ID* » représentant l'identifiant de l'opération dans la fonction, la propriété « *Operator* » représentant le nom de l'opérateur sur profil d'apprenant trouvé dans la description des opérateurs sur des profils d'apprenant. Il y a aussi des propriétés optionnelles comme la propriété « *Desc* ». (cf. *Figure 4*)

Figure 4: La balise "Operation"

La balise « *Input* » signale la donnée d'entrée pour l'exécution de la fonction. Cette balise est déclarée seulement au niveau de la fonction. On l'utilise pour définir les entrées qui peuvent être spécifiées seulement au moment de l'exécution de la fonction. Il y a aussi des propriétés obligatoires comme la propriété « *ID* » représentant l'identifiant de la balise, la propriété « *DataType* » représentant le type de données de l'entrée et la propriété optionnelle « *Desc* » représentant la description. (cf. *Figure 5*)

Figure 5: La balise "Input"

La balise « *InputRef* » spécifie la référence à une balise « *Input* ». La valeur de cette balise est obligatoire et indique l'identifiant de la balise « *Input* » référencée. (cf. *Figure 6)*

Figure 6: La balise "InputRef"

La balise *« Parameter »* représente un paramètre d'une opération. La donnée entrée pour le paramètre peut être des valeurs spécifiées directement par l'utilisateur, le résultat d'une opération exécutée précédemment, ou la référence à la balise « *Input* ». La seule propriété qui est obligatoire dans cette balise est la propriété « *Name* » représentant le nom du paramètre trouvé dans la description des opérateurs. On a trois types de paramètre : le paramètre du type 1 contenant des valeurs constantes (cf. *Figure 7*), le paramètre du type 2 contenant des valeurs référencées aux résultats d'une opération exécutée précédemment (cf. *Figure 8*), et le paramètre du type 3 contenant des valeurs référencées à une balise « *Input* » (cf. *Figure 9*).

Figure 7: La balise "Parameter" du type 1

Figure 8: La balise "Parameter" du type 2

Figure 9: La balise "Parameter" du type 3

La balise « *ResultOf* » signale que la valeur de cette balise est le résultat d'une opération exécutée précédemment à l'opération contenue. La valeur textuelle de cette balise est l'identifiant de l'opération dont le résultat est récupérer (cf. *Figure 10*).

Figure 10: La balise "ResultOf"

3.3.2 Les types de données proposés utilisés par les fonctions

Nous définissons ici les types de données utilisés par les fonctions. Actuellement, il y a treize types de donnée proposés qui sont décrits dans le tableau suivant (cf. *Table 1*).

Table 1. Les types de données utilisés dans le modèle des fonctions.

Nom	Description	Inline
StructP	structure de profil d'apprenant	Oui
DataP	données de profil d'apprenant	Oui
Element	élément de la structure de profil d'apprenant	Non
ListElems	liste des éléments de la structure de profil d'apprenant	Non
Eval	évaluation de profil d'apprenant	Non
ListEvals	liste des évaluations de profil d'apprenant	Non
Function	instance d'une fonction de comparaison utilisée par des opérateurs	Non
Converter	instance d'une fonction de conversion utilisée par des opérateurs	Non
Date	date	Oui
String	chaîne des caractères	Oui
ListString	liste de chaînes des caractères : chaque chaîne des caractères est séparée de la suivante par un point-virgule	Oui
Double	nombre décimal	Oui
Int	nombre entier	Oui

Dans le tableau, la colonne « Inline » indique que les valeurs de ce type peuvent être directement évaluées à la compilation de la fonction.

4 Implémentation

Dans cette partie, nous présentons comment nous mettons en œuvre nos propositions théoriques au sein du module GROUPE : Nous présentons l'architecture côté serveur et l architecture côté client qui permettent de s'adapter à plusieurs environnements informatiques d'apprentissage humain (EIAH) sans ayant besoin de modifier les traitements principaux sur les profils d'apprenant. Les diagrammes de classes peuvent être trouvés dans (Annexes 3). Nous commerçons par présenter l'architecture du module côté serveur.

4.1 L'architecture du module côté serveur

Les opérateurs sur des profils d'apprenant sont centralisés sur le serveur et distribués utilisateurs sous forme de services web.

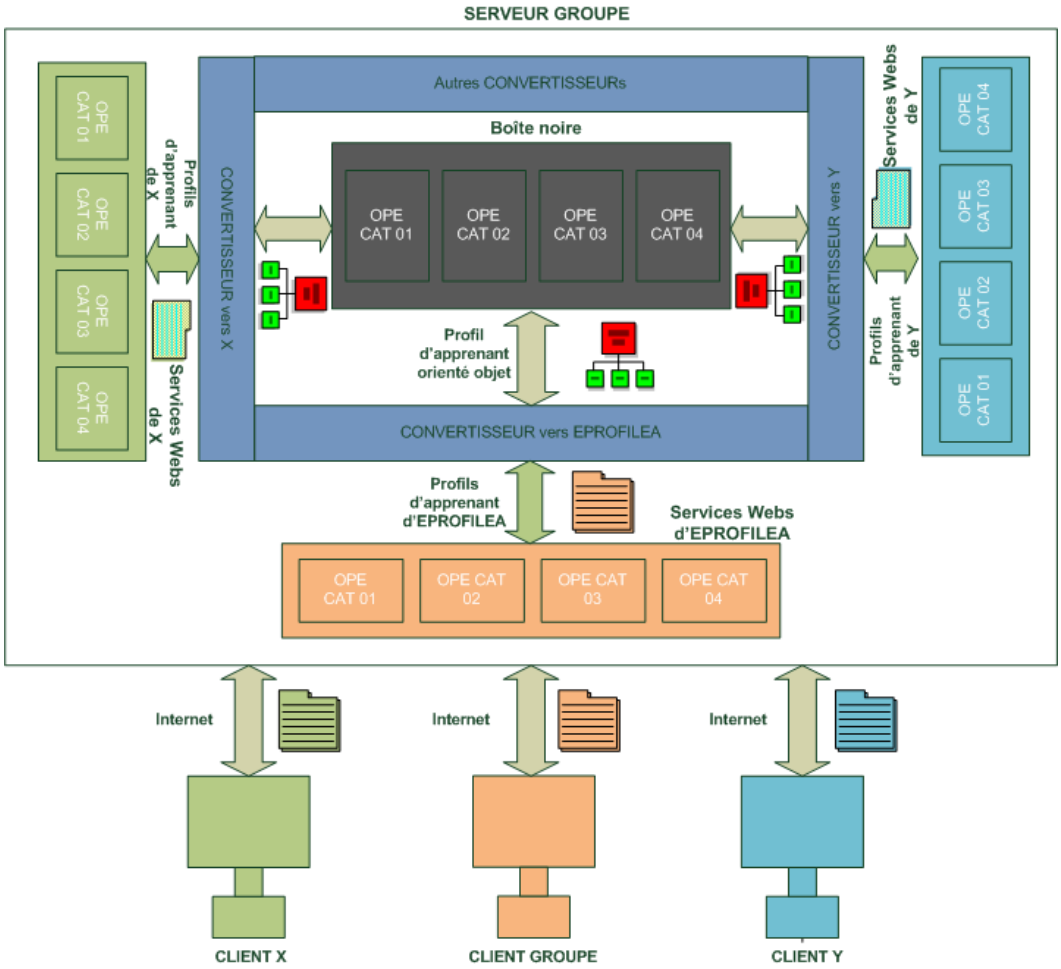


Figure 11: L'architecture du module GROUPE côté serveur

La Figure 2 illustre l'architecture du module GROUPE côté serveur. Cette architecture permet des utilisations des opérateurs à distance et fournit des traitements sur des profils d'apprenant de manière équivalent, même dans des contextes différents. Les notions « OPE CAT 01 », « OPE CAT 02 », « OPE CAT 03 », « OPE CAT 04 » sont des opérateurs sur des profils d'apprenant présentés précédemment.

Au centre de l'architecture, nous construisons une *boîte noire* qui contient des traitements sur des profils d'apprenant. Les profils d'apprenant utilisés dans cette boîte noire sont des profils d'apprenant orientés objets comme des nœuds, des arbres, des pointeurs. La structure orientée objet des profils d'apprenant nous permet de réaliser plus facilement des traitements complexes sur des profils d'apprenant comme l'union de deux structures de profils ou la synthèse des profils d'apprenant.

Autour de la boîte noire, nous définissons des *convertisseurs* qui transforment des profils d'apprenant venant de plusieurs logiciels d'apprentissage (EIAH) en profils d'apprenant orientés objets. Selon chaque EIAH, un convertisseur spécifique est construit pour la transformation des profils. Actuellement, pour le projet PERLEA, nous avons construit un convertisseur appelé *convertisseur vers EPROFILEA* pour la transformation des profils d'apprenant venant de l'environnement EPROFILEA en des profils d'apprenant utilisables par la boîte noire.

Finalement, autour des convertisseurs, des *services web* seront utilisés directement par des utilisateurs. Ces services web sont construits en se basant sur des traitements de la boîte noire. Ils reçoivent des profils d'apprenant envoyés par l'utilisateur, utilisent les traitements appropriés dans la boîte noire pour traiter ces profils et puis renvoient le résultat à l'utilisateur. Pendant l'exécution, les profils d'apprenant envoyés par l'utilisateur sont transformés en profils d'apprenant orientés objets qui seront à nouveaux transformés dans le format initial grâce aux convertisseurs appropriés.

A côté de l'avantage de l'adaptation à plusieurs systèmes, l'utilisation des traitements à distance, cette architecture permet aussi la collaboration entre des systèmes d'apprentissage informatique. Un EIAH peut utiliser les traitements sur ses profils d'apprenant, puis transformer les résultats en profils d'apprenant d'un autre système grâce à la boîte noire et les convertisseurs afin de lui permettre de réutiliser ces résultats. Un système peut donc utiliser des profils d'apprenant venant des autres systèmes ou émettre ses profils d'apprenant pour des autres systèmes à les réutiliser.

4.2 L'architecture du module côté client

Comme nous l'avons présenté dans la partie précédente, le serveur des traitements sur des profils d'apprenant peut accepter beaucoup de clients qui utilisent les services web du serveur. Parmi ces clients, le module GROUPE de l'environnement EPROFILEA est le client qui a été implémenté dans le cadre de ce travail. Avant de parler de l'architecture du module côté client, nous présentons globalement le module GROUPE de l'environnement EPROFILEA.

4.2.1 Le module GROUPE au sein de l'environnement EPROFILEA

L'environnement EPROFILEA est composé de plusieurs modules parmi lesquels le module GROUPE.

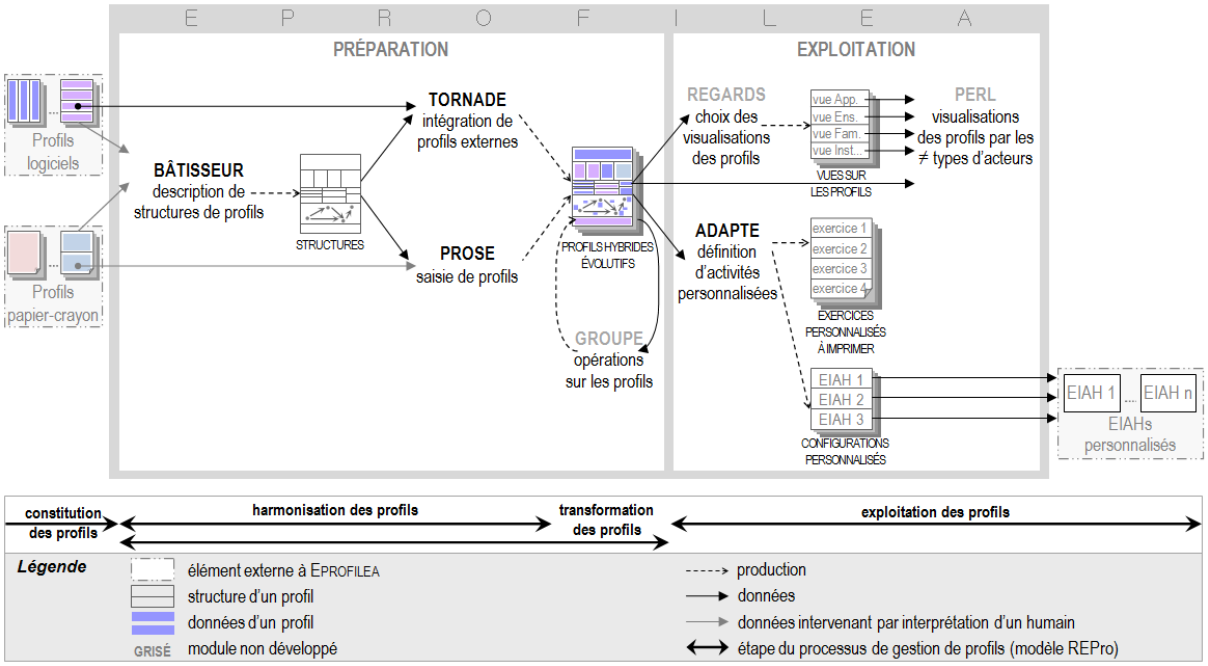


Figure 12: Le module GROUPE de l'environnement EPROFILEA

Dans l'environnement EPROFILEA, le rôle du module GROUPE permet à l'enseignant d'effectuer des opérations sur des profils d'apprenant. Il se trouve dans la phase de préparation contenant des modules effectuant la création et la manipulation des profils d'apprenant. Puisque le module GROUPE est un module indépendant des autres modules, l'utilisateur peut lancer ce module au moment où il veut. Les résultats des traitements sur des profils d'apprenant de ce module peuvent être réutilisés par la suite par les autres modules. Le module GROUPE joue aussi un rôle de pont reliant la phase de préparation et la phase d'exploitation.

4.2.2 L'architecture du module GROUPE côté client

Nous avons construit le module GROUPE comme une « *applet* » qui peut être lancée par un navigateur Web avec la machine virtuelle JAVA installée chez utilisateur. Le module peut à la fois communiquer avec des services web du serveur et avec le système de fichiers stocké en local.

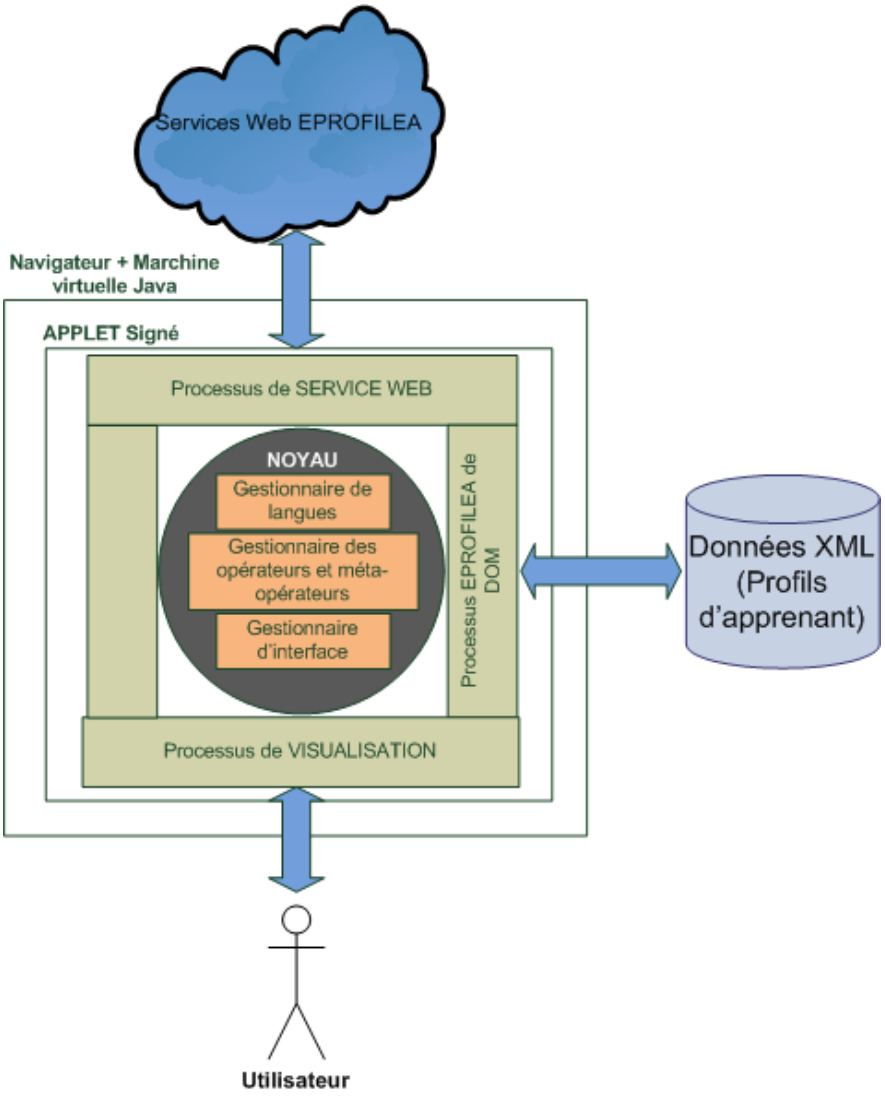


Figure 13: L'architecture du module côté client

Actuellement, il y a trois types d'acteurs possibles qui interagissent avec le module GROUPE. L'acteur *Services Web EPROFILEA* fournit des traitements sur des profils d'apprenant en recevant les profils d'apprenant envoyés par le module et retournant les résultats au module près avoir traité sur les profils d'apprenant reçus. L'acteur *Données XML* est le système des fichiers XML contenant les informations de profil d'apprenant. Et l'acteur *Utilisateur* est l'utilisateur d'EPROFILEA.
Le module GROUPE utilise trois *Processeur* pour communiquer avec les trois acteurs présentés en Français au-dessus. Le *Processeur de SERVICE WEB* se compose de composants qui prennent des profils d'apprenant, les transforment en format transférable à travers l'Internet, invoquent des services web pour réaliser des traitements et revoient le résultat après l'avoir transformé en format lisible par le module. Le *Processeur EPROFILEA de DOM* contient des traitements sur les fichiers XML qui sont la lecture et l'écriture des profils d'apprenant dans les fichiers XML. Le *Processeur de VISUALISATION* est utilisé pour offrir une interface riche à l'utilisateur. Il comporte la visualisation des profils d'apprenant, les éléments d'interface et les messages affichés pendant l'exécution.

Au centre de cette architecture, un ensemble des traitements très important appelé le *Noyau* qui permet le bon fonctionnement du module. Le noyau contient trois *Gestionnaires*. Le *Gestionnaire de langues* gère les langues du module. Il permet à l'utilisateur de choisir la langue pour le module. Pour l'instant, nous fournissons l'Anglais et le Français comme les options de langue du module. Le *Gestionnaire d'interface* gère tous les objets d'écran dans le module. Il permet à l'utilisateur de réaliser les traitements sur des profils d'apprenant l'étape par l'étape ou de retourner à l'étape précédent. Le *Gestionnaire des opérateurs et fonctions* est utilisé pour effectuer des prétraitements ou post-traitements sur des opérateurs ou des fonctions de composition d'opérateurs, par exemple, pour la préparation des entrées ou le traitements sur des résultats retournés.

4.3 Le module GROUPE du point de vue de l'utilisateur

Dans cette partie, nous présentons le module GROUPE afin d'illustrer la mise en œuvre de nos modèles dans l'environnement EPROFILEA. Actuellement, nous avons implémenté vingt-trois écrans permettant de manipuler sur les vingt-trois opérateurs ainsi que trois écrans permettant de manipuler les fonctions.

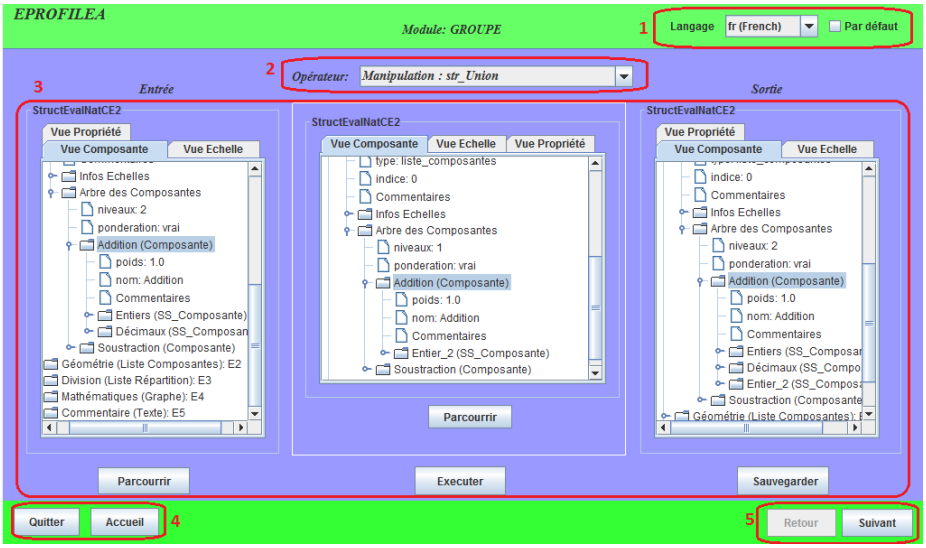


Figure 14: Écran de l'opérateur str_Union

La première zone est la *zone de langue* (cf.1. *Figure 14*). Cette zone permet à l'utilisateur de choisir la langue utilisée à l'interface et de la mettre comme langue par défaut pour le module. La deuxième zone est la *zone des choix des opérateurs* (cf.2 *Figure 14*). Cette zone permet à l'utilisateur de sélectionner l'opérateur à appliquer sur les profils d'apprenant. La troisième zone est la *zone de contenu de l'opérateur* (cf.3. *Figure 14*) permettant à l'utilisateur de choisir les entrées et de voir le résultat retourné. La quatrième zone à gauche est *la zone de contrôle général* (cf.4. *Figure 14*). L'utilisateur peut choisir de quitter le module ou de retourner à l'écran d'accueil en utilisant deux buttons dans cette zone. Enfin, la cinquième zone est la *zone de gestion de l'exécution* (cf.5. *Figure 14*). Cette zone permet à l'utilisateur de choisir de passer à l'étape suivante ou de retourner à l'étape précédente dans le sens de l'exécution. Le profil d'apprenant résultat de chaque étape est bien gardé pour des étapes suivantes.

La « *Figure 14* » présente l'écran correspondant à l'utilisation de l'opérateur « *str_Union* ». Dans la zone de contenu de l'opérateur, il y a deux entrées sous forme de deux instances structurelles de profil d'apprenant dont on souhaite faire l'union et une sortie sous forme du résultat de l'union de deux instances entrées.

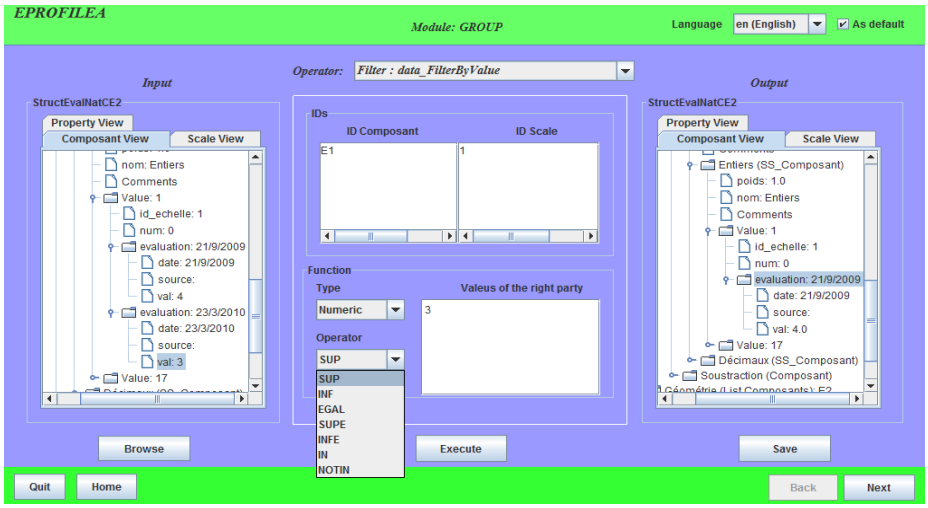


Figure 15: Écran de l'opérateur data_FilterByValue

La « *Figure 15* » montre l'écran de l'opérateur « *data_FilterByValue* ». Cet écran permet à l'utilisateur de faire le filtrage selon des valeurs sur une instance de données de profil d'apprenant. L'utilisateur peut choisir le profil d'apprenant entré, spécifier les paramètres pour exécuter l'opérateur comme la liste des identifiants de composant, la liste des identifiants d'échelle, et appliquer la fonction de filtrage sur des valeurs. Actuellement, nous avons défini deux types de fonctions : une fonction de type textuel traitant des valeurs textuelles et une fonction du type numérique traitant des valeurs numériques, cinq opérateurs arithmétiques : SUP (supérieur), INF (inférieur), EGAL (égal), SUPE (supérieur ou égal), INFE (inférieur ou égal) et deux opérateurs arithmétiques et textuels : IN (existe) et NOTIN (n'existe pas).

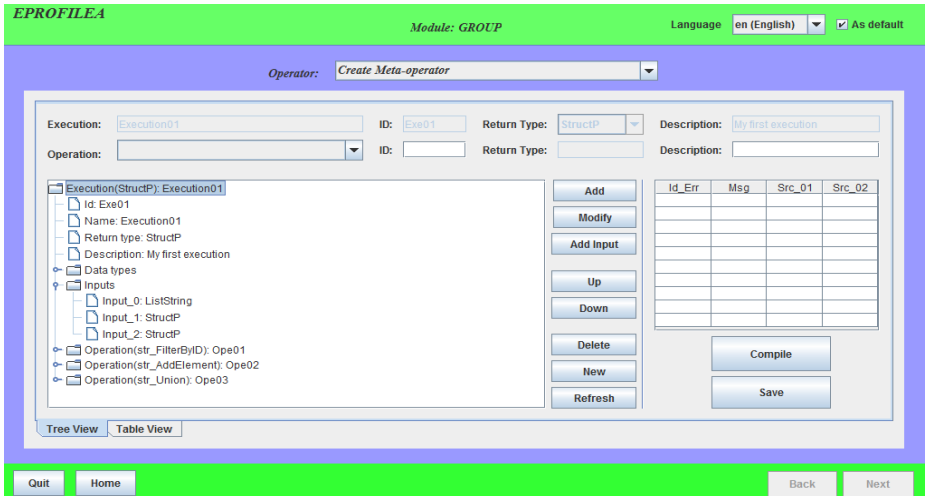


Figure 16: Écran de GROUPE permettant de créer des fonctions

La « Figure 16 » présente l'écran permettant de créer une fonction. L'utilisateur peut spécifier le nom, l'identifiant, le type de données résultat, et la description de la fonction. Pour chaque opération ajoutée, l'utilisateur peut aussi choisir dans la liste l'opérateur souhaité qui est désigné par son nom, son identifiant et sa description. Il peut également spécifier des entrées pour la fonction. Par rapport aux paramètres des opérations, utilisateur peut les spécifier de façon très simple et efficace. Après avoir défini toutes les informations pour la fonction, l'utilisateur peut choisir la fonction de compiler pour vérifier s'il y a des erreurs dans la fonction. Il peut aussi modifier des informations qui ont été saisies ou sauvegarder la fonction sous forme de fichier XML.

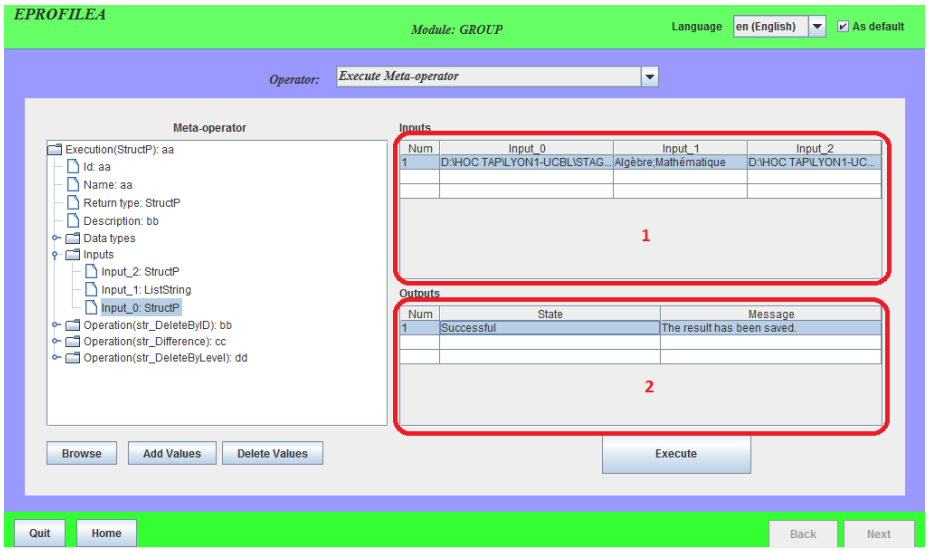


Figure 17: Écran d'exécution d'une fonction

La « *Figure 17* » montre l'écran qui permet à l'utilisateur d'exécuter une fonction. L'utilisateur peut choisir la fonction à exécuter et cette fonction est compilée automatiquement par le module au moment de son lancement. Si la compilation finit avec succès, l'utilisateur peut définir des entrées pour l'exécution de la fonction. Les entrées de la fonction sont détectées automatiquement par le module et sont présentées par des colonnes dans le tableau des entrées (cf.1. *Figure 17*). Après d'avoir spécifié tous les entrées nécessaires, l'utilisateur peut lancer l'exécution de la fonction. Les résultats de l'exécution sont affichés dans le tableau des sorties (cf.2. *Figure 17*).

5 Conclusion et perspectives

Le travail de recherche présenté dans ce rapport se compose de deux parties. La partie théorique consistant à étudier le concept d'opérateurs sur des profils d'apprenant au sein du projet PERLEA et à proposer des modèles décrivant les traitements applicables à des profils d'apprenants écrits dans le langage de description de profils PMDL. La partie pratique visait à mettre en œuvre des opérateurs proposés dans un module permettant d'effectuer des traitements sur les profils d'apprenant dans l'environnement EPROFILEA. Au-delà de cet objectif initial, nous avons également étendu les modèles décrivant des opérateurs avec des fonctions permettant de créer des fonctions combinant plusieurs opérateurs.

En ce qui concerne le travail théorique, nous avons présenté notre modèle des opérateurs qui propose de nombreux traitements sur des profils d'apprenant venant de l'environnement EPROFILEA. Le nombre actuel d'opérateurs est de trente-huit et pourra être augmenté si de nouveaux besoins sont identifiés. Grâce aux opérateurs, l'utilisateur peut réaliser des traitements aussi bien sur des instances structurelles que sur des instances de donnée des profils d'apprenant.

En ce qui concerne le travail pratique, nous sommes arrivées à mettre en place sur un serveur tous les opérateurs proposés et les mettre en disposition à l'utilisateur sous forme de services web. Côté client, nous avons construit le module GROUPE qui permet à l'utilisateur d'EPROFILEA, surtout des enseignants, d'exploiter et d'exécuter les opérateurs. Nous avons aussi proposé et implémenté un modèle des fonctions qui permet à l'utilisateur de définir des fonctions en combinant les opérateurs pour effectuer automatiquement une chaîne de traitements sur des profils d'apprenant.

Bien que nous utilisions le langage de programmation JAVA pour la première fois pour construire une « *applet* », nous avons offert à l'utilisateur une application entièrement fonctionnelle permettant d'afficher des informations en multi-langue. Pendant l'implémentation des opérateurs sous forme services web, nous avons rencontré un problème sérieux de transformation des profils d'apprenant en format transférable à travers l'Internet qui est finalement résolu par le « *JAXB Binding* » [20].

Selon Senach [26], le concept de l' « utilité » est défini : « *L'utilité détermine si le système permet à l'utilisateur de réaliser sa tâche, s'il est capable de réaliser ce qui est nécessaire à l'utilisateur. L'utilité couvre la capacité fonctionnelle, les performances du système, les qualités d'assistance.* ». Et selon la norme ISO 9241-18 [27], le concept de l'« utilisabilité » est défini ainsi: « *Une technologie est utilisable lorsqu'elle permet à l'utilisateur de réaliser sa tâche avec efficacité, efficience et satisfaction dans un contexte d'utilisation spécifié.* ». Le module GROUPE satisfait le concept de l' « *utilité »* et nous allons en modifier l'ergonomie afin qu'il satisfasse également le concept de l' « *utilisabilité* ». Par ailleurs, GROUPE devra également être expérimenté par des enseignants.
Pour ce travail, nous avons mis en œuvre des opérateurs sous forme de services web utilisables par les utilisateurs d'EPROFILEA. Pour l'instant, un convertisseur est construit côté serveur pour la transformation des profils externes provenant de l'environnement d'EPROFILEA en profils orientés objets afin d'être traités par la boîte noire. Pour d'autres utilisations venant d'autres EIAH comme nous l'avons proposé dans l'architecture (cf. figure 11), il reste à développer des convertisseurs selon le format des profils d'apprenant de chaque système. Nous espérons que les concepts d'ontologie et le RDF pourront être utilisés pour créer un mécanisme de transformation des profils d'apprenant de manière automatique facilitant l'adaptation des opérateurs à plusieurs EIAH.

6 Références

[1] Jean-Daubias, S., Eyssautier-Bavay, C., Lefevre, M.: Modèles et outils pour rendre possible la réutilisation informatique de profils d'apprenants hétérogènes, Sticef – Recueil (2009)

[2] Pease, A., Niles, I., Li, J.: The Suggested Upper Merged Ontology: A Large Ontology for the Semantic Web and its Application, AAAI Technical Report WS-02-11 (2002)

[3] Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., Wilamowitz-Moellendorff, M.: GUMO – the General User Model Ontology, In: Proc. of the 10th Int. Conf. on User Modeling, Edinburgh, UK (2005) 428–432

[4] Heckmann, D.: Ubiquitous User Modeling. PhD thesis, Akademische Verlagsgesellschaft Aka GmbH, Berlin, ISBN 3-89838-297-4 and ISBN 1-58603-608-4 (2006)

[5] Assad, M., Carmichael, D., Kay, J., Kummerfeld, B.: Personisad: Distributed, active, scrutable model framework for context-aware services. (2007) 55–72

[6] Abel, F., Henze, N., Krause, D., Plappert, D.: User modeling and user profile exchange for semantic web applications. In: 16th Workshop on Adaptivity and User Modeling in Interactive Systems, Wuerzburg, Germany. (2008)

[7] Eyssautier-Bavay, C.: Modèles, langage et outils pour la réutilisation de profils d'apprenants. PhD thesis, Université Joseph Fourier – Grenoble 1 (2008)

[8] Sowa, J.: Knowledge Presentation, Brooks/Cole, Pacific Grove, CA (2000)

[9] Borgo, S., Guarino, N., Masolo, C.: A Pointless Theory and Space Base on Strong Connection and Congruence, in L.C. and J.Doyle (eds.),Principles of Knowledge Representation and Reasoning(KR96), MorganKaufmann (1996)

[10] Allen, J.: Towards a general theory of action and time,ArtificialIntelligence,23,123-154. (1984)

[11] Smith,B.: Fiat Objects, Parts and Wholes: Conceptual Part-Whole Relations and Formal Mereology,11thEuropean Conferenceon Artificial Intelligence, Amsterdam,8 August 1994,15–23. (1994)

[12] Smith,B.: Mereotopology: A Theory of Parts and Boundaries, Data and Knowledge Engineering,20,287–303. (1996)

[13] Heckmann, D.: Introducing situational statements as an integrating data structure for user modeling, context-awareness and resource-adaptive computing. In: ABIS2003, Karlsruhe, Germany 283–286 (2003)

[14] Teissier, J.: Les arbres de connaissances: objet de controverse et expérimentation à suivre, Publications du Céreq (1998)

[15] Authier, M., Levy, P.: Les arbres de connaissances, La Découverte, coll. Essais, 174 pages (1993)

[16] Jalbert, P., Le portolio scolaire : une autre façon d'évaluer les apprentissages, Vie pédagogique, Vol. n° 103, p. 31-33 (1997)

[17] Stiggins, R.J, Student-Centered Classroom Assessment, Merrill Publishing Co (1994)

[18] Barrett, H.C., Electronic Portfolios, A chapter in Educational Technology: An Encyclopedia, ABC-CLIO (eds) (2001)

[19] Chantal, N., Le portfolio électronique : Quoi ? Pourquoi ? Comment ?, L'école branchée, Vol. fascicule 4 (2004)

[20] The Java™ Web Services Tutorial, For Java Web Service Developper's Pack, v2.0 (February 17, 2006)

[21] Arbre de connaissances, http://fr.wikipedia.org/wiki/Arbre_de_connaissances

[22] Peter Naur, Backus, J.W., Katz, C., Rutishauser, H., Wegstein, J.H., Bauer, F.L., McCathy, J., Samelon, K., Van Wijngaarden, A., Green, J., Perlis, A.J., Vauquoi, B., Woodger, M: Revised Report on the Algorithmic Language, ALGO 60, Communication of the ACM, Vol .6, No .1, (1963)

[23] Spécification finale v1.0 d'IMS-LIP (Mars 2001), http://www.imsglobal.org/profiles/lipbest01.html (Dernière consultation 2007)

[24] Zapata-Rivera, J. D., Greer, J.: Interacting with Inspectable Bayesian Student Models. (IJAIED), Vol. 14, p.1-37 (2004)

[25] Jim Reye, A Belief Net Backbone for Student Modelling, Proceedings of the Third International Conference on Intelligent Tutoring Systems, p.596-604 (1996)

[26] Senach B. (1990). Evaluation ergonomique des IHM : revue de la littérature, rapport Inria n°1180

[27] ISO 9241-18, Ergonomics of human-system interaction, http://www.iso.org/iso/catalogue_detail.htm?csnumber=40727

[28] Rueda, U., Larrañaga, M., Arruarte, A., Elorriaga, J.A.: DynMap+: A Concept Mapping Approach to Visualize Group Student Models. *Innovative Approaches for Learning and Knowledge Sharing,* (EC-TEL '06), Crete, Greece, p.383-379 (2006)

7 Annexe 1 : des opérateurs proposés sur profils d'apprenant

Dans cette partie, nous présentons en détail des opérateurs proposés sur profils d'apprenant. Pour chaque opérateur, il est détaillé en 4 parties : *Description* décrit la description générale de l'opérateur, *Rôle* décrit le but de l'opérateur, *Utilisation* décrit l'utilisation de l'opérateur, *Exemple* donne un exemple sur l'utilisation de l'opérateur. Pour tous les exemples sur l'opérateur, nous utilisons trois instances de profils d'apprenant : Struc Eval Nat CE2 (cf. *Figure 18*) est une instance structurelle, Struct CE2_V2 (cf. *Figure 19*) est une autre instance structurelle de profil d'apprenant et Eval Nat CE2 (cf. *Figure 20*) est une instance de données de profil d'apprenant.

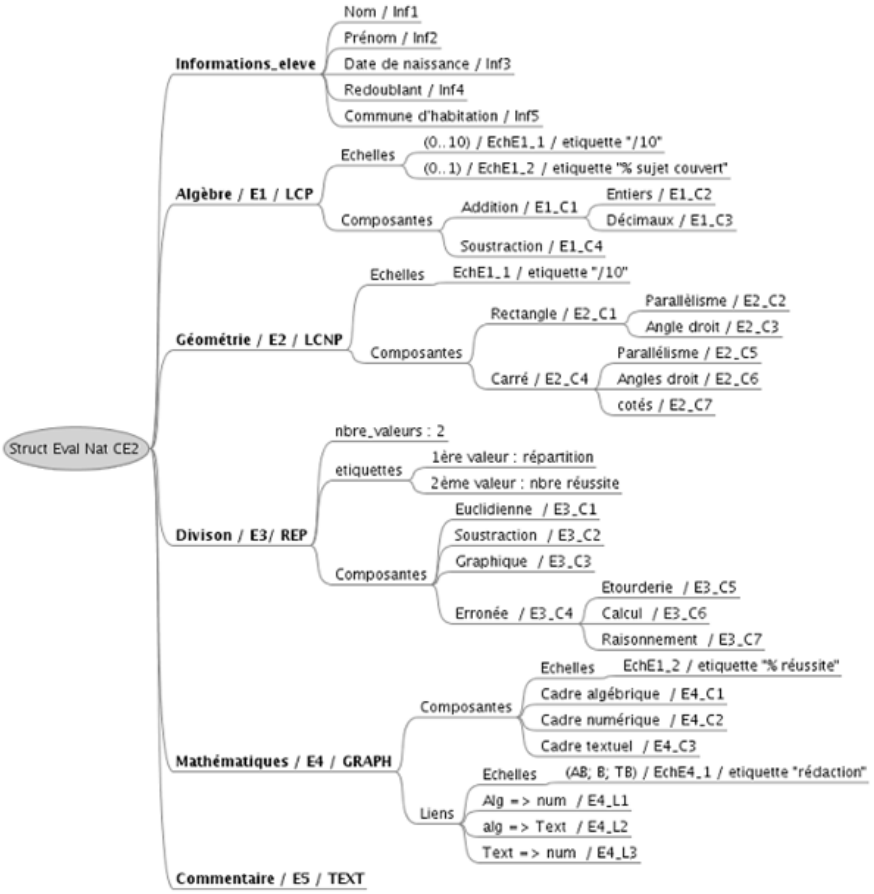


Figure 18: Struc Eval Nat CE2

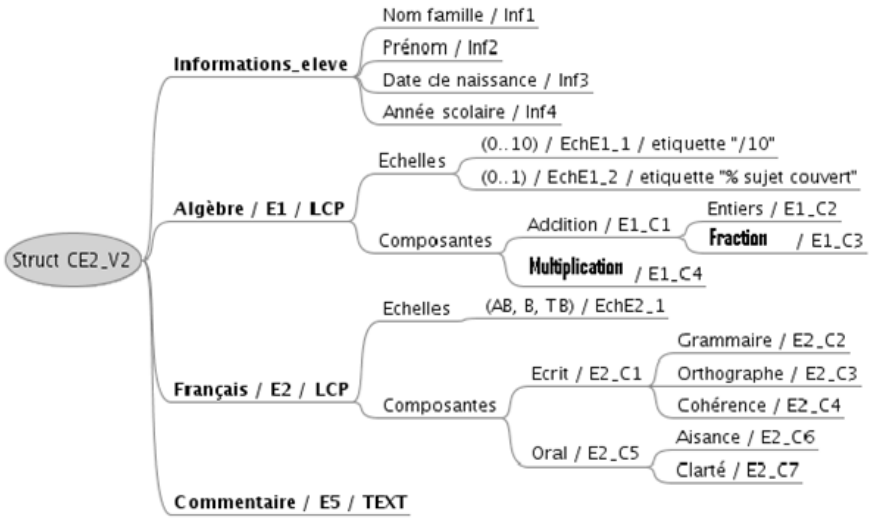


Figure 19: Struct CE2_V2

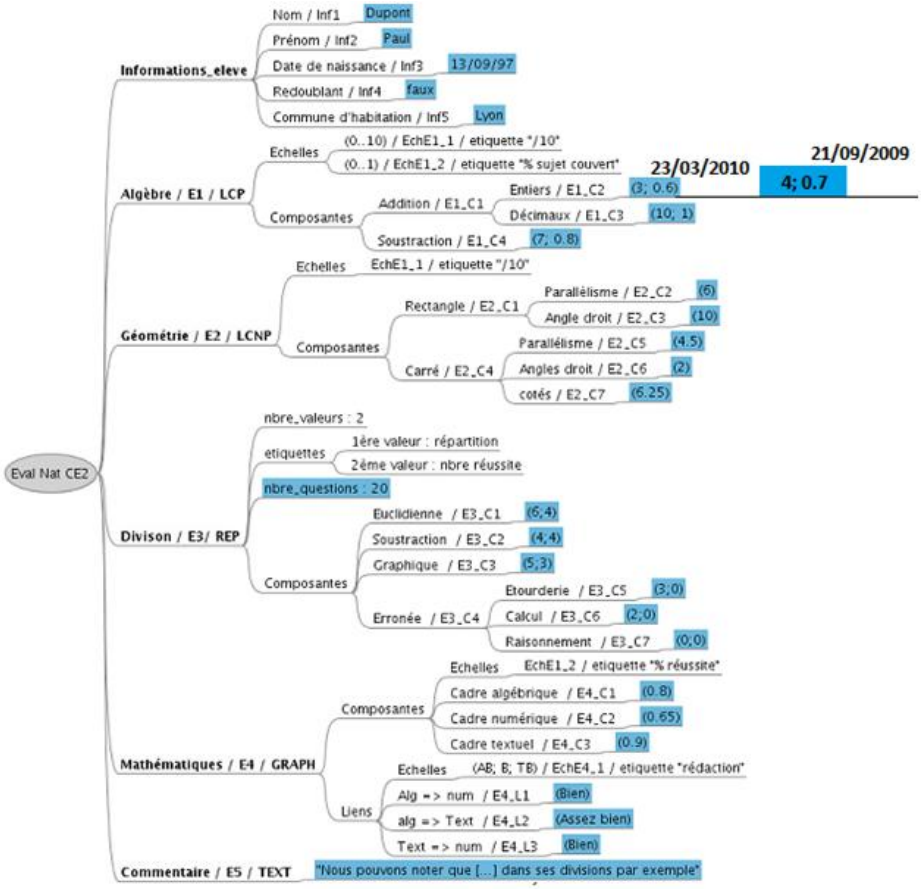


Figure 20: Eval Nat CE2

1. str-GetElementByID

Description

« str-GetElementByID » est un opérateur sur la structure de profil d'apprenant. Il prend une structure de profil d'apprenant et l'identifiant de la composante souhaité comme les entrées et puis renvoie la structure de la composante dont l'identifiant est similaire à l'identifiant entré.

Rôle

« str-GetElementByID » permet de récupérer la structure d'un élément souhaité par l'identifiant donné.

Utilisation str-GetElementByID(strucP, *list_of_IDs*)

Paramètre

strucP : La structure du profil. Type de donnée : StrucPro

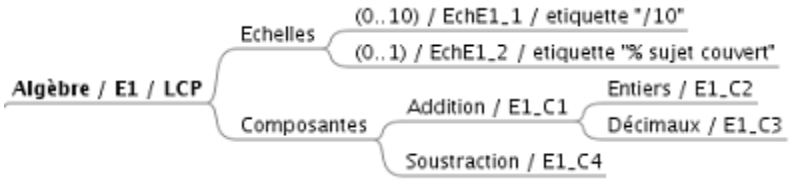
list_of_IDs: L'identifiant de l'élément souhaité. Type de donnée : chaîne des caratères

Valeur retournée : la structure des composantes trouvée ou nulle. Type de donnée retourné dépend du type de la composante trouvée. Les types de donnée retournés possible : StrucElem, StrucCompL, StrucCompL_W, StrucCompL_NotW, StrucCompR, StrucGraph, StrucLink, StrucScal

Exemple

str-GetElementByID (Struc Eval Nat CE2, {« E2 »})

Résultat :



2. str-FilterByID

Description

« str-FilterByID » est un opérateur sur la structure de profil d'apprenant. Il prend une structure de profil d'apprenant et une liste des identifiants de composante comme les entrées et donne une autre structure de profil qui ne contient que les composantes dont l'identifiant est comme un identifiant dans la liste.

Rôle

« str-FilterByID » permet de récupérer la structure de profil qui ne contient que les composantes dont l'identifiant se trouve dans la listes des identifiants entrée.

Utilisation

str-FilterByID(strucP, *list_of_IDs*)

Paramètre

strucP : la structure de profil où on a besoin de faire un filtre par les identifiants de composante. Type de donnée : StrucPro

list_of_IDs : la liste qui contient les identifiants qu'on veut utiliser pour le filtre. Type de donnée : liste des chaînes des caratères.

Valeur retournée : la structure de profil qui ne contient que les composantes dont l'identifiant se trouve dans la *liste_of_IDs*. Type de donnée : StrucPro

Note

« str-FilterByID » est appliqué sur tous les briques.

Exemple

str-FilterByID(Struct Eval Nat CE2, {'E2_C4'}) --> Struc Eval Nat CE2 filtre_V2

3. str-FilterByLevel

Description

« str-FilterByLevel » est un opérateur sur la structure de profil d'apprenant. Il prend une structure de profil d'apprenant et une valeur numérique indiquant le niveau souhaité comme les entrées et donne une autre structure de profil qui ne contient les composantes dont le niveau est inférieur ou égal au niveau entré.

Rôle

« str-FilterByLevel » permet de récupérer la structure de profil dont les composantes ont le niveau inférieur ou égal au niveau spécifié.

Utilisation

```
str-FilterByLevel( strucP,intLevel)
```

Paramètre

strucP : la structure de profil où on a besoin de faire un filtre par niveau. Type de donnée : StrucPro

intLevel : le niveau qu'on utiliser pour faire le filtre. Type de donnée : integer

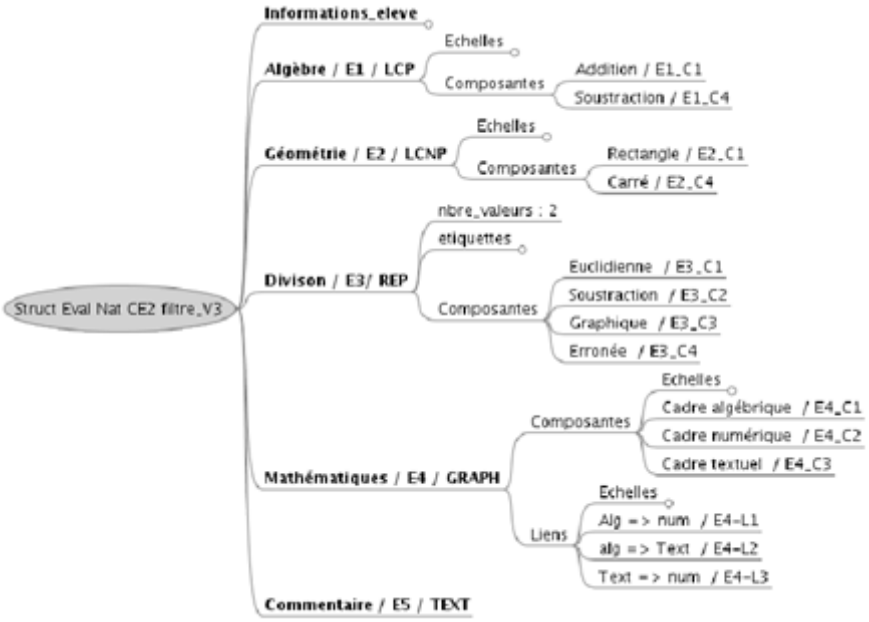
Valeur retournée : la structure de profil qui ne contient que les composantes dont le niveau est inférieur ou égal au niveau spécifié. Type de donnée : StrucPro

Note

« str-FilterByLevel » ne concerne que les briques du type *ListeComposante* et *ListeRepatition.*

Exemple

str-FilterByLevel(Struct Eval Nat CE2, 1) --> Struct Eval Nat CE2 Filtre_V3



4. str-FilterByScale

Description

« str-FilterByScale » est un opérateur sur la structure de profil d'apprenant. Il prend une structure de profil d'apprenant, et la liste des identifiants d'échelle comme les entrées. Il rend une autre structure de profil qui ne contient que les composantes ayant l'identifiant d'échelle qui a été spécifié dans la liste entrée.

Rôle

« str-FilterByScale » permet de filtre les composantes ayant l'identifiant d'échelle spécifié.

Utilisation

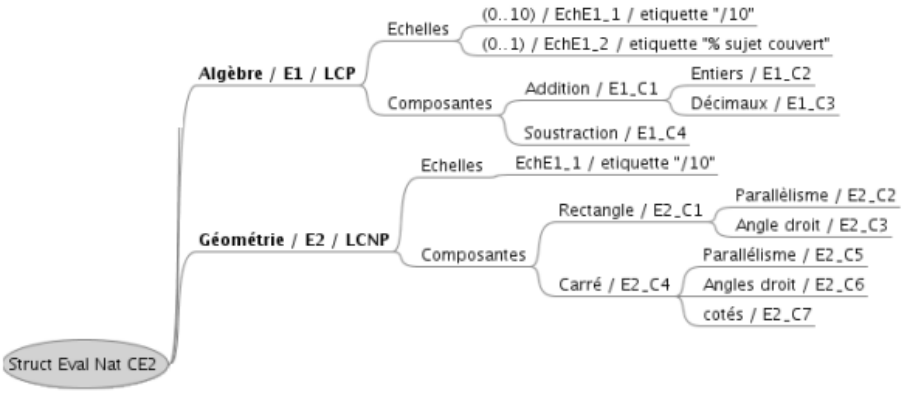
```
str-FilterByScale( strucP, list_of_idScale)
```

Paramètre

strucP : la structure de profil d'apprenant. Type de donnée : StrucPro

list_of_idScale : la liste des identifiants d'échelle souhaitée. Type de donnée : liste des chaînes des caractères

Valeur retournée : une structure de profil d'apprenant qui ne contient que les composantes ayant l'identifiant d'échelle qui se trouve dans la liste. Type de donnée : StrucPro
Note
« str-FilterByScale » ne concerne que les briques du type *ListeComposante* and *Graphe* et *ListRepartition*
ID Echelle de ListRepartion = 18
Exemple
str-FilterByScale(Struc Eval Nat CE2, {'EchE1_1'})



5. str-Union

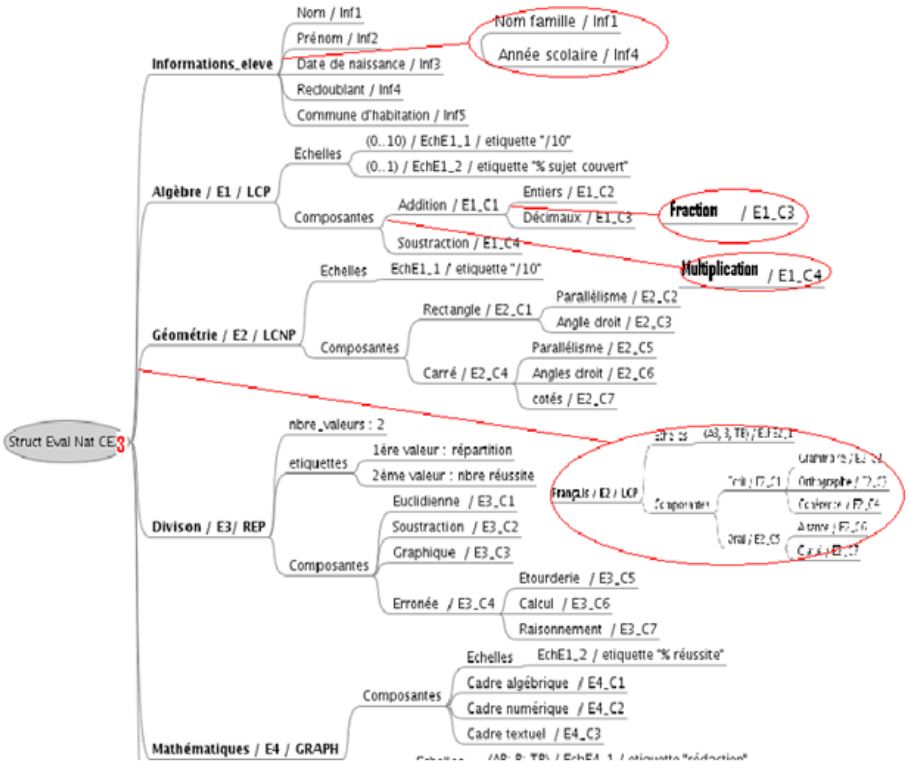
Description
« str-Union » est un opérateur sur la structure de profil d'apprenant. Il prend deux structures de profil d'apprenant comme les entrées et donne une autre structure de profil d'apprenant comme l'union de deux structures entrées.
Rôle
« str-Union » permet de faire l'union des deux structures de profil en former une troisième. L'union se fait niveau par niveau. D'abord, on transmet tous les composantes dans la première structure à la troisième structure. Puis, on parcourt du niveau de racine au niveau de feuille. Si dans un niveau il y a deux composantes similaires par rapport à la troisième structure et la deuxième structure, l'union se fait pour les enfants de ces composantes. Si une composante dans la deuxième structure n'est semblable à aucune composante dans la troisième structure dans le même niveau, on ajoute cette composante dans l'ensemble des composantes de la troisième structure. Par rapport aux échelles, on n'ajoute que les échelles différentes de la deuxième structure à l'ensemble des échelles de la troisième structure.
Comment ça marche
Le « str-Union » ne concerne pas les briques de TEXTE
Ajouter les infoEleves différentes.
Pour les briques de LISTE REPARTITION, il fait :
Comparer le nom de la brique
Comparer le nombre des valeurs
Comparer le nombre des questions
Comparer l'arbre des composantes. Pour une branche, la comparaison s'arrête au nœud différent, tous les nœuds enfants sont considèrent différents.

S'il y a une différence trouvée, la comparaison s'arrête et l'opérateur ajoute cette brique à la liste des briques résultats.
Pour les briques de LISTE COMPOSANTE, il fait :
Comparer le nom des briques
Comparer la valeur indiquant la pondération
Comparer les échelles
S'il y a une différence trouvée, l'opérateur ajoute cette brique à la liste des briques résultats et la comparaison s'arrête.
Comparer l'arbre des composantes, s'il y a une différence trouvée, le nœud courant est ajouté au le nœud parent et la comparaison s'arrête pour les nœuds enfants.
Pour les briques de GRAPHE, il fait :
Comparer tous les échelles des composantes
Comparer tous les échelles des liens
Comparer le nom de chaque composante
Pour chaque lien, comparer le nom du nœud SRC, le nom du nœud DES
S'il y a une différence dans toutes les comparaisons, la comparaison s'arrête et le Graphe est ajouté à la liste résultat.
Utilisation

str-Union(strucP1, strucP2)

Paramètres
strucP1, strucP2 : les structures de profil d'apprenant utilisées pour faire l'union. Type de donné : StrucPro
Valeur retournée : Une structure de profil d'apprenant qui se compose des composantes des deux structures entrées. Type de donné : StrucPro

Exemple
str-Union (Struct Eval Nat CE2, Struct CE2_V2) --> Struct Eval Nat CE3



6. str-Intersection

Description

« str-Intersection » est un opérateur sur la structure de profil d'apprenant. Il prend deux structures de profil d'apprenant comme entrées et donne une autre structure de profil d'apprenant comme l'intersection de deux structures entrées.

Rôle

« str-Intersection » permet de faire l'intersection des deux structures de profil en former une troisième. La structure formée ne se compose que des composantes similaires par rapport à la première et la deuxième structure. La similarité ici prend en compte les échelles des composantes.

Comment ça marche

Le « str-Intersection » ne concerne pas les briques de TEXTE.

Garder les infoEleves similaires.

Pour les briques de LISTE REPARTITION, il fait :

Comparer le nom de la brique

Comparer le nombre des valeurs

Comparer le nombre des questions

Comparer l'arbre des composantes. Pour une branche, la comparaison s'arrête au nœud différent, tous les nœuds enfants sont considèrent différents.

S'il n'y a pas de la différence trouvée, l'opérateur ajoute cette brique à la liste des briques résultats.

Pour les briques de LISTE COMPOSANTE, il fait :

Comparer le nom des briques

Comparer la valeur indiquant la pondération

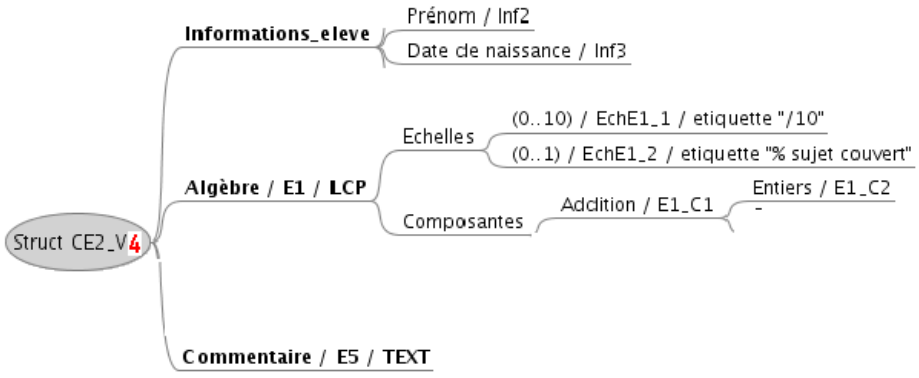
Comparer les échelles

S'il y a une différence trouvée, l'opérateur ajoute cette brique à la liste des briques résultats et la comparaison s'arrête.
Comparer l'arbre des composantes, s'il y a une différence trouvée, la comparaison s'arrête et les nœuds enfants sont supprimés dans l'arbre résultat.
Pour les briques de GRAPHE, il fait :
Comparer tous les échelles des composantes
Comparer tous les échelles des liens
Comparer le nom de chaque composante
Pour chaque lien, comparer le nom du nœud SRC, le nom du nœud DES
S'il n'y a pas de la différence dans toutes les comparaisons, on dit que les deux Graphes sont similaires.

Utilisation

str-Intersection (strucP1, strucP2)

Paramètre
strucP1, strucP2 : les structures de profil d'apprenant utilisées pour faire l'union. Type de donnée: StrucPro
Valeur retournée : la structure de profil formé. Type de donnée : StrucPro
Exemple
str-Intersection (Struct Eval Nat CE2, Struct CE2_V2) --> Struct Eval Nat CE4



7. str-Difference

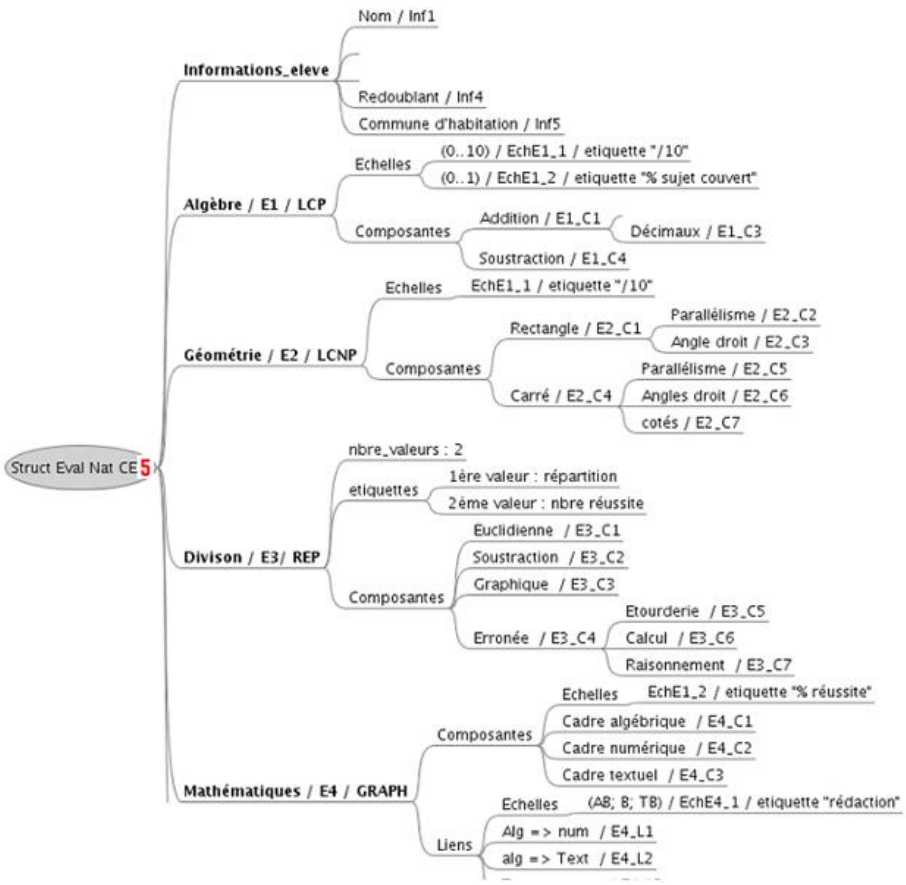
Description
« str-Difference » est un opérateur sur la structure profil d'apprenant. Il prend deux structures de profil d'apprenant comme entrées et donne une autre structure de profil d'apprenant comme l'différence de deux structures entrées.
Rôle
« str-Difference » permet de faire la différence des deux structures de profil en former une troisième. La structure formée ne contient que des composantes de la première structure qui ne se trouve pas dans la deuxième structure. La différence tient en compte des échelles des composantes.
Comment ça marche
« str-Difference » ne concerne pas les briques du type TEXTE
Pour les briques REPARTITION, il fait :
Comparer le nom de brique

Comparer le nombre de valeur et le nombre de question
Comparer des arbres de composantes
S'il y a une différence, la comparaison s'arrête et cette brique est ajoutée à la liste résultat
Pour les briques LISTE COMPOSANTE, il fait :
Comparer le nom de brique
Comparer les échelles
S'il y a une différence, la comparaison s'arrête et cette brique est ajoutée à la liste résultat
Pour l'arbre des composantes, il supprime les nœuds non-identiques.
Pour les briques GRAPHE, il fait :
Comparer tous les échelles des composantes
Comparer tous les échelles des liens
Comparer le nom de chaque composante
Pour chaque lien, comparer le nom du nœud SRC, le nom du nœud DES
S'il y a une différence, cette brique est ajoutée à la liste résultat.
Utilisation

str-Difference (strucP1, strucP2)

Paramètres
strucP1, strucP2 : les structures de profil d'apprenant utilisées pour faire l'union. Type de donnée: StrucPro
Valeur retournée : la structure de profil formé. Type de donnée : StrucPro

Exemple
str-Difference (Struct Eval Nat CE2, Struct CE2_V2) --> Struct Eval Nat CE5



8. str-AddElemment

Description

« str-AddElement » est un opérateur sur la structure de profil d'apprenant. Il prend une structure de profil d'apprenant et une structure de composante qu'on va ajouter. Il rend une autre structure où on a déjà ajouté la structure de la composante.

Rôle

« str-AddElement » permet d'ajouter un élément à une structure de profil. Il faut noter qu'on doit éviter les conflits des ID entre l'élément ajouté avec les éléments existants dans la structure.

Utilisation

```
str-AddElement (strucP, elemAdd)
```

Paramètre

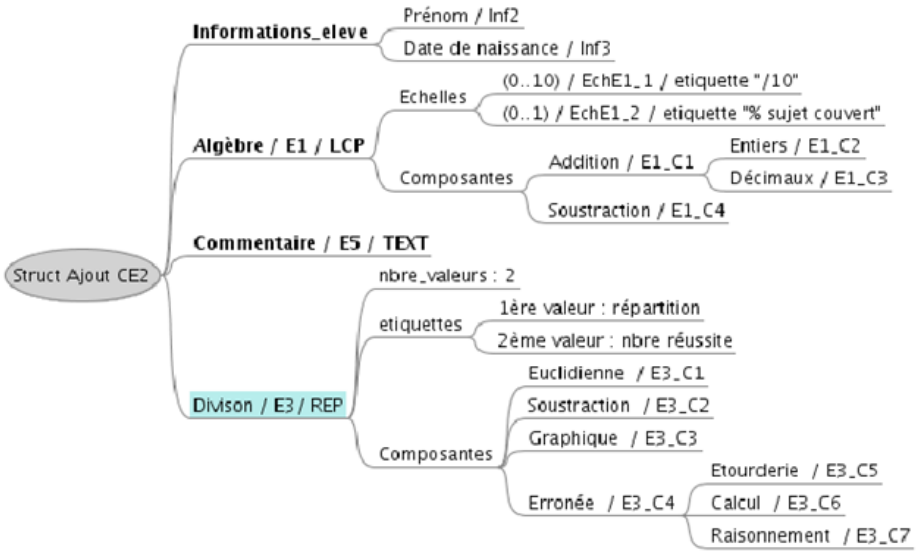
strucP : la structure de profil qui contient l'élément supprimé. Type de donnée : StrucPro

elemAdd : l'élément ajouté.

Valeur retournée : la structure après d'avoir ajouté l'élément Add à l'élément Parent. Type de donné : est le type de donnée de l'élément Parent

Exemple

str-AddElement (Struct Intersection CE2, Division)



9. str-DeleteByIDs

Description

« str-DeleteByIDs» est un opérateur sur la structure de profil d'apprenant. Il prend une structure de profil d'apprenant et l'identifiants de la composante qu'on veut supprimer. Il rend une autre structure dont la composante a été supprimée.

Rôle

« str-DeleteByIDs » permet de supprimer des éléments ou des composantes de la structure de profil donnée. L'élément supprimé est identifié par son identifiant donné

Utilisation

```
str-DeleteByIDs (strucP,list_of_strID)
```

Paramètre

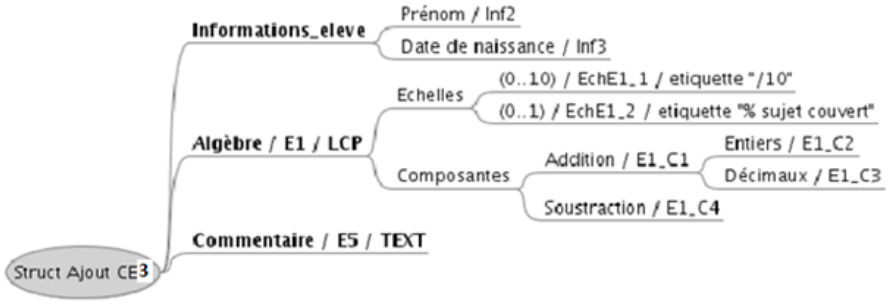
strucP : la structure de profil qui contient l'élément supprimé. Type de donnée : StrucPro

list_of_strID: l'identifiants de l'élément supprimé. Type de donnée : chaîne des caractères

Valeur retournée : la structure de profil après la suppression de l'élément donné. Type de donnée : StrucPro

Exemple

str-DeleteByID (Struc Ajout CE2, {«E3 »})



10. str-ReplaceElementByID

Description

« str-ReplaceElementByID » est un opérateur sur la structure de profil d'apprenant. Il prend une structure de profil d'apprenant, un identifiant de la composante remplacée, une structure de la composante pour remplacer comme les entrées. Cet opérateur rend une autre structure de profil d'apprenant dont la composante a été remplacée.

Rôle

« str-ReplaceElementByID » permet de remplacer une composante par une autre composante d'une structure de profil d'apprenant.

Utilisation

str-ReplaceElementByID (strucP,idComp, strucComp)

Paramètre

strucP : la structure de profil d'apprenant. Type de donnée : StrucPro

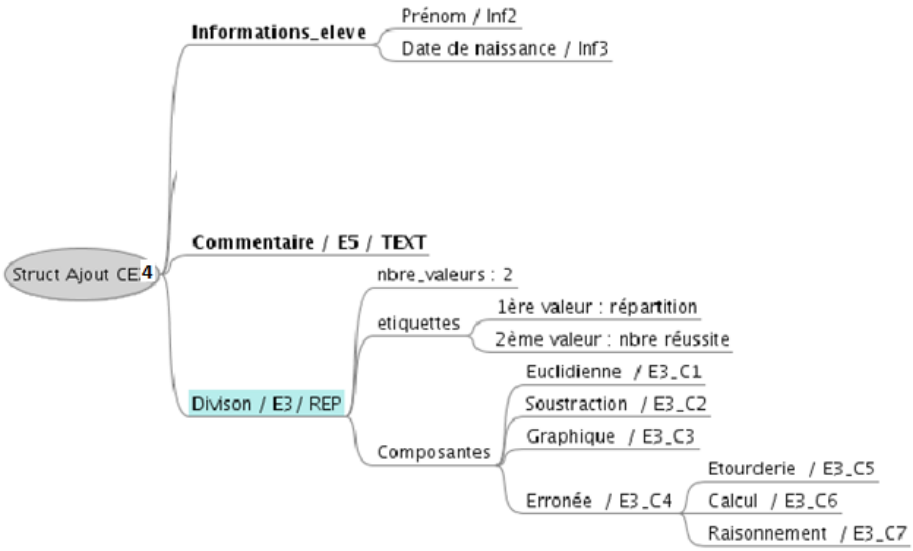
idComp : l'identifiant de la composante qui est remplacée. Type de donnée : chaîne des caractères

strucComp : la structure de la composante qu'on utilise pour remplacer. Type de donnée : StrucElem

Valeur retournée : une autre structure de profil d'apprenant dont la composante a été remplacée.

Exemple

str-ReplaceElementByID(Struc Ajout CE3, 'E1', Division)



11. str-DeleteByLevel

Description

« str-DeleteByLevel » est un opérateur sur la structure de profil d'apprenant. Il prend une structure d'un profil d'apprenant et une valeur numérique indiquant le niveau souhaité comme les entrées. Il rend une autre structure de profil d'apprenant dont tous les composantes ayant le niveau supérieur ou égal au niveau souhaité sont supprimées.

Rôle

« str-DeleteByLevel » permet de supprimer plusieurs éléments en même temps. Les éléments supprimés sont identifiés par le niveau donné.

Note : utilisé seulement pour les éléments du type ListeComposante et ListeRepartition

Utilisation

str-DeleteByLevel (strucP, list_of_idElems, intLevel)

Paramètre

strucP : la structure de profil qui contient l'élément supprimé. Type de donnée : StrucPro

list_of_idElems: liste des identifiants de la brique.

intLevel : le niveau où on veut supprimer tous les éléments. La racine a le niveau 0. Type de donnée : integer ≥ 1

Valeur retournée : la structure de profil après la suppression tous les éléments ayant le niveau donné. Type de donnée : StrucPro

Exemple

str-DeleteByLevel (Struct Ajout CE2, 2)

12. data-GetValueByID

Description

« data-GetValueByID » est un opérateur sur la donnée de profil d'apprenant. Il prend l'instance de profil d'apprenant (la partie donnée de profil d'apprenant), l'identifiant de la composante dont on veut récupérer les valeurs et deux dates (date 1 , date2) constituant l'intervalle de date comme les entrées. L'intervalle de date nous permet de spécifier les valeurs dont la date est comprise dans cet intervalle. Si on met la valeur nulle à la date 1, ça veut dire qu'on ne veut que les valeurs dont la date est inférieure ou égale à la date 2. Si on

met la valeur nulle à la date 2, ça veut dire qu'on ne veut que les valeurs dont la date est supérieure ou égale à la date 1. Si on met la valeur nulle à la date 1 et la date 2, ça veut dire qu'on veut tous les valeurs n'importe la date. Cet opérateur rend les triples de (valeur, date, identifiant d'échelle) dont la date est satisfaite l'intervalle spécifié.

Rôle

« data-GetValueByID » permet de récupérer tous les triples de valeur et date et l'identifiant d'échelle d'une composante.

Utilisation

data-GetValueByID (dataP, IDCompF, NumValue)

Paramètre

dataP : l'instance du profil. Structure de donnée : DataPro

IDCompF : l'identifiant de la composante **FEUILLE** où on veut récupérer ses valeurs. Type de donnée : chaîne des caractères

NumValue: le numéro de la valeur dont les évaluations qu'on veut récupérer. Type de donnée : chaîne des caractères

Valeur retournée : une liste des triple (valeur, date, identifiant d'échelle) et de la composante qui sont satisfait la condition de date. Type de donnée : liste des triples de valeur (valeur, date, l'identifiant d'échelle)

Exemple

data-GetValueByID(Eval Nat CE2, 'E1_C2', '0') (3 ,'23/03/2010', 'EchE1_1')

13. data-GetValueByDate

Description

« data-GetValueByDate » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant, un identifiant de la composante souhaitée, et les dates constituant l'intervalle de date. Il rend les triples de valeur dont la date est comprise dans l'intervalle spécifié.

Rôle

« data-GetValueByDate » permet de récupérer les triples de valeur dont la date est comprise dans l'intervalle de date spécifié.

Utilisation

data-GetValueByDate (dataP, IDCompF, NoValue, Date1, Date2)

Paramètre

dataP : l'instance du profil. Structure de donnée : DataPro

IDCompF : l'identifiant de la composante **FEUILLE** où on veut récupérer la valeur. Type de donnée : chaîne des caractères

NoValue : le numéro de la valeur dont les évaluations qu'on veut récupérer. Type de donnée : chaîne des caractères

Date1, Date2 : les dates constituant l'intervalle de date. Type de donnée : Date

Valeur retournée : les triples de valeur dont la date est comprise dans l'intervalle de date.

Exemple

data-GetValueByDate(Eval Nat CE2, 'E1_C2', '0', null, 01/01/2010)
(3 ,'23/03/2010', 'EchE1_1')

14. data-GetValueByValue

Description

« data-GetValueByValue » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant, un identifiant de la composante souhaitée, une fonction d'évaluation des valeurs comme les entrées. Il rend les triples de valeur qui satisfont la fonction d'évaluation.

Rôle

« data-GetValueByValue » permet de récupérer les triples de valeurs qui satisfont une fonction d'évaluation de valeur donnée.

Utilisation

data-GetValueByValue (dataP, IDCompF, NoValue, func)

Paramètre

dataP : l'instance du profil. Structure de donnée : DataPro

IDCompF : l'identifiant de la composante **FEUILLE** où on veut récupérer la valeur. Type de donnée : chaîne des caractères

NoValue : le numéro de la valeur dont les évaluations qu'on veut récupérer. Type de donnée : chaîne des caractères

func : la fonction d'évaluation de valeur. Type de donnée : fonction XML de la comparaison

Valeur retournée : les triples de valeur qui satisfont la fonction d'évaluation.

Exemple

data-GetValueByValue(Eval Nat CE2, 'E1_C2', '0', 'EchE1_1, func)

Avec func :

```
<Func>
        <Src><Value>var</Value></Src>
        <Des><Value>3</Value></Des>
<Operator>SUP</Operator>
        <Func>
```

Résultat: (4 ,'21/09/2009', 'EchE1_1')

15. data-GetMaxValue

Description

« data-GetMaxValue » est un opérateur sur la donnée de profil d'apprenant. Il prend l'instance de profil d'apprenant, l'identifiant de la composante souhaitée, l'identifiant d'échelle souhaitée, et les deux dates constituant l'intervalle de date comme les entrées. Il rend le triple (valeur, date, identifiant d'échelle) dont la valeur est la plus grande et la date est satisfaite l'intervalle donné.

Rôle

« data-GetMaxValue » permet de récupérer le triple de valeur d'une composante dont la valeur est la plus grande.

Utilisation

data-GetMaxValue (dataP, IDCompF, noScale, Date1, Date2)

Paramètre

dataP : l'instance du profil. Structure de donnée : DataPro

IDCompF : l'identifiant de la composante où on veut récupérer la valeur. Type de donnée : chaîne des caractères
noScale : l'identifiant d'échelle des valeurs qu'on veut récupérer
Date1, Date2 : les dates constituant l'intervalle de date. Type de donnée : Date
Valeur retournée : le triple (valeur, date, identifiant d'échelle) dont la valeur est la plus grande. Type de donnée : le triple (valeur, date, l'identifiant d'échelle)
Exemple
data-GetMaxValue(Eval Nat CE2, 'E1_C2', '0', 'EchE1_1', null, null)

Résultat : (4 ,'21/09/2009', 'EchE1_1')

16. data-GetMinValue

Description
« data-GetMinValue » est un opérateur sur la donnée de profil d'apprenant. Il prend l'instance de profil d'apprenant, l'identifiant de la composante souhaitée, l'identifiant d'échelle souhaitée, et les deux dates constituant l'intervalle de date comme les entrées. Il rend le triple (valeur, date, identifiant d'échelle) dont la valeur est la plus petite et la date est satisfaite l'intervalle donné.
Rôle
« data-GetMinValue » permet de récupérer le triple de valeur d'une composante dont la valeur est la plus petite.
Utilisation
data-GetMinValue (dataP, IDCompF, noScale, Date1, Date2)
Paramètre
dataP : l'instance du profil. Structure de donnée : DataPro
IDCompF : l'identifiant de la composante où on veut récupérer la valeur. Type de donnée : chaîne des caractères
noScale : l'identifiant d'échelle des valeurs qu'on veut récupérer
Date1, Date2 : les dates constituant l'intervalle de date. Type de donnée : Date
Valeur retournée : le triple (valeur, date, identifiant d'échelle) dont la valeur est la plus petite. Type de donnée : le triple (valeur, date, l'identifiant d'échelle)
Exemple
data-GetMinValue(Eval Nat CE2, 'E1_C2', 'EchE1_1', null, null)

Résultat : (3 ,'23/03/2010', 'EchE1_1')

17. data-GetAvgValue

Description
« data-GetAvgValue » est un opérateur sur la donnée de profil d'apprenant. Il prend l'instance de profil d'apprenant, l'identifiant de la composante souhaitée, l'identifiant d'échelle souhaitée, et les deux dates constituant l'intervalle de date comme les entrées. Il rend une valeur numérique indiquant la valeur moyenne de tous les valeurs de la composante spécifiée.
Rôle
« data-GetAvgValue » permet de récupérer la valeur moyenne des valeurs d'une composante spécifiée
data-GetAvgValue (dataP, idComp, idScale, Date1, Date2)

Utilisation

Paramètre

dataP : l'instance de profil. Type de donnée : DataPro

idComp : l'identifiant de la composante où on veut calculer la valeur moyenne. Type de donnée : chaîne des caractères

idScale : l'identifiant d'échelle dont la valeur est utilisée pour le calcul

Date1, Date2 : les dates constituant l'intervalle de date. Type de donnée : Date

Valeur retournée : une valeur numérique indiquant la valeur moyenne. Type de donnée : Double

Exemple

data-GetAvgValue(Eval Nat CE2, 'E1_C2', 'EchE1_1', null, null)

Résultat : 3.5

18. data-GetFeqValue

Description

« data-GetFeqValue » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant, un identifiant de la composante souhaitée, une fonction d'évaluation des valeurs, et les dates constituant l'intervalle de date. Il calcule la fréquence des valeurs qui satisfont la fonction d'évaluation et l'intervalle de date par rapport à tous les valeurs.

Rôle

« data-GetFeqValue » permet de calculer la fréquence de valeurs concernées d'une composante d'une instance de profil d'apprenant.

Utilisation

data-GetFeqValue (dataP, idComp,idScale,func, Date1, Date2)

Paramètre

dataP : instance de profil d'apprenant. Type de donnée : DataPro

idComp : l'identifiant de la composante. Type de donnée : chaîne des caractères

func : la fonction d'évaluation des valeurs. Type de donnée : fonction d'XML

Date1, Date2 : les dates constituant l'intervalle de date. Type de donnée : Date

Valeur retournée : une valeur numérique indiquant la fréquence des valeurs. Type de donnée : Double

Exemple

data-GetFeqValue(Eval Nat CE2, 'E1_C2', 'EchE1_1', func, null, null)

Avec

```
<Func>
    <Src><Value>var</Value></Src>
    <Des><Value>3</Value></Des>
<Operator>SUP</Operator>
    <Func>
```

On a une valeur qui satisfait la function au dessus et pour l'échelle 'EchE1_1' on a deux valeurs. Alors, le résultat est ½ ou 0.5

19. data-GetSumValue

Description

« data-GetSumValue » est un opérateur sur la donnée de profil d'apprenant. Il prend l'instance de profil d'apprenant, l'identifiant de la composante souhaitée, l'identifiant d'échelle souhaitée, et les deux dates constituant l'intervalle de date comme les entrées. Il rend une valeur numérique indiquant la somme de tous les valeurs de la composante spécifiée.

Rôle

« data-GetSumValue » permet de calculer la somme des valeurs d'une composante dont les évaluations ont l'identifiant d'échelle qui est spécifié dans le paramètre entré

Utilisation

data-GetSumValue (dataP, idComp,idScale, Date1, Date2)

Paramètre

dataP : l'instance de profil. Type de donnée : DataPro

idComp : l'identifiant de la composante où on veut calculer la somme des valeurs. Type de donnée : chaîne des caractères

idScale : l'identifiant d'échelle dont la valeur est utilisée pour le calcul

Date1, Date2 : les dates constituant l'intervalle de date. Type de donnée : Date

Valeur retournée : une valeur numérique indiquant la somme des valeurs. Type de donnée : Double

Exemple

data-GetSumValue(Eval Nat CE2, 'E1_C2', 'EchE1_1', null, null)

Résultat : 7

20. data-GetVoVValue

Description

« data-GetVoVProfile » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant, l'identifiant de la composante souhaitée, l'identifiant d'échelle concerné, et les dates constituant l'intervalle de date. Cet opérateur calcule la variance des valeurs d'évaluation (dispersion absolue) qui représente la stabilité d'une compétence d'élève. On utilise la formule qui résulte du théorème de Koenig :

$$S^2 = \frac{1}{n}\sum_{i=1}^{i=n} x_i^2 - \bar{x}^2$$

$$S = \sqrt{S^2}$$

Rôle

« data-GetVoVProfile » permet de calculer la variance des valeurs d'une composante d'une instance de profil d'apprenant.

Utilisation

data-GetVoVValue (dataP, idComp,idScale, Func, Date1, Date2)

Paramètre

dataP : l'instance de profil. Type de donnée : DataPro
idComp : l'identifiant de la composante où on veut évaluer le progrès. Type de donnée : chaîne des caractères
idScale : l'identifiant d'échelle dont la valeur est utilisée pour le calcul
Date1, Date2 : les dates constituant l'intervalle de date. Type de donnée : Date
Valeur retournée : une valeur numérique indiquant le progrès d'élève. Type de donnée : Double
Exemple
data-GetVoVProfile(Eval Nat CE2, 'E1_C2', 'EchE1_1', null, null)
On a

$$\overline{X} = \frac{3+4}{2} = 3.5$$

$X_1 = 3$; $X_2 = 4$; $n = 2$
Alors

$$S^2 = \frac{\left(3^2 - 3.5^2\right) + \left(4^2 - 3.5^2\right)}{2} = 0.25$$

Alors

S = 0.5

21. data-GetCVProfile

Description
« data-GetCVProfile » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant, un identifiant de la composante souhaitée, un identifiant d'échelle, et les dates constituant l'intervalle de date. Cet opérateur calcule le **coefficient de variation** des valeurs (dispersion relative) de cette composante par la formule :

$$C.V. = \frac{100.S}{\overline{x}}$$

Rôle
« data-GetCVProfile » permet de calculer le coefficient de variation des valeurs d'une composante d'une instance de profil. On peut utiliser cette valeur pour comparer la variance des valeurs d'évaluation appartenant à deux échelles hétérogènes.
Utilisation

```
data-GetCVProfile(dataP, idComp, idScale, Date1, Date2)
```

Paramètre
dataP : l'instance de profil. Type de donnée : DataPro
idComp : l'identifiant de la composante où on veut évaluer le progrès. Type de donnée : chaîne des caractères
idScale : l'identifiant d'échelle dont la valeur est utilisée pour le calcul
Date1, Date2 : les dates constituant l'intervalle de date. Type de donnée : Date
Valeur retournée : une valeur numérique indiquant le coefficient de variation. Type de donnée : Double
Exemple
data-GetCVProfile(Eval Nat CE2, 'E1_C2', 'EchE1_1', null, null)
On a

S = 0.5 ; x = 3.5
Alors

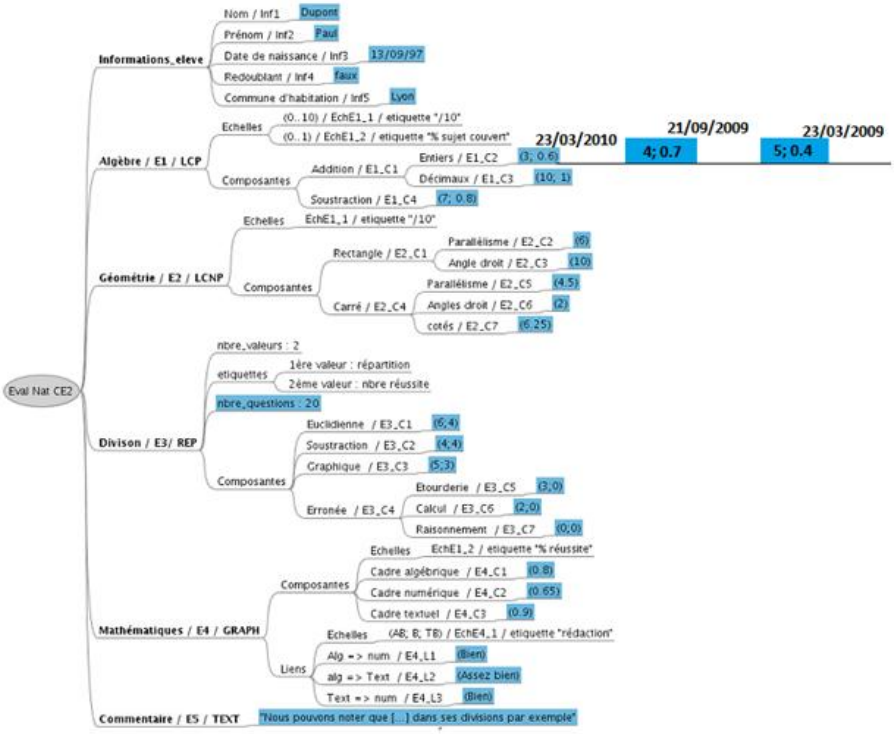
C.V. = (100 x 0.5) / 3.5 = 14.28

22. data-AddValue

Description
« data-AddValue » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant, un identifiant de la composante souhaitée, et la partie de la nouvelle valeur comme la valeur d'évaluation, l'identifiant d'échelle, la source d'évaluation, la date d'évaluation, le commentaire. Il va vérifier les règles (règle de la date, règle d'échelle) avant d'ajouter la nouvelle valeur .Cet opérateur rend une autre instance de profil d'apprenant dont la composante spécifiée a été ajoutée la partie de nouvelle valeur.
Rôle
« data-AddValue » permet d'ajouter une nouvelle valeur dans une composante spécifiée d'une instance de profil d'apprenant.
Utilisation data-AddValue(dataP, idComp, NoValue, { value, source,

Paramètre
dataP : l'instance de profil d'apprenant. Type de donnée : DataPro
idComp : l'identifiant de la composante (**ou de la lien dans le graphe ou l'identifiant de l'élément TEXTE**). Cette composante doit être une feuille ou une composante du graphe. Type de donnée : chaîne des caractères
NoValue : le numéro de la valeur où les évaluations sont ajoutées. Type de donnée : chaîne des caractères.
Paire de l'évaluation
value : la valeur de l'évaluation.
source : la source d'évaluation. Type de donnée : chaîne des caractères
date : la date de la valeur. Type de donnée : Date
Valeur retournée : une instance de profil d'apprenant qui a été ajoutée la nouvelle valeur. Type de donnée : DataPro
Exemple
data-AddValue(Eval Nat CE2, E1_C2, EchE1_1, {5, 'Test', '23/03/2009'})
data-AddValue(Eval Nat CE2, E1_C2, EchE1_2, {0.4, 'Test', '23/03/2009'})

Résultat



23. data-DeleteValueByDate

Description

« data-DeleteValueByDate » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant, l'identifiant de la composante souhaitée, la date où on veut supprimer la valeur comme les entrées. Il rend une autre instance de profil d'apprenant qui a été supprimée la valeur donnée.

Rôle

« data-DeleteValueByDate » permet de supprimer une valeur de la composante d'une instance de profil d'apprenant. On peut l'utiliser pour modifier la donnée d'un profil d'apprenant.

Utilisation

data-DeleteValueByDate (dataP, *list_of_IDComps, list_of_IDScales*, date1, date2)

Paramètre

dataP : l'instance de profil d'apprenant. Type de donnée : DataPro

list_of_idComp : la liste des identifiants des composantes qu'on veut faire la suppression. **Mettre étoile(« * ») si on veut effectuer la suppression sur tous les composantes.** Structure de donnée : liste des chaînes des caractères

list_of_idEchs : la liste des id échelles . **Mettre étoile(« * ») si on veut effectuer la suppression sur tous les valeurs.** Structure de donnée : liste des chaînes des caractères.

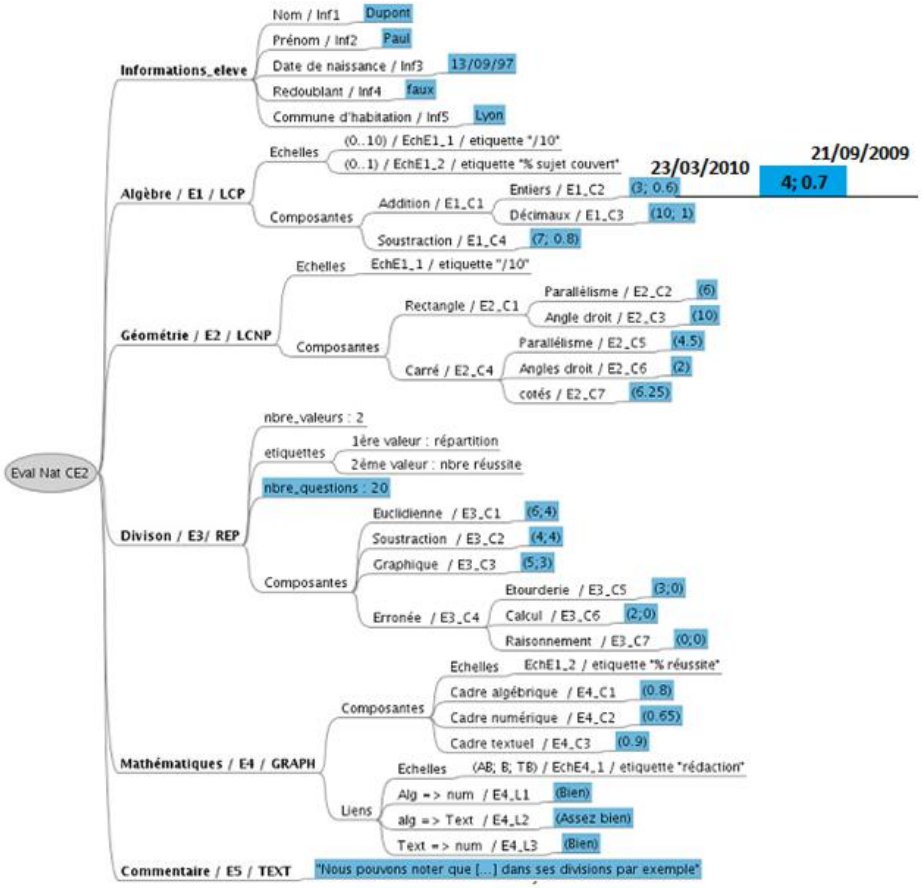
date1, date2 : la date où on veut supprimer la valeur. Type de donnée : Date

Valeur retournée : l'instance de profil qui a été supprimée la valeur par la date donnée.

Exemple

data-DeleteValueByDate(Eval Nat CE2, 'E1_C2', '01/01/2009', '01/04/2009')

Résultat



24. data-DeleteValueByValue

Description

« data-DeleteValueByValue » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant, l'identifiant de la composante souhaitée, la valeur et la fonction d'évaluation (on supprime la valeur si et seulement si cette fonction évalue à TRUE). Cet opérateur rend une autre instance de profil d'apprenant dont les valeurs ont été supprimées.

Rôle

« data-DeleteValueByValue » permet de supprimer les valeurs de la composante d'une instance de profil d'apprenant par une fonction d'évaluation des valeurs.

Utilisation

data-DeleteValueByValue (dataP, *list_of_IDComps, list_of_IDScales*, func)

Paramètre

dataP : l'instance de profil d'apprenant. Type de donnée : DataPro

IDComp : l'identifiant de la composante. Type de donnée : chaîne des caractères

func : la fonction d'évaluation.

Valeur retournée : l'instance de profil d'apprenant dont les valeurs ont été supprimées.

Exemple :

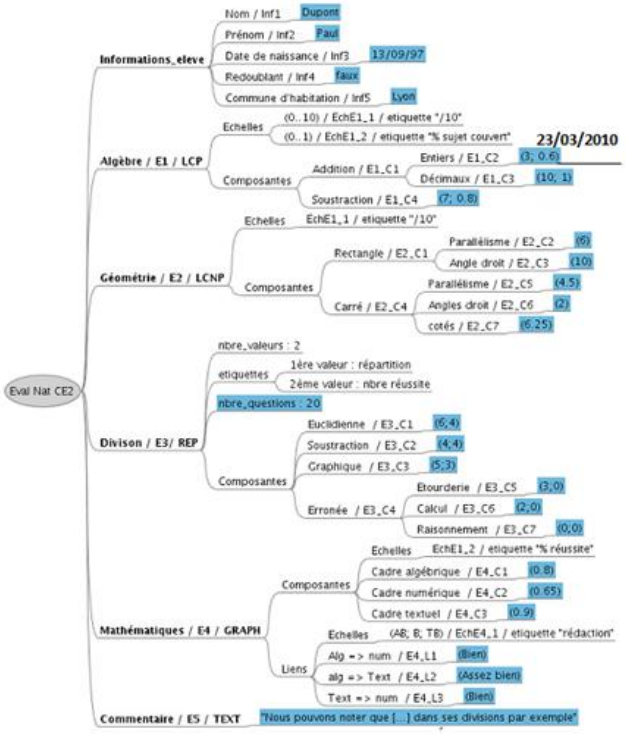
data-DeleteValueByValue(Eval Nat CE2,{ 'E1_C2'}, {'EchE1_1'}, func)

Avec

<Func>

<Src><Value>var</Value></Src>
<Des><Value>3</Value></Des>
<Operator>SUP</Operator>
<Func>

Résultat



25. data-ReplaceValueByDate

Description

« data-ReplaceValueByDate » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant, un identifiant de la composante souhaitée, un identifiant d'échelle concernée, une nouvelle valeur pour le remplacement, et les dates constituant l'intervalle de date. Il rend une autre instance de profil d'apprenant dont les valeurs de la composante sont remplacées par la nouvelle valeur.

Rôle

« data-ReplaceValueByDate » permet de remplacer les valeurs dont la date est comprise dans l'intervalle de date spécifié par une nouvelle valeur.

Utilisation

data-ReplaceValueByDate (dataP, *list_of_idComp, list_of_idScale*, newValue, Date1, Date2)

Paramètre

dataP : l'instance de profil d'apprenant. Type de donnée : DataPro

idComp : l'identifiant de la composante. Type de donnée : chaîne des caractères

idScale : l'identifiant d'échelle. Type de donnée : chaîne des caractères

newValue : la nouvelle valeur. Type de donnée : dépend sur l'échelle spécifiée

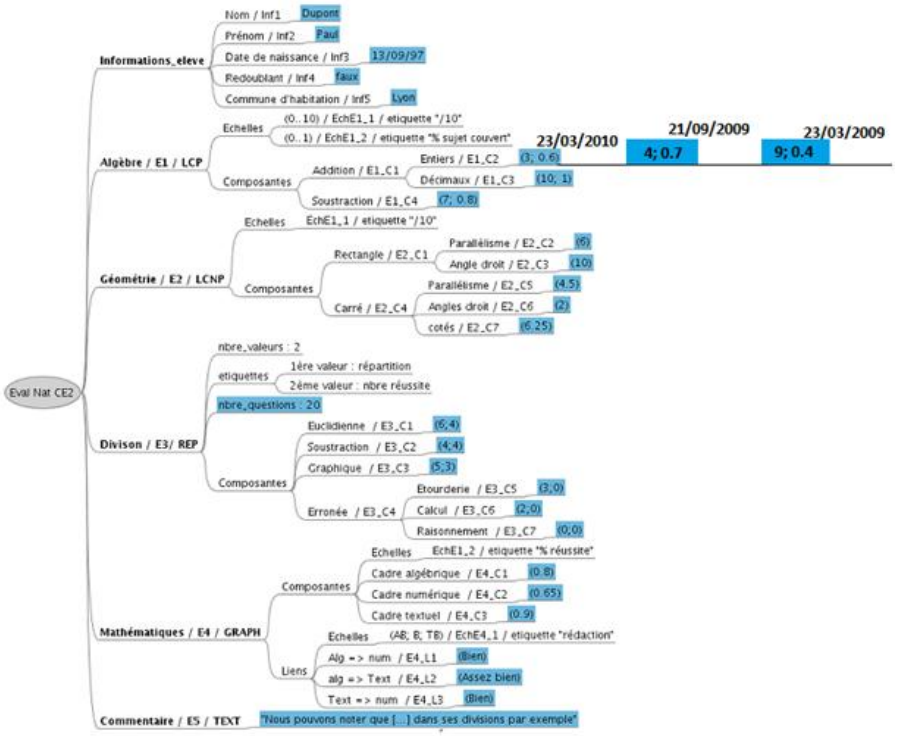
Date1, Date2 : les dates constituant l'intervalle de date. Type de donnée : Date

Valeur retournée : une instance de profil d'apprenant dont les valeurs sont remplacées par la nouvelle valeur.

Exemple

data-ReplaceValueByDate(Eval Nat CE2, ‘E1_C2’, ‘EchE1_1’, 9, ’01/01/2009’, ‘01/04/2009’)

Résultat



26. data-ReplaceValueByValue

Description

« data-ReplaceValueByValue » est un opérateur sur la donnée de profil d’apprenant. Il prend une instance de profil d’apprenant, un identifiant de la composante souhaitée, un identifiant d’échelle, une nouvelle valeur, et la fonction d’évaluation de valeur. Il rend une instance de profil d’apprenant dont les valeurs qui satisfont la fonction d’évaluation sont remplacées par la nouvelle valeur.

Rôle

« data-ReplaceValueByValue » permet de remplacer les valeurs spécifiées par une fonction d’évaluation de valeur par une nouvelle valeur donnée.

Utilisation

data-ReplaceValueByValue (dataP, *list_of_idComps, list_of_idScale*s, newValue, *func*)

Paramètre

dataP : l’instance de profil d’apprenant. Type de donnée : DataPro

idComp : l’identifiant de la composante. Type de donnée : chaîne des caractères

idScale : l’identifiant d’échelle. Type de donnée : chaîne des caractères

newValue : la nouvelle valeur. Type de donnée : dépend sur l’échelle spécifiée

func : la fonction d’évaluation de valeur. Type de donnée : fonction d’XML

Valeur retournée : une instance de profil d’apprenant dont les valeurs sont remplacées par la nouvelle valeur.

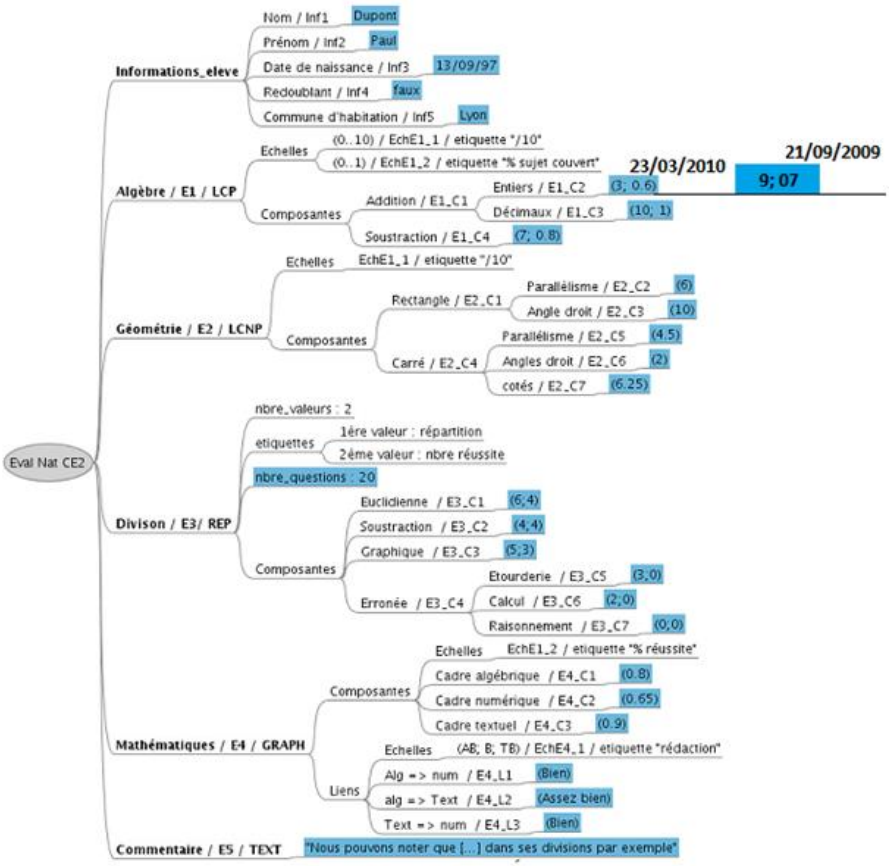
Exemple

data-ReplaceValueByValue (Eval Nat CE2, ‘E1_C2’, ‘EchE1_1’, 9, func)

Avec

<Func>
<IDScale>EchE1_1</IDScale>
<Src><Value>var</Value></Src>
<Des><Value>3</Value></Des>
<Operator>SUP</Operator>
<Func>

Résultat



27. data-CompressByAvg

Description

« data-CompressByAvg » est un opérateur sur la donnée de profil d’apprenant. Il prend l’instance de profil d’apprenant, l’identifiant de la composante souhaitée, les deux dates constituant l’intervalle de date, et une valeur booléen indiquant le traitement sur les commentaires comme les entrées. Il rend une autre instance de profil d’apprenant dont les valeurs de la composante spécifiée sont remplacées par une valeur moyenne.

Rôle

« data-CompressByAvg » permet de compresser un profil ou une composante en remplaçant tous les valeurs d’une composante dont la date est comprise dans l’intervalle de date donné par une valeur moyen des valeurs remplacées.

Utilisation

data-CompressByAvg (dataP, *list_of_IDComp*, *list_ofIDScale*,Date1, Date2)

Paramètre

dataP : l'instance du profil. Structure de donnée : DataPro

list_of_IDComp: la liste des identifiants de la composante qu'on veut la compresser. **Mettre étoile(« * ») si on veut effectuer le filtrage sur tous les composantes.**. Structure de donnée : Chaîne des caractères

Date1, Date2 : les dates indiquant l'intervalle. Pour un intervalle valide on doit mettre la Date1 inférieur à la Date2. Sinon, l'opérateur va compresser tous les valeurs de composante. Type de donnée : Date

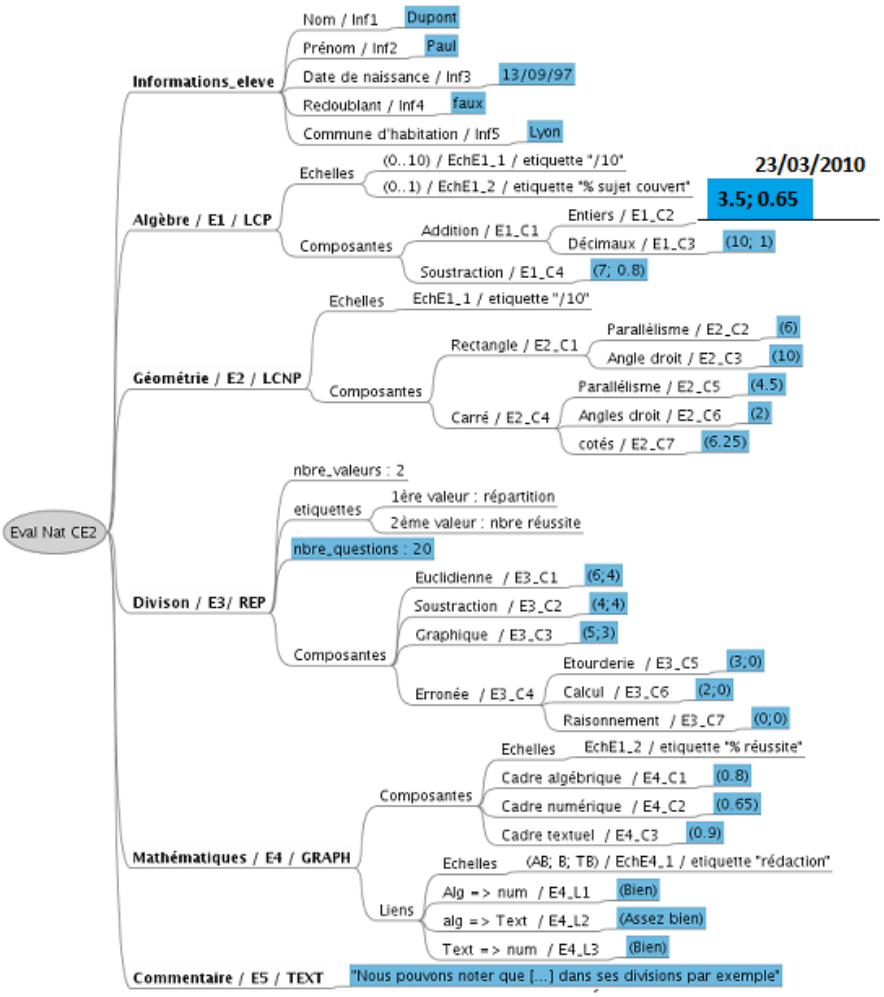
Les types de composante de la compression : DataCompL(DataCompL_W, DataCompL_NotW), DataCompR, DataGraph (DataCompG, DataLink)

Valeur retournée : l'instance de profil après d'avoir compressé. Structure de donnée : DataPro

Exemple

data-CompressByAvg(Eval Nat CE2, 'E1_C2', TRUE, null, null)

Résultat



28. data-CompressBySum

Description

« data-CompressBySum » est un opérateur sur la donnée de profil d'apprenant. Il prend l'instance de profil d'apprenant, l'identifiant de la composante souhaitée, les deux dates constituant l'intervalle de date, et une valeur booléen indiquant le traitement sur les commentaires comme les entrées. Il rend une autre instance de profil d'apprenant dont les valeurs de la composante spécifiée sont remplacées par la somme.

Rôle

« data-CompressBySum » permet de compresser les évaluations d'une composante ou toutes les composantes d'une instance de profil. La compression remplace les valeurs dans une évaluation par une valeur qui est la somme des valeurs remplacées. On peut indiquer aussi un intervalle de date.

Utilisation

data-CompressBySum (dataP, *list_of_IDComp*, *list_ofIDScale*, Date1, Date2)

Paramètre

dataP : l'instance de profil où la compression a lieu. Type de donnée : DataPro

idComp : l'identifiant de la composante qu'on veut compresser. Type de donnée : chaîne des caractères

boolComment : la valeur indiquant si on veut garder tous les commentaires. Type de donnée : Boolean

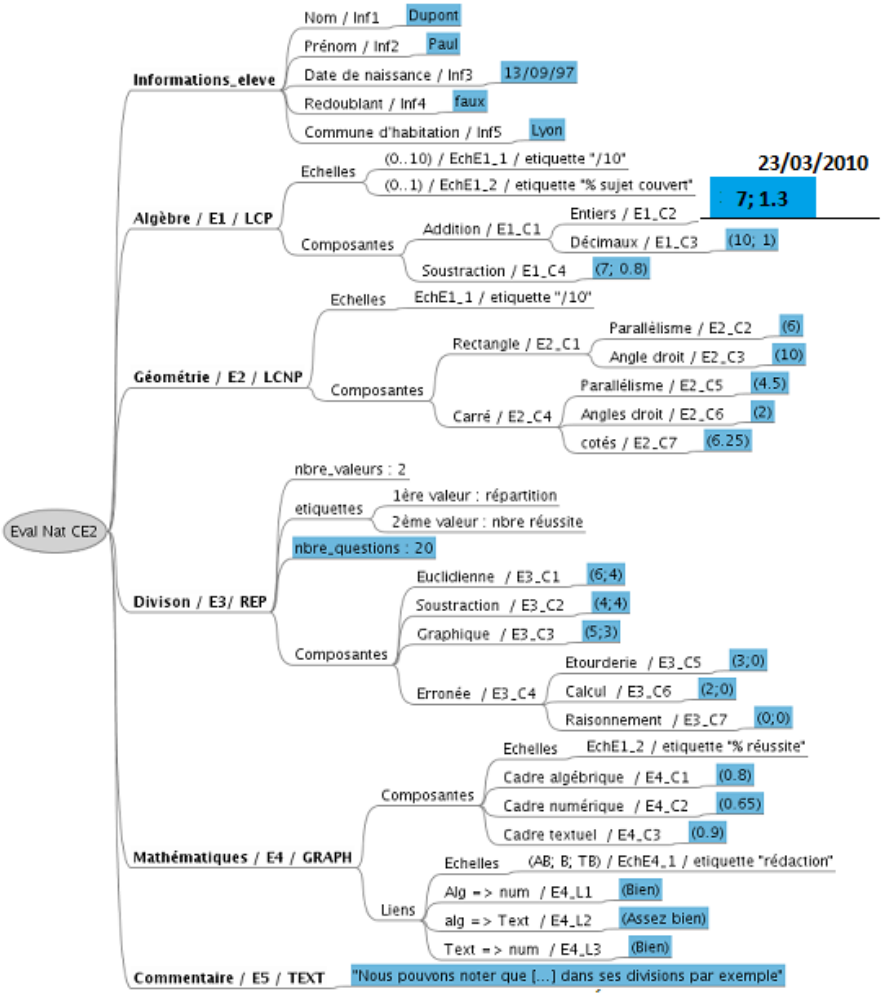
Date1, Date2 : les dates constituant l'intervalle de date

Valeur retournée : l'instance de profil après d'avoir compressé. Type de donnée : DataPro

Exemple

data-CompressBySum(Eval Nat CE2, 'E1_C2', TRUE, null, null)

Résultat



29. data-FilterByID

Description

« data-FilterByID » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant, une liste des identifiants des composantes comme les entrées. Il rend une autre instance de profil d'apprenant qui ne contient que les composantes dont l'identifiant est comprise dans la liste.

Rôle

« data-FilterByID » permet de filtre les valeurs des composantes par les identifiants de composante donnés.

Utilisation

data-FilterByID (dataP,*list_of_idComp*)

Paramètre

dataP : instance de profil d'apprenant. Type de donnée : DataPro

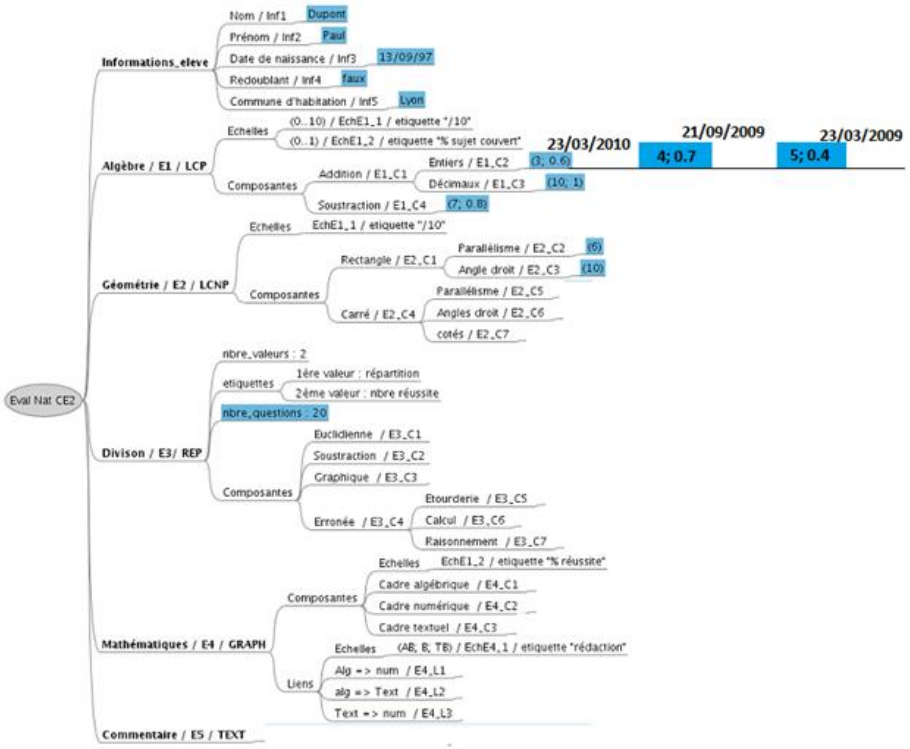
list_of_idComp : liste des identifiants de composante. Type de donnée : liste des chaînes des caractères

Valeur retournée : une instance de profil d'apprenant qui ne contient que les valeurs des composantes dont l'identifiant est comprise dans la liste.

Exemple

data-FilterByID(Eval Nat CE2, {'E1', 'E2_C1' })

Résultat



30. data-FilterByDate

Description

« data-FilterElementByDate » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant, l'identifiant de la composante souhaité et les deux dates constituant l'intervalle de date comme les entrées. Il rend une autre instance de profil d'apprenant dont on ne garde que les valeurs ayant la date satisfaite l'intervalle de date spécifié.

Rôle

« data-FilterElementByDate » permet de filtrer les valeurs des composantes d'une instance de profil ou filtrer les valeurs d'une composante de l'instance de profil.

Utilisation

data-FilterByDate (dataP, *list_of_idComp*, Date1, Date2)

Paramètre

dataP : l'instance du profil. Structure de donnée : DataPro

list_of_idComp : la liste des identifiants des composantes qu'on veut faire le filtrage. Si on veut filtrer tout le profil, on met une chaîne des caractères vide (''). **Mettre étoile(« * ») si on veut effectuer le filtrage sur tous les composantes.** Structure de donnée : liste des chaînes des caractères

Date1, Date2 : les dates indiquant l'intervalle. Pour un intervalle valide on doit mettre la Date1 inférieur à la Date2. Sinon, l'opérateur va compresser tous les valeurs de composante.

Type de donnée : Date

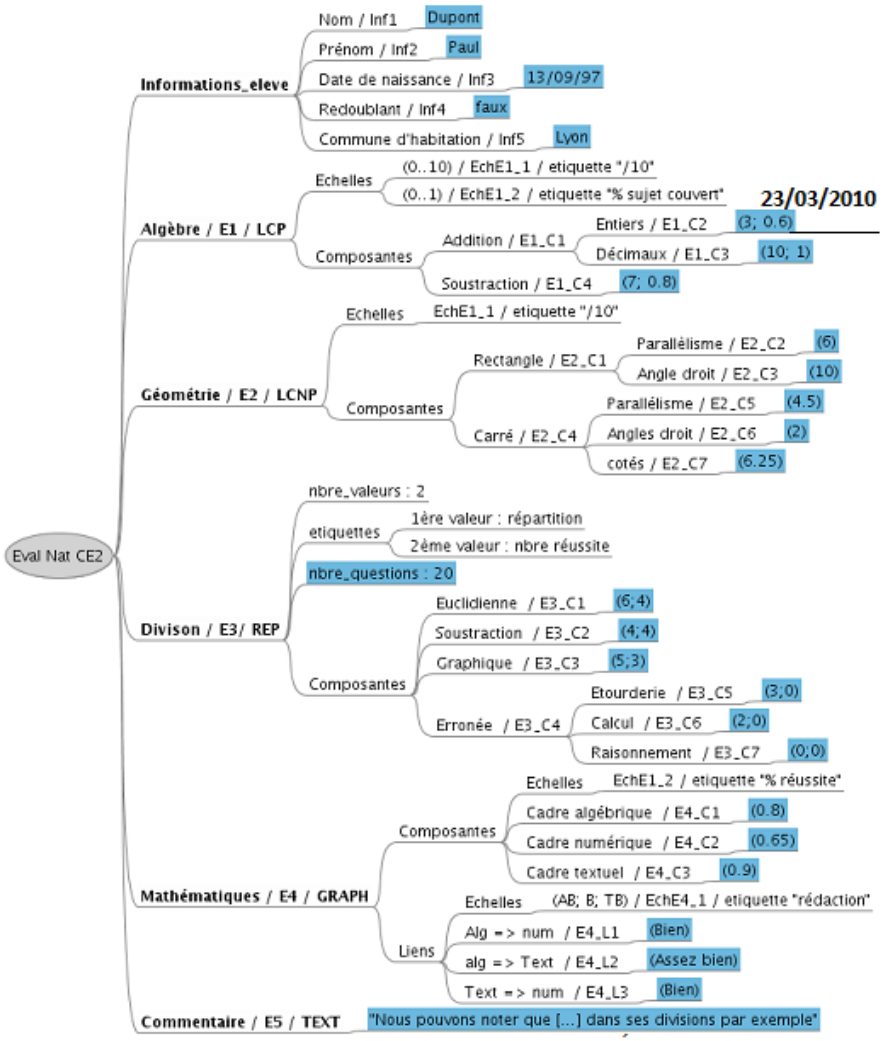
Les types de composante concernés : tous les composantes qui sont évolutives

Valeur retournée : l'instance de profil après d'avoir compressé. Structure de donnée : DataPro

Exemple

data-FilterByDate(Eval Nat CE2, {'E1_C2'}, '01/01/2010', null)

Résultat



31. data-FilterByValue

Description

« data-FilterByValue » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant, l'identifiant de la composante souhaité et une fonction de filtrage utilisée pour évaluer les valeurs. Il rend une autre instance de profil d'apprenant dont les valeurs de la composante sont évaluées vrai par la fonction de filtrage.

Rôle

« data-FilterByValue » permet de filtrer les valeurs d'évaluation d'une instance de profil. L'instance de profil résultat ne contient que les évaluations satisfaites la fonction de filtrage.

Utilisation

data-FilterByValue (dataP, *list_of_idComp*, *list_of_idEchs,* foncFiltre)

Paramètre

dataP : l'instance de profil où on fait le filtrage. Type de donnée : DataPro

list_of_idComp : la liste des identifiants des composantes qu'on veut faire le filtrage. **Mettre étoile(« * ») si on veut effectuer le filtrage sur tous les composantes.** Structure de donnée : liste des chaînes des caractères

list_of_idEchs : la liste des id échelles . **Mettre étoile(« * ») si on veut effectuer le filtrage sur tous les valeurs.** Structure de donnée : liste des chaînes des caractères.

foncFiltre : la fonction de filtrage indiquant quelle opération utilisée pour le filtrage. Type de donnée : fonction XML

Valeur retournée : l'instance de profil après du filtrage. Type de donnée : DataPro

Notes :

Pour les éléments du type Graphe, on s'intéresse que l'identifiant d'élément par rapport au filtrage des ids.

Les éléments du type Texte ne sont pas effectués.

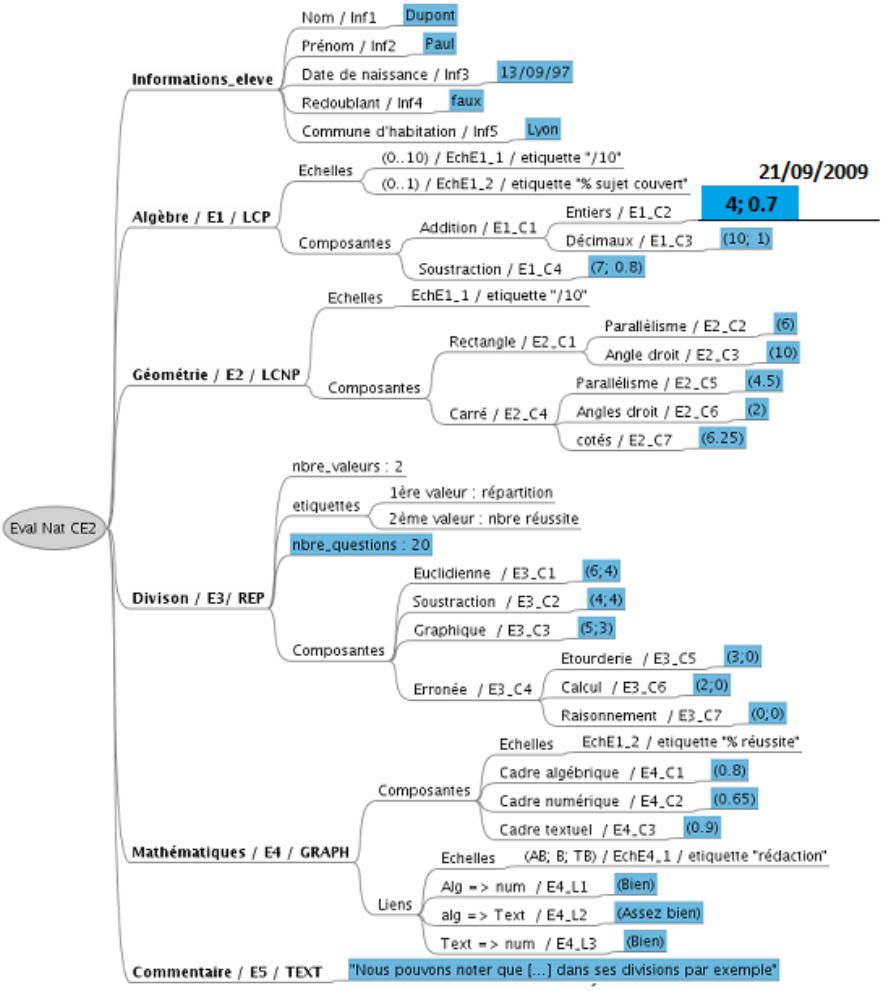
Exemple

data-FilterByValue(Eval Nat CE2, {'E1_C2'}, func)

Avec

```
<Func>
        <IDScale>EchE1_1</IDScale>
        <Src><Value>var</Value></Src>
        <Des><Value>3</Value></Des>
<Operator>SUP</Operator>
        <Func>
```

Résultat



32. data-FilterByScale

Description

« data-FilterByScale » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant, la liste des identifiants d'échelle souhaitée comme les entrées. Cet opérateur rend une autre instance de profil d'apprenant qui ne contient que les composantes ayant l'identifiant d'échelle qui se trouve dans la liste spécifiée.

Rôle

« data-FilterByScale » permet de filtre les composantes dans une instance de profil d'apprenant par les identifiants d'échelle.

Utilisation

data-FilterByScale (dataP, *list_of_idComp*, *list_of_idScale*)

Paramètre

dataP : l'instance de profil d'apprenant. Type de donnée : DataPro

list_of_idComp : la liste des identifiants des composantes qu'on veut faire le filtrage. Si on veut filtrer tout le profil, on met **Etoile (*)**. Structure de donnée : liste des chaînes des caractères

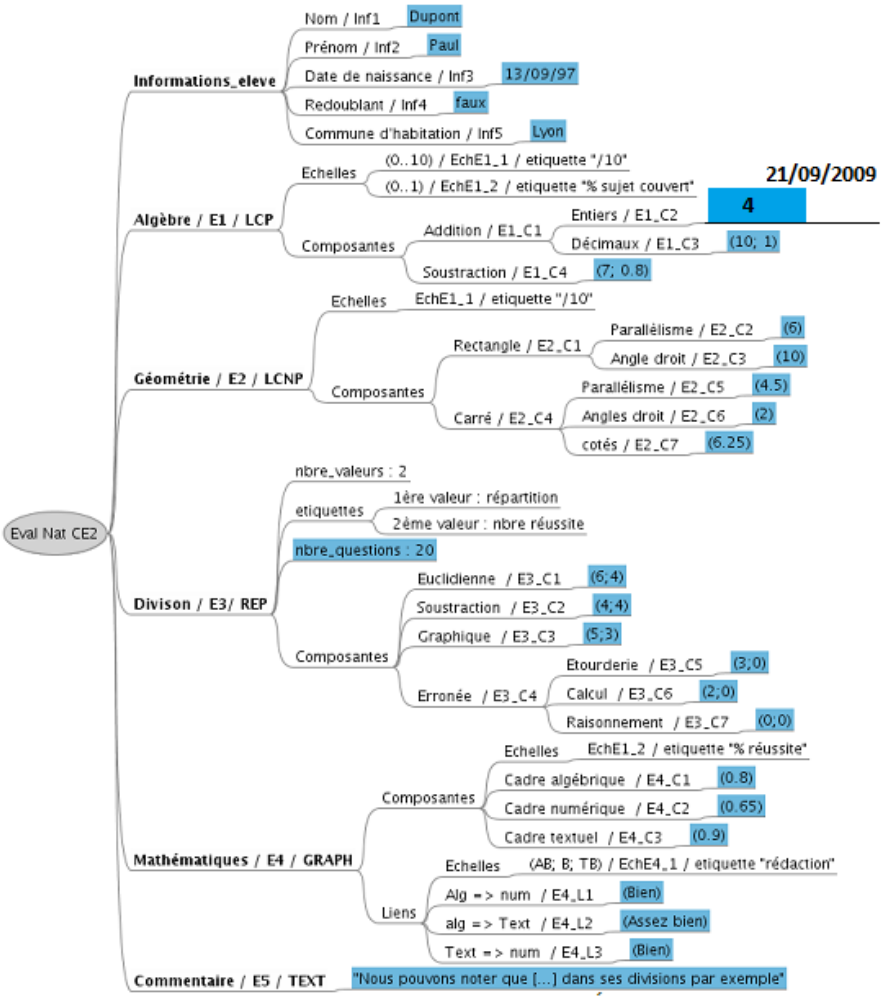
list_of_idScale : la liste des identifiants d'échelle souhaitée. Type de donnée : liste des chaînes des caractères

Valeur retournée : une instance de profil d'apprenant qui ne contient que les composantes ayant l'identifiant d'échelle qui se trouve dans la liste. Type de donnée : DataPro

Exemple

data-FilterByScale(Eval Nat CE2, {'E1_C2'}, {'EchE1_1'})

Résultat



33. data-MoreThan

Description

« data-MoreThan» est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant comme la source, une autre instance de profil d'apprenant comme la cible, les deux dates constituant l'intervalle de date. La source et la cible doivent avoir une même structure. Cet opérateur rend une autre instance de profil d'apprenant qui ne contient que les valeurs de la source qui sont supérieures aux valeurs correspondantes de la cible. La comparaison a lieu sur les valeurs dont la date est comprise dans l'intervalle de date.

Rôle

« data-MoreThan» permet de trouver tous les évaluations de l'instance de profil qui sont supérieures aux évaluations correspondantes de l'autre instance de profil. Tous les deux profils doivent avoir une même structure. On peut indiquer un intervalle de date.

Utilisation

data-MoreThan (dataPSrc, dataPDes, Date1, Date2)

Paramètre

dataPSrc : l'instance de profil qu'on veut trouver les évaluations qui sont supérieur aux évaluations correspondantes de dataPDes

dataPDes : l'instance de profil qu'on utilise pour faire la comparaison

Date1, Date2 : les deux dates constituant l'intervalle de date

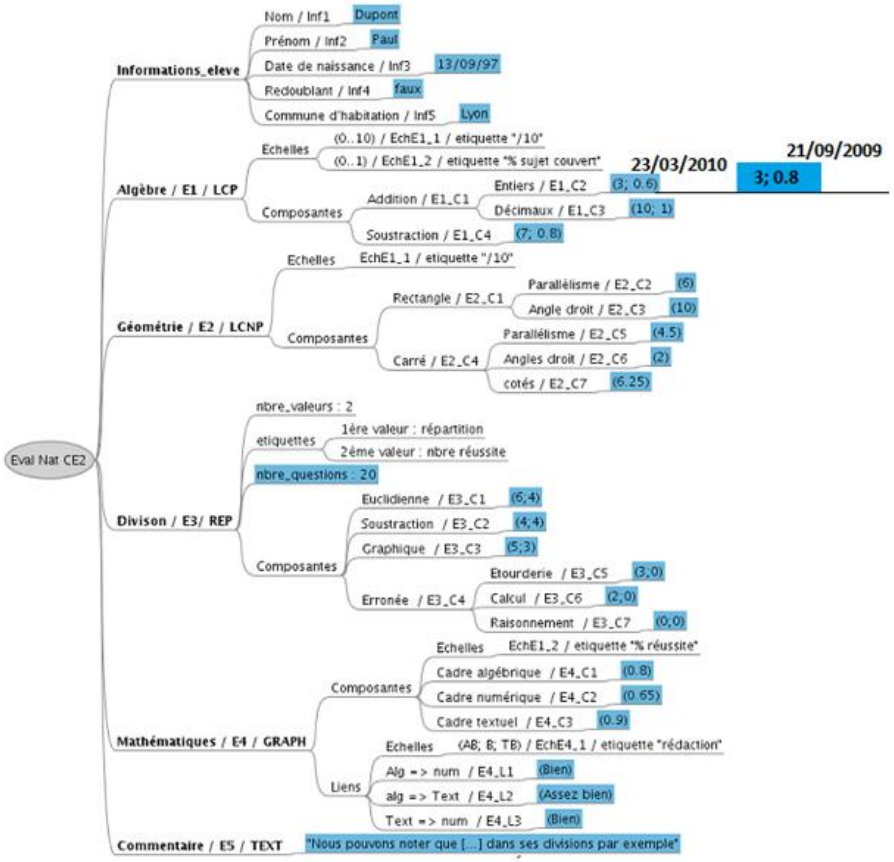
Valeur retournée : l'instance de profil qui ne contient que les évaluations supérieures aux évaluations correspondantes de dataPDes

Exemple

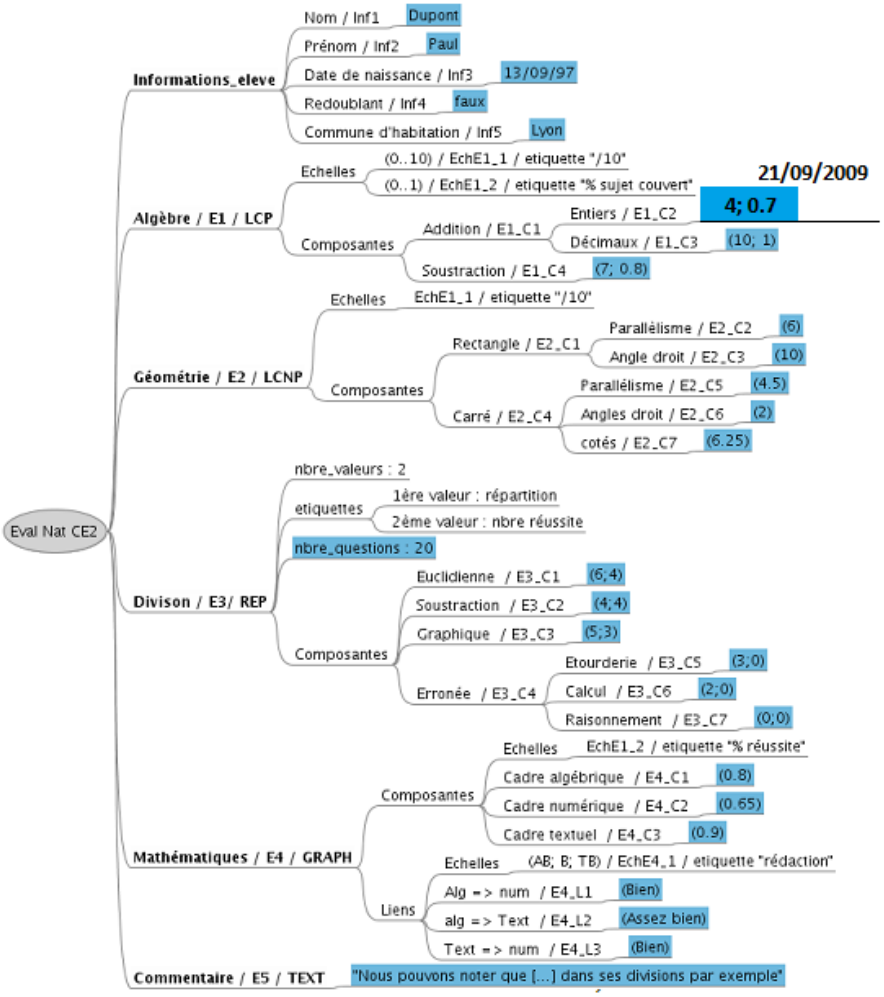
data-MoreThan(Eval Nat CE2, dataPDes, {'E1_C2'}, 'EchE1_1', null, null)

Avec

dataPDes :



Résultat :



34. data-LessThan

Description

« data-LessThan » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant comme la source, une autre instance de profil d'apprenant comme la cible, les deux dates constituant l'intervalle de date. La source et la cible doivent avoir une même structure. Cet opérateur rend une autre instance de profil d'apprenant qui ne contient que les valeurs de la source qui sont inférieures aux valeurs correspondantes de la cible. La comparaison a lieu sur les valeurs dont la date est comprise dans l'intervalle de date.

Rôle

« data-LessThan » permet de trouver tous les évaluations de l'instance de profil qui sont inférieures aux évaluations correspondantes de l'autre instance de profil. Tous les deux profils doivent avoir une même structure. On peut indiquer un intervalle de date.

Utilisation

data-LessThan(dataPSrc, dataPDes, Date1, Date2)

i sont inférieur aux évaluations correspondantes de dataPDes

dataPDes : l'instance de profil qu'on utilise pour faire la comparaison

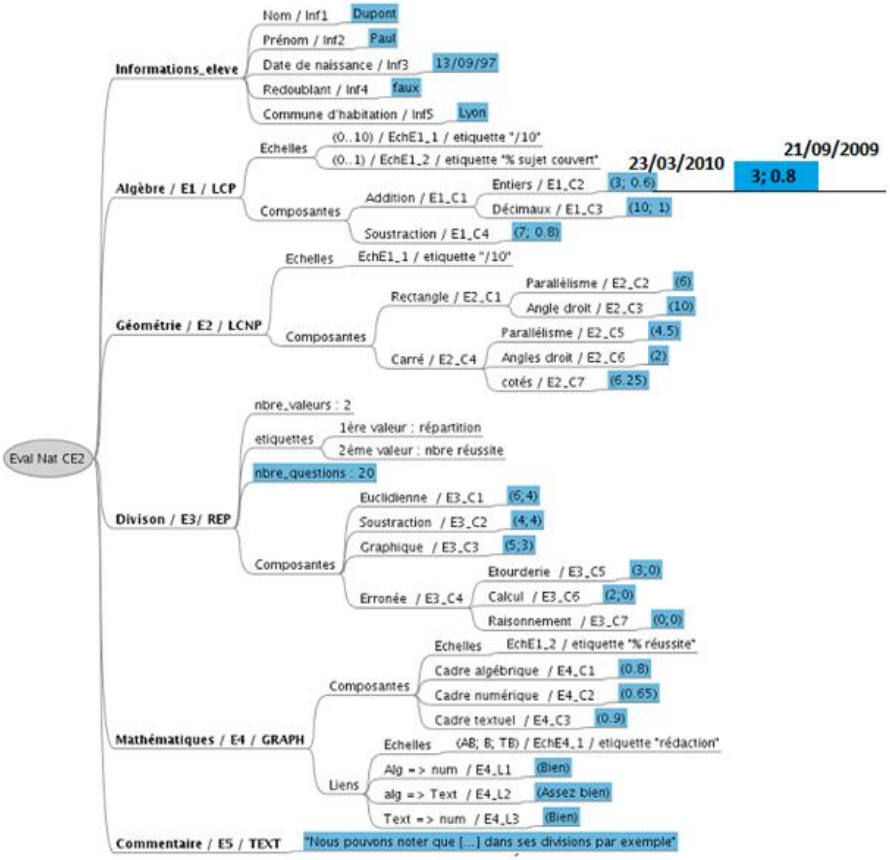
Date1, Date2 : les deux dates constituant l'intervalle de date

Valeur retournée : l'instance de profil qui ne contient que les évaluations inférieures aux évaluations correspondantes de dataPDes

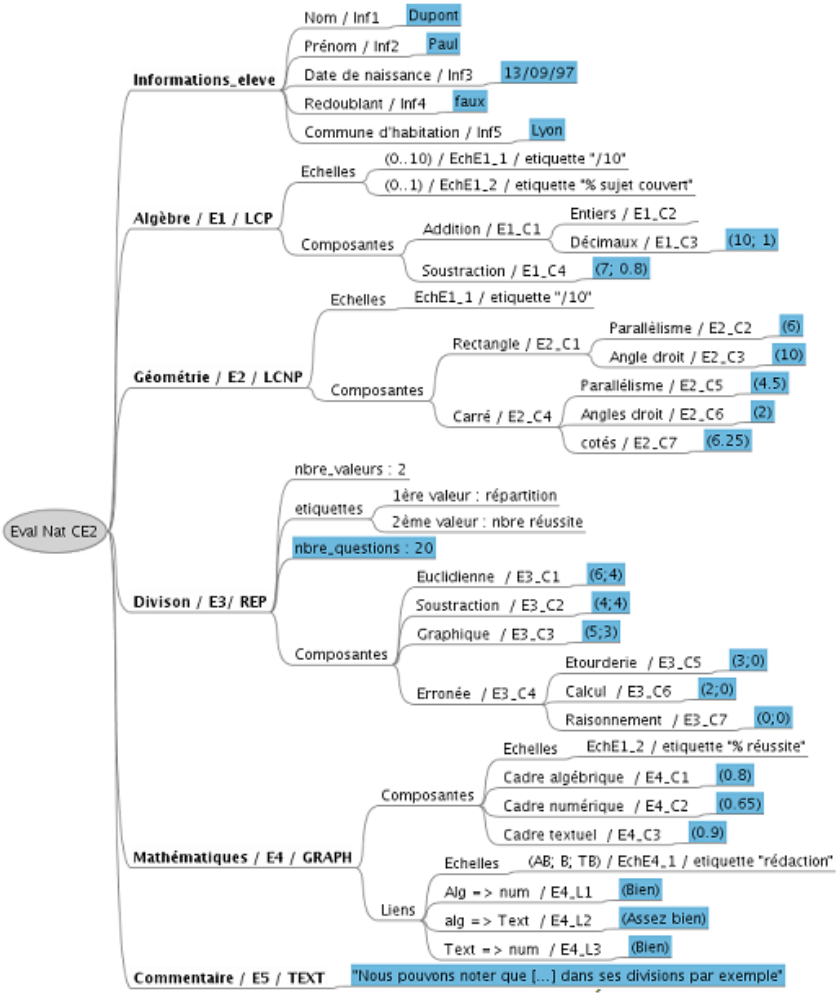
Exemple

data-LessThan(Eval Nat CE2, dataPDes, {'E1_C2'}, 'EchE1_1', null, null)

Avec dataPDes :



Résultat :



35. data-Equal

Description

« data-Equal» est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant comme la source, une autre instance de profil d'apprenant comme la cible, les deux dates constituant l'intervalle de date. La source et la cible doivent avoir une même structure. Cet opérateur rend une autre instance de profil d'apprenant qui ne contient que les valeurs de la source qui sont égales aux valeurs correspondantes de la cible. La comparaison a lieu sur les valeurs dont la date est comprise dans l'intervalle de date.

Rôle

« data-Equal» permet de trouver tous les évaluations de l'instance de profil qui sont égales aux évaluations correspondantes de l'autre instance de profil. Tous les deux profils doivent avoir une même structure. On peut indiquer un intervalle de date.

Utilisation

Pa

```
data-Equal(dataPSrc, dataPDes, Date1, Date2)
```

dataPSrc : l'instance de profil qu'on veut trouver les évaluations qui sont égales aux évaluations correspondantes de dataPDes

dataPDes : l'instance de profil qu'on utilise pour faire la comparaison

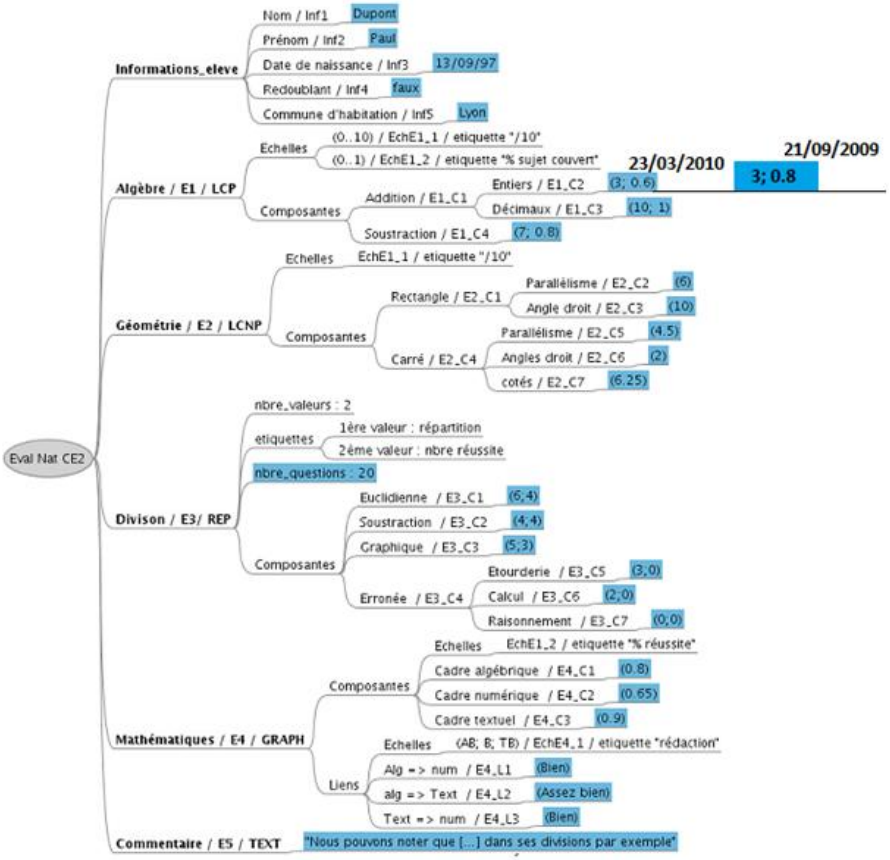
Date1, Date2 : les deux dates constituant l'intervalle de date

Valeur retournée : l'instance de profil qui ne contient que les évaluations égales aux évaluations correspondantes de dataPDes

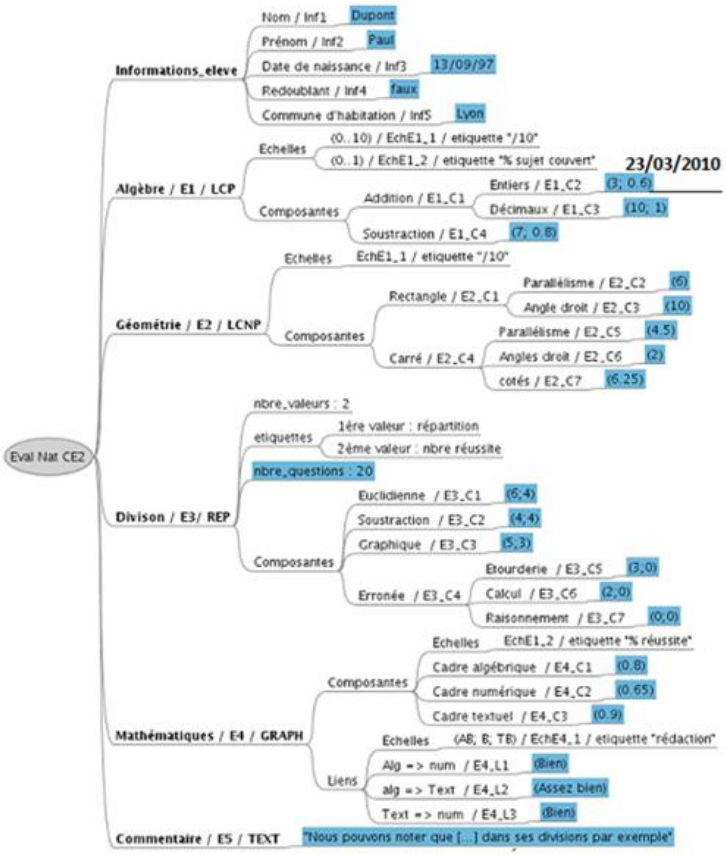
Exemple

data-Equal(Eval Nat CE2, dataPDes, {'E1_C2'}, 'EchE1_1', null, null)

Avec dataPDes :



Résultat :



36. data-CompareProfile

Description

« data-CompareProfile » est un opérateur sur la donnée de profil d'apprenant. Il prend deux instances de profil d'apprenant pour faire la comparaison, l'identifiant de la composante concernant, l'identifiant d'échelle des valeurs où on veut appliquer la comparaison, deux dates indiquant l'intervalle de date. Si on a plusieurs valeurs concernant, on calcule la valeur moyenne et puis on fait la comparaison. Cet opérateur rend une valeur numérique indiquant le résultat de la comparaison.

Rôle

« data-CompareProfile » permet de comparer les deux instances de profil d'apprenant. On peut spécifier aussi un intervalle de date

Utilisation

```
data-CompareProfile(dataP1, dataP2, idComp, idScale, Date1,Date2)
```

Paramètre

dataP1, dataP2 : les instances de profil d'apprenant. Type de donnée : dataP

idComp : l'identifiant de la composante où la comparaison a lieu. Type de donnée : chaîne des caractères

idScal : l'identifiant de l'échelle dont les valeurs sont utilisées pour la comparaison. Cette échelle doit être du type numérique. Type de donnée : chaîne des caractères

Date1, Date2 : deux dates constituant l'intervalle de date. Type de donnée : Date

Valeur retournée : le résultat de la comparaison

« -2 » : erreur (on ne peut pas faire la comparaison)

« -1 » : dataP1 est inférieur à dataP2

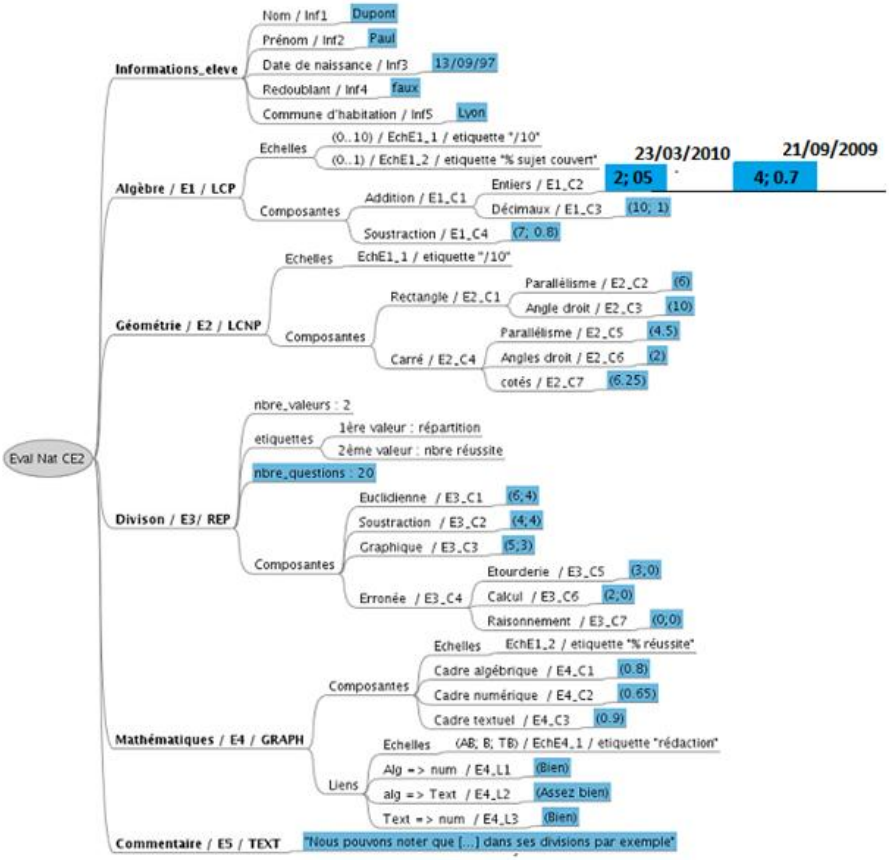
« 0 » : dataP1 est égal à dataP2

« 1 » : dataP1 est supérieur à dataP2

Exemple

data-CompareProfile(Eval Nat CE2, dataP2, ‘E1’, ‘EchE1_1’, null, null)

Avec dataP2 :



Donc, on a

Pour l’Eval Nat CE2 : $\overline{X} = \frac{3+4+10+7}{4} = 6$

Pour le dataP2 : $\overline{X} = \frac{2+4+10+7}{4} = 5.75$

Alors le résultat est 1

37. data-GroupProfile

Description

« data-GroupProfile » est un opérateur sur la donnée de profil d’apprenant. Il prend deux instances de profil d’apprenant pour faire la synthèse et deux dates indiquant l’intervalle de date. Les deux instances de profil doivent avoir une même structure. La synthèse est la valeur moyenne entre deux valeurs correspondantes de deux instances. Cet opérateur rend une autre instance de profil qui ne contient que les valeurs moyennes.

Rôle

« data-GroupProfile » permet de créer un profil d'apprenant qui est la synthèse des profils d'apprenant entrés par moyen. Les profils entrés doivent avoir une même structure. On peut spécifier un intervalle de date pour faire la synthèse.

Utilisation

```
data-GroupProfile(dataP1, dataP2, Date1, Date2)
```

Paramètre

dataP1, dataP2 : les instances de profil d'apprenant entrées. Ces instances doivent avoir une même structure. Type de donnée : DataPro

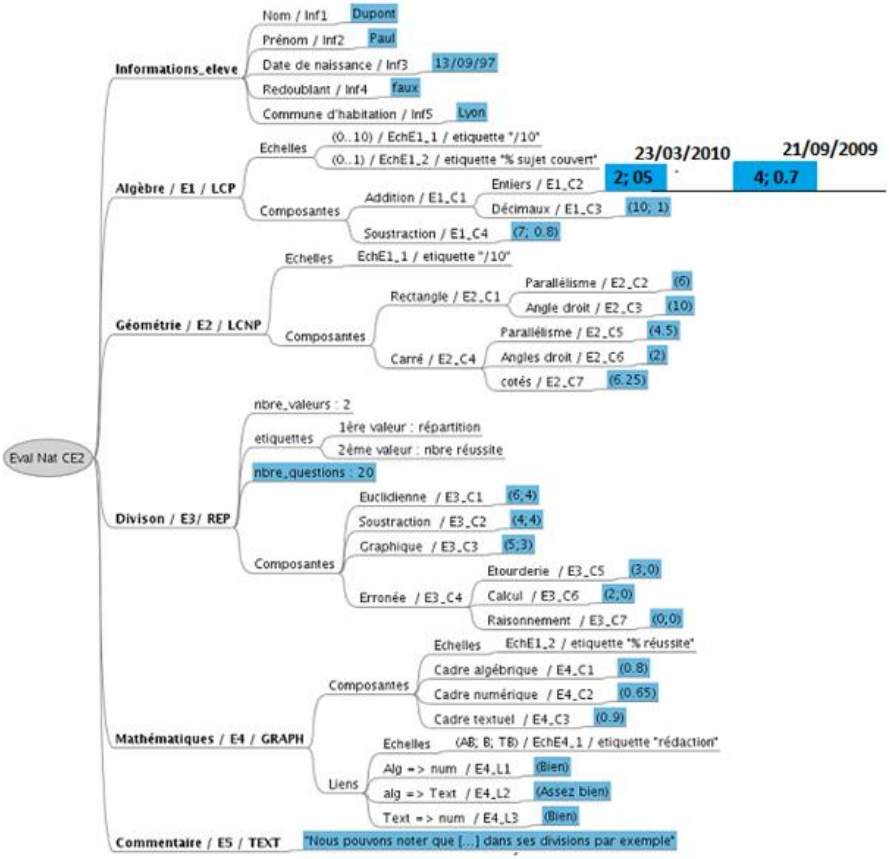
Date1, Date2 : deux dates constituant l'intervalle de date. Type de donnée : Date

Valeur retournée : la synthèse des instances de profil d'apprenant. Type de donnée : DataPro

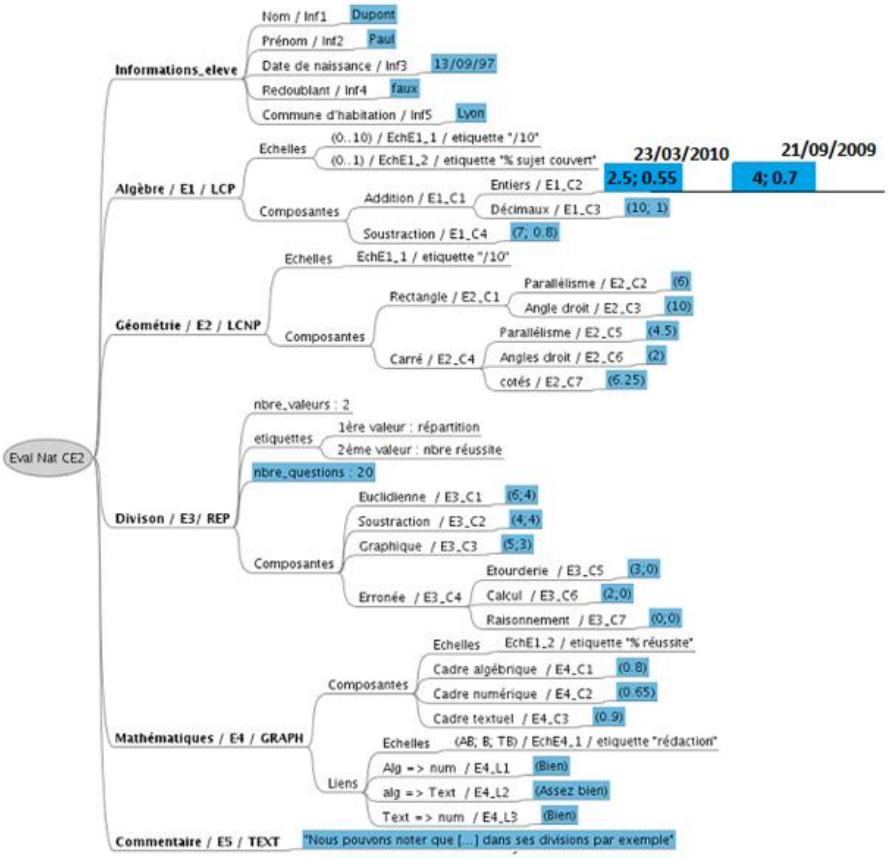
Exemple

data-Synthetize(Eval Nat CE2, dataP2, '01/01/2010', null)

Avec dataP2 :



Résultat :



38. data-ConvertProfile

Description

« convertPro » est un opérateur sur la donnée de profil d'apprenant. Il prend une instance de profil d'apprenant, l'identifiant de la composante souhaitée, l'identifiant d'échelle dont les valeurs ont besoin de convertir, l'identifiant d'échelle des valeurs après la conversion, et la fonction qui faire la conversion. Cet opérateur rend une autre instance de profil dont les valeurs de la composante spécifiée contiennent les nouvelles valeurs.

Rôle

« convertPro » permet de convertir les valeurs d'une instance d'une composante dans une échelle aux valeurs dans autre échelle. Il faut avoir une fonction de convertir d'une valeur de l'échelle d'origine à une valeur correspondante de l'échelle de destination.

Utilisation

data-ConvertProfile(dataP, *list_of_idElement*, foncConvert)

Paramètre

dataP : l'instance de profil d'apprenant. Type de donnée : DataPro

list_of_idElement : liste des identifiants des éléments qu'on souhaite faire la conversion. Type de donnée : chaîne des caractères

foncConvert : une fonctionne de la conversion. Type de donnée : fonction sous forme XML

Valeur retournée : l'instance de profil après d'avoir fait la conversion

8 Annexe 2 : exemple de fonction

Ici, nous montrons un exemple d'une fonction sous forme de balises.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Execution desc="" id="func01" name="Function 01" returntype="StructP" saveas="">
<Input datatype="StructP" id="Input_2"/>
<Input datatype="ListString" id="Input_1"/>
<Input datatype="StructP" id="Input_0"/>
<Operation desc="op01" id="ope01" name="str_DeleteByID">
<Parameter name="strucWP"><InputRef>Input_0</InputRef></Parameter>
<Parameter name="listOfIDs"><InputRef>Input_1</InputRef></Parameter>
</Operation>
<Operation desc="" id="ope02" name="str_Difference">
<Parameter name="strucWP2"><ResultOf>ope01</ResultOf></Parameter>
<Parameter name="strucWP1"><InputRef>Input_2</InputRef></Parameter>
</Operation>
<Operation desc="" id="ope03" name="str_DeleteByLevel">
<Parameter name="strucWP"><ResultOf>ope02</ResultOf></Parameter>
<Parameter name="listOfIDs">a;b;c;d</Parameter>
<Parameter name="level">2</Parameter>
</Operation>
</Execution>
```

Dans cet exemple, on a une fonction qui s'appelle « *Function 01* » avec l'identifiant « *func01* ». Dans cette fonction, on a trois entrées : « *Input_0* » du type « *StructP* », « *Input_1* » du type « *ListString »*, et « *Input_2* » du type « *StructP »*. Cette fonction contient trois opérations : « *ope01* » contenant l'opérateur « *str_DeleteByID* », « *ope02* » contenant l'opérateur « *str_Difference* », et « *ope03* » contenant l'opérateur « *str_DeleteByLevel* ». Pour chaque opération, on a spécifié ses paramètres. Pour l'opération « *ope01* », la valeur du paramètre « *strucWP »* est la valeur de l'entrée « *Input_0* », la valeur du paramètre « *listOfIDs »* est la valeur de l'entrée « *Input_1* ». Pour l'opération « *ope02 »*, on a la valeur du deuxième paramètre « *strucWP2 »* est le résultat de l'opération « *ope01* ». Pour l'opération « *ope03* », la valeur du paramètre « listOfIDs » est spécifiée directement par une liste de chaînes des caractères séparé par des points-virgules. Cette fonction retourne le résultat est une instance structurelle de profil d'apprenant.

9 Annexe 3 : diagrammes de classes pour l'implémentation

Ici, nous présentons les diagrammes de classes utilisés pour l'implémentation du module GROUPE.

1. Les diagrammes de classes pour l'implémentation côté serveur

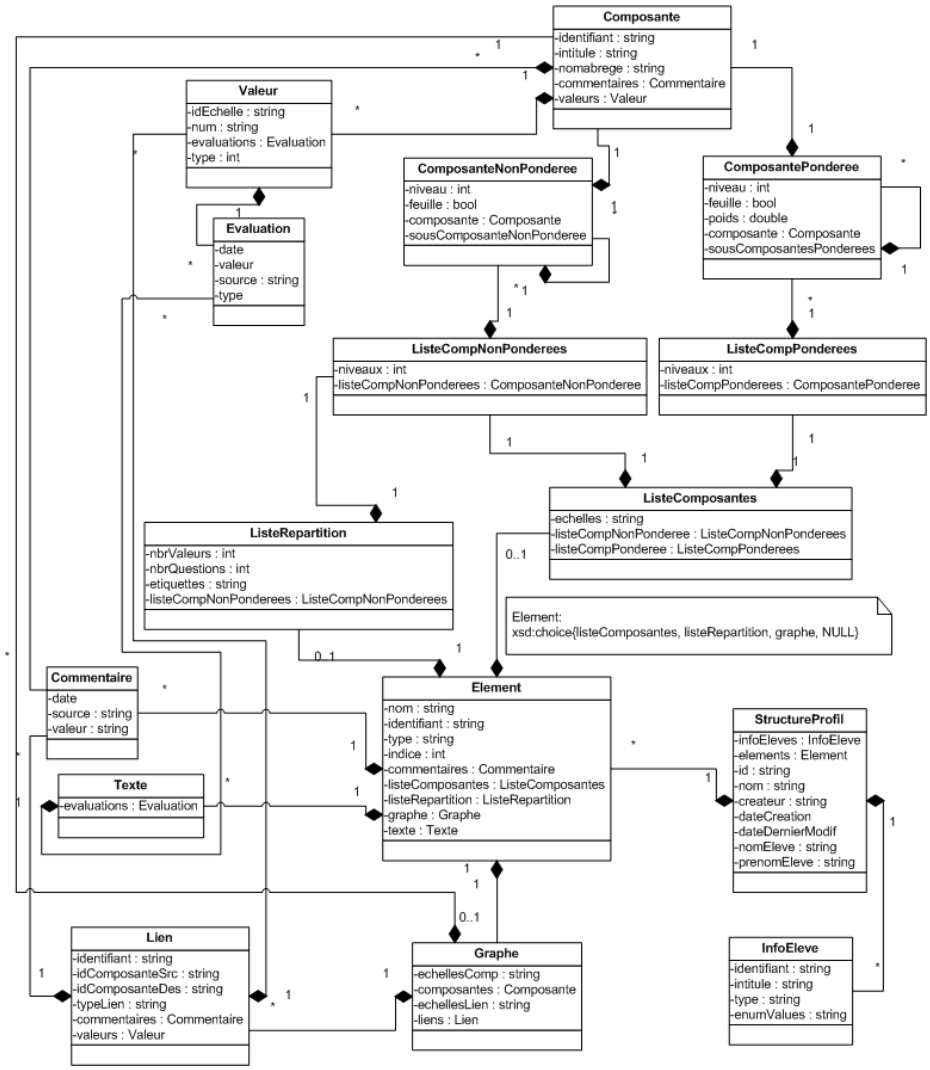


Figure 21: Le diagramme de classes des profils d'apprenant EPROFILEA

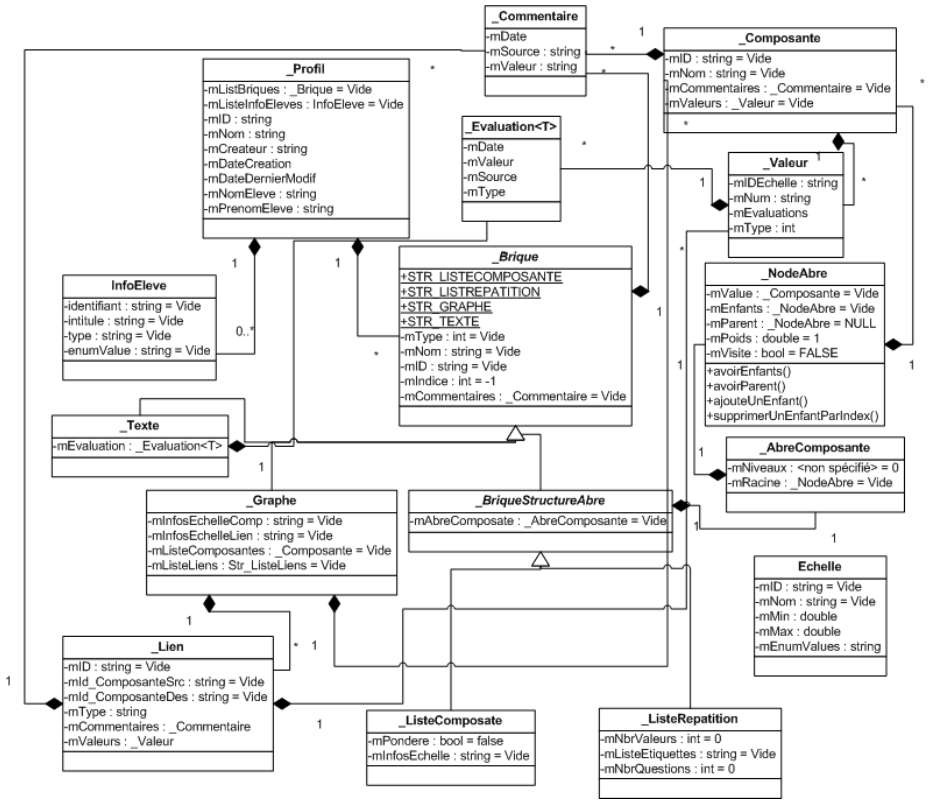


Figure 22: Diagramme de classes des profils d'apprenant dans la boîte noire

2. Le diagramme de classes pour l'implémentation côté client

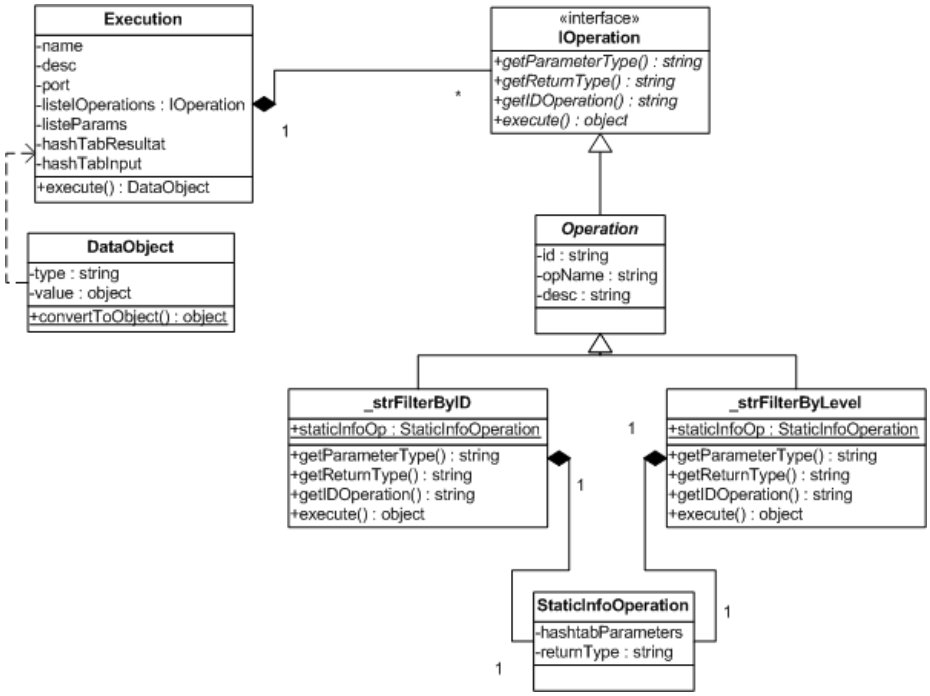


Figure 23: Le diagramme de classes manipulant des fonctions

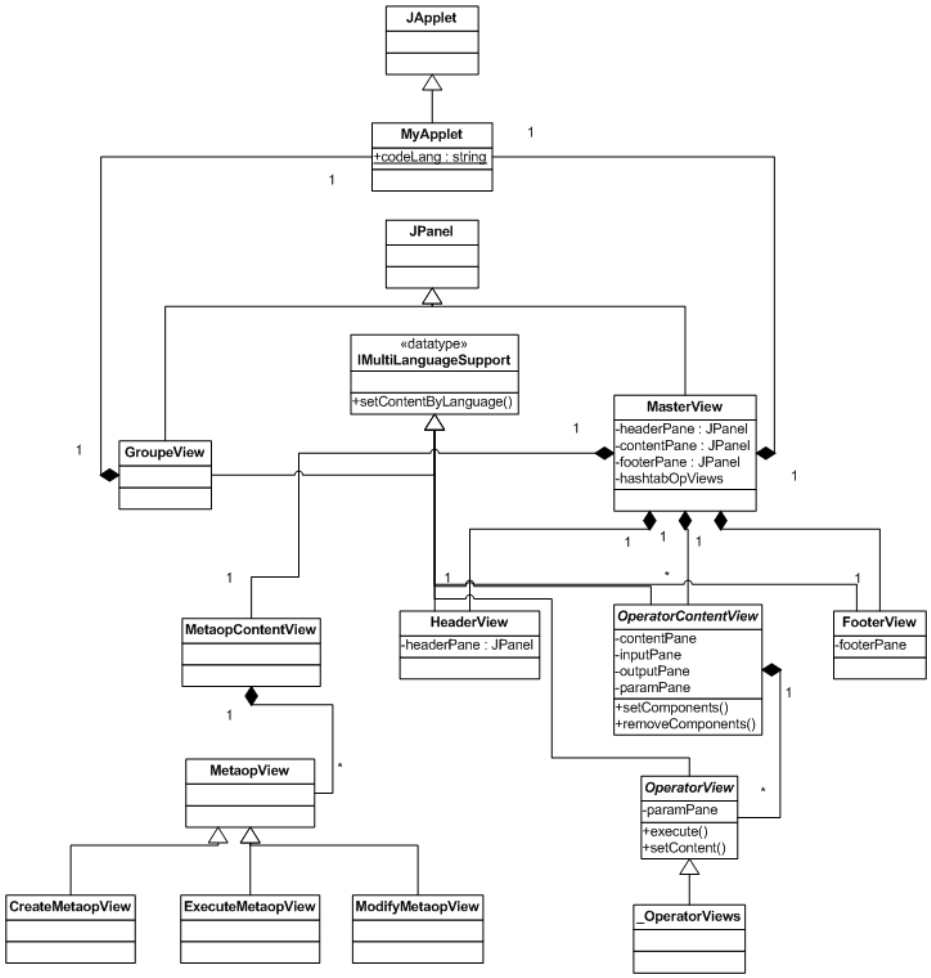


Figure 24: Le diagramme de classes manipulant l'interface

10 Annexe 4 : configurations pour le lancement du module GROUPE

Avant de lancer le module GROUPE, l'utilisateur doit faire quelques configurations :

Créer un variable de l'environnement du système d'exploration avec le nom : EPROFILEA et la valeur : le chemin de l'endroit où le dossier racine d'EPROFILEA se trouve.

Modifier l'URL de services web utilisés par le module dans le fichier EPROFILEA\Fichiers systeme\ParametresGROUPE\Configuration.xml (EPROFILEA : le chemin de l'endroit où le dossier racine d'EPROFILEA se trouve).