REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université Mouloud MAMMERI de Tizi-Ouzou

Faculté génie électrique et informatique

Département informatique

Mémoire de master

Thème

Détection des difficultés d'apprentissage à travers l'analyse sémantique du parcours à base de trace

Proposé et encadré par :

M^me^ AIT ADDA SAMIA

Présenté par :

M^lle^ HADJ AHMED DJOUHAR

M^lle^ LADJIMI DJAMILA

Année Universitaire ***2011/2012***

Remerciements

Avant tout, nous tenons à remercier le Bon Dieu pour le courage, l'aide et la patience qu'Il nous a donné pour mener ce projet à terme.

Que notre enseignante et promotrice Mlle Samia AIT ADDA trouve ici, l'expression de notre large reconnaissance.

Nous remercions très respectueusement les membres du jury d'avoir accepter d'évaluer notre travail.

Nous remerciement vont aussi à nos très chers parents aux quel on doit beaucoup et sans eux ce présent travail ne serait jamais abouti.

Enfin, que tous ceux qui ont contribué de prés ou de loin à l'accomplissement de ce travail, trouvent ici l'expression de nos très profonds respects.

Dédicaces

Je dédie ce travail

A la mémoire de mon cher et tendre frère "Yuva"

A mes très chers parents

A mon frère et ma sœur

A ma copine Farida et à toute sa famille

A toute ma famille

A tous mes amis

Djamila

Dédicaces

Je dédie ce travail avec considération et respect

À tous ceux que j'aime et j'apprécie

À mes très chers parents

À mes frères et mes soeurs

À mes très chers neveux Elyas ❤, sara ❤

Serine💙, Adem💙, Agnes💚, yanis💚& sami ❤

À ma copine Djamila et à toute sa famille

À tous mes amis

À toute ma famille

Djouhar ☺

Liste des figures

Introduction générale

Introduction générale

1. Introduction générale

L'enseignement à distance par Internet, appelé EIAH (Environnement Informatique pour l'Apprentissage Humain), constitue une avancée pédagogique importante. L'EIAH utilise le web (structure hypertexte, capacités multimédias, etc.) comme support de diffusion des connaissances et d'interaction entre les différents acteurs (enseignants, apprenants, etc.). Plusieurs plateformes d'EIAH ont été développées et plusieurs sont disponibles sur le web en libre accès. Ces plateformes sont des environnements qui permettent à un enseignant de créer et de gérer très facilement un cours sur Internet, en lui laissant le libre choix de la méthode pédagogique, et sans nécessiter de compétences informatiques particulières. Elles offrent aussi des outils de communication (forums, chat), des instruments d'évaluation (exercices, sondages, travaux), et la possibilité de déposer des ressources pédagogiques (fichiers PDF, séquences vidéo, etc.). [1]

L'objectif des EIAH est de répondre aux attentes des différents acteurs participants au processus d'apprentissage. La personnalisation et l'adaptation de l'apprentissage sont parmi les attentes les plus importantes.

L'adaptation est définie par H. Dietrich, U. Malinowski, T. Kühme et M. Schneider comme « une tentative de modifier le comportement interactif d'un système en considérant à la fois les besoins individuels des utilisateurs humains et les conditions propres à l'environnement de l'application ».

Adapter le cours aux besoins des apprenants nécessite la détection des apprenants en obstacle par l'analyse sémantique du parcours d'un ou d'un groupe d'apprenants, tout en intégrant la sémantique aux EIAH généralement guidée par les ontologies qui sont « une spécification explicite et formelle d'une conceptualisation partagée » [2]. L'utilisation d'ontologie en EIAH a pour finalité de spécifier des connaissances qui seront interprétables à la fois par l'homme et par la machine. La sémantique constitue la notion principale représentant un mécanisme que nous utilisons à la modélisation de notre proposition.

1.1. Contexte et problématique

Le travail présenté dans ce mémoire s'inscrit dans le cadre générale des Environnements Informatiques d'Apprentissage Humain (EIAH) et le Web sémantique. Particulièrement, nous nous intéressons à la détection des difficultés des apprenants dans (EIAH). Une analyse des traces d'interactions de l'apprenant durant son parcours d'apprentissage permettra d'affiner et de compléter les manques qu'il soit au niveau de la structuration du cours, suivi ou le tutorat au sein des EIAH. L'optique du Web sémantique dans les EIAH est d'expliciter la connaissance contenue dans les cours et de la formaliser afin de pouvoir l'exploiter pour mieux répondre aux besoins des apprenants. D'une manière générale, le contexte scientifique de l'approche proposée est celui de l'analyse d'activités des apprenants à l'extérieur du système d'apprentissage, puis le calcul d'indicateurs pertinents qui permet de synthétiser des statistiques sur les traces de l'espace de travail des apprenants (du groupe) pour déterminer les apprenants en obstacles ainsi que le contexte de ses difficultés (les concepts non maitrisés durant l'apprentissage) [1].

Dans ce contexte, plusieurs questions se posent au sujet de l'amélioration du processus d'apprentissage. La problématique auxquelles nous cherchons à trouver des solutions dans le cadre de ce mémoire comprend deux facettes : 1) comment récupérer et restructurer les traces brutes issues de la source de traçage pour donner naissance à de nouvelles traces modélisées nommées *traces premières*, 2) comment définir les transformations à partir de la trace première et exploiter l'ontologie afin d'énumérer ce que l'apprenant n'a pas compris de ce qu'il a compris et trouver les concepts du cours enseignés qui sont peu, ou pas maîtrisés pour pouvoir lui faire adapter l'apprentissage.

Ainsi la détection des liens sémantiques entre les pages Web visitées par un apprenant et le cours qui lui est proposé dans le cadre de l'enseignement en ligne nous mène à poser ces questions : pourquoi l'apprenant a recours aux pages web pendant son apprentissage ? Est-ce qu'elles sont liées au contenu du cours ? Si oui, quels sont les concepts du cours auxquelles ces pages sont alors liées avec?

1.2. Organisation du mémoire

L'ensemble des chapitres composant ce mémoire sont organisés en trois parties : La première partie est un état de l'art, constituée des chapitres 1, 2 et 3. Le premier chapitre présente la notion de trace fichier log. Dans le deuxième chapitre nous aborderons les différentes méthodes de représentation des documents sous une forme dont l'ordinateur pourra comprendre et ce par les types d'annotation et d'indexation des granules documentaires où nous décrivons les types et quelques algorithmes d'indexation existants. Les approches basées sur l'espace vectoriel pour le calcul de similarité entre documents y sont également présentées. Le troisième chapitre présente le web sémantique où nous présentons aussi la notion d'ontologie et son apport avec les environnements d'apprentissage en ligne.

Dans la deuxième partie, nous effectuons une étude conceptuelle du système à développer. Cela consiste entre autre en l'identification des acteurs et des fonctionnalités à leurs fournir et le choix de l'environnement de travail.

Enfin, la dernière partie est en rapport avec l'implémentation des besoins identifiés durant l'étude conceptuelle. Des annexes ont été ajoutées afin d'apporter de plus amples informations sur certains points décrits dans le rapport.

Chapitre1
Les traces et les fichiers logs

Chapitre 1

Les traces et les fichiers logs

1. Introduction

Les expériences vécues par un individu constituent son histoire, elles sont un matériau pour son futur. Ces expériences peuvent être mobilisées, après qu'elles aient eu lieu, dans d'autres situations. Nombre de recherches se sont intéressées à l'utilisation des expériences passées, en particulier des recherches sur le processus de conception.

Comme notre application consiste à réutiliser l'expérience d'interaction de l'utilisateur avec le système, ce chapitre est réservé pour les traces et les fichiers logs. Pour cela nous allons d'abord définir la trace et les systèmes qui tracent les interactions des utilisateurs système puis on introduira le concept de métacognition et réflexivité, par la suite nous allons parler du raisonnement à partir de l'expérience tracée et définir c'est quoi un indicateur, et pour terminer nous allons aborder la notion de fichier log.

2. Traces

2.1. A la recherche d'une Définitions [3]

Lorsque l'on aborde la question des traces, il faut tout d'abord pouvoir s'entendre sur la définition que l'on associe au terme trace. En effet, il existe actuellement plusieurs points de vue sur ce que pourrait être cette définition d'une trace.

D'après P. Jermann [4], une trace est une observation ou un enregistrement de l'interaction de l'apprenant avec un système en vue d'une analyse. Dans le même sens, J-P. Pernin [5] définit une trace comme un indice de l'activité des acteurs d'une situation d'apprentissage, qu'elle soit ou non instrumentée. Il complète, par ailleurs, sa définition en précisant qu'il s'agit d'un résultat obtenu au cours ou au terme d'une activité, d'un événement ou d'un ensemble d'évènements relatifs au déroulement de la situation d'apprentissage. Dans une optique légèrement différente, P-A. Champin [6] parle d'une séquence d'états et de transitions représentant l'activité de l'utilisateur : «la séquence temporelle des objets et opérations mobilisés par l'utilisateur lorsqu'il utilise le système est appelée trace d'utilisation ». Dans ces trois « définitions », une trace est une trace d'activité, d'utilisation, d'interaction. On parle alors de traces primaires, brutes, de base ou de « bas niveau ». IL ressort également de ces

définitions qu'une trace est temporellement marquée, plus particulièrement lorsque Champin parle de séquence temporelle.

Ces *traces brutes* ne portant aucune interprétation, des traces de « plus haut niveau » sont construites par agrégation ou structuration de traces de « bas niveau », ainsi que par l'application d'opérations (par exemple : des données statistiques) sur celles-ci. Ces traces sont porteuses d'une information plus complexe que les traces primaires. Jermann et Pernin avancent alors le terme d'*indicateurs* pour désigner ces traces secondaires.
On y retrouve plusieurs définitions concernant les traces. Notamment, on y considère qu'une trace est une donnée de base enregistrée par un système.
Les données obtenues en effectuant une opération (calcul, ajout de sémantique) sur une trace ne sont pas considérées comme des traces, elles sont désignées sous le terme d'*indicateurs.*

En conception logicielle, l'enregistrement de l'expérience inscrite et « capitalisée » dans des traces des interactions utilisateur-système est un domaine de recherche actif, à l'origine du développement de systèmes qui tracent ces interactions. Nous présentons ci-dessous les différents types de systèmes « traçants ».

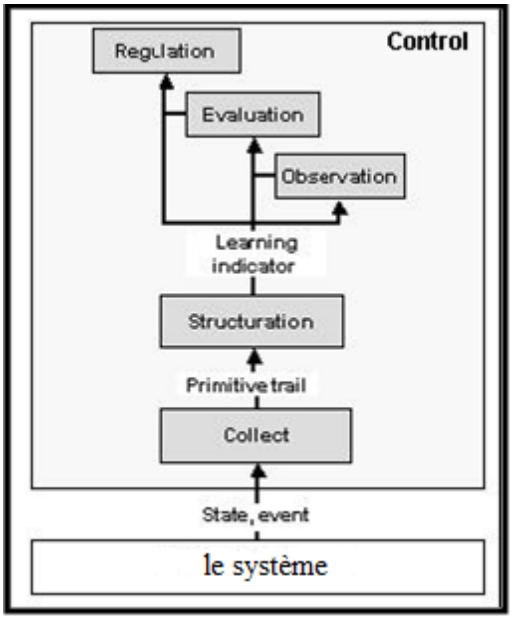


Figure-1-: Le processus général de gestion de trace [7]

2.2. Types de systèmes qui tracent les interactions utilisateur système [8] :

Dans le domaine de l'interaction homme-machine, le traçage des interactions utilisateur système et l'utilisation des traces comme outils de recherche existent depuis longtemps [9]. Ces traces sont des historiques, utilisés pour comprendre la situation d'interaction ou pour assister l'utilisateur dans sa tâche.

Selon le fait que les situations d'utilisations des histoires interactionnelles sont ou non présentées à l'utilisateur il y a quatre types de système qui traces les interactions utilisateur système.

2.2.1. Systèmes utilisant l'histoire interactionnelle sans la présenter aux utilisateurs

Le premier groupe des systèmes identifiés est celui des systèmes qui ne présentent pas l'histoire interactionnelle aux utilisateurs bien qu'ils l'utilisent. Ces systèmes utilisent les traces des interactions entre utilisateurs et système, mais ne les exploitent pas sous forme de visualisation. Des calculs sont réalisés sur ces traces (fichiers logs), en vue de prévoir, conformément à un modèle implicite, les actions futures des utilisateurs et ainsi modifier l'interface pour qu'elle corresponde au comportement « prédit ». Les informations relatives aux interactions sur lesquelles les calculs sont effectués correspondent aux informations de types suivants : accès à des ressources, consultations des écrans, clics effectués, temps passés aux opérations, choix effectués, réponses données aux éventuelles questions, etc. Ces traitements sont automatiques et prévus par le programme. Les actions effectives de l'utilisateur sont comparées à un modèle d'actions prévues. De tels systèmes s'intéressent ainsi aux préférences des utilisateurs pour personnaliser l'interface. Certains de ces systèmes proposent des interfaces qui « donnent et prennent conseils » [10] en interaction avec l'utilisateur. Ces interfaces reflètent les calculs faits sur les interactions utilisateur-système stockées en mémoire, en proposant à l'utilisateur des suggestions d'actions.

2.2.2. Systèmes présentant une visualisation de l'histoire interactionnelle destinée à l'analyste de la situation

Le deuxième groupe de systèmes est celui des systèmes proposant une visualisation des traces d'interactions utilisateurs-système à un analyste de la situation, qui n'est pas l'utilisateur du système. Dans le cadre d'analyses des usages en situations interactionnelles, il peut être intéressant que l'analyste de la situation, par exemple le chercheur, ait accès aux traces des interactions entre utilisateurs et système. Depuis longtemps, les traces informatiques sont

utilisées par les chercheurs pour « espionner » la manière dont les sujets se comportent dans une situation donnée ou utilisent un système, qui peut précisément être le système sur lequel est installé le logiciel étudié. Ce genre d'études a existé en ergonomie, en sciences de l'éducation, en psychologie, et en communication. Pour une situation d'apprentissage instrumenté, Després [11] a développé un système basé sur les traces d'interactions, permettant au tuteur de percevoir l'état d'avancement du travail des apprenants. En interaction homme-machine, un « espion » bien connu est *PlayBack* [12] Les traces d'interactions enregistrées peuvent être à l'origine de comptages divers : temps passés, fréquences d'utilisation, fonctionnalités utilisées ou inutilisées, erreurs, taux de réussite, etc. [13] citent d'autres mesures plus spécifiques telles que le taux de répétition, le taux de composition et la localité [14]. Pour classifier les traces, des méthodes issues des travaux sur la reconnaissance de formes sont appliquées : réseaux de neurones et recherche de séquences répétées.

2.2.3. Systèmes présentant une visualisation de l'histoire interactionnelle destinée à l'utilisateur et lui proposant la possibilité d'y naviguer

Le troisième groupe de systèmes offre une visualisation des traces d'interactions aux utilisateurs, et leur permet de naviguer dans ces informations. Ces systèmes présentent l'histoire interactionnelle aux utilisateurs en vue de supporter leur activité. Les possibilités des utilisateurs d'interagir avec cet historique sont limitées à de la navigation et ne concernent pas le déclenchement de nouvelles actions ni la saisie d'informations déclenchant des actions. Certains systèmes concernent la navigation, d'autres sont destinés à des situations d'apprentissage.

2.2.4. Systèmes présentant une visualisation de l'histoire interactionnelle pour l'utilisateur et proposant la possibilité d'agir dessus

Le quatrième groupe de systèmes concerne les systèmes qui présentent une visualisation de l'histoire interactionnelle à l'utilisateur et qui lui offrent la possibilité d'agir dessus. Ces systèmes utilisent l'histoire interactionnelle comme un outil pour les utilisateurs, pour entrer des données ou des commandes.

Les chercheurs qui développent des systèmes présentant une visualisation de l'histoire interactionnelle pour les utilisateurs font l'hypothèse que cette présentation va permettre aux utilisateurs de prendre de la distance sur leur activité et susciter ainsi une activité sur

l'activité. Ceci les conduit à mobiliser le concept de métacognition que nous expliquant dans le paragraphe ci-dessous.

2.3. Métacognition et réflexivité

Bon nombre de travaux utilisant les traces informatiques pour les présenter aux utilisateurs convoquent le concept de métacognition pour tantôt justifier leur proposition de visualisation des traces, tantôt nommé les activités que la présentation des traces suscite.

2.3.1. Concept de métacognition

La métacognition désigne la connaissance que le sujet a de ses propres processus de pensée et de ceux d'autrui, ainsi que le contrôle qu'il exerce sur ses propres processus cognitifs [15]. Depuis les premières études sur la métacognition, ce concept renvoie à deux types de compétences bien distinctes : les connaissances déclaratives sur le système cognitif et sa nature, et le contrôle et la régulation effective de ce système. Cette distinction connaissance /contrôle est classiquement retrouvée dans les définitions de la métacognition. Selon Flavell, la métacognition réfère aux connaissances que possède un sujet concernant ses propres processus et produits cognitifs. Elle réfère au monitoring actif, à la régulation et à l'orchestration de ces processus et produits, en général en relation avec un but ou un objectif. La définition proposée par [16] attribue ainsi deux dimensions essentielles à la métacognition : les connaissances métacognitives et les régulations métacognitives. Les connaissances métacognitives d'un individu portent sur sa propre cognition et sur la Cognition en général. Les régulations métacognitives réfèrent aux activités supportant le contrôle individuel de la pensée ou de l'apprentissage [17].

2.3.2. La réflexivité

Les systèmes présentés aux 2.2.3 et 2.2.4 sont des systèmes qui présentent une visualisation de l'histoire interactionnelle pour les utilisateurs, en s'appuyant sur la métacognition, ont pour objectif de favoriser la prise de conscience réflexive par les utilisateurs de leur activité. Par activité réflexive, nous entendons davantage qu'une activité réfléchie, c'est-à-dire tournée vers soi. Nous désignons une activité se prenant elle-même pour objet. L'idée est que le système informatique peut servir de « miroir doté de mémoire » pour l'utilisateur, par le biais de la présentation des traces informatiques, ces dernières suscitant chez l'utilisateur une prise de distance par rapport à son activité à l'origine d'une prise de conscience de nature *meta*. Une des méthodes réflexives utilisées en ergonomie consiste à utiliser les traces de l'activité

d'opérateurs (*via* des enregistrements vidéo) comme un outil de construction de savoirs nouveaux [18], par la mise à distance temporelle et physique du sujet face à son activité. Nous présentons ci-dessous une approche informatique pour la conception et l'évaluation « centrées utilisateur » de systèmes visant une réelle coopération entre la machine et l'utilisateur, le raisonnement à partir de l'expérience tracée.

2.4. Raisonnement à partir de l'expérience tracée

L'idée du raisonnement à partir de l'expérience est de concevoir des systèmes qui s'enrichissent au fil de leurs utilisations, à partir du traçage des interactions avec les utilisateurs. Le principe est d'utiliser ces traces comme explicitation des expériences entre utilisateurs et système pour soutenir les utilisateurs dans leur activité. Dans cette partie, nous exposons premièrement en quoi la question du niveau d'abstraction des informations tracées est à prendre en compte. Nous présentons ensuite les fondements du raisonnement à partir de l'expérience tracée, qui a pour objectif de modéliser l'utilisation d'un système par un (ou des) utilisateur(s) pour pouvoir la tracer.

2.4.1. De la nature des traces d'interaction : niveau d'abstraction des informations tracées

Différents types de traces informatiques peuvent être utilisés comme soutien à l'activité. En particulier, le niveau d'abstraction des informations recueillies peut varier. Dans le cas où les traces sont utilisées par le système pour assister l'utilisateur, mais sans qu'elles lui soient présentées, les calculs étant faits sur les fichiers logs, le système récupère les informations issues des calculs pour agir sur les interfaces. Dans ce cas, la question du niveau d'abstraction des traces ne se pose donc pas puisque ce n'est pas un humain qui les reçoit. Si il existe un modèle de l'utilisateur, ou bien si il y a un moteur de raisonnement à partir de cas, le système pourra prédire ce que l'utilisateur fera, soit pour avoir une session d'utilisation « réussie », soit en fonction de ce qu'il a fait auparavant, ou bien encore selon ce que d'autres utilisateurs avec un « profil » similaire ou dans une situation proche ont fait. Il pourra ensuite lui faire des suggestions d'actions. En revanche, dans le cas où les traces sont présentées soit à l'analyste de la situation ou bien à l'utilisateur, les questions de leur niveau d'interprétation, de leur mise en forme puis de leur visualisation se posent. Dans le cas de traces présentées au chercheur ou, de manière plus générale, à l'analyste de la situation, les traces d'interactions peuvent être simplement reconstruites ou plus finement interprétées par le système, en fonction des

objectifs d'analyse. Si l'analyste a des hypothèses de recherche pré expérimentales, il peut faire un modèle de traces en fonction de ses attentes et l'implémenter afin que le système n'enregistre puis ne lui « sorte » que les informations qui l'intéressent, filtrées, mises en forme voire directement annotées. Dans le cas où les traces sont adressées à l'utilisateur, elles peuvent également être plus ou moins interprétées par le système en fonction du type d'utilisateur (âge, expertise éventuelle, proximité culturelle avec la tâche, etc.), en fonction de ce que l'on suppose que cela va susciter chez lui, et également en fonction de l'approche théorique d'assistance que l'on adopte. Dans certains systèmes, on trouve des traces interprétées qui sont des « profils ». Pour une application donnée, ces traces contiennent des informations sur l'utilisateur, sur les dernières actions qu'il a effectué avec l'application. L'idée des profils est que la présentation à l'utilisateur d'une trace interprétée de son activité va le guider et susciter une prise de conscience de son activité. Les traces peuvent aussi être présentées à l'utilisateur dans leur forme « brute ».

Ce paragraphe montre qu'il existe différents niveaux d'abstraction dans les traces d'interactions qui sont utilisées dans les systèmes informatiques, et que ce niveau varie selon l'usage qui est fait de ces traces.

2.4.2. Raisonner à partir de l'expérience tracée

Nous allons présenter le principe du raisonnement à partir de l'expérience tracée, puis exposer quels sont ses liens avec le raisonnement à partir de cas.

2.4.2.1. Principe général du Raisonnement à partir de l'Expérience Tracée :(RàPET)

L'objectif du raisonnement à partir de l'expérience tracée est de participer à la conception de systèmes informatiques capables de fonctionner « en intelligence » avec l'utilisateur [19]. L'idée principale est que le système « apprenne » progressivement à partir des interactions de l'utilisateur médiées par l'environnement informatique. Il s'agit d'utiliser les traces informatiques des interactions utilisateur(s)-système pour les assister dans leur(s) activité(s) en profitant du fait que ces informations peuvent être familières aux utilisateurs puisqu'elles sont constituées de leur(s) expériences(s). Dans la signification coutumière du terme « expérience », Mille rappelle que « toute connaissance trouverait sa source de manière plus ou moins directe dans l'expérience : raisonner consiste en effet à mobiliser des connaissances préalablement établies pour produire un résultat satisfaisant un but particulier ». C'est le niveau d'explicitation de l'expérience qui permet de distinguer le RàPET d'un autre type de processus de raisonnement. L'expérience ne peut être appréhendée que par lestraces laissées

dans l'environnement en tant qu'inscriptions de connaissances émergées ou mobilisées en cours d'expérience, celles-ci devenant sources d'émergence de nouvelles connaissances. Dans le domaine de l'ingénierie des connaissances, l'inscription des « connaissances » dans les systèmes informatiques constitue un champ de recherches à l'origine de traçages de différents types d'informations interactionnelles. Ces informations appelées « inscripteurs de connaissances » sont enregistrées dans un format exploitable par un environnement informatique, c'est-à-dire sous une forme computationnelle dont on pourrait extraire la sémantique des informations par des calculs. Les systèmes experts, les systèmes à base de connaissances, les systèmes basés sur le paradigme du raisonnement à partir de cas (RàPC) sont des systèmes visant l'inscription des connaissances pour assister les utilisateurs.

2.4.2.2. Liens avec le raisonnement à partir de cas

Utilisé en intelligence artificielle depuis le début des années 1980, le raisonnement à partir de cas (RàPC) est une des méthodes de raisonnement par analogies, inspirée de la psychologie cognitive. Il a introduit l'idée de la réutilisation de l'expérience dans le domaine de la résolution de problèmes en intelligence artificielle. Il permet de raisonner à partir de cas ou d'expériences anciennes pour résoudre un problème, de critiquer des solutions, d'expliquer des situations inattendues ou d'interpréter des situations nouvelles [20].

D'après tous ce qu'on a parlés sur les traces on peut dire qu'elles sont des indicateurs qui peuvent subir des traitements pour obtenir des indicateurs de plus grande sémantique.

2.5. Définition d'un indicateur [21]

Selon [22] un indicateur est une variable au sens mathématique à laquelle est attribuée une série de caractéristiques**.** C'est une **variable** qui prend des valeurs représentées par une **forme** numérique, alphanumérique ou même graphique.... La valeur possède un **statut** : elle peut être brute (sans unité définie), calibrée ou interprétée. Le statut identifie une caractéristique bien précise : celle du **type d'assistance** offert aux utilisateurs. Chaque indicateur peut dépendre d'autres variables .comme le temps et, ou même d'autres indicateurs.

Exemple d'indicateur dans les EIAH (environnement informatique d'apprentissage humain) :

Indicateur sur le pourcentage de participation (Participation per-centage PART) :

L'indicateur PART implémenté dans plateforme MODELLINGSPACE représente le taux de participation des acteurs dans n'importe quel type d'activité. Cet indicateur donne le pourcentage des agents qui ont agit dans l'intervalle de temps considéré. La formule de calcul

est la suivante: Par exemple, si PART=0,5 alors la moitié des agents ont interagi dans l'ensemble des modules sur un intervalle de temps. Si PART = 0 alors aucun acteur n'a agit dans le groupe.

- *Les traces de l'utilisation d'un système informatique sont de deux types :*

_ Le logiciel (tel quel ou après avoir été instrumenté) enregistre dans des fichiers (appelés classiquement fichiers de log ou logs) des informations sur ce qu'a fait l'utilisateur.

_ Il est aussi envisageable de filmer l'utilisateur en train d'utiliser le système ; la trace se présente alors sous la forme d'un vidéogramme.

Parce que l'utilisation de la deuxième technique est, en pratique, très contraignante et très coûteuse en temps de traitement, les systèmes d'analyse de traces à partir de fichiers log sont les plus fréquemment mis en œuvre.

3. Fichier log

3.1. Définition [23]

Le terme « log » provient de la langue anglaise (journal de bord des navires). Il est notamment employé en informatique pour désigner un historique d'événements et par extension le fichier contenant cet historique.

Le terme « log » en tant que tel n'apparaît pas dans la réglementation française. Les textes applicables, qu'il s'agisse de textes nationaux ou de directives communautaires, retiennent les termes suivants :

- ✓ « données relatives au trafic »;
- ✓ « données de nature à permettre l'identification » ;
- ✓ « données de connexion à des services de communications électroniques » ;
- ✓ « données de connexion » ;
- ✓ « fichiers de journalisation des connexions

3.2. Le contenu d'un fichier log [24]

Les fichiers logs sont typologiquement sont en nombre de quatre, à savoir:

3.2.1. Les logs de transferts

Ils enregistrent tous les transferts de fichier résultant d'une requête d'un client à un serveur, ils contiennent principalement le nom et le type (image, vidéo…) du fichier demandé ainsi que l'adresse de la machine hôte.

3.2.2. Les logs d'erreurs

Ils conservent la trace incidente survenue lors d'une transaction entre le client et le serveur. Principal intérêt de ces fichiers logs d'erreurs en terme de mesure d'audience d'un site web, réside dans le fait qu'ils donnent la possibilité de savoir dans qu'elle fichier ou téléchargement a été interrompue et donc repéré les pages sur les quelles les utilisateurs quittent le site internet.

3.2.3. Les logs référentiels

Ils facilitent l'identification à la fois du site depuis lequel le client est arrivé(par exemple un moteur de recherche ou un site contenant un lien avec le site étudié) et de la page du site étudié sur lequel le client est arrivé.

Par exemple, la ligne ("http://sitededépart.fr/lien->/accueil.html''signifie que le client est arrivé depuis page lien d'un site appelé sitededépart.fr sur la page appelée accueil de site étudié.

L'analyse de ces fichiers permet donc d'analyser le parcours de navigation des internautes.

3.2.4. Les logs d'agent

Ils archivent notamment les informations portant sur l'équipement informatique et les types des navigateurs utilisées par chaque client.

3.3 Utilité des fichiers logs

Les fichiers logs tracent tous les événements qui arrivent pendant l'activité d'un système. Ils peuvent contenir la preuve en détail de toute activité exceptionnelle, suspecte ou non désirée. Les fichiers logs issus des différents composants d'un réseau peuvent indiquer si la sécurité du réseau est compromise ou en voie de compromission. Par exemple l'utilité d'un fichier log pour un serveur web est :

- ✓ Déterminer les pages (ou produits) les plus et moins populaires ;
- ✓ Déterminer les types de clientèle (localisation, langue, etc. .) ;
- ✓ Suivre l'évolution de l'achalandage /implantation des nouvelles technologies ;
- ✓ Déterminer comment les clients sont arrivés sur votre site web (mots clés dans les moteurs de recherches, référencement par d'autres sites, etc. .) ;
- ✓ Déterminer le parcours des clients dans le site web ;
- ✓ Identifier des problèmes techniques sur le site.

3.4. Quelques formats de fichier log

L'analyse du contenu d'un fichier log exige une bonne compréhension du format et une structure appropriée facilitant son analyse.

En effet, des formats standards ont été développés. Parmi les quels, on peut citer : le format W3C Etendu et le format log commun

3.4.1. Le format W3C Etendu (Word Wide Web Consortium)

C'est un format ASCII adapté avec une variété de champs. L'utilisation de ce format permet d'inclure des champs importants comme elle peut omettre des champs non désirés. Les champs sont séparés par un espace, le temps est enregistré en GMT (Greenwich Mean Time). Ce format est disponible pour les serveurs Web et les serveurs FTP.

Exemple d'une entrée

172.16.25.10 02-05-1998 17:42:15 GET /default.html 200 HTTP/1.0

Cette entrée signifie que le 02 mai 1998 à 17h 42 mn 15s (GMT), un utilisateur ayant l'adresse IP 172.16.25.10 et en utilisant HTTP 1.0 a lancé une requête GET/default.html. Cette requête signifie le téléchargement de la page Web default.html. La requête est exécutée avec succès. La valeur 200 étant le code de l'état du service, cette valeur indique le succès de la requête.

3.4.2. Le format log commun

C'est un format ASCII fixé disponible uniquement pour les serveurs Web. Il a été développé par NCSA (National Center for Supercomputing Applications) à l'université d'Illinois à Urbana-Champaign. Chaque entrée contient les champs suivants :

- ✓ Le nom de l'hôte ou l'adresse IP de l'hôte ;
- ✓ Le nom de l'utilisateur ;
- ✓ La date de soumission de la requête ;
- ✓ Le temps de soumission de la requite ;
- ✓ Le contenu de la requête envoyée par le client ;
- ✓ Le code de l'état HTTP retourné à l'utilisateur : c'est le code de la réponse http envoyé par le serveur au client ;
- ✓ La taille en octets des informations envoyées par le serveur Les champs sont séparés par un espace, le temps enregistré est le temps local.

Exemple d'une entrée:

172.21.13.45 FRED 08-04-1998 17 :39 :10 GET/scripts/iisadmin/ism.dll 200 3401

Cette entrée indique qu'un utilisateur appelé FRED utilisant une adresse IP 172.21.13.45 a envoyé au serveur Web une requête en utilisant la commande HTTP qui est « GET » pour télécharger le fichier scripts/iisadmin/ismi.dll. Cette requête a été soumise au serveur le 08 Avril 1998 à 17h39mn 10 s temps local et elle a été retournée avec succès (code d'état HTTP= 200). Les données envoyées à l'utilisateur FRED ont une taille de 3401 octets.

Remarque :

La réponse d'un serveur à un client peut être faite avec succès ou échec. Le code de l'état de la réponse peut renseigner sur ce résultat.

▪ Si le code d'état ∈ {200, 201, 202, 203, 204, 300, 301, 302, 303, 304}, Alors le serveur a répondu au client avec succès.

▪ Si le code d'état ∈ {400, 401, 402, 403, 404, 500, 501, 502, 503}, Alors le serveur n'a pas répondu à la requête, il y a eu un échec et par conséquent la taille des données transférées au client est égale à 0.

3.5. Les logiciels d'analyse de fichier log

Ces fichiers se présentent sous la forme de lignes de codes laissées sur les serveurs. Pour bénéficier d'une présentation plus digeste, certains logiciels ont été conçus pour faire une synthèse graphique des données.

• Emplacement: logiciel à installer sur le serveur ou sur votre ordinateur • Prix : gratuit et jusqu'à plusieurs milliers de francs • Exemple : Webtrends, Analog,

3.6. Les problèmes liés aux fichiers logs

Malgré l'importance des fichiers logs, néanmoins certains problèmes demeurent posés.

Nous citons :

- ✓ Les fichiers logs consomment un espace disque très grand.
- ✓ Les fichiers logs contiennent beaucoup d'information, ils sont immenses et par conséquent l'analyse de leurs contenus devient une tâche très difficile.
- ✓ Les fichiers logs menacent la vie privée (privacy) de l'utilisateur. Un utilisateur refuse l'idée que toutes ses activités soient enregistrées.
- ✓ Les fichiers logs peuvent être menacés comme d'autres formes de données dans le réseau ou dans un système. Un attaquant qualifié pénétrant dans un système peut effacer les fichiers logs ou modifier leur contenu. Il peut même arrêter le mécanisme d'enregistrement.

4. Conclusion

Nous avons vu dans ce chapitre c'est quoi une trace d'interaction, l'intérêt de tracer l'utilisateur, Nous avons cité une classification des logiciels qui tracent les interactions utilisateurs-système, selon l'usage qui est fait des traces dans ces systèmes. Certains de ces systèmes proposent une visualisation des traces à l'utilisateur, en vue de leur permettre d'avoir une activité m*eta* sur leur activité, et font appel aux concepts de métacognition et de réflexivité.

Nous avons vu que les traces informatiques peuvent avoir différents niveaux d'abstraction, selon le niveau d'interprétation souhaité par le concepteur du logiciel. Comme on a vu le Principe général du Raisonnement à partir de l'Expérience Tracée, et enfin, nous avons vu la définition des fichiers logs, qui est le moyen le plus concret pour tracer.

Comme les adresses des pages visitées sur le web contenues dans la trace sont non porteuses d'informations, l'aspiration de ces pages est nécessaire pour récupérer leurs différents mots clés (leurs contenues), qui seront utilisés par la suite dans le calcule la similarité entre le cours et les pages web visitées. Le chapitre suivant sera réservé pour l'indexation et mesure de similarité.

Chapitre2
L’indexation et mesure de similarité

Chapitre2
Indexation et mesure de similarité

1. Introduction

Pour pouvoir réaliser l'adaptation, l'évaluation des situations d'apprentissage dans les environnements informatiques pour l'apprentissage humain (EIAH) par l'analyse du parcours sémantique d'un ou d'un groupe d'apprenants est nécessaire.

L'analyse sémantique du parcours de navigation de l'apprenant fait appel à l'indexation. L'indexation permet d'identifier la connaissance contenue dans les pages consultées par un apprenant et de la représenter par des mots clés appelés descripteurs afin de détecter les liens sémantiques entre ces pages, et le contenu du cours proposé dans le cadre de l'enseignement en ligne.

Dans ce chapitre nous allons présenter d'abord les différentes méthodes de représentation du contenu des documents, ensuite nous détaillerons les types d'indexation et d'annotations et quelques algorithmes d'indexations. Enfin nous terminerons avec les approches basées sur l'espace vectoriel pour le calcul de similarité entre documents.

2. Les méthodes existantes de représentation du contenu des documents [25]

L'ordinateur ne peut pas comprendre le contenu du document, on doit donc représenter ces documents sous une forme que l'ordinateur peut comprendre et peut traiter. Il y a plusieurs méthodes de représentation de documents textuels et nous avons représenté les plus courantes dans ce qui suit.

2.1. Modèle de l'espace vectoriel (VSM)

Le modèle de l'espace vectoriel (Vector Space Model – VSM) sert de base à la représentation des données textuelles par des vecteurs dans l'espace euclidien. Selon ce modèle, l'élément sémantique de chaque document est le terme. Un terme peut être un mot simple ou un mot composé (un groupe de mots). À base de cette caractéristique, chaque document est représenté par un vecteur des termes. Chaque élément dans le vecteur de terme est la valeur du poids de

chaque terme. Soit, on peut donner le poids du terme en utilisant une mesure, soit on peut l'enregistre simplement comme « présent » / « non présent » en utilisant le codage binaire (1 et 0).

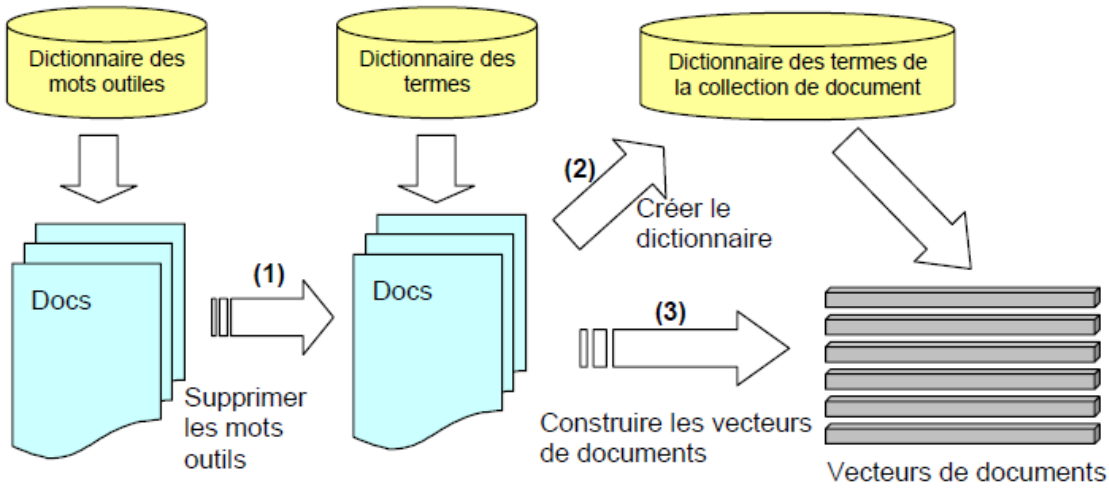


***Figure -2-* :** *Modèle de l'espace vectoriel* [25]

L'algorithme de représentation des documents selon ce modèle se compose de 3 étapes suivantes :

- *Première étape* : On supprime les mots outils dans la collection de documents à l'aide d'un dictionnaire de mots outils. Ce dictionnaire est prédéfini. Les mots outils sont les mots vides qui ne contiennent aucun contenu (par exemple : le, la, les, à, de,...) ou les mots de liaison (par exemple : ce, cette, enfin, cependant, ...), etc.

- *Deuxième étape* : Supposons qu'on a un dictionnaire des termes. À parti de ce dictionnaire et l'ensemble de documents obtenu après l'étape au-dessus, on construit un nouveau dictionnaire des termes qui contient tous les termes apparus dans ces données textuelles. La taille de ce nouveau dictionnaire est inférieure ou égale la taille du dictionnaire précédent.

- *Dernière étape* : En utilisant le nouveau dictionnaire de terme, on calcule le poids de chaque terme dans les documents et construit les vecteurs qui les représentent. La dimension du vecteur représentant des documents est égale le nombre de termes dans le dictionnaire utilisé.

Mais cette approche a quelques inconvénients. Premièrement, chaque document peut contenir beaucoup de termes. Le dictionnaire de terme qui est produit à partir de l'ensemble des documents peut avoir une taille très grande. Le vecteur représentant ces documents pourra avoir donc une dimension très grande. Il est difficile ainsi de traiter un ensemble des grands vecteurs quand le nombre de vecteurs est augmenté et on risque de perdre beaucoup de temps pour faire tout ce traitement. En autre, cette approche ne s'intéresse pas au contexte et à la sémantique de chaque mot dans le document

2.2. Le modèle « Latent Semantic Indexing » (LSI)

Ce modèle est développé à base du modèle de l'espace vectoriel. L'objectif de ce modèle est de diminuer la taille de vecteur représentant des documents. Premièrement, on utilise le modèle de l'espace vectoriel pour construire les vecteurs représentant des documents. On dépose ces vecteurs sur une matrice dont les colonnes sont les vecteurs représentant des documents où chaque ligne contient le poids de chaque terme dans ces documents. On appelle cette matrice: la matrice « terme-document »

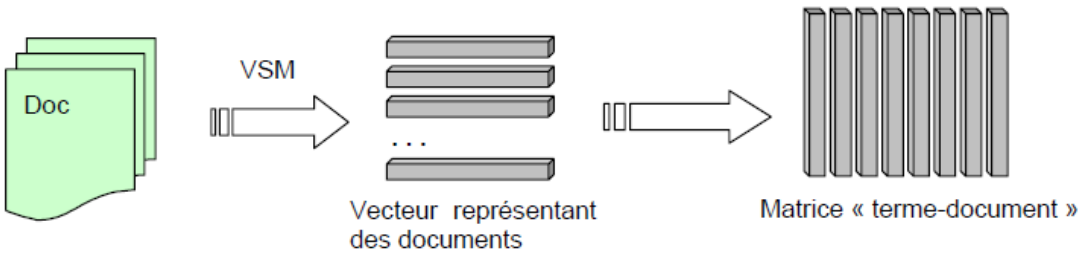


Figure -3- **:** *Génération la matrice « terme-document »* [25]

Généralement, chaque terme n'apparaît pas dans tous les documents, il apparaît seulement dans certains documents. La matrice « terme-document » contient donc beaucoup de composants ayant la valeur zéro. L'idée est de diminuer le nombre de composant zéro et diminuer la taille des vecteurs colonnes en utilisant des méthodes très complexe ce qui nécessite beaucoup de calculs et cela peut prendre énormément de temps pour une grande collection de documents.

2.3. Présentation des documents en utilisant une ontologie

Dans ce modèle, il est nécessaire d'utiliser une ontologie pour pouvoir déterminer les concepts des documents. Le processus est représenté par la figure suivante :

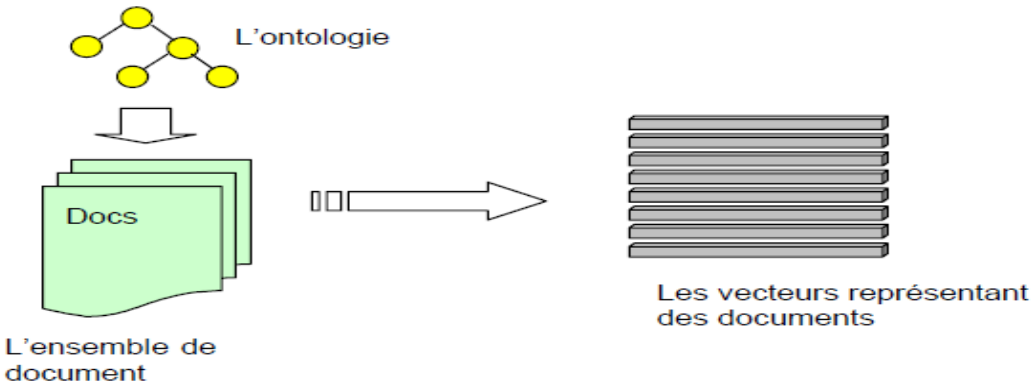


Figure -4- **:** *Représentation des documents en utilisant une ontologie* [25]

Selon cette approche, les caractéristiques de document sont les concepts. Pour chaque document dans la collection, on compte l'apparition des termes représentant chaque concept de l'ontologie et calcule le poids de ce concept en utilisant la mesure ***cf.idf*** *(cf. annexe5)*. Chaque concept dans l'ontologie est représenté par plusieurs termes. Si un des sous concepts de concept A est apparu dans un document, on dit que le concept A est apparu aussi dans ce document. Le vecteur représentant un document est le vecteur des poids de concepts (vecteur concept).

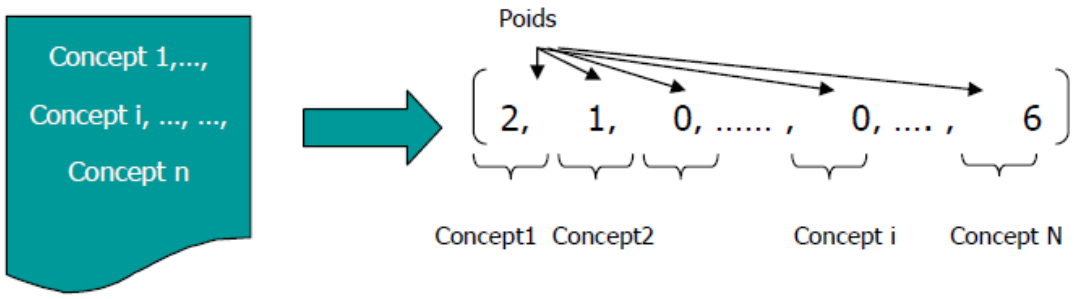


Figure -5- **:** *Représentation un document par un vecteur des concepts* [25]

Grâce à chaque concept contenant un ensemble des termes, la dimension de vecteur représentant le document est plus petite que celle dans autre méthode et elle est fixée (elle est

égale au nombre de concepts dans l'ontologie). Dans notre cas d'études nous penchons sur ce type de représentation des documents.

Représenter les pages nécessite leurs annotation, qui est le sujet de la section suivante.

3. Annotation des documents [26] [27]

3.1. Présentation

L'annotation d'une page web constitue l'outil qui permet d'associer une sémantique au contenu de la page. Annoter c'est accompagner un texte des informations relatives au média (date de création, taille, format d'encodage), des métadonnées présentes dans les documents (auteurs, date de production), des indexes (les descripteurs de contenu du document), de l'identifiant du document par le système (emplacement) et une vue sur le contenu (résumé ou extraits) [28]. Actuellement et avec ce grand volume d'information, il est difficile d'annoter manuellement des millions de ressources mises à la disposition des utilisateurs.

3.2. Types d'annotations

De manière générale, on peut distinguer les annotations externes des annotations internes, selon que l'on a à faire à des métadonnées créées *a priori* en accompagnement de la ressource électronique (interne) ou que l'on a à faire à des métadonnées qui sont retrouvées et combinées *a posteriori* par des systèmes de recherche.: les annotations internes et les annotations externes.

Dans notre cas d'étude, nous nous intéressons exclusivement au premier type (les annotations internes) et cela pour plusieurs raisons. D'abord, [29] affirme que « pouvoir attacher une annotation à une partie spécifique d'un document semble être une fonctionnalité primordiale ». De son côté [30] a montré que la quasi-totalité des annotations faites sur papier font référence à une partie précise du document. [31] affirme que « *les annotations qui font partie du document sont plus utiles, car dans ce cas on peut apercevoir l'annotation et le document* ». [32] dans un autre article affirme que les gens aiment annoter directement sur le document lu, et qu'ils veulent que ces annotations puissent être distinguables de ce document. Le même auteur [33], qualifie la nature des annotations de « unselfconsciousness » dans le document,

ce qui signifie que les annotations ne sont pas autonomes en termes de sémantique car elles ont besoin du contexte de leur ancre pour pouvoir être interprétée par le lecteur.

Nous retenons que les études d'usages ont montré que les utilisateurs préfèrent créer des annotations internes qui soient liées directement au passage annoté. En effet, le contexte physique de l'annotation permet de se rappeler et de retrouver plus facilement « pourquoi » un annotateur l'a posée dans cette partie du document.

En réalité, il existe une bonne partie des pages web qui ne sont pas annotées. Nous distinguons alors deux cas de figure: le premier est celui où les pages Web sont accompagnées de métadonnées. Dans ce cas, les mots clés sont extraits directement des pages. Dans le deuxième cas, les pages Web ne contiennent pas de métadonnées. Ainsi, l'extraction des métadonnées se fait par des algorithmes d'indexation que nous expliquerons par ailleurs. Cependant il existe plusieurs types d'indexation que nous détaillerons dans ce qui suit.

4. Indexation de granules documentaires [35]

4.1. Définition

L'indexation d'un texte, consiste à repérer dans son contenu certains mots ou expressions particulièrement significatifs (Appelés termes d'indexation) dans un contexte donné qui ont un caractère réducteur car tous les termes d'un document ne sont pas importants à prendre en compte et à créer un lien entre ces termes et le texte d'origine. [27]

4.2. Les types d'indexation

Traditionnellement l'indexation est réalisée par des documentalistes de manière *manuelle*. L'analyse du document est donc réalisée par une personne et non par une machine, ce qui est très coûteux en temps car l'indexeur doit lire et comprendre un document avant de pouvoir l'indexer correctement. Avec l'augmentation de la quantité de documents à indexer, l'indexation tend à être automatisée. *L'indexation automatique* consiste à extraire les mots les plus utilisés dans un texte, car on considère qu'un mot fréquemment utilisé est important. Ces termes sont alors qualifiés de mots-clés, ou de descripteurs lorsqu'ils sont issus d'un vocabulaire contrôlé.

Mais lors de l'indexation, on doit éviter de sélectionner un mot vide (le, de, un…) comme descripteur ou mot-clé. C'est la raison pour laquelle, lors d'une indexation automatique, on ignore généralement les mots courts, car ils sont considérés comme des mots vides par défaut. Cependant, cela peut entrainer une perte d'information. En effet cette méthode par exemple ignore le mot « os » alors que ce dernier est un mot important si on travaille dans le domaine de l'archéologie. Un deuxième problème de l'indexation automatique se situe au niveau de la sémantique des mots, car le processus de l'indexation automatique compte le nombre d'occurrences d'un mot sans prendre en compte le sens de chaque mot. Par exemple, le mot « la » désigne à la fois un article ou une note de musique, de même le mot « orange » désigne une couleur et un fruit. Si les deux sens du mot sont utilisés dans un même texte le mot sera compté deux fois alors qu'il n'a pas le même sens. Un autre problème rencontré par l'indexation automatique se situe au niveau des mots composés. Par exemple, considérons le mot composé « pomme de terre ». Ce dernier sera indexé au niveau de « pomme » et de « terre » mais pas au niveau de « pomme de terre » qui est son sens initial.

Pour améliorer l'indexation automatique, il est possible d'utiliser des dictionnaires de conjugaison. Ces dictionnaires ont pour but de retrouver l'infinitif de chaque verbe afin de trouver le nombre de fois qu'un verbe apparait sans tenir de sa conjugaison.

Cependant même si l'indexation automatique semble la plus appropriée lorsqu'un très grand nombre de documents doit être analysé, elle ne constitue pas la solution idéale. En effet en raison des problèmes qui viennent d'être soulevés, les résultats obtenus sont bien moins pertinents que ceux résultant d'une indexation manuelle.

Une méthode d'indexation intermédiaire existe, c'est *l'indexation semi-automatique* ou indexation contrôlée. Dans un premier temps, un programme d'indexation automatique analyse les documents et propose des mots-clés, c'est-à-dire les mots qui sont les plus souvent répétés dans le texte, suivant le principe de l'indexation automatique. Ensuite dans un deuxième temps, une personne chargée de l'indexation relie les mots-clés proposés par le programme d'indexation et confirme ceux considérés comme pertinents. Cette méthode d'indexation permet de filtrer les résultats obtenus à partir de l'indexation automatique en supprimant des mots vides par exemple. Cette méthode semi-automatique a l'avantage d'améliorer la rapidité de l'indexation grâce à l'analyse automatique tout en conservant une

précision élevée garantie par la validation humaine. Le schéma ci-dessous résume les trois types d'indexation.

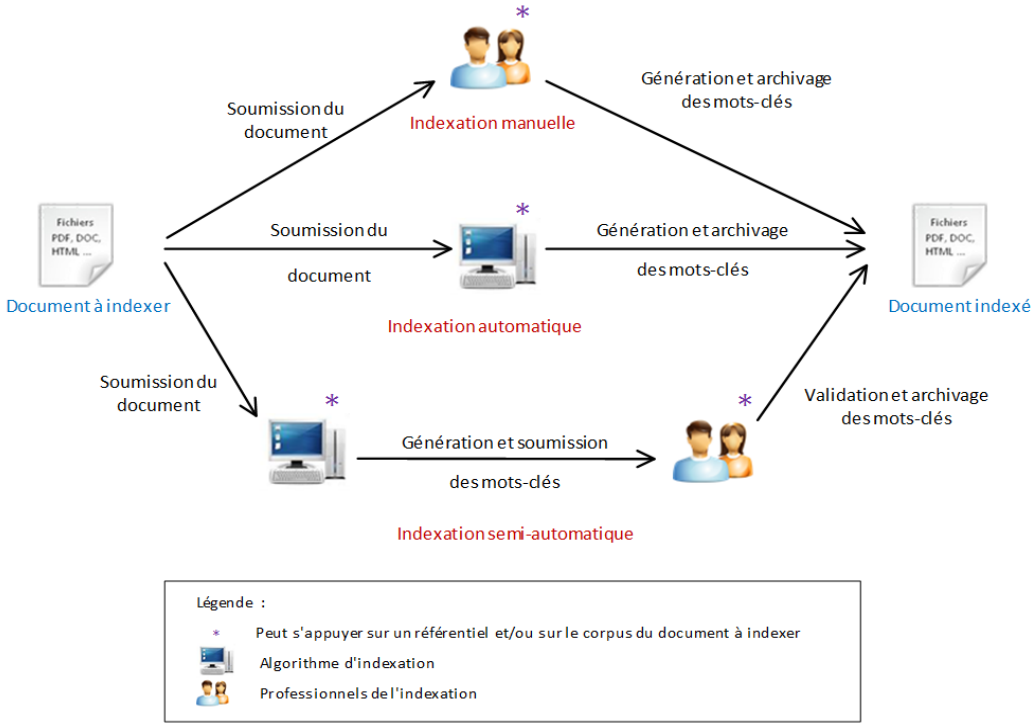


Figure -6-: *Schéma des différents types d'indexation [35]*

On distingue aussi deux types de langages d'indexation :

- Langage contrôlé : Utilise un lexique de descripteurs figé, l'indexation est le plus souvent manuelle (un professionnel choisit un ou plusieurs descripteurs pour représenter le document), parfois semi-automatique.

- Langage libre : Les descripteurs sont extraits automatiquement des documents (le plus souvent un document est indexé par la liste des mots qui le composent).

5. Quelques algorithmes d'indexation [27]

5.1. Indexation par la sémantique latente, vers une indexation conceptuelle

L'indexation à partir de la sémantique latente est une approche statistique qui vise à extraire la sémantique implicite contenue dans les documents. Cette approche regroupe les termes ayant des caractéristiques communes dans les documents et considère que les regroupements sont

des unités de sens. L'approche a pour but d'éviter la polysémie et la synonymie des termes retenus comme descripteurs par les approches statistiques classiques en regroupant les termes ayant des caractéristiques communes dans leur apparition dans les documents. La méthode a été créée à l'origine pour permettre une représentation des documents pouvant s'appliquer à des collections spécifiques en s'adaptant aux variations lexicales.

5.2. Indexation sémantique

C'est une autre approche d'indexation vise à s'appuyer sur des ontologies pour représenter les granules documentaires, ce type d'indexation s'appelle l'indexation sémantique. L'indexation sémantique repose sur l'intuition suivant laquelle le sens des informations textuelles (et des mots qui composent les granules) dépend des relations conceptuelles entre les objets du monde auxquels elles font référence plutôt que des relations linguistiques et contextuelles trouvées dans leur contenu [36]. L'indexation sémantique n'est possible que par l'existence et l'utilisation de ressources décrivant explicitement l'information correspondant aux objets. Deux types de démarches dans l'indexation sémantique : la démarche issue de la recherche d'information (RI) et la démarche issue du Web Sémantique.

5.2.1. Types de démarches dans l'indexation sémantique

5.2.1.1. La démarche issue du domaine de la RI

Consiste à choisir comme langage de représentation des documents, l'ensemble des concepts et instances de l'ontologie. L'utilisation d'ontologies sous forme de hiérarchies de concepts. Les descripteurs ne sont plus choisis directement dans les documents ou dans un vocabulaire contrôlé (ou thésaurus) mais au sein même de l'ontologie. Les granules documentaires sont alors indexés par des concepts qui reflètent leur sens plutôt que par des mots bien souvent ambigus [37]. Il convient dans ce cas d'utiliser une ontologie reflétant le ou les domaines de connaissance abordés dans la collection documentaire. Il est en effet nécessaire de retrouver dans l'ontologie les concepts présents dans la collection pour indexer les documents à partir de toutes les thématiques abordées.

5.2.1.2. La démarche orientée Web Sémantique

L'indexation sémantique est un type d'indexation qui s'inscrit également dans la démarche orientée Web Sémantique. Les précurseurs de cette nouvelle version du Web considèrent que les ressources participant au Web Sémantique seront toutes reliées entre elles par des relations sémantiques. Plus précisément, les données présentes sur le Web Sémantique seront modélisées sous forme d'ontologies où chaque ressource apparaît comme un élément de ces ontologies au même titre que la connaissance qui les décrit. L'objectif est donc d'ajouter au contenu du Web une structure formelle et de la sémantique (à travers des métadonnées et de la connaissance) dans le but de permettre une meilleure gestion et un meilleur accès aux informations. Cette démarche repose sur des ontologies modélisant les objets du monde à travers les acteurs et entités que les documents constituent et comportent [38]. Elles peuvent être vues comme une représentation des métadonnées explicitement ou implicitement présentes dans les documents. La mise en place de cette nouvelle vision du Web dépend de la présence de ces métadonnées. Un enjeu actuel du Web Sémantique est de définir des techniques permettant de les extraire [38] [39]. La démarche orientée Web Sémantique a donc un double objectif : indexer le contenu des documents à partir des ressources permettant d'en extraire les concepts et instances mais aussi représenter les ressources en générant les métadonnées correspondantes.

5.3. La différence entre l'indexation sémantique et conceptuelle

Certains auteurs différencient l'indexation sémantique de l'indexation conceptuelle [40]. L'indexation conceptuelle repose, pour eux, sur des hiérarchies de concepts ou ontologies de domaine, alors que l'indexation sémantique repose sur l'utilisation d'ontologies génériques telles que WordNet. L'ontologie WordNet est limitée par rapport à la sémantique qu'elle peut contenir par contre les ontologies de domaine peuvent par leur formalisation représenter des ressources impliquant un engagement sémantique plus fort. Donc l'indexation sémantique est l'indexation de granules documentaires à partir de n'importe quelle ontologie de domaine. L'indexation sémantique se fait en deux étapes. La première étape consiste à identifier les concepts ou instances de l'ontologie dans les granules. La deuxième étape pondère les concepts pour chaque document en fonction de la structure conceptuelle dont ils sont issus [36].

5.4. Indexation multilingue [26]

L'étape la plus importante de l'indexation est le choix des entités d'indexation représentant le contenu du document c'est-à-dire ses descripteurs. Dans le cadre de l'indexation multilingue, cette phase délicate se complexifie car elle passe obligatoirement par une étape de « traduction ». Avant de traduire véritablement un terme, il faut passer par une étape de reconnaissance du concept identifié par celui-ci. Il ne faut donc plus rechercher des termes mais des concepts [42]. Pour cela, on doit lever au moins trois types d'ambiguïtés sémantiques [43]. Nous illustrerons les ambiguïtés de sens par des exemples tirés d'un corpus français/anglais.

La polysémie : Un même terme peut avoir différents sens. Le terme anglais "*plant"* illustre bien ce problème car il possède au moins 3 sens différents (la plante, l'installation technique, le coup monté). Par contre, "*power plant"* n'a qu'un seul sens. Donc, si l'on tient compte des termes voisins au terme ambigu, c'est-à-dire son contexte, on arrive à déterminer son sens exact.

L'homographie : deux mots différents s'écrivent de la même façon. Par exemple, "*livre"* est soit la conjugaison du verbe livré, soit le nom synonyme d'ouvrage. La catégorie lexicale du terme (nom, verbe, adjectif) permet de lever cette ambiguïté de sens.

Le sens large : un terme qui a un sens très large, (exemple : « *air »* peut prendre un sens particulier dans certain domaine (« *air bag* »). Il se peut que dans une autre langue, un concept particulier résultant de l'association de termes généraux, dans un syntagme nominal (SN)[1], soit décrit par une toute autre combinaison de terme. *« Air bag »* se traduit par « *sac gonflable de protection »*. Pour résoudre ce problème, il faut identifier la totalité de l'expression, parmi les composantes du syntagme nominal pour déterminer le concept spécifique qui se cache derrière la combinaison des termes.

Il existe plusieurs approches pour le calcul de lien sémantique entre documents et nous allons présenter les plus utilisées dans la section suivante.

[1] syntagme (nom masculin) [linguistique] Groupe de morphèmes, de termes dont la succession a un sens et qui forment une unité fonctionnelle.

6. Mesure de similarité [44]

Soit X l'espace des données ou univers, une mesure de similarité est usuellement définie de la manière suivante :

6.1. Définition : *Une mesure de similarité* S *est une fonction* $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *qui satisfait les propriétés suivantes :*

– *Positivité :* $\forall x, y \in \mathcal{X}, S(x, y) \geq 0$

– *Symétrie :* $\forall x, y \in \mathcal{X}, S(x, y) = S(y, x)$

– *Maximalité :* $\forall x, y \in \mathcal{X}, S(x, x) = S(y, x)$

D'autres propriétés peuvent être requises comme la normalisation qui impose que les valeurs appartiennent à l'intervalle [0,1].

6.2. Similarité entre documents

Les documents sont représentés par des ensembles de vecteurs de concepts, et la similarité entre les documents est calculée par une agrégation des similarités entre les vecteurs. Les similarités entre vecteurs sont calculées par des mesures basées sur l'espace vectoriel. Ces approches utilisent un vecteur caractéristique, dans un espace dimensionnel, pour représenter chaque objet et calculent la similarité en se basant sur des mesures. Le modèle de l'espace vectoriel est employé pour un arrangement des objets complexes en les représentants comme des vecteurs de k-dimensions. La définition de la similarité entre deux vecteurs d'objets est obtenue par leurs contenus internes. Parmi les approches citées dans la littérature :

6.2 .1.Similarité de Jaccard

La mesure de similarité de Jaccard est définie par le nombre des objets communs divisé par le nombre total des objets moins le nombre d'objets communs :

$$\text{Simj}(Q,D) = \frac{\sum_{i=1}^{N} q_i \cdot d_i}{\sum_{i=1}^{N} q_i^2 + \sum_{i=1}^{N} d_i^2 - \sum_{i=1}^{N} q_i \cdot d_i}$$

Où q_i et d_i représentent les éléments des vecteurs Q et D de taille N.

6.6.2. Similarité de Cosine

Cette mesure utilise la représentation vectorielle complète, c'est-à-dire la fréquence des objets (mots). Deux objets (documents) sont similaires si leurs vecteurs sont confondus. Si deux objets ne sont pas similaires, leurs vecteurs forment un angle (V1,V2) dont le cosinus représente la valeur de la similarité. La formule est définie par le rapport du produit scalaire des vecteurs $v1_i$ et $v2_i$ et le produit de la norme de $v1_i$ et de $v2_i$.

$$\text{cosinus}(V1,V2) = \frac{\sum_{i=1}^{m} v1_i.v2_i}{\sqrt{\sum_{i=1}^{m} v1_i.v1_i} * \sqrt{\sum_{i=1}^{m} v2_i.v2_i}}$$

Où *v1i* et *v2i* représentent les éléments des vecteurs V1 et V2 de taille m. La mesure de Cosine quantifie donc la similarité entre les deux vecteurs comme le cosinus de l'angle entre les deux vecteurs.

6.2 .3. Similarité de Dice

La similarité de Dice est définie par le nombre des objets communs multipliés par 2 sur le nombre total d'objets. La mesure de Dice est donc définie par la formule suivante :

$$\text{Sim}(X,Y) = \frac{\mathbf{2}.\sum_{i=1}^{n} \boldsymbol{xi}.\boldsymbol{yi}}{\sum_{i=1}^{n} \boldsymbol{xi^2} + \sum_{i=1}^{n} \boldsymbol{yi^2}}$$

Dans le cadre de notre travail, nous nous sommes basées sur la méthode Jaccard pour mesurer la similarité entre les documents visités à l'extérieur du cours et celle du cours. Ce choix est arbitraire, car les trois méthodes que nous avons signalées ne se distinguent pas beaucoup, car la mesure dépond du contenu des documents à comparer.

7. Conclusion

Nous avons vu dans ce chapitre les différentes méthodes de représentation des documents, les types d'annotations et d'indexations existantes, et quelques approches de mesures de similarité entre documents. Le but du chapitre était de présenter l'intérêt de l'indexation des granules documentaires qui est l'identification de la connaissance contenue dans un texte et de la représenter par des mots clés appelés descripteurs afin de l'utiliser dans l'évaluation de similarité entre les concepts qui décrivent le cours et les pages Web consultées durant une session d'apprentissage.

Nous nous sommes basées sur la représentation sémantique de document, désigné souvent par le terme indexation sémantique ou conceptuelle. Le but principal est d'identifier les concepts caractérisant le contenu du document. Le challenge est de s'assurer que les concepts non pertinents ne seront pas pris et que les concepts pertinents ne seront pas omis.

Dans le chapitre suivant nous allons détailler le concept du web sémantique et ontologie que nous avons par ailleurs mentionnée au cours de ce chapitre dans l'indexation sémantique et conceptuelle sans aucune définition préalable. Dans le même contexte, nous allons montrer aussi l'apport du web sémantique dans les technologies de l'apprentissage en ligne.

Chapitre3

Le web sémantique et E-learning

Chapitre 3

Le web sémantique et E-learning

1. Introduction

Dans ce chapitre nous intéressons aux apports des technologies du web sémantique aux systèmes e-learning, nous allons présenter la notion de web sémantique et comme il repose sur la notion d'ontologie nous présenterons cette dernière, et nous terminons par l'apport des ontologies aux systèmes e-learning, que nous présentons par ailleurs durant ce chapitre.

2. Web Sémantique [45]

2.1. Présentation générale

D'un point de vue technique, le développement du Web est mené par un consortium international nommé le W3C (World Wide Web Consortium). Ce consortium regroupe des membres d'universités ou de grandes entreprises, discutant et orientant les évolutions techniques du Web. Le W3C est un organisme important et écouté. Parmi ses membres, le plus connu est sûrement Tim Berners-Lee, inventeur du Web. En 2001, dans un article [46] fondateur, Tim Berners-Lee et ses collaborateurs présentent leur vision du futur Web, appelée « Web Sémantique », c'est-à-dire un Web dont le contenu est compréhensible par les machines.

Actuellement, une page Web est réalisée pour que le contenu soit compris par les humains (*Des pages Web pour les humains*) alors que les machines n'en perçoivent que la forme. La *Figure-1-* illustre cette différence. Un humain lit un texte qui évoque des concepts tandis qu'une machine perçoit juste des mots, successions de caractères alphabétiques séparées par des espaces. Une machine peut réaliser des traitements simples sur ces mots. Par exemple, reconnaître que les mot2 et mot10 sont identiques.

Néanmoins, cette reconnaissance syntaxique est insuffisante quand un ensemble de mots désigne un objet du monde réel ou quand un mot peut posséder plusieurs sens. Par exemple, toujours dans la *Figure-7-*, un humain comprend aisément que les mots « Gérard » et « Dutilleul » désignent un être humain précis alors qu'une machine aura plus de difficultés si elle ne possède pas l'information que « Gérard » est un prénom français courant. De plus, le mot « cinquième », isolé, est ambigu et une machine peut difficilement savoir qu'il désigne un niveau scolaire.

En conclusion, une machine est limitée pour interpréter des contenus textuels. Il lui manque des connaissances extérieures aux contenus et des références non-ambiguës vers ces connaissances.

Leçon de mathématique Par Gérard Dutilleul Cinquième D Théorème de Pythagore	mot1 mot2 mot3 mot4 mot5 mot6 mot7 mot8 mot9 mot10 mot11

Figure-7- **:** Extrait de contenu pédagogique. Ce que lit un humain, à gauche, et ce que perçoit une machine, à droite [45].

2.2. Définition du Web Sémantique

Le Web Sémantique est un courant technologique reposant sur des représentations informatisées des connaissances appelées « ontologies » (cf. paragraphe 2). Généralement, ces représentations se font à l'aide de concepts (cf. paragraphe 3.3.1) et de relations entre concepts (cf. paragraphe 3.3.2). Le but final est de permettre des raisonnements automatisés sur ces connaissances. Par exemple, imaginons quelques experts en pédagogie qui se réunissent pour créer une ontologie du domaine pédagogique. Ils définissent trois concepts : « document », « devoir » et « contrôle ». Ils créent ensuite des relations entre ces concepts ; ils expriment que le concept de « document » est plus générique que les deux autres, et que les concepts de « devoir » et de « contrôle » sont similaires. Dans cette ontologie, ils formalisent une partie de leurs connaissances du domaine qui pourra alors être traitée par des machines.

Pour la recherche de documents, l'existence des ontologies permet d'indexer les documents non plus selon des mots-clefs mais selon des concepts reliés entre eux. Le moteur de recherche pouvant s'appuyer sur l'ontologie précédente peut donc déduire que si un utilisateur recherche un « devoir », il peut être utile de lui fournir aussi les documents indexés par « contrôle ». Grâce aux ontologies, les machines disposent ainsi d'une connaissance du domaine pouvant améliorer les résultats des recherches. Comme le web sémantique repose sur les ontologies, nous allons présenter cette notion dans le paragraphe qui suit.

3. Notion d'ontologie [47]

Historiquement, l'ontologie est un concept philosophique. Il désigne la science de l'être en général. L'ontologie décrit une théorie à propos de la nature de l'existence selon le paradigme " *On ne cherche pas à comprendre le monde mais à le représenter* " [48]. Plus tard, l'ontologie est apparue en pleine lumière dans le domaine de l'intelligence artificielle, afin de résoudre les problèmes de modélisation des connaissances et, plus précisément, en ingénierie des connaissances. Ceci a engendré de nombreuses définitions que nous allons résumer dans les paragraphes suivants en présentant les points de vue philosophique et informatique.
Les ontologies se construisent dans un domaine donné afin d'atteindre un objectif spécifique.

3.1. Définitions issues de la philosophie

Dans le domaine de la philosophie, l'ontologie est considérée comme une branche de la métaphysique qui s'intéresse à la nature et l'organisation de la réalité. Elle a une signification plus large, celle de « *science de ce qui existe* » dans laquelle on ne cherche pas à expliquer le monde, mais à le représenter. Elle s'applique à « *l'être en tant qu'être* », indépendamment de ses déterminations particulières. Cette définition philosophique est tirée de deux définitions issues respectivement du dictionnaire de l'Académie française et du dictionnaire Webster.

• En tant que discipline de la philosophie, intéressons à la toute dernière définition de l'Académie française. " *ONTOLOGIE XVIIe siècle*" *Emprunté du latin scientifique Ontologie, lui-même composé à l'aide de Onto-, tiré du grec* ôn, ontos, *« étant, ce qui est », et Logia-, tiré du grec* logos, *« discours, traité ». Partie de la philosophie qui a pour objet l'être en tant qu'être, qui étudie les propriétés générales de l'être*" [49].

La définition de l'Académie a le mérite d'être claire. Mais un sens nouveau est apparu. Il s'agit d'une représentation formalisée d'ensembles de concepts d'un domaine donné, assortis de leurs relations, utilisée en informatique. Le terme s'utilise le plus souvent au pluriel, des *ontologies*. Au singulier, l'ontologie désigne la discipline qui traite de la modélisation des concepts et de leurs relations en vue de leur exploitation par des machines.
Cette définition est plus proche du point de vue de l'intelligence artificielle, comme on le verra dans ce qui suit.

3.2. Définitions issues de l'intelligence artificielle

L'intelligence artificielle a permis de représenter les connaissances d'un domaine sous forme de base, dite base de connaissances, et d'automatiser leur utilisation et la résolution de

problèmes autour, à travers des inférences de données. Néanmoins, les bases de connaissances ne sont, globalement, pas réutilisables, ce qui limite leur utilisation. La notion d'*ontologie* a été introduite, entre autres, pour pallier cette limite.

D'une manière générale, une ontologie est vue comme un ensemble de concepts permettant de modéliser un ensemble de connaissances dans un domaine donné. Un concept peut présenter plusieurs sens thématiques. Les concepts sont liés entre eux par des relations sémantiques, des relations de composition et d'héritage. Afin de préciser cette notion, de nombreux chercheurs ont proposé des définitions qu'il est utile d'examiner :

• Une définition générale a été donnée par Thomas R. Gruber où il décrit une ontologie comme une spécification explicite d'une conceptualisation modélisant des concepts et les relations entre concepts [50].

• Plus tard, John F. Sowa a spécifié de façon plus précise cette notion. Dans sa définition, l'ontologie est vue comme un catalogue de types issus de l'étude des catégories d'entités abstraites et concrètes qui existent ou peuvent exister dans un domaine [51].

• Après, afin de compléter le sens philosophique originel, Guarino a introduit la notion d'ontologie formelle, qui est définie en tant que modélisation conceptuelle, ou une représentation de cette modélisation. " *Une ontologie est un accord sur une conceptualisation partagée et éventuellement partielle* " [52].

• De même, Uschold définit une ontologie comme une description formelle d'entités et de leurs propriétés, relations, contraintes et comportements. De plus, les auteurs ont introduit, dans [53], la notion de l'ontologie explicite "*An explicit ontology may take a variety of forms, but necessarily it will include a vocabulary of terms and some specification of their meaning*".

• Enfin, Christophe Roche a donné une définition générique et simple "*Une ontologie est une conceptualisation d'un domaine à laquelle sont associés un ou plusieurs vocabulaires de termes. Les concepts se structurent en un système et participent à la signification des termes. Une ontologie est définie pour un objectif donné et exprime un point de vue partagé par une communauté. Une ontologie s'exprime dans un langage (représentation) qui repose sur une théorie (sémantique) qui garantit des propriétés de l'ontologie en termes de consensus, cohérence, réutilisation et partage*" [48].

Nous retenons cette dernière définition car elle englobe et résume les définitions précédentes.

Dans ce qui suit, nous présenterons, d'une manière non exhaustive, les éléments qui constituent l'ontologie.

3.3. Eléments constitutifs de l'ontologie

Les ontologies sont, à l'heure actuelle, au cœur des travaux menés en ingénierie des connaissances. Elles permettent de représenter les connaissances et les manipuler automatiquement, tout en gardant leur sémantique. Les connaissances sont définies à travers des *concepts*. Les liens entre concepts sont appelés *relations*. Afin de relier les concepts, l'ontologie se présente, généralement, sous forme d'une organisation hiérarchique des concepts.

3.3.1. Concepts

Selon Uschlod et Grüninger [53], un concept peut représenter un objet matériel, une notion ou une idée. C'est une représentation de l'esprit qui abrège et résume une multiplicité d'objets empiriques ou mentaux par abstraction et généralisation des traits communs identifiables.

L'ensemble des propriétés d'un concept s'appelle sa *compréhension* ou son *intension*, et l'ensemble des objets ou êtres qu'il englobe, son *extension*. L'intension du concept peut généraliser un ensemble de propriétés qualitatives ou fonctionnelles communes aux individus auxquels il s'applique. L'extension du concept est représentée par un ensemble d'objets qui sont appelés *instances du concept*.

Exemple. : Une bibliothèque scolaire est caractérisée par un bibliothécaire, des livres de différents domaines, des magazines, etc. On peut définir un concept "*C* " pour désigner les livres, par exemple. L'intension de ce concept peut être LIVRE, et son extension pourra être son domaine : livre de programmation, livre d'ingénierie des connaissances, etc.

Les concepts sont organisés en taxonomie au sein d'un réseau de concepts et peuvent être structurés hiérarchiquement. Un concept est caractérisé par un ensemble de propriétés. Dans [54], l'auteur s'inspirait des propriétés proposées par Guarino. Les plus intéressantes:

• Un concept est *générique* s'il n'admet pas d'extension. La vérité, par exemple, n'a pas d'extension.

• Un concept porte une propriété d'*identité* si cette propriété permet de différencier deux instances de ce concept. Par exemple, dans un système de gestion de fichier, un nom désigne d'une manière unique un fichier ou un répertoire. Le nom est une identité du fichier ou répertoire.

• Un concept est *rigide* s'il ne peut pas être une instance d'autres concepts. Par exemple, l'être vivant est un concept rigide, mais un " être humain " n'est pas un concept rigide, car l'humain est une instance du concept " être vivant ".

• Un concept est *anti-rigide* s'il peut être une instance pour d'autres concepts. Comme le cas de l'être humain dans l'exemple précédent.

Les concepts peuvent être *équivalents* s'ils ont la même extension. Ils peuvent être *disjoints* ou incompatibles si leurs extensions sont disjointes. Ils peuvent aussi être *dépendants* : un concept *C*1 est dépendant de *C*2 si, pour toute instance de *C*1, il existe une instance de *C*2.

3.3.2. Relations entre concepts

Les concepts (respectivement, les instances) peuvent être reliés entre eux par des relations au sein d'une ontologie. Une relation est définie comme une notion de lien entre des entités, exprimée souvent par un terme ou par un symbole littéral ou autre. Généralement, les liens sont classés en deux catégories : des liens hiérarchiques et des liens sémantiques. La relation hiérarchique reprend la structuration d'*Hyperonymie*/*Hyponymie*, tandis que la relation sémantique lie les concepts par un lien, dit *partie-tout* ou (whole-part), qui correspond à la structuration d'*Holonymie*/*Meronymie*.

Une relation *hiérarchique* lie un élément supérieur, dit l'hyperonyme, et un élément inférieur, dit l'élément hyponyme, ayant les mêmes propriétés que le premier élément avec au moins une en plus. Dans certains cas, le couple (*Hyperonymie*, *Hyponymie*) s'interprète par (*Type*, *Sous-Type*). L'hyperonyme englobe l'hyponyme. On pourra alors écrire « HYPONYME est une sorte de HYPERONYME ». *Exemple* : Le chat est une sorte d'animal, donc, " animal " est l'hyperonyme de chat.

Une relation *sémantique*, comme la relation partie-tout, parfois appelée " Partie-de ", ou (holonyme / méronyme) [2] est une relation hiérarchique qui existe entre un couple de concepts dont l'un dénote une partie et l'autre dénote le tout. La relation partie-tout est différente de celle d'hyponymie par le fait qu'un hyperonyme impose ses propriétés à ses hyponymes, par contre le TOUT dispose des propriétés qui ne sont pas obligatoirement transmises à ses parties.

Exemple : Dans le corps humain, la tête et les jambes font partie du corps mais elles ne disposent pas des mêmes propriétés. Tête n'est pas une sorte de Corps (voir la figure -2-).

Comme les concepts, les relations peuvent aussi avoir des propriétés. Ces dernières peuvent être *algébriques* (symétrie, réflexivité, transitivité). Elles peuvent être des propriétés de

[2] X est un méronyme de Y si X est une partie de Y.
Y est un holonyme de X si X est une partie de Y.

cardinalité, comme par exemple, un ordinateur qui dispose, d'au moins, un disque dur. En général, ces relations sont binaires.

Afin de décrire les concepts et relation d'une ontologie, celle-ci s'exprime dans un langage et repose sur un formalisme, que nous expliquerons dans la partie suivante du chapitre.

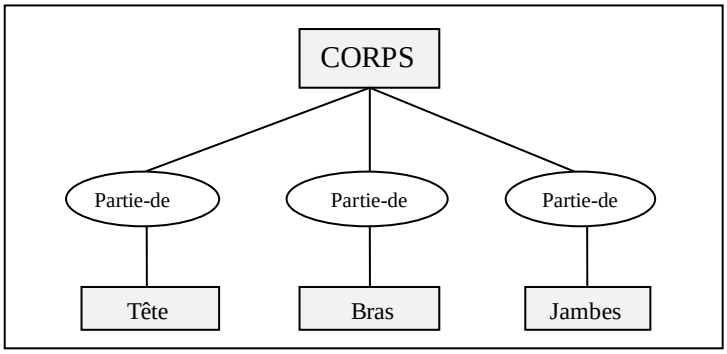


Figure-8- **:** Exemple de la relation partie-de [47]

3.4. Langages et plateformes pour les ontologies

Il existe de nombreux langages informatiques, plus ou moins récents, spécialisés dans la création et la manipulation des ontologies. Nous en décrivons quelques-uns dans ce qui suit.

3.4.1. XML, RDF, OWL et SKOS

Les ontologies peuvent être décrites en XML (Extensible Markup Language). C'est un langage qui permet de décrire des métadonnées en facilitant leurs traitements et leurs échanges.

D'autre part, RDF (Resource Description Framework) est un modèle de graphe destiné à décrire, de façon formelle, les ressources Web et leurs métadonnées et permettre le traitement automatique de telles descriptions.

DFS (Resource Description Framework Schema) est un langage extensible qui permet la représentation des connaissances. Il appartient à la famille des langages du Web sémantique publiés par le W3C. Il fournit des éléments de base pour la définition d'ontologies ou de vocabulaires destinés à structurer des ressources RDF. Dublin Core est un schéma générique de métadonnées qui permet de décrire des ressources numériques ou physiques et d'établir des relations avec d'autres ressources. Les déclarations de termes du Dublin Core sont représentées en RDFS.

Cependant, RDF et RDFS souffrent de limites, comme l'impossibilité de raisonner et de mener des raisonnements automatisés sur les modèles de connaissances établis à l'aide de ces langages. En conséquence, un nouveau langage, OWL (Web Ontology Language), est apparu.

OWL (Web Ontology Language), est un dialecte XML fondé sur une syntaxe RDF. Il fournit les moyens pour définir des ontologies Web structurées. Il se différencie du couple RDF / RDFS par le fait que c'est un langage d'ontologies, contrairement à RDF.

Si RDF et RDFS apportent à l'utilisateur la capacité de décrire des classes et des propriétés, OWL intègre, en plus, des constructeurs de comparaison des propriétés et des classes : identité, équivalence, contraire, cardinalité, symétrie, transitivité, disjonction, etc. Ainsi, OWL offre aux machines une plus grande capacité d'interprétation du contenu web que RDF et RDFS, grâce à un vocabulaire plus large et à une vraie sémantique formelle.

SKOS (Simple Knowledge Organisation System) [3](cf. annexe1) désigne une famille de langages formels permettant une représentation standard des thésaurus, classifications ou tout autre type de vocabulaire contrôlé et structuré. SKOS est construit sur la base du langage RDF, dont il est une application, et son principal objectif est de permettre la publication facile de vocabulaires structurés pour leur utilisation dans le cadre du Web sémantique.

De nombreux éditeurs d'ontologies sont apparus. *Protégé*[4] est l'un des éditeurs d'ontologie les plus utilisés. Il peut lire et sauvegarder des ontologies dans la plupart des formats d'ontologies : RDF, RDFS, OWL, SKOS.

3.4.2. LOOM

LOOM est une plate-forme pour la représentation des connaissances. Son objectif principal est de construire des applications intelligentes. Les connaissances déclaratives dans LOOM sont composées de définitions, de règles, de faits, etc. Pour compiler les connaissances déclaratives, LOOM utilise un moteur déductif. Ce dernier est un classifieur qui utilise le chaînage-avant, l'unification sémantique et des technologies orientées objet.

SUMO est l'une des ontologies utilisées dans LOOM par l'intermédiaire d'un outil *SUMO2LOOM* [68]. Motta à, lui aussi, montré qu'il est plus facile de compléter une ontologie existante que de partir de rien. Il utilise le langage OCML. Le projet, appelé *WebOnto*, consiste en une application Java couplée à un serveur Web qui permet de naviguer et d'éditer des modèles de connaissance [55].

[3] http://www.w3.org/2004/02/skos/

[4] http://protege.standford.edu

3.4.3. ONTOLINGUA

Ontolingua est un mécanisme qui permet aux utilisateurs de créer et manipuler des ontologies[5]. Il supporte les ontologies portables pour qu'elles soient traduites dans différents systèmes. Ontolingua est basé sur le langage d'interchange KIF (Knowledge Interchange Format).

Celui-ci est conçu pour l'échange de connaissances entre des systèmes informatiques répartis. Thomas Gruber a introduit la syntaxe et la sémantique utilisées dans KIF dans [50]. Ontolingua permet aussi de traduire des ontologies génériques en LOOM, KIF, etc.

3.4.4. OIL

OIL (Ontology Inference Layer) est un langage dédié à la spécification et à l'échange des ontologies sur le Web[6]. Il permet la représentation et l'inférence d'ontologies, en combinant des primitives de modélisation des langages de frame avec la sémantique formelle et les modes de raisonnement des logiques descriptives. Ainsi, il représente une ontologie par un conteneur (ontology container) et des définitions ontologiques (ontology definition). Pour cela, il se base sur des formalismes tels que RDF/RDFS et XML, ce qui garantit sa totale compatibilité avec ces formalismes standards ou des formalismes en cours de standardisation. Les liens existant entre la structure d'un document et la modélisation du domaine couvert par ce document sont étudiés dans [56] au travers d'une comparaison entre OIL et les schémas XML.

3.4.5. SHOE

SHOE (Simple HTML Ontology Extensions) est une extension du langage HTML qui permet aux auteurs de pages Web de générer une annotation de leurs documents, compréhensible par la machine[7]. Ce langage peut être utilisé par des agents pour la gestion des pagesWeb [57].

En effet, autant le langage HTML est utilisé pour rendre la connaissance facilement lisible par un humain, autant il n'est pas adapté pour permettre cette lisibilité à un système informatique. Un agent chargé d'extraire la sémantique d'un document a beaucoup de difficulté à le faire, car les données et leur présentation sont entremêlées. SHOE évite ce problème grâce à sa propriété d'inclusion des données directement lisibles et exploitables dans les pages Web.

---

[5] http://www.ksl.stanford.edu/software/ontolingua/

[6] http://www.ontoknowledge.org/oil/

[7] http://www.cs.umd.edu/projects/plus/SHOE/

3.5. Typologie des ontologies

Les ontologies peuvent être classifiées selon plusieurs critères. Ces derniers sont explicités dans [58], afin de déterminer une typologie d'ontologies :

• l'objet de conceptualisation ;

• le niveau de détail ;

• le niveau de complétude ;

• le niveau de formalisme de représentation.

Comme notre travail se base sur une ontologie de domaine, qui est classifiée selon l'objet de conceptualisation, nous limitons à présenter les catégories de cette typologie qui sont au moins de quatre.

3.5.1. Typologie selon l'objet de conceptualisation

Par rapport à l'objet de la conceptualisation de l'ontologie, quatre catégories au moins peuvent être identifiées :

3.5.1.1. Les ontologies de domaine

Décrivent le vocabulaire lié à des domaines particuliers comme la physique, la mécanique, la chimie, la médecine et la modélisation d'entreprise. Selon Guarino [52], les ontologies de domaine sont définies de deux manières :

• des ontologies spécifiques à un domaine particulier, contenant le vocabulaire spécifique à un domaine précis ;

• des ontologies de tâches qui contiennent l'ensemble des tâches réalisées dans un domaine particulier. Elles décrivent un vocabulaire en relation avec une tâche ou une activité générique. Elles fournissent un ensemble de termes au moyen desquels on peut décrire, au niveau générique, comment résoudre un type de problème. D'après Mizoguchi [59], l'ontologie de tâche caractérise l'architecture computationnelle d'un système à base de connaissances qui réalise une tâche.

Les ontologies ont été employées dans divers domaines et pour différents objectifs. Elles sont devenues un thème grandissant et d'actualité au sein des travaux de recherche menés dans l'EIAH [60] Il y a de plus en plus de projets mettant en œuvre des ontologies traitant différents aspects de la formation à distance via le web.

3.5.1.2. Ontologie d'application

Contrairement a l'ontologie de domaine, l'ontologie d'une application donnée ne peut pas être réutilisée pour d'autre application, elle sert à décrire des conceptualisations de domaine spécifique à l'application en question.

3.5.1.3. Ontologie générique (ontologie de haut niveau)

Cette ontologie a l'objectif d'exprimer les connaissances acceptables par différents domaines, elle permet de catégoriser les choses du monde, par exemple, les relations, les actions, l'espace, le temps, etc.

3.5.1.4. Ontologie de représentation des connaissances (méta ontologie)

Elle décrit les concepts utilisés par les langages de représentation des ontologies.

4. Les ontologies pour le e-learning [61]

En Juin 2000 la commission européenne définit le e_learning comme : « *l'utilisation des nouvelles technologies multimédias et de l'Internet, pour améliorer la qualité de l'apprentissage en facilitant l'accès à des ressources et des services, ainsi que les échanges et la collaboration à distance* ». Le e-learning et comme tout autre services sur le Web peut bénéficier de la nouvelle vision du Web sémantique tout en reposant particulièrement sur le potentiel des ontologies.

Cette partie sera consacrée à illustrer l'apport des ontologies aux systèmes e-learning, mais au départ on présentera quelques notions relatives au domaine du e-learning.

4.1. Définitions

4.1.1. Objet pédagogique

Un objet pédagogique, ou Learning Object (LO), peut être défini comme "*toute entité, sur un support numérique ou non, pouvant être utilisée pour l'apprentissage, l'enseignement ou la formation*". Pour Yolaine Bourda et Marc Hélier, "*les objets pédagogiques peuvent être, par exemple, des transparents, des notes de cours, des pages Web, des logiciels de simulation, des programmes d'enseignement, des objectifs pédagogiques, etc.*". [62] Un objet pédagogique

peut être réutilisé pour différentes fins. Par exemple, un exercice peut bien servir dans une série de TD que dans le cadre d'un examen.

4.1.2. Profil utilisateur

Il s'agit d'un ensemble de données persistantes qui caractérisent un utilisateur ou un groupe d'utilisateurs particuliers. Un tel modèle peut contenir des caractéristiques sur les connaissances, les préférences, les centres d'intérêts, etc... d'un utilisateur. [48]

4.1.3. Plate forme de formation

Une plate-forme de formation est un logiciel qui assiste la conduite des formations à distance. Elle est basée sur des techniques de travail collaboratif et regroupe les outils nécessaires aux trois principaux acteurs de la formation : apprenant, tuteur, administrateur. Une plate forme utilise des moyens de travail et de communication : visioconférence, e-mail, forums, chats, etc. L'usage de ces systèmes est relativement standard, le tuteur crée des parcours de formation, incorpore des ressources pédagogiques et effectue un suivi des activités des apprenants. L'apprenant, peut consulter en ligne ou télécharger les contenus pédagogiques, effectuer des exercices, s'auto-évaluer et transmettre des travaux à son tuteur pour les corriger. Les apprenant et les tuteurs communiquent individuellement ou en groupe, et peuvent créer des thèmes de discussion. L'administrateur, de son côté, assure l'installation et la maintenance du système. [62]

Aujourd'hui il existe plusieurs choix concernant les plates formes de formation, on peut citer : Plone, Moodle, Gannisha, Cartable électronique... etc.

4.2. Besoin des systèmes e-learning

On aborde ici les différents besoins d'un système e-learning, ainsi que le rôle des ontologies à satisfaire ces besoins :

4.2.1. Besoin en archivage et recherche

Une application e-learning est mise en ligne via l'utilisation du Web. Compte tenu de la diversité et la croissance exponentielle des ressources pédagogiques utilisées dans le cadre d'une formation de type e-learning, il est de plus en plus difficile de trouver les documents

pédagogiques pertinents. Une application e-Learning partage donc le même problème de pertinence avec le Web lorsque les apprenants veulent accéder au savoir mis à leur disposition. [62]

L'approche basée sur la recherche d'informations dans la ressource même est limitée d'une part, par l'absence de certaines informations qui ne sont pas généralement contenues dans la ressource et d'autre part, par la nature multimédia des ressources : la plupart des ressources de type image ou son ne contiennent pas d'information textuelle.

La communauté du e-learning a convenu d'utiliser des métadonnées. Les métadonnées fournissent un ensemble commun de balises qui peuvent être appliquées à n'importe quelle ressource, ce qui permet aux organisations de décrire, indexer et rechercher leurs ressources.

Dans la perspective du Web sémantique, qui est en voie de devenir une assise pour les environnements de formation à distance, les ontologies offrent de façon spécifique une *sémantique riche*, mieux que toute autre méthode de représentation des connaissances connue.

Dans une problématique de recherche d'un contenu pédagogique sur une plate forme d'enseignement, reposer sur le vocabulaire conceptuel définit dans une ontologie peut aider à améliorer la précision de cette recherche en évitant des ambiguïtés au niveau terminologique et en autorisant des inférences diminuant le bruit et augmentant la pertinence.

Prenons l'exemple simple d'une conversation entre un étudiant et un enseignant : - « Vous pouvez me conseiller un livre sur les équations différentielles ? »

- « Il y a le manuel de cours du professeur Carman sur l'analyse à la bibliothèque. » - « Merci » Dans une conversation aussi banale, l'étudiant a généralisé sa requête au concept de « livre », qui représente la catégorie la plus abstraite recouvrant toutes les informations de réponses acceptables pour lui. L'enseignant, sans même y prêter réellement attention, a utilisé sa taxonomie de concepts pour en déduire qu'un « manuel de cours » est un « livre », que les équations différentielles » font partie de « l'analyse » et que par conséquent sa réponse est pertinente. Le fait que la taxonomie soit partagée apparaît implicite puisque le professeur suppose que sa réponse sera comprise et qu'elle est effectivement. [63]

De plus la définition et l'utilisation d'une ontologie, sont une réponse pour résoudre les problèmes de variabilité linguistique en permettant de contextualiser la requête, c'est-à-dire en lui associant une sémantique précise et bien définie. [64]

4.2.2. Besoin de partage

Le problème de compréhension commune dans le e-learning survient sur plusieurs niveaux orthogonaux qui décrivent les différents aspects d'usage des documents :

- **Lors de la création du contenu**

La probabilité que deux auteurs de contenus expriment différemment le même concept est très élevée. En d'autres termes, chacun peut fournir le contenu mais en utilisant des mots-clés différents. Par exemple, le premier peut utiliser le mot « auteur » alors que le deuxième utilise le terme « créateur » pour référencer un acteur qui a fournit une ressource d'apprentissage.

- **Lors de l'accès et la recherche du contenu par un utilisateur**

Il existe un problème concernant les mots-clés à employer pour faire la recherche du matériel d'apprentissage. [65]

La construction d'une ontologie se fait par la voie d'un *consensus*, et représente ainsi la compréhension partagée a priori d'un groupe ou d'une communauté, au lieu, comme c'est le cas dans la plupart des systèmes, de reposer sur une signification donnée par quelques individus ou par une autorité, à laquelle tous doivent s'ajuster. [66]

De ce fait les fournisseurs de contenu d'apprentissage et les apprenants seront en quelque sorte sur la même longueur d'onde (un vocabulaire commun) et peuvent ainsi mieux partager le matériel

4.2.3. Besoin en réutilisation des objets pédagogiques

Devant le volume de plus en plus croissant des documents pédagogiques disponible sur le net, peu d'objets pédagogiques sont réutilisables. La recherche et la sélection des fragments de texte pertinents, des figures, des exercices, à partir d'un document dans l'objectif de leurs réutilisation dans un nouveau document est devenu presque impossible, de ce fait il est

nécessaire que les concepteurs des documents pédagogiques aient à leurs disposition un moyen d'accès rapide et flexible aux objets pédagogiques pertinents.

4.3. Exemples d'utilisation des ontologies

Nous décrivons dans cette section quelques exemples d'utilisation d'ontologies dans des systèmes de formation e-learning :

4.3.1. IMAT (Integrating Manuals And Training)

IMAT repose sur l'utilisation d'ontologies pour la conception de manuels de formation à partir de manuels constructeurs dans le domaine technique de la maintenance. Pour réutiliser les versions numériques des manuels techniques fournis par les constructeurs, ceux-ci sont découpés en un ensemble de fragments indexés selon des ontologies génériques, des ontologies domaines et autres pédagogiques. Les fragments indexés sont à la disposition des formateurs pour une recherche rapide et flexible basée sur les ontologies. [67].

4.3.2. QBLS (*Question Based Learning System*)

Pour l'étudiant, QBLS est un outil d'aide à la résolution de questions de Travaux Dirigés (TD) reliées aux connaissances clés du cours (*les notions, les thèmes, et le sujet de cours*), des accès aux fiches contenant les connaissances utiles sont conseillées pour répondre aux questions. QBLS est un moyen d'inciter les étudiants à aller chercher activement les connaissances. Dans cette optique, QBLS utilise une ontologie pédagogique selon laquelle le document de cours est vu comme un réseau de fiches *(définition, exemple, formalisation, précision) et de ressources abstraites (thème, notion, sujet de cours),* où chaque ressource abstraite réfère une ou plusieurs fiches [62].

4.3.3. VIUM (Projet de repérage et de visualisation du modèle de l'apprenant)

Dans le cadre de ce projet, un outil de repérage et de visualisation, *VIUM,* a été conçu pour permettre à l'utilisateur de sélectionner un concept central sur l'écran. Une ontologie est utilisée pour s'assurer que les concepts les plus proches sémantiquement sont visibles. Cette sélection de concepts rendus visibles est une partie essentielle de la visualisation qui assiste les apprenants dans l'exploration de domaines comprenant des centaines de concepts. La tâche particulière ici, est de montrer à un utilisateur ce qu'une ontologie computationnelle permet

d'inférer à partir d'informations comme l'évaluation de l'apprenant sur sa propre connaissance [62].

5. Conclusion

Dans ce chapitre nous avons présenté la notion du web sémantique, et celle d'ontologie, ainsi que langage et plate forme pour les ontologies, de plus nous avons montré l'apport des ontologies aux systèmes e-learning, et nous avons même fait allusion à quelques projets e-learning qui ont été réalisés à base des ontologies.

Le chapitre suivant sera réservé pour la modélisation de notre solution à l'aide du langage de modélisation UML.

Chapitre 4
Etude conceptuelle du système

Chapitre 4

Étude conceptuelle du système

1. Introduction

Dans l'introduction générale, nous avons relevé un certains nombre de questions, dont les réponses aideront à concevoir un système de déduction automatique des apprenants en obstacle basé sur l'analyse des comportements des apprenants, dans un environnement de formation sur le Web.

Ces questions sont dépendantes. Par exemple, pour répondre à la question relative à la détermination des apprenants en difficultés nous devons d'abord récupérer et restructurer les traces brutes issues de source de traçage.

En effet, notre motivation n'est pas de rajouter un système de détection automatique des apprenants en obstacles à base de traces, mais d'exploiter une nouvelle approche permettant de fournir aux enseignants des informations, ou des indicateurs, sur les difficultés rencontrées par des apprenants aux cours de session d'apprentissage. Ces informations servent à enrichir le modèle de l'apprenant et à fournir un service aux enseignants dans leurs tâches de tutorat et de suivi, et aux concepteurs des EIAH dans leurs tâches d'adaptation et de personnalisation des environnements de formation.

Pour cela, une étude conceptuelle du système avant son élaboration est nécessaire afin de mieux comprendre son fonctionnement, ainsi de bien maitriser sa complexité et d'assurer sa cohérence. Pour ce faire, le chapitre est subdivisé en deux sections distinctes.

La première section: on y trouve la présentation du projet, le schéma général de la solution et la description des différentes étapes pour la détermination des apprenants en difficultés.

La deuxième section: présente la conception, par la modélisation UML, de la solution mise en place.

2. Présentation du projet [41]

Nous nous situons dans le cadre d'une formation dispensée sur une plate-forme de formation à distance. L'étude des différentes interactions possibles de l'apprenant nous permet de classer ses activités selon leur intégration dans le processus de la formation en deux catégories : les activités élaborées au sein de la plate-forme de formation, et celles effectuées en dehors de la plate-forme, comme nous le constatons sur la figure suivante:

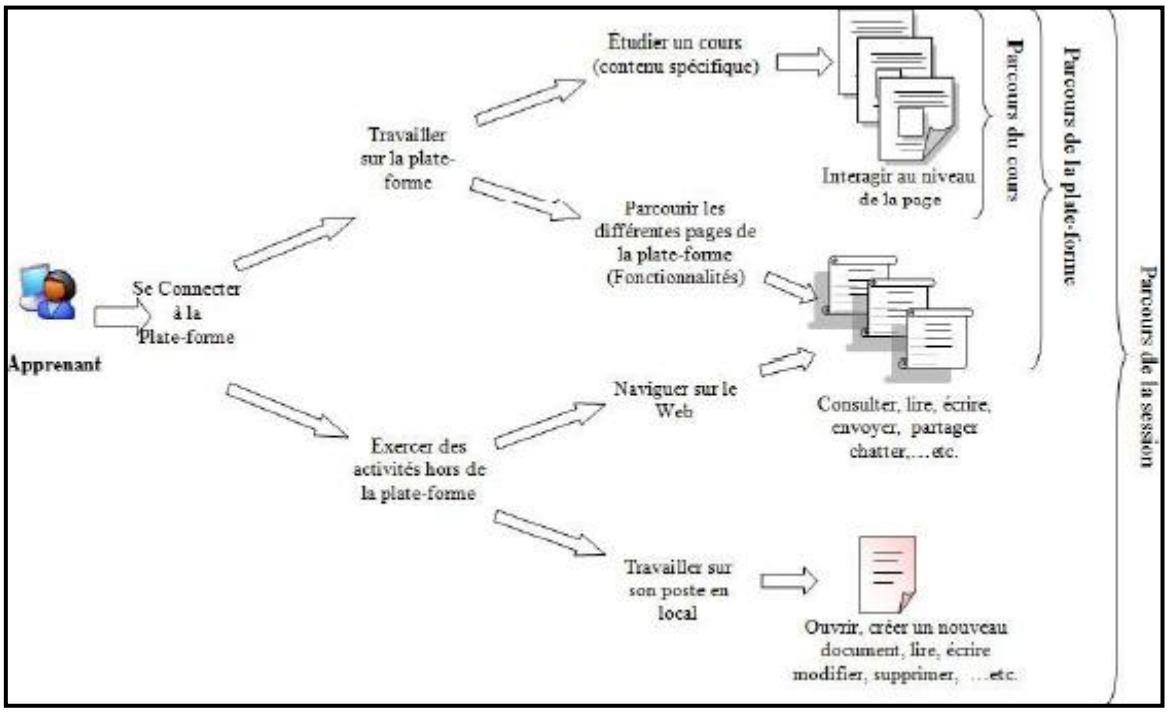


Figure-9- **:** Interactions de l'apprenant durant une session d'apprentissage [69]

Dans le cadre de notre travail nous allons s'intéresser aux activités élaborées en dehors de la plate-forme de formation, elles peuvent être parfois liées indirectement au processus de formation et permettent de renseigner sur les difficultés rencontrés par l'apprenant durant sa formation. Par conséquent, nous pouvons distinguer encore deux niveaux d'analyse du parcours de l'apprenant :

- *Parcours Web* : lors de la consultation de son cours à distance, l'apprenant peut naviguer sur le Web : ouvrir sa boite de messagerie, ouvrir d'autres pages, d'autres documents, faire de la recherche, consulter des sites de référence … etc.
- *Travail effectué en local* : l'apprenant peut aussi travailler sur sa machine en local ; ouvrir ou écrire dans un document Word, PDF… préparer des questions, préparer des travaux, prendre des notes lors de la consultation d'un contenu. Il peut aussi faire des commentaires ou

des remarques, etc. La trace de ce dernier type d'activités nécessite la collecte de données au niveau utilisateur, bien entendu avec l'accord de ce dernier.

Lors de la navigation de l'apprenant en dehors de la plate-forme de formation, que cela soit en local sur son poste, ou sur Internet, le parcours s'effectue dans un espace documentaire, dont le principal défaut est de n'être ni structuré, ni hiérarchisé. Pour cela, il est utile de déterminer l'ensemble des pages et des sites visités afin de déterminer les difficultés rencontrées par l'apprenant en calculant le lien sémantique entre les pages visitées et le cours ainsi les concepts non maitrisés durant une session d'apprentissage. Ceci se fait, par la collecte de toutes les traces enregistrables.

2.1. Collecte de traces

Nous pouvons identifier trois modes :

- *Une collecte manuelle* : procédée par un observateur humain, acteur ou non dans la situation d'apprentissage, ou par des questionnaires remis aux acteurs.
- *Collecte audiovisuelle* : exécutée par un outil d'enregistrement visuel et audio de la situation d'apprentissage.
- *Collecte numérique :* menée en utilisant un environnement informatique sauvegardant l'activité de l'apprenant. Les résultats de la collecte numérique constituent une trace numérique.

Environ 78 % des environnements de formation ouverte et à distance intègrent un outil de capture des traces numériques [70]. Ces systèmes utilisent différentes approches de collecte selon les sources d'observation : le serveur, le poste client, ou des mécanismes d'observation spécifiques au système d'apprentissage.

2.2. Les approches de collecte selon les sources d'observation

2.2.1. Approches centrées serveur

L'approche centrée serveur s'intéresse à la recherche des motifs de navigation des utilisateurs sur un site donné en se basant sur l'analyse des *logs* des serveurs Web. Ces *logs* contiennent l'ensemble des actions effectuées sur le serveur. La création des traces à partir de ces *logs* est

un processus complexe qui nécessite de nombreuses opérations (filtrage, recomposition en sessions, etc.

Dans notre contexte de formation en ligne, si la plate-forme d'apprentissage est hébergée sur un serveur apache, par exemple, les données, présentent dans le fichier de *log*, peuvent être utilisées pour : évaluer l'efficacité d'un cursus en ligne ; quantifier les interactions entre les utilisateurs et les pages du cours ou de l'environnement de formation, etc. Ces données peuvent ainsi répondre à certaines questions [71]: les pages *Web* du cours sont-elles adaptées au navigateur le plus utilisé par les apprenants ? Les apprenants accèdent-ils facilement aux pages essentielles du cours ? Une page spécifique est-elle réellement nécessaire à ce cours ? Le temps passé par les apprenants sur une page particulière est-il adapté pour atteindre l'objectif visé ? etc.

Ces approches sont intéressantes. Cependant, elles tiennent pour acquis que le contenu des pages est connu. En effet, les informations enregistrées dans les fichiers *log*, côté serveur, ne renseignent que les activités liées au serveur en question. Les données produites par les applications s'exécutant sur le poste client, ou celles issues d'applications produisant du contenu dynamique, ne sont pas collectées, empêchant ainsi de connaître les informations réellement consultées par les utilisateurs [72]. De plus, ces fichiers ne sont généralement pas disponibles à n'importe quel utilisateur (juste les administrateurs réseau et système).

Pour pallier aux inconvénients des approches fondées sur l'analyse des fichiers de *log côté serveur*, d'autres approches proposent des mécanismes d'observation côté utilisateur.

2.2.2. Approches centrées utilisateur

Si, pendant un exercice, l'apprenant effectue des recherches sur le Web, cette interaction n'est pas observée sur le serveur. Pourtant cette interaction peut être un élément important d'explication du parcours de l'apprenant. Il est donc intéressant d'instrumenter le poste client afin d'observer toutes les interactions propres à l'apprenant. Cette démarche n'est pas largement utilisée. En effet, si le recueil de données de trafic centrées-serveur est maintenant relativement standardisé, la collecte d'informations au niveau des postes utilisateurs est encore un domaine d'activité, du fait des différents choix technologiques possibles et du degré de finesse des informations que l'on souhaite recueillir [73].

Parmi les travaux adoptant cette approche, nous pouvons citer *WebIC* [74], qui est un système de recommandation de liens, focalisé sur la recherche d'information. Son objectif est de proposer à l'utilisateur les liens qui semblent le mieux convenir à ses attentes, en fonction des recherches qu'il a effectuées dans des moteurs de recherche et des liens qu'il suit.

Dans le domaine des EIAH, [75] propose un agent *keylogger(ex: MiniKeyLog)* capable d'enregistrer tous les évènements clavier et souris produits par l'apprenant, les processus qui sont exécutés ainsi que les titres et le contenu des boites de dialogue affichées sur l'écran. Néanmoins, très peu de travaux en EIAH implémentent cette approche. Généralement, ils tracent uniquement les actions des apprenants dans l'environnement de formation en utilisant les *logs* serveur ou des outils spécifiques intégrés à l'environnement tracé.

2.2.3. Approches basées sur des logiciels spécifiques

Au-delà des approches qui distinguent la source d'observation, selon que cela soit fait sur le poste client ou sur le serveur, d'autres approches se focalisent sur l'identification de l'interaction au moment de la collecte à travers un outil spécifique à l'environnement tracé. Parmi ces travaux, nous pouvons citer eMédiathèque[8], un outil de travail collaboratif où toutes les interactions de l'utilisateur sont observées et traitées à partir de deux modèles [76]: l'un définit les objets observables (les outils mis à disposition tels que le navigateur Internet, la messagerie instantanée, ou les ressources telles que les textes, les images, etc.) ; et l'autre spécifie les actions réalisables par l'utilisateur (création, modification, suppression de contenus, etc.). Les traces sont ensuite créées en fonction des activités des utilisateurs, stockées dans un composant interne de l'application, et affichées en temps réel à l'utilisateur.

3. Le schéma général de l'application

Pour atteindre notre objectif qui est la détermination des apprenants en difficultés, nous proposons l'architecture présentée dans la figure -10-. Dans ce système, Sur le poste client des apprenants de la plateforme moodle, un collecteur de traces (MinKeyLog) est installé. Une application de collecte de traces est activée au début de la session par l'apprenant. Elle enregistre toutes les interactions réalisées par un apprenant dans un fichier de *log* au format texte. À la fin de la session, ce fichier est traité par le module "Traitement des fichiers logs" ce qui génère un autre fichier de format XML qui va être sauvegardé dans une base de données. Le fichier XML généré est traité afin de créer le corpus des pages html et fichiers

[8] http://www.elycee.com

PDF consultés par l'apprenant durant la session d'apprentissage, ensuite le corpus va être sauvegardé dans la base de données.

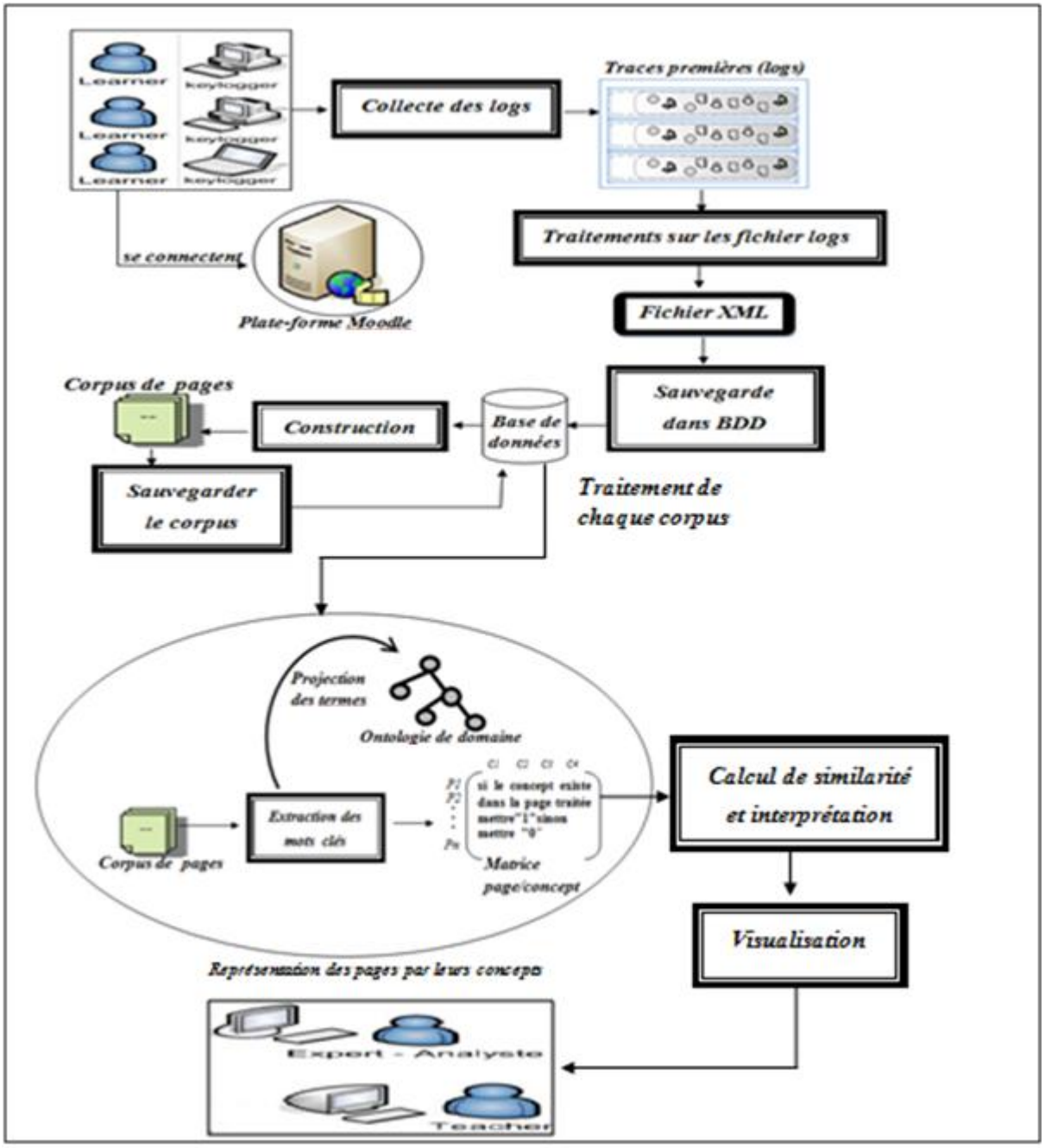


Figure -10- **:** Architecture générale du système

Le corpus est soumis à un processus de traitement afin d'avoir la matrice page/concepts qui est utilisée pour le calcul de la similarité sémantique de visite. Les mesures permettent de déduire l'ensemble des apprenants en obstacle. Ces résultats sont visualisés par l'enseignant ou le tuteur.

Ces différentes étapes vont être détaillées ci-dessous.

4. Description des composants

Afin de déduire les apprenants en obstacles à base de traces nous considérons les sous étapes suivantes :

4.1. Collecte de traces

Suite à la comparaison entre les trois approches de collecte dont nous avons parlé avant et vu la nécessité de collecter les traces des interactions des apprenants en dehors de la plate-forme de formation pour la détermination de l'ensemble des apprenants en difficultés, nous avons opté pour l'approche centrée utilisateur à travers un programme installé sur le poste de l'apprenant. Toutefois, nous nous intéressons principalement aux données de navigation Web. L'outil MiniKeyLog[9], permet de collecter les traces de navigation et offre également une fenêtre de configuration pour interdire la collecte de traces concernant certains actions ainsi que les mots de passes, pour des raisons de confidentialité et de protection de la vie privée des apprenants comme nous le constatons sur la figure -11-.

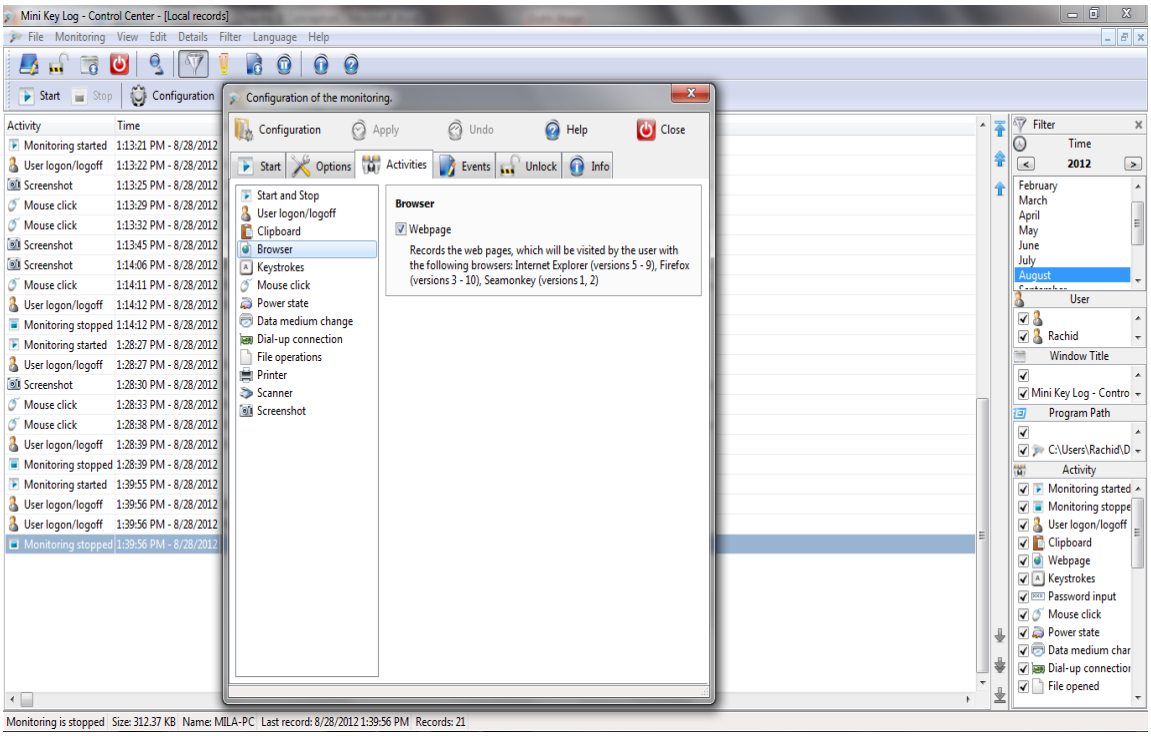
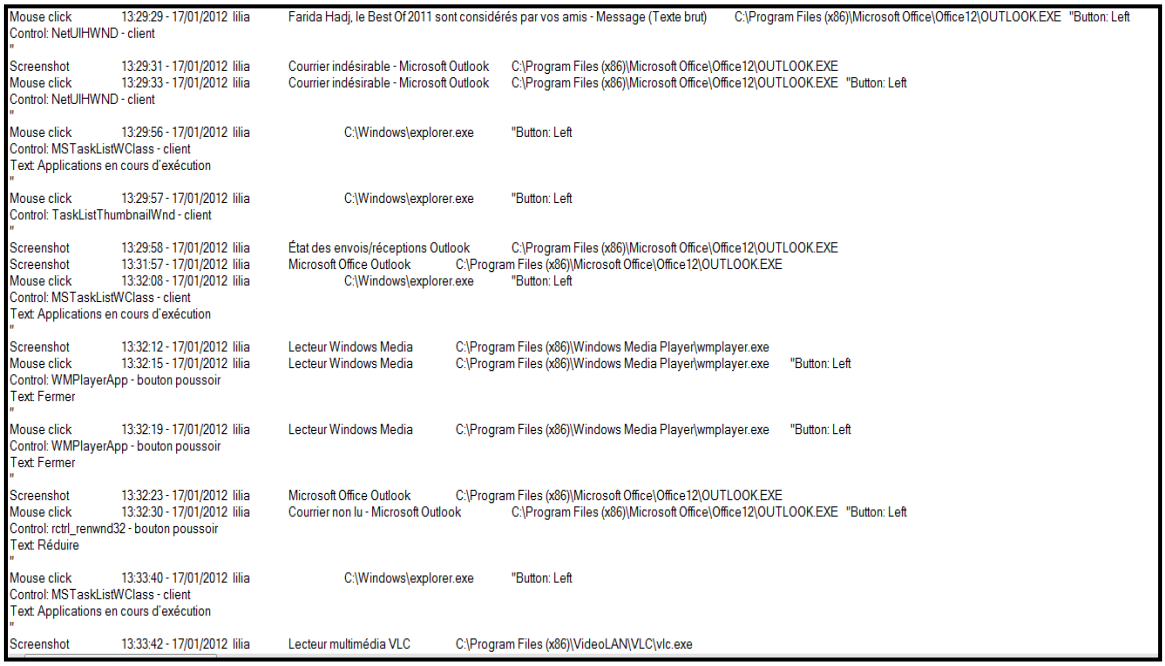


Figure - 11- **:** *La fenêtre de configuration du MiniKeyLog*

[9] http://www.blue-series.de/

Les traces collectées sont donc enregistrées au format TXT pour chaque apprenant (*user*) entre les moments de début et de fin de la session d'apprentissage (*time_start*, *time_end*). La figure -12- présente un extrait d'un fichier trace généré par l'outil.



```
Mouse click 13:29:29 - 17/01/2012 lilia Farida Hadj, le Best Of 2011 sont considérés par vos amis - Message (Texte brut) C:\Program Files (x86)\Microsoft Office\Office12\OUTLOOK.EXE "Button: Left
Control: NetUIHWND - client
"
Screenshot 13:29:31 - 17/01/2012 lilia Courrier indésirable - Microsoft Outlook C:\Program Files (x86)\Microsoft Office\Office12\OUTLOOK.EXE
Mouse click 13:29:33 - 17/01/2012 lilia Courrier indésirable - Microsoft Outlook C:\Program Files (x86)\Microsoft Office\Office12\OUTLOOK.EXE "Button: Left
Control: NetUIHWND - client
"
Mouse click 13:29:56 - 17/01/2012 lilia C:\Windows\explorer.exe "Button: Left
Control: MSTaskListWClass - client
Text: Applications en cours d'exécution
"
Mouse click 13:29:57 - 17/01/2012 lilia C:\Windows\explorer.exe "Button: Left
Control: TaskListThumbnailWnd - client
"
Screenshot 13:29:58 - 17/01/2012 lilia État des envois/réceptions Outlook C:\Program Files (x86)\Microsoft Office\Office12\OUTLOOK.EXE
Screenshot 13:31:57 - 17/01/2012 lilia Microsoft Office Outlook C:\Program Files (x86)\Microsoft Office\Office12\OUTLOOK.EXE
Mouse click 13:32:08 - 17/01/2012 lilia C:\Windows\explorer.exe "Button: Left
Control: MSTaskListWClass - client
Text: Applications en cours d'exécution
"
Screenshot 13:32:12 - 17/01/2012 lilia Lecteur Windows Media C:\Program Files (x86)\Windows Media Player\wmplayer.exe
Mouse click 13:32:15 - 17/01/2012 lilia Lecteur Windows Media C:\Program Files (x86)\Windows Media Player\wmplayer.exe "Button: Left
Control: WMPlayerApp - bouton poussoir
Text: Fermer
"
Mouse click 13:32:19 - 17/01/2012 lilia Lecteur Windows Media C:\Program Files (x86)\Windows Media Player\wmplayer.exe "Button: Left
Control: WMPlayerApp - bouton poussoir
Text: Fermer
"
Screenshot 13:32:23 - 17/01/2012 lilia Microsoft Office Outlook C:\Program Files (x86)\Microsoft Office\Office12\OUTLOOK.EXE
Mouse click 13:32:30 - 17/01/2012 lilia Courrier non lu - Microsoft Outlook C:\Program Files (x86)\Microsoft Office\Office12\OUTLOOK.EXE "Button: Left
Control: rctrl_renwnd32 - bouton poussoir
Text: Réduire
"
Mouse click 13:33:40 - 17/01/2012 lilia C:\Windows\explorer.exe "Button: Left
Control: MSTaskListWClass - client
Text: Applications en cours d'exécution
"
Screenshot 13:33:42 - 17/01/2012 lilia Lecteur multimédia VLC C:\Program Files (x86)\VideoLAN\VLC\vlc.exe
```

***Figure -12-* :** extrait du fichier log

Dans ce cas, les traces générées selon le modèle de cet outil, doivent être transformées vers notre modèle de trace pour pouvoir être traitées. De plus, elles doivent subir un processus de prétraitement afin d'éliminer le bruit. Cette tâche est réalisée grâce au module de traitement des fichiers logs.

4.2. Traitements des fichiers logs:

La phase de collecte produit généralement une très grande masse de données. Afin de raffiner l'analyse, il est possible de réduire cette quantité de données, en considérant seulement les données nécessaire pour le calcul de la similarité sémantique de visite. Comme la structure de fichier log est variable car elle dépond de la source de traçage proposée pour la collecte, nous proposons de structurer les données essentielles dans un fichier XML pour avoir une structure fixe. La structure des traces est présentée dans la figure suivante :

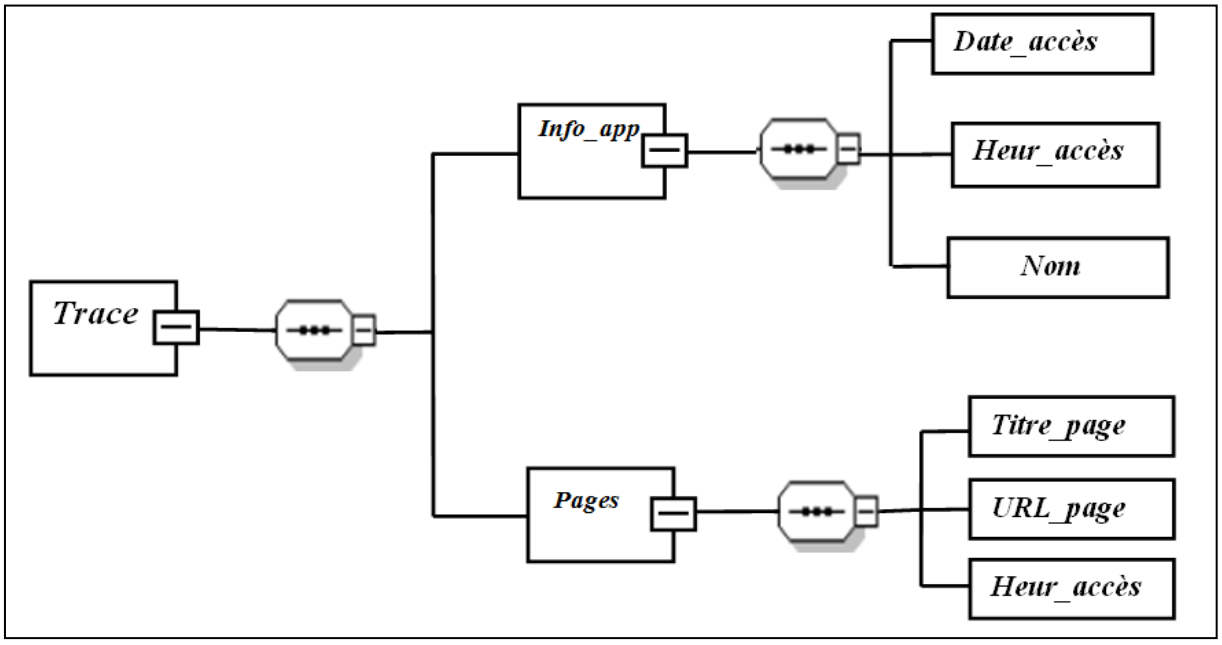


***Figure -13-* :** *Modèle de trace*

La racine de modèle est la trace, nous avons choisi d'encapsuler tous les autres composants décrits par le modèle dans cet élément.

La trace est associé a un apprenant qui est observé durant sa session d'apprentissage et elle est composé de deux éléments, info_apprenant et pages, le premier possède un nom de l'apprenant et l'heur d'accès a la formation, et le deuxième possède l'ensemble des pages visitées durant la session. Toute page est représentée par l'heur d'accès à la page, le titre et l'dresse complète de la page.

Vue la complexité du processus d'indexation, nous allons se limiter aux pages HTML et aux fichiers PDF.

Voici un exemple d'un fichier XML suivant le modèle que nous avons proposé:

```
<?xml version="1.0" encoding="UTF-8"?>
- <users>
  - <internaute_info>
      <date_accés>2012-05-24</date_accés>
      <heure_accés>16:09</heure_accés>
      <nom>Djamila</nom>
    </internaute_info>
  - <pages>
    - <page>
        <lien_consulter>http://www.clubic.com/lancer-le-telechargement-213524-0-mini-key-log.html</lien_consulter>
        <heure>16:16</heure>
        <titre_page>Mini Key Log 2.9 gratuit à télécharger/Download (2003, XP, 2000, Me, 98, 95, NT) sur Clubic !</titre_page>
      </page>
    - <page>
        <lien_consulter>http://forum.hardware.fr/hfr/Programmation/Java/resolu-afficher-jeditorpane-sujet_66287_1.htm</lien_consulter>
        <heure>16:22</heure>
        <titre_page>Afficher une page html avec CSS dans un JEditorPane [JAVA] [Résolu]</titre_page>
      </page>
    - <page>
        <lien_consulter>http://forum.hardware.fr/hfr/Programmation/Java/resolu-afficher-jeditorpane-sujet_66287_1.htm</lien_consulter>
        <heure>16:27</heure>
        <titre_page>Afficher une page html avec CSS dans un JEditorPane [JAVA] [Résolu]</titre_page>
      </page>
    - <page>
        <lien_consulter>http://www.mlab.im.dendai.ac.jp/~yamada/ir/HTMLParser/SimpleParser.html</lien_consulter>
        <heure>16:30</heure>
        <titre_page>SimpleParser</titre_page>
      </page>
    </pages>
</users>
```

Figure -14- **:** Exemple d'un fichier XML

4.3. Création d'une base de trace

Pour pouvoir manipuler plusieurs fichiers xml générés après traitement des fichiers logs, nous devons sauvegarder leurs chemins dans une base de trace que nous allons présenter la base en détail ci-dessous.

4.3.1. Dictionnaire de données

Le tableau suivant représente un descriptif du codage des informations sauvegardées dans la base de données :

Code	Désignation	Type	Taille
Apprenant_i	Représente le N de l'apprenant	Varchar	30
Ident	Identifiant de la session	Varchar	32
Nom_app	Le nom de l'apprenant	Varchar	50
Chemin_xml	Le chemin de fichier xml	Varchar	50
Chemin_corpus	Le chemin de corpus des pages visitées	Varchar	50

Tableau-1- **:** Le tableau descriptif du codage des informations

4.3.2. Le modèle physique des données :

Ce modèle nous donne la représentation physique de table de la base de données du système étudié :

Table *Fichiers_xml*:

Nom du champ	Null
Apprenant_i	Not null
Ident	Not null
Nom_app	Not null
Chemin_xml	Not null
Chemin_corpus	Null

***Tableau -2-* :** *La table Fichiers_xml dans la base du système*

4.4. Construction des corpus

Cette étape consiste à construire le corpus des pages visitées durant la session d'apprentissage pour chaque apprenant, afin d'avoir le code des pages html en local, Pour cela, nous considérons trois sous étapes:

- ✓ soumettre les fichiers xml sauvegardés dans la base à un traitement afin d'extraire les urls des pages visitées,
- ✓ chaque page est consulté pour une durée considérable, si cette durée est supérieure à la durée moyenne de consultation des pages, la page va être soumise au module de l'aspiration des pages,
- ✓ mise à jour de la base de trace par l'ajout des chemins des corpus construits dans le champ chemin_corpus (initialement ce champ est nul, voir tableau -2-).

4.5. Représentation des pages par leurs concepts

Nous allons indexer les pages de chaque corpus en se basant sur l'approche métadonnées et non pas sur l'approche automatique car cette approche en elle-même est un thème de recherche.

L'approche métadonnées il s'agit d'exploiter les pages html aspirées pour extraire les mots clés(les métadonnées) qui se trouvent dans la balise méta "*keywods*" du chaque page aspirée comme la montre la figure suivante, et à propos des métadonnées des fichiers pdf aspirés nous

allons intégrer un module d'extraction, et cela se fait seulement pour les fichiers qui possèdent des mots clés.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- saved from url=(0045)http://axiomcafe.fr/les-structures-de-
donnees -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
"http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd"><HTML
xmlns="http://www.w3.org/1999/xhtml" xmlns:og =
"http://opengraphprotocol.org/schema/"><HEAD>
<META content="text/html; charset=utf-8" http-equiv="Content-
Type"><TITLE>Les
structures de données</TITLE><LINK rel="prev" href="/tutoriel-
documenter-un-code-avec-doxygen">
<LINK rel="up" href="/section-informatique"><LINK rel="next"
href="/les-piles">
<LINK rel="shortcut icon" type="image/x-icon"
href="/sites/axiomcafe.fr/files/favicon.png">
<LINK rel="schema.dc" h                    l.org/dc/elements/1.1/">
<LINK rel="canonical"
href="http://axiomcafe.f                  structures-de-donnees">
<META name="keywords"
content="vulgarisation,science,scientifique,explication,decouvert
e,diffusion,connaissance,mediation,savoir,structure,donnee,algori
thme,programmation,Informatique">
<META name="copyright" content="AxiomCafe est mis à disposition
selon les termes de la licence Creative Commons Attribution - Pas
d'Utilisation Commerciale - Pas de Modification 3.0 France.">
```

Les métadonnées

Figure -15- **:** Extrait de code html

Pour effectuer la projection nous avons opté pour une ontologie de domaine de type "skos" (cf. annexe1) qui a été crée par l'éditeur "protégé" (*cf. annexe2*) et qui permet de représenter des termes et des concepts multilingues, dans notre cas nous avons choisit français et anglais comme langues, ainsi illustrer dans la figure -16-.

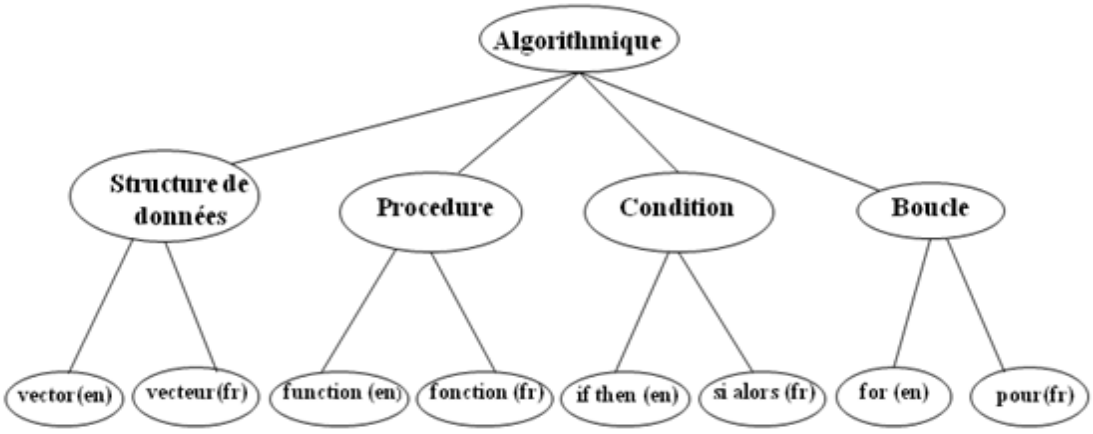


Figure -16- : Extrait d'ontologie du cours "algorithmique" de type skos

La phase de projection consiste à marquer l'ensemble des concepts qui correspondent aux termes de chaque page aspirée, c'est-à-dire que les termes de ces pages vont être comparés aux altLabels (alternative labels) de l'ontologie et dans le cas où les termes existent, les

concepts correspondants vont avoir la valeur "1" dans la matrice où les lignes représentent l'ensemble des pages constituant le corpus et les colonnes sont les concepts de cours.

La figure ci-dessous montre ce processus de projection :

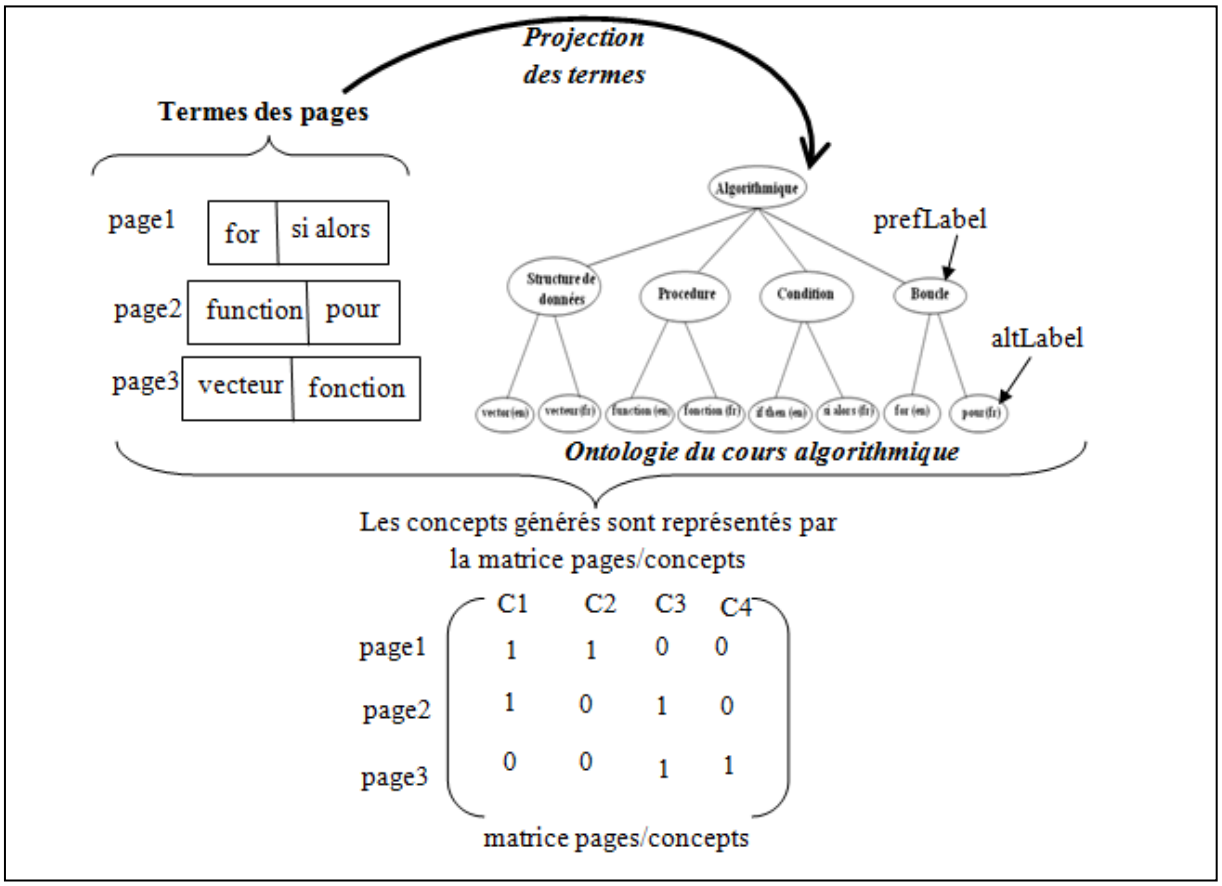


Figure 17 : Indexation des pages par l'approche métadonnées

4.6. Calcul de la mesure sémantique et interprétation

Pour le calcul de la mesure en appliquant la formule de Jaccard *(cf. chapitre2)* qui est une approche basée sur les vecteurs.

Nous considérons que le cours est un vecteur qu'on initialise à 1 et qui représente l'ensemble de concepts de l'ontologie crée, et les pages consultées est un vecteur comprend les valeurs 0 et 1.

Ce vecteur de pages est construit, en deux sous étapes: la première étape consiste à effectuer la somme entre les valeurs de chaque colonne de la matrice générée dans l'étape précédente, pour avoir un vecteur contenant les fréquences des concepts des pages qui appartiennent à l'intervalle [0 - nombre de page constituant le corpus] et la deuxième étape permet d'écraser

les valeurs du vecteur de fréquences qu'ils sont supérieurs à zéro en mettant la valeur 1 pour chaque case traitée afin d'avoir le vecteur de pages contenant que des valeurs binaires (0 et 1).
Par la suite nous allons appliquer la formule de Jaccard entre le vecteur de pages et le vecteur de cours pour avoir la mesure qui représente le degré de similarité entre le cours et les pages consultées en dehors de la plateforme durant la session d'apprentissage.
Le même traitement sera effectué pour chaque matrice générée.
Suite à l'étape de calcul, les valeurs des vecteurs de fréquences et les mesures de similarité sémantique calculées vont être comparé à des seuils afin de déterminer les concepts non maitrisés pour chaque apprenant et l'ensemble des apprenants en difficultés. La détermination des seuils nécessite l'utilisation d'un questionnaire pour l'ensemble des apprenants, pour cette raison que nous préférons les laissés en paramètre.

4.7. Visualisation

La visualisation est déclenchée par l'enseignant ou le tuteur afin de détecter les difficultés d'apprentissage en vue d'une amélioration et une préalable adaptation. Dans notre cas nous allons choisi la valeur numérique pour la représentation des résultats.

Dans ce qui suit nous élaborons les différents diagrammes qui correspondent aux différents vues de notre système.

5. Identification des besoins

Vu le besoin de réaliser notre travail dont nous venons d'exposer ses principales fonctionnalités, nous avons fait une étude conceptuelle, qui consiste à faire une simulation permettant de détecter les difficultés d'apprentissage.

L'objectif final de traitement des traces collectées est :

La détermination des concepts non maitrisés durant la session d'apprentissage par les apprenants, et la détection l'ensemble des apprenants en difficultés

5.1. Identifier les acteurs

Dans le langage de modélisation UML« Unified Modeling Language », le concept acteur représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositif matériel) qui interagissent avec le système étudié.

En analysant le contexte du problème, nous avons conclu que les acteurs qui interagissent sont les suivants :

L'enseignant, le tuteur et l'apprenant: sont des entités externes au système en interaction avec ce dernier.

5.2. Les messages

Un message représente la spécification d'une communication entre objets qui transporte de l'information avec l'intention de déclencher une activité chez le récepteur. La réception d'un message est considérée comme un événement. Cette notion de message est applicable pour décrire les interactions de plus haut niveau entre les acteurs et le système.

5.3. Identification de cas d'utilisation

Le tableau ci-dessous établit le résultat de ce travail :

Cas d'utilisation	**Acteurs principaux**	**Message (émis, reçue)**
	Enseignant/tuteur	Emet : obtenir les différents indicateurs calculés Reçoit : indicateurs affichés
	Apprenant	Emet : obtenir les traces des apprenants Reçoit: traces collectées

***Tableau-3-* :** tableau descriptif de cas d'utilisation

5.4. Diagramme et description de cas d'utilisation

Titre : **Les mesures et leurs interprétations**
Auteurs principaux : l'enseignant, le tuteur
Intention : Détection des difficultés rencontrées par les apprenants durant la session d'apprentissage
Enchainements nominaux : Enchainement: afficher l'ensemble des indicateurs calculés
Besoin d'IHM : Interfaces pour afficher les résultats obtenus

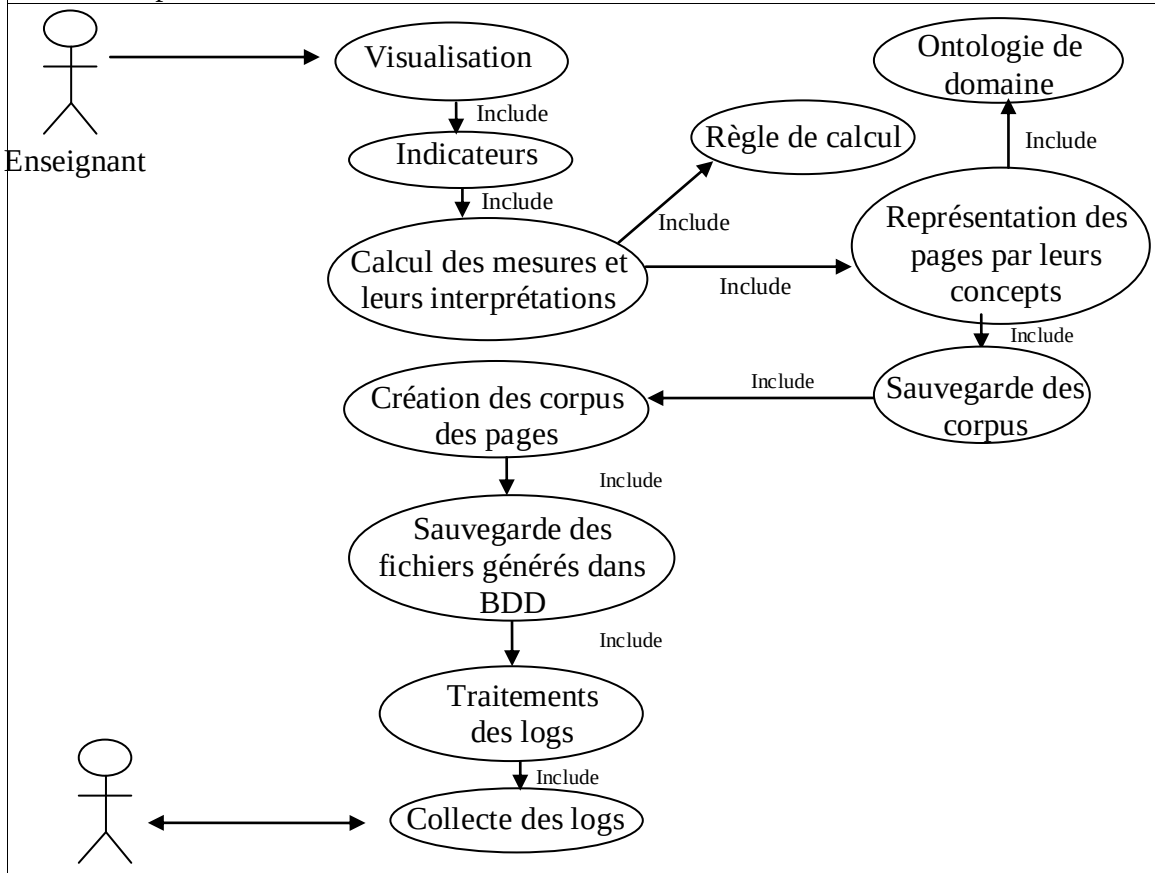


Figure -18- **:** Diagramme principal de cas d'utilisation

Le rôle de l'apprenant est passive qui consiste en l'objet de la collecte des traces d'interactions. L'enseignant et le tuteur peuvent visualiser tout les indicateurs calculés suite à un processus de traitement.

5.5. Diagramme de séquence système

Vu le besoin de compléter et de bien éclairer la description de cas d'utilisation précédent, nous allons ajouter le diagramme de séquence pour le cas d'utilisation, afin d'illustrer les différents scenarios nominaux.

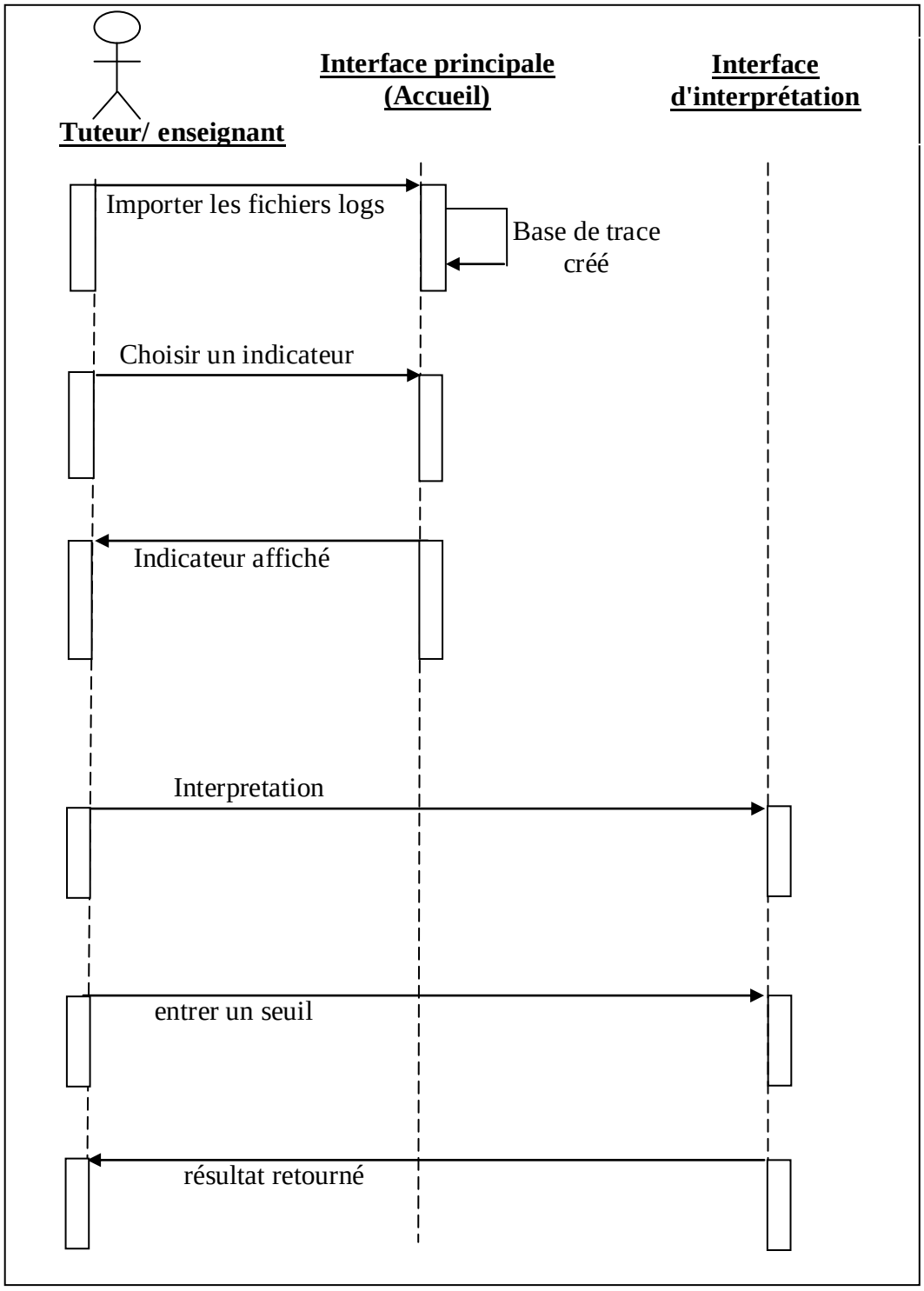


***Figure -19-*:** Diagramme de séquence pour le cas d'utilisation

Cette figure représente le scénario d'exécution du système. Le premier scenario est l'importation des logs afin de créer une base de trace. Par la suite nous pouvons avoir les différents indicateurs et ces derniers s'affichent de manière synchrone. Le dernier scénario est l'interprétation des mesures calculées précédemment, l'enseignant ou le tuteur doit faire entrer les seuils afin de déterminer les concepts non maitrisés ainsi l'ensemble des apprenants en obstacle.

5.6. Elaboration du diagramme de classes :

Afin de compléter les différentes étapes et diagrammes précédents, nous allons rajouter le diagramme de classe qui représente l'aspect logique de notre système.

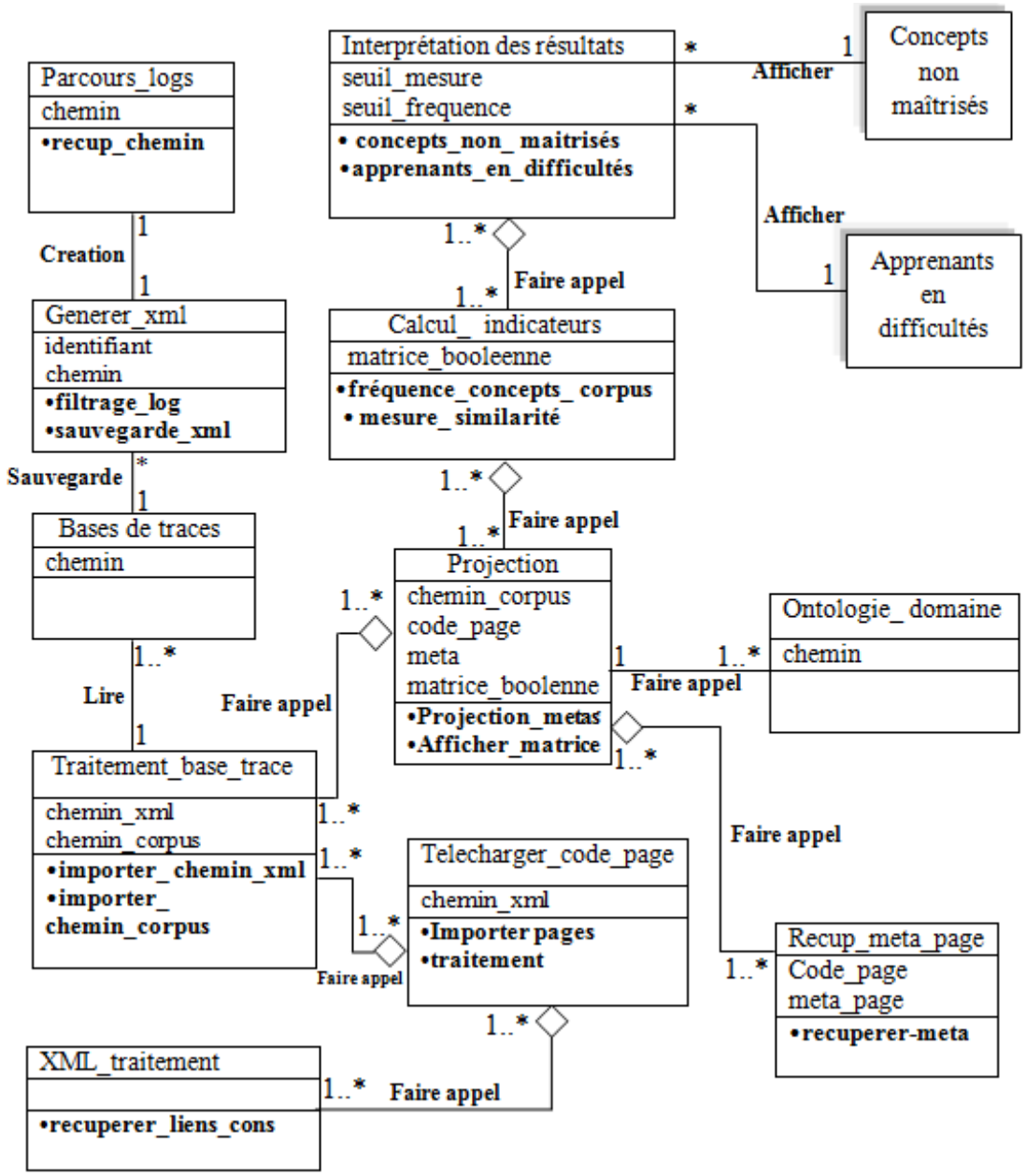


Figure -20- **:** Diagramme de classes

La classe "parcours log" permet de récupérer les chemins des logs à filtrer puis la classe "Generer_xml" effectue le traitement sur les fichiers sélectionnés et crée des fichiers XML suivant le modèle proposé, ensuite ces derniers seront sauvegardés dans la base de trace qui a le rôle de la classe "Base de trace". La classe "Telecharger_code_page" interroge la base de trace pour récupérer les chemins des différents fichiers_xml et des corpus sauvegardés par la classe "Traitement_base_trace", par la suite elle fait appel à la classe XML_traitement pour récupérer les liens (URL) des pages consultées de chaque corpus, dans le but de créer le corpus des pages. Le corpus est soumis à un processus de traitement par la classe "Projection"

afin de créer la matrice pages/concepts en faisant appel à la classe "Recup_méta_page" pour récupérer les métadonnées de chaque page et à la classe "Ontologie_domaine" qui représente l'ontologie de domaine de type "SKOS" afin d'avoir les concepts qui correspondent aux termes des pages du corpus traité. La classe "Calcul_indicateurs" permet d'effectuer un traitement sur la matrice générée précédemment pour avoir les fréquences des concepts du corpus et les mesures de similarités, par la suite ces indicateurs seront interprétés par la classe "Interprétation des résultats" afin de déduire les concepts non maitrisés par chaque apprenant ainsi que l'ensemble des apprenants en difficultés (ceux qui ont sollicité ses concepts).

6. Conclusion

A travers ce chapitre, nous avons présenté notre démarche d'analyse des comportements des apprenants pour l'identification automatique des apprenants en obstacle. Nous avons également présenté l'étude conceptuelle de notre application pour mener bien nos expérimentations.

La prochaine étape sera donc d'implémenter la solution, que nous présenterons dans le chapitre suivant.

Chapitre 5
La réalisation

Chapitre 5

La réalisation

1. Introduction

Il est bien évident qu'une conception bien faite facilite beaucoup la réalisation du projet en tenant compte de ce qu'on a modélisé dans les étapes précédentes. Nous allons décrire durant ce chapitre le fonctionnement de notre application ainsi que quelques explications nécessaires pour comprendre son fonctionnement à travers les différentes interfaces qui la composent.

2. Description de l'environnement de développement

2.1. Aspect matériel

Pour la réalisation de notre projet, nous avons utilisé une machine qui possède les caractéristiques suivantes :

- *Microprocesseur AMD Sempon(tm) SI-42 2.10 GHZ*
- *3.00 GO de RAM (2.75 Go utilizable)*
- *116 GO de capacité du disque*

2.2. Aspect logiciel

2.2.1. Langage et outils de programmation

2.2.1.1. Langages

Langage java:

Le choix du langage adéquat est une étape très délicate dans le développement des logiciels, pour le développement de notre logiciel (ASTAD), nous avons choisi le langage de programmation Java[10] qui est un langage très puissant pour le développement de très nombreuses applications surtout dans le domaine des réseaux et internet.

Langage XML (cf. Annexe3)

Le langage SQL :

SQL est un langage de gestion de bases de données relationnelles. Il permet :

- L'interrogation de bases de données.
- La manipulation des données (mise à jour, suppressions…).
- La définition des données (création de bases de données et de tables relationnelles).

[10] www.java.sun.com

2.2.1.2. Les Outils

Eclipse

Eclipse[11] est un environnement de développement intégré libre extensible, universel et polyvalent, permettant de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. **E**clipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions.

MySQL :

MySQL[12] est un système de gestion de base de données (SGBD), distribué sous une double licence GPL et propriétaire. MySQL supporte le langage de requête SQL et fonctionne sur de nombreux systèmes d'exploitation.

Apache :

Apache est un logiciel de serveur HTTP distribué sous licence libre, nommée licence Apache. Il est l'un des logiciels les plus populaires du Web.

phpMyAdmin :

phpMyAdmin[13] est un outil d'administration de serveurs MySQL. Il est sous licence open source, et possède principalement les fonctionnalités suivantes:

- Création, modification et suppression de bases de données et de tables.
- Gestion des utilisateurs et de leurs privilèges.
- Exécution de requêtes SQL.

WampServer :

WampServer[14] est une plateforme de développement Web, distribuée sous licence open source. Elle permet de faire fonctionner localement (sans se connecter à un serveur externe) des applications web. WampServer intègre les serveurs Apache, et MySQL, le langage PHP, ainsi que l'outil phpMyAdmin.

Nous avons utilisé aussi protégé (cf. Annexe2) et MiniKeyLog.

---

[11] www.eclipse.org

[12] www.mysql.com

[13] www.phpmyadmin.net

[14] www.wampserver.com

2.2.2. Système d'exploitation

En utilisant le langage java qui offre une compatibilité totale d'exécution sur différents systèmes d'exploitation, notre produit fonctionne sur n'importe quelle plateforme. Ce logiciel est testé sur les systèmes d'exploitation suivants : Windows XP et windows7.

4. Présentation du logiciel ASTAD (Analyseur Sémantique de Traces pour la détection des Apprenants en Difficultés)

Dans ce qui suit nous allons décrire brièvement l'interface du logiciel et les différentes fenêtres d'utilisation :

4.1. Interface de démarrage

Dés que vous démarrez le logiciel, la fenêtre de l'interface de démarrage s'affiche comme illustrer sur la figure suivante :

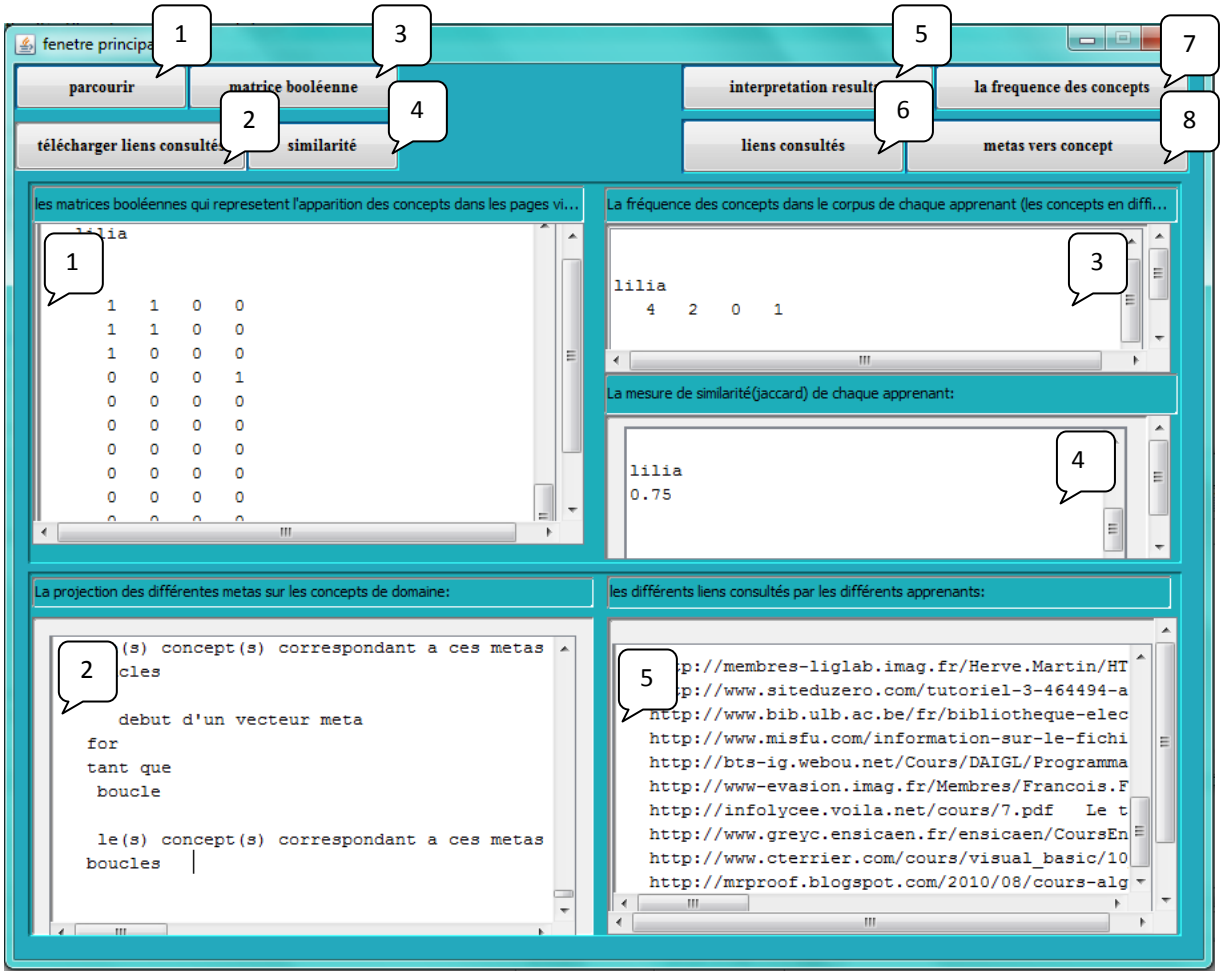


Figure-21- **:** Fenêtre de démarrage

4.1.1. Description de la fenêtre de démarrage

Elle est composée de huit boutons :

- ✓ Un clic sur le bouton (1) permet d'afficher une boite de dialogue permettant de parcourir afin de récupérer les logs, et d'effectuer leurs filtrage et sauvegarder les résultats (fichiers xml) dans la base de données;
- ✓ Un clic sur le bouton (2) permet d'aspirer les pages consultées par les apprenants pendant sa session d'apprentissage, et affiche vers la fin une fenêtre d'information (Figure-24-), pour indiquer que le téléchargement est achevé;
- ✓ Le clic sur le bouton (3) permet d'afficher les matrices booléennes des pages/concepts où chaque matrice représente l'apparition/ou non des concepts du domaine dans les différentes pages visitées par les apprenants ;
- ✓ Le clic sur le bouton (4) permet d'afficher les mesures de similarité entre le cours et les pages visitées par tous les apprenant pendants la session d'apprentissage ;
- ✓ clic sur le bouton (5) permet de quitter la fenêtre du démarrage et d'afficher une autre fenêtre, qui est la fenêtre d'interprétation des résultats calculés (Figure --) ;
- ✓ clic sur le bouton (6) permet d'afficher tout les liens consultés (URL) par les apprenants pendant leur session d'apprentissage ;
- ✓ clic sur le bouton (7) permet d'afficher la fréquence d'apparition des différents concepts dans les pages visitées (dans le corpus) en dehors de la plate forme par les apprenants pendant la session d'apprentissage
- ✓ clic sur le bouton (8) permet l'affichage les différents mots clés des pages web visitées, ainsi que les concepts correspondant à ces mots clés (la projection des termes/concept).

Cette fenêtre de démarrage contient par conséquence 5 zones d'affichage :

- ✓ Zone (1) pour l'affichage des matrices booléennes ;
- ✓ Zone (2) pour l'affichage de la projection métas/concepts ;
- ✓ Zone (3) pour l'affichage de la fréquence des concepts ;
- ✓ Zone (4) pour l'affichage de la mesure de similarité ;
- ✓ Zone (5) et la dernière pour l'affichage des différents liens visités et l'heur d'accès à ces liens.

La fenêtre d'information qui apparait après avoir cliquer sur le bouton (2) pour indiquer la fin de téléchargement du code est présentée dans la figure qui suit :

Figure -22- **:** Fenêtre de la boite d'information 'fin téléchargement'

4.2. La boite de dialogue du choix d'un ou plusieurs fichiers logs

Cette fenêtre apparait lorsqu'on clic sur le bouton (1) de la figure (21), elle permet à l'utilisateur de sélectionner le(s) fichier(s) log(s) à analyser (à filtrer et sauvegarder le résultat dans la base de données), cette interface est indiqué par la figure suivante :

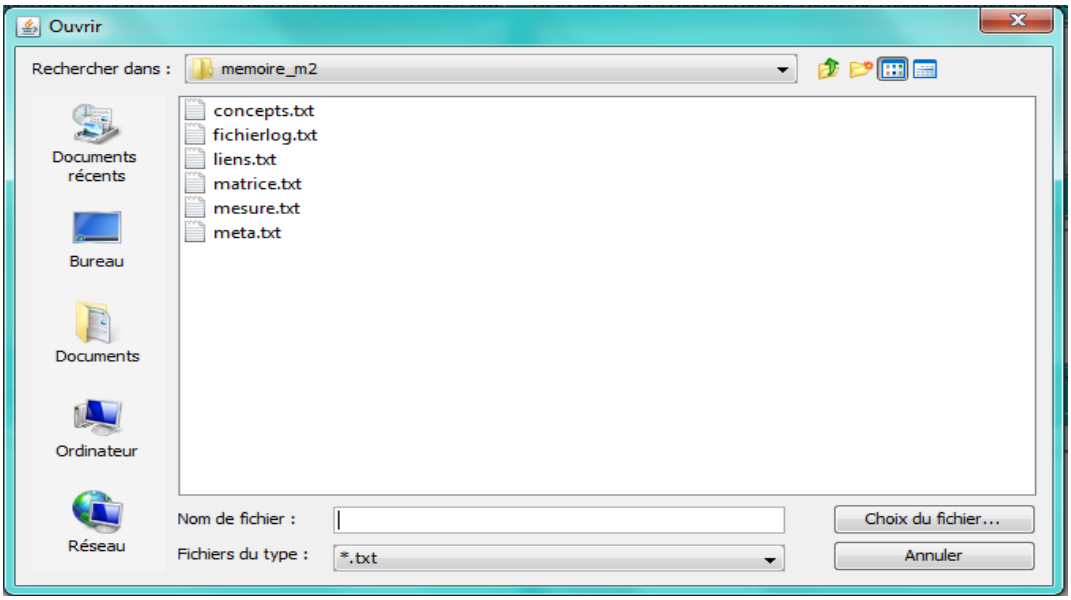


Figure -23- **:** Fenêtre de la boite de la sélection de (s) log (s)

Après la sélection de(s) fichier(s) log(s), une boite d'information apparait pour avertir l'utilisateur que le chargement et le traitement du fichier(s) log(s) s'est terminé(s) et le résultat est sauvegardé dans la base de données. La figure suivante représente cette boite d'information :

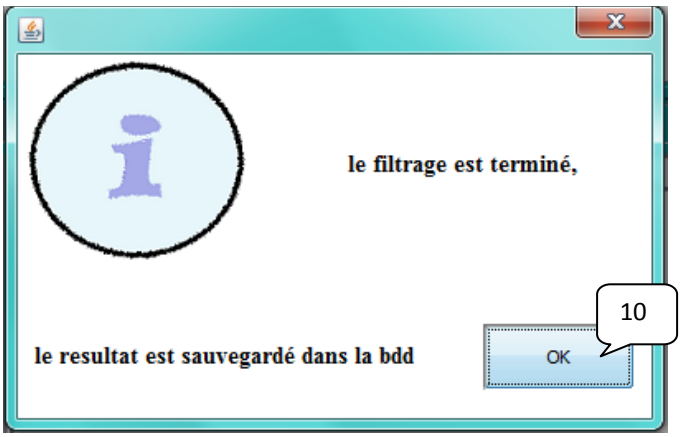


Figure -24- **:** Fenêtre de la boite d'information 'fin filtrage'

Un clic sur (10) et cette fenêtre va se disparaitre (elle se ferme).

4.3. Interface d'interprétation des résultats

Cette fenêtre apparait après un clic sur le bouton (5) toujours de l'interface de la figure (21), elle se présente comme exposée sur la figure ci dessous :

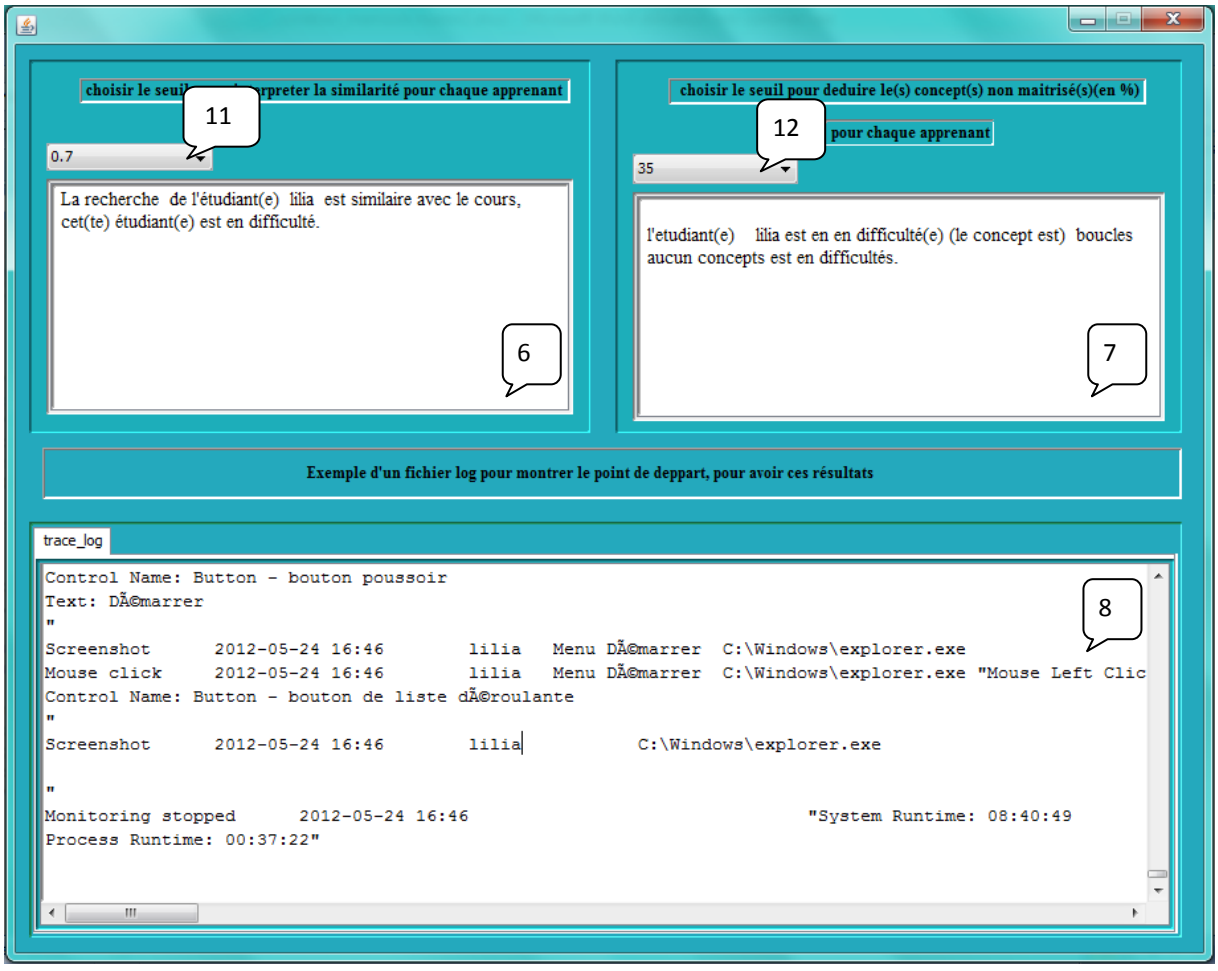


Figure -25- **:** Fenêtre d'interprétation des résultats

4.3.1. Description de la fenêtre d'interprétation des résultats

Cette fenêtre est composée de deux listes déroulantes (11) et (12) et de trois zones d'affichage (6), (7), et (8) :

- ✓ (11) permet la sélection du seuil de la similarité entre le cours et les pages visitées par l'apprenant en dehors de la plateforme, durant la session d'apprentissage ;
- ✓ (12) permet la sélection du seuil pour le nombre de concepts en difficultés à prendre en considération pour la prise de décision. En effet, se seuil sera comparé avec les fréquences des concepts calculées, et affiche les différents concepts qui ne sont pas ou peu maitrisés par les apprenants;
- ✓ (6) Zone d'affichage pour l'interprétation des mesures de similarité entre le cours et les différentes pages visitées par l'apprenant à l'extérieur de la plateforme pendant la session d'apprentissage par l'apprenant ;
- ✓ (7) Zone d'affichage des différents concepts non maitrisés ;
- ✓ (8) Zone d'affichage d'un exemplaire du fichier log, pour montrer la structure de notre matière première, sur laquelle nous avons basé pour avoir tout les résultats interprétable (la détection des difficultés d'apprentissage à travers le recueille et l'interprétation des traces d'apprentissage).

5. Conclusion

Dans ce chapitre nous avons présenté l'environnement du développement de l'application réalisée. L'objectif principal de cette application est de calculer la similarité entre le cours et les pages web visitées par l'apprenant pendant sa session d'apprentissage en dehors de la plateforme, et trouver les différents concepts dont il lui pose problème, en se basant sur un modèle de trace que nous avons par ailleurs proposé lors de la conception, le fruit de la trace d'interaction (fichier Log) entre l'apprenant et sa machine, collectés par l'outil Minkeylog.
Nous avons présenté aussi les différentes interfaces de l'application qui regroupent les fonctionnalités qu'offre cette dernières.

Conclusion générale

Conclusion générale et perspectives

Au terme de ce mémoire nous tentons de tirer des conclusions du travail présenté et de discuter les perspectives d'amélioration de nos propositions.

Au début de ce manuscrit, nous avons mentionné que notre principal objectif est de déterminer, concevoir et mettre en œuvre une approche pour la perception des comportements des apprenants durant une session d'apprentissage, basée sur l'analyse sémantique des traces et plus précisément sur le parcours Web, dans une perspective de détection automatique des apprenants en obstacles.

Pour cela, nous avons apporté des éléments de réponses aux différentes questions posées dans la problématique. Ces réponses résument nos principales contributions.

Nous avons perçu dès le départ, que la majorité des plates-formes de formation existantes ne fournissent pas toutes les informations nécessaires au suivi de l'apprentissage essentiellement lorsque le parcours est externe à la plateforme. Nous avons constaté le besoin d'adopter une collecte de traces de navigation basée sur l'approche centrée utilisateur (peu utilisée dans le domaine) pour avoir une perception réelle de toute l'activité de l'apprenant durant sa session d'apprentissage, même en dehors de la plate-forme de formation.

Nous avons proposé une méthode pour représenter les pages consultées en dehors de la plateforme d'apprentissage qui consiste à représenter chaque page par un vecteur de concepts qui ont été générés par la projection des termes (par lesquels les pages Web ont été annotées) sur l'ontologie du cours représentée par le format SKOS. Nous avons déterminé par la suite les concepts non maitrisés par l'apprenant ainsi que les liens sémantiques entre le contenu du cours (ontologie du domaine) et les pages visitées sur le web.

Par ailleurs, la validation des résultats nécessite l'utilisation d'un questionnaire pour comparer les résultats obtenus à ceux de questionnaire.

Concernant les perspectives, vue la complexité de ce domaine, nous pensons qu'il reste énormément de chose à faire et nous avons touché seulement qu'a une partie de son introduction. Cependant dans le contexte de notre problématique nous imaginons toujours d'autres améliorations, nous citons quelques une dans ce qui suit:

- Nous pensons qu'il est préférable d'utiliser une méthode d'indexation sémantique multilingue pour extraire les termes que représente chaque page.
- L'utilisation d'une ontologie riche qui comprend des contraintes de cardinalité, une taxonomie de relations, des axiomes/héritages sémantiques (logique de description, logique de propositions, logiques d'ordre plus élevé).
- Des expérimentations réelles pour confirmer les seuils indiqués sur l'application.

Annexes

Annexe1

SKOS (Simple Knowledge Organization System) [77]

1. Présentation

SKOS (Simple Knowledge Organization System ou Système simple d'organisation des connaissances) est un langage de représentation de schémas de concepts, qui recouvre les langages documentaires tels que les thésaurus, classifications, etc.

Son nom a été choisi pour mettre en évidence l'objectif même visé par ce langage : « proposer un système permettant d'exprimer et de gérer des modèles interprétables par les machines dans la perspective du web sémantique. » (Source : Michèle Lénart, la revue « Documentaliste » Volume 44, N° 1, paru le 28 février 2007)

Ce modèle est défini comme « simple » par opposition à d'autres modèles, comme OWL, plus à même de représenter des structures sémantiques plus riches telles que les ontologies, mais de ce fait également plus complexes à utiliser.

SKOS est depuis le 18 août 2009, une recommandation du *World Wide Web Consortium** (W3C).

2. Principes de représentation de SKOS

Le formalisme de représentation utilisé par SKOS repose sur les graphes RDF. Le concept constitue le centre du graphe auquel peuvent notamment être attachés en tant que propriétés RDF :

- les indications portant sur le concept lui-même :
  - ✓ des termes préférentiels ou alternatifs, les équivalents dans d'autres langues,
  - ✓ les termes cachés, très pratiques pour gérer des variantes correspondant à des fautes d'orthographes courantes, ce qui permettra de les prendre en compte en recherche sans qu'elles apparaissent en affichage ou en impression du thésaurus,
  - ✓ la représentation par une image ;
- les différents types de notes : notes de définition et d'application (*scope note*), exemples, notes historiques, etc. ;
- les relations sémantiques : hiérarchie et association.

3. Composition de SKOS

L'élément essentiel est le « *SKOS Core* », ou *le noyau* de SKOS. Ce terme de *noyau* est à prendre au sens propre car il s'agit bien des classes et des propriétés de base. Elles peuvent être complétées par les « *SKOS Extensions* », les *extensions* de SKOS, qui permettent de:

- ✓ représenter les relations de manière plus fine : il est possible, par exemple, de préciser si la nature d'une relation de hiérarchie est de type tout/partie ou classe/instance ;
- ✓ préciser certains attributs d'un concept : une note historique, par exemple.

Le SKOS Mapping qui est Vocabulaire pour exprimer des correspondances (alignements exacts ou correspondances approximatives) entre concepts provenant de schémas différents.

Certains vocabulaires ont été mis au format SKOS et sont disponibles pour le public, notamment les thésaurus multilingues européens AGROVOC[15] et GEMET[16].

4. Les principaux éléments

Classes

- ✓ skos:Concept : descripteur, catégorie, rubrique
- ✓ skos:ConceptScheme : ensemble de concepts organisé

Propriétés

- ✓ skos:prefLabel : terme préférentiel (un par langue)
- ✓ skos:altLabel : terme alternatif, synonymes
- ✓ skos:definition ; skos:scopeNote : définition et note d'application
- ✓ skos:broader ; skos:narrower : relation entre concept générique et concept spécifique
- ✓ skos:related : relations entre termes associés (pas de sémantique particulière)

Exemple

La figure ci-dessous présente un exemple, issu du SKOS, de modèle de représentation d'un concept, complété par le fichier RDF correspondant.

[15] www.fao.org/agrovoc

[16] www.eionet.europa.eu/gemet

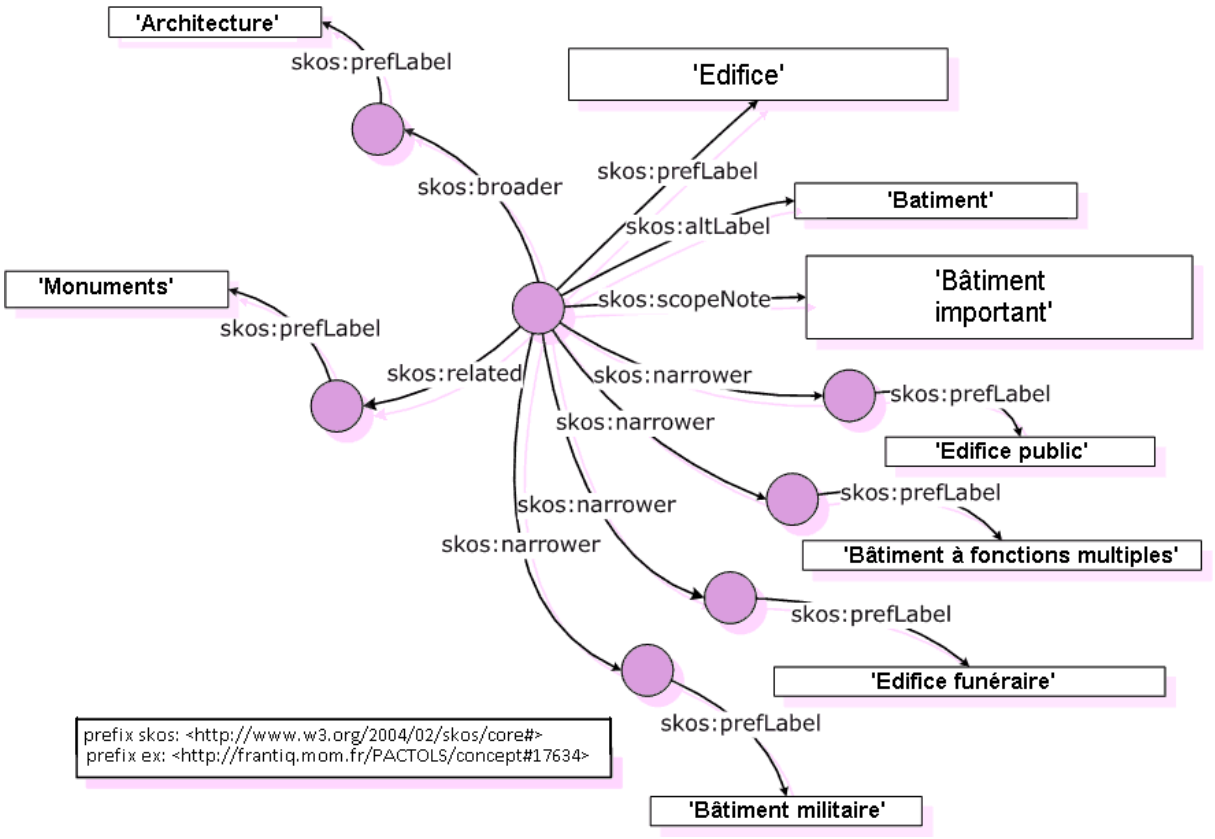


***Figure -26-*:** Modèle de représentation du concept Edifice [77]

Extrait du fichier RDF correspondant :

```
<skos:Concept rdf:about="http://frantiq.mom.fr/PACTOLS/concept#17634">
<skos:prefLabel xml:lang="fr">édifice</skos:prefLabel>
<skos:altLabel xml:lang="fr">bâtiment</skos:altLabel>
<skos:broader rdf:resource="http://frantiq.mom.fr/PACTOLS/concept#13289"/> [architecture]
<skos:narrower rdf:resource="http://frantiq.mom.fr/PACTOLS/concept#13685"/> [bât à FM]
<skos:narrower rdf:resource="http://frantiq.mom.fr/PACTOLS/concept#13684"/> [bâtiment militaire]
<skos:narrower rdf:resource="http://frantiq.mom.fr/PACTOLS/concept#17635"/> [edifice funéraire]
<skos:narrower rdf:resource="http://frantiq.mom.fr/PACTOLS/concept#17636"/> [edifice public]
<skos:related rdf:resource="http://frantiq.mom.fr/PACTOLS/concept#12243"/> [monuments]
<skos:scopeNote xml:lang="fr">Bâtiment important (Rob.)</skos:scopeNote>
< /sko  s:Concept>
```

Figure-27- : Extrait du fichier RDF correspondant

5. Définitions de quelques termes cités dans les paragraphes ci-dessus

5.1. Taxonomie

D‘après Karl Dubost, « dans une taxonomie, le vocabulaire contrôlé est organisé sous forme hiérarchique simple. Cette hiérarchisation correspond souvent à une spécialisation. Il existe donc un lien précis entre un terme du vocabulaire et ses enfants. Ce lien donne un sens supplémentaire, une signification. D'un vocabulaire contrôlé, on passe à un vocabulaire organisé.

Par exemple, dans une classification animale, nous aurons les vertébrés, invertébrés et puis sous les vertébrés nous auront les mammifères, les ovipares, etc. Tous ces termes nous permettront de classifier les animaux. On pourra donc dire que les mammifères représentent une sous-catégorie (sous-classe) des vertébrés. »

Les taxonomies permettent de traduire des relations hiérarchiques de type généralisation ou spécialisation entre les descripteurs.

5.2. Thésaurus

Un thésaurus est un vocabulaire contrôlé (ensemble de descripteurs) et organisé (ensemble de relations entre les descripteurs) servant à représenter des concepts.

Les relations existantes entre les descripteurs sont des relations d'équivalence (synonymes), des relations de hiérarchisation (spécification ou généralisation) et des relations d'association (du type "relatif à" ou "similaire à").

Ainsi lorsque l’on recherche un mot avec un thésaurus, la recherche s’étendra à tous les mots équivalents, parents et associés. L’utilisateur aura donc plus de chance de trouver un résultat correspondant à sa recherche.

Par exemple un thésaurus reliant « vente » à « production », « voiture » à « véhicule », et « France » à « Europe », permettra pour une question portant sur les ventes de voitures en France de trouver des ressources indexées avec « production » « véhicule » « Europe ».

5.3. URI (Uniform Ressource Identifier)

Une URI est un identifiant uniforme de ressource. D’après le site descripteurs, « L’URI est le protocole qui normalise la syntaxe de la chaîne de caractères qui identifie une ressource physique (image, document sur le web) ou abstraite (concepts) ». Une URI doit permettre d’identifier une ressource de manière unique et pérenne sur un réseau (par exemple le web).

D'après le site descripteurs, « cet identificateur permet aussi de distinguer des ressources entre elles. Parmi les URI, on peut distinguer: l'URL (Uniform resource locator = Localisation de ressource uniforme) » qui identifie une ressource sur un réseau, la localise et permet d'en obtenir une représentation. Il existe également l'URN (Universal Resource Name = Nom de ressource uniforme) qui identifie la ressource indépendamment de sa localisation. Le code ISBN, qui est l'identifiant unique d'un livre et permet de le retrouver dans n'importe quelle librairie ou bibliothèque dans le monde entier, est aussi une forme d'URI.

Annexe2

Protégé

Présentation générale de l'éditeur Protégé

Protégé [78], développé à l'Université de Stanford[17], est sans aucun doute l'environnement d'édition d'ontologies le plus utilisé aujourd'hui. Son noyau est basé sur le modèle des frames [79] mais les ontologies développées avec cet environnement peuvent être exportées dans plusieurs formats (RDF(S), OWL, XML Schema).

L'environnement d'édition offre deux possibilités de construction d'ontologies :

1. avec l'éditeur à base de frames, il permet la construction et l'instanciation d'ontologies basées sur le modèle des frames, en conformité avec le protocole OKBC [80] ; dans ce modèle, une ontologie est composée d'un ensemble de classes, organisé sous la forme d'une hiérarchie de subsumption ; d'un ensemble de slots associés aux classes, décrivant leurs propriétés et relations ; et enfin d'un ensemble d'instances des classes définies.

2. avec l'éditeur OWL, il permet la construction d'ontologies pour le Web Sémantique et particulièrement la construction d'ontologies SKOS (voir *Figure* -28-). Il offre avec cet éditeur tous les outils nécessaires pour l'édition des différents éléments d'une ontologie OWL (concepts, propriétés, instances), avec la possibilité de spécifier des contraintes et d'utiliser des moteurs d'inférence externes tels que Racer [81] ou Pellet [82] pour vérifier la consistance de l'ontologie et d'inférer de nouvelles connaissances.

Protégé s'enrichit régulièrement de l'apport de la communauté des utilisateurs et développeurs grâce au système de plugins qui permettent de rajouter de nouvelles fonctionnalités à l'outil.

L'outil permet d'intégrer plusieurs ontologies et de gérer les différentes versions d'une même ontologie.

[17] Http : //protege.stanford.edu/

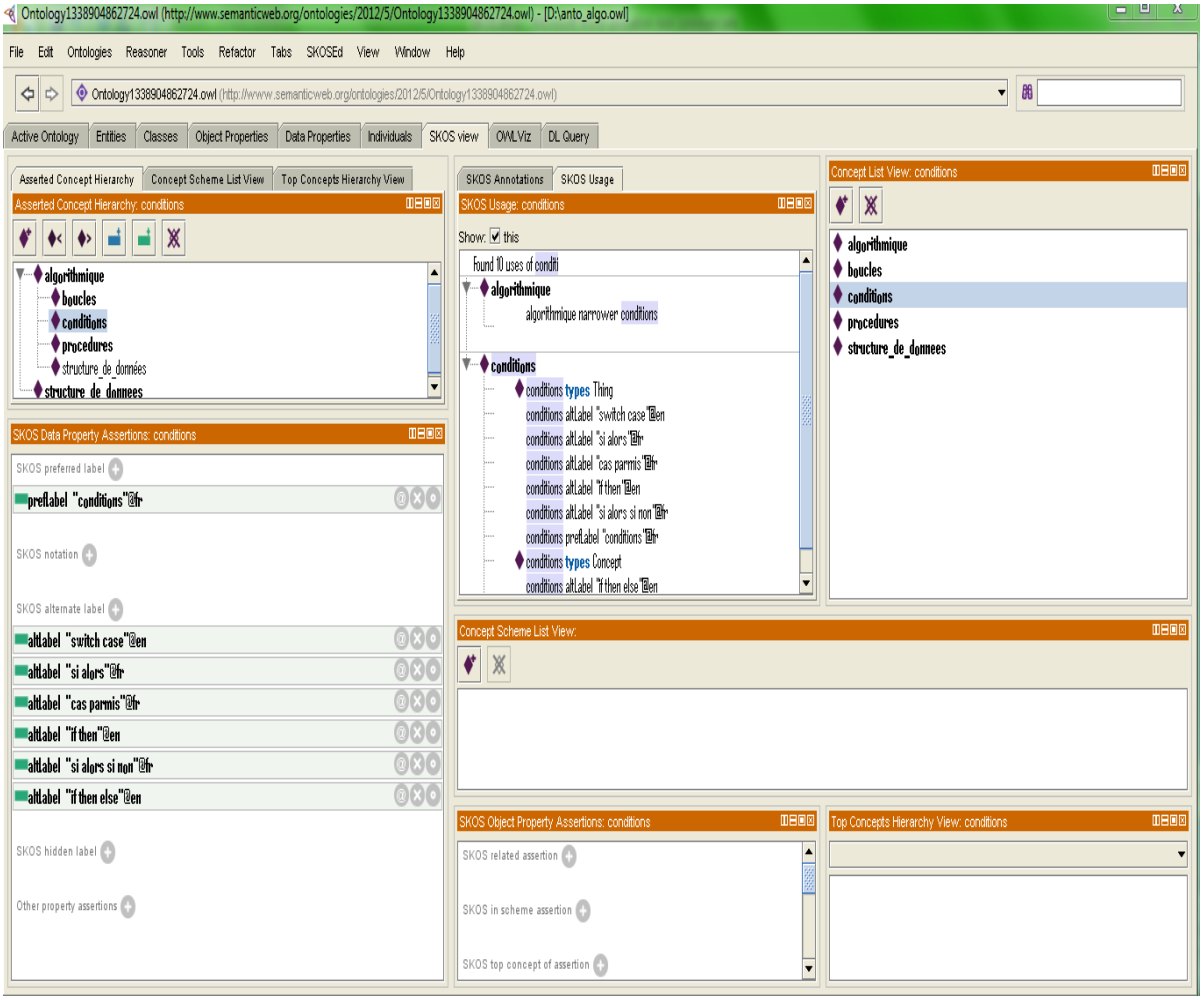


Figure-28- **:** Interface d'édition d'une ontologie SKOS sous Protégé

Annexe 3

JDOM

JDOM est une API du langage Java développée indépendamment de *Sun Microsystems*. Elle permet de manipuler des données XML plus simplement qu'avec les API classiques. Son utilisation est pratique pour tout développeur Java et repose sur les collections XML de Sun. En septembre 2004 JDOM est disponible en version 1.0 et est compatible avec les versions 1.1 du JDK et supérieures.

JDOM utilise des collections SAX (cf. annexe4) pour parser les fichiers XML. Il s'inspire largement de DOM pour manipuler les éléments d'un *Model Object Document* spécifique crée grâce à un constructeur basé sur SAX. On peut donc construire des documents, naviguer dans leur structure, ajouter, modifier, ou supprimer soit des éléments soit du contenu. Selon les auteurs de JDOM, pour un programmeur Java, JDOM est plus simple que les API classiques DOM[18] car il est très laborieux de développer des applications complexes autour d'XML avec DOM.

Dans la mesure où l'API JDOM ne fait pas partie de la machine virtuelle Java, il faut, pour pouvoir l'utiliser, l'importer dans votre projet : *import jdom.jar*.

Les principales fonctionnalités de JDOM sont les suivantes.

- Créer un arbre en mémoire puis en faire un fichier XML :

✓ Créer une arborescence simple, La constitution d'un fichier XML en partant de zéro est plus simple. Il suffit de construire chaque élément et de les ajouter les uns aux autres de façon logique. Un noeud est un Element.

✓ Afficher et enregistrer son fichier XML, une unique classe pour ces deux flux de sortie: XMLOutputter(), qui prend entre autre en argument un Format de sortie. En plus de 3 formats par défaut, la classe Format contient une panoplie de méthodes pour affiner votre sérialisation.

- Parcourir un fichier XML chargé en mémoire :

✓ Parser un fichier XML, revient à transformer un fichier XML en une arborescence JDOM grâce au constructeur SAXBuilder, basé comme son nom l'indique sur l'API SAX.

[18] *www.encyclopédie.com* ; « Les API pour XML », 65 pages, document de F. Rossi, apiacoa.org/contact.html

✓ Parcourir une arborescence – utilisation pour cela de deux classes apartenant à Java.util (Java.util.List ; Java.util.Iterator) pour créer une liste basée sur les noeuds de l'arborescence puis la parcourir grâce à un iterator.

✓ Filtrer les données de l'arborescence

- Modifier une arborescence JDOM :

✓ Modifie des Elements ;

- Transformer JDOM

✓ Passer de DOM à JDOM et l'inverse ;

✓ JDOM et XSLT[19] : il est très facile de faire des transformations XSLT sur un document JDOM

- Support de XPath[20] 1.0.

JDOM modélise un document XML comme un objet Document qui contient éventuellement des objets « liste de commentaires » et des objets « *Processing Instruction* » et un unique objet *Element* qui est en fait l'élément racine.

Chaque objet Element contient une liste de ses enfants : objet *Comment*, objet *Processing Instruction*, objet *Text*, et objet *Element*. Il a de plus une liste séparée des attributs et des espaces de noms additionnels. Tout ces enfants sont optionnels. La liste des enfants est accessible au travers de la méthode *getContent()*. Un contenu peut être ajouté à un élément ou à un document grâce à la méthode surchargée *addContent()* ou bien retiré de l'arbre en utilisant une autre méthode surchargée r*emoveContent*. D'autres opérations transformant l'arbre lui-même, telle qu'insérer un noeud au début ou au milieu des enfants d'un *Element* requiert d'utiliser des méthodes de *Java.util.List*.

A l'exception du *Namespace*, les classes "noeud" de JDOM sont un constructeur public et peuvent être construite sensiblement de toutes les façons possibles.

---

[19] XSLT (Extensible Style Language Transformations) est un langage destiné à transformer un fichier XML en quelque chose d'autre. Ce quelque chose d'autre sera le plus souvent un fichier XML ou HTML. Mais ce pourra être tout aussi bien un fichier d'un autre format : par exemple du texte pur...

[20] XPath, ou XML Path**,** est une syntaxe (non XML) pour désigner une portion d'un document XML.

La classe *Content*

Content est la super classe des objets JDOM qui peuvent être des enfants légal d'un noeud Parent. Elle est la classe abstraite dont hérite les classes Comment, DocType, Element, EntityRef, Processing Instruction, Text. Elle étend Java.lang.Object et implémente Java.lang.Cloneable, Java.io.Serializable. Elle n'est dotée que d'une seule propriété : elle a un parent (Parent).

Les méthodes de la classe Content sont :

- **clone**() : retourne un Object correspondant à une copie indépendante des enfants et de leur descendants détachés d'un parent ou d'un document ;
- **detach**() : retourne un Content correspondant à l'enfant détaché de son parent, ne fait rien si l'enfant n'a pas de parent ;
- **equals**(Java.lang.Object ob) : test l'égalité de ce contenu avec le contenu de l'Object ob, retourne un boolean ;
- **getDocument**() : retourne le Document auquel est rattaché ce contenu enfant ;
- **getParent**() : retourne le Parent de ce contenu enfant ou null si cete enfant est non attaché à un parent ;
- **getParentElement**() : retourne l'Element parent de cet Element ou null si l'élément n'est pas attaché ou s'il est l'Element root ;
- **getValue**() : retourne un string correspondant au chemin XPath 1.0
- **hashCode**() : retourne un int correspondant au "hash code" pour ce contenu
- **setParent**(Parent parent) : indique le parent de ce contenu

L'interface *Parent*

L'interface « Parent » est implémentée par les classes « Document » et « Element » et offre les méthodes suivantes :

- **clone**() : retourne un clone de ce parent et de ses enfants (Java.lang.Object) ;
- **cloneContent**() : retourne un clone du contenu (i.e. une liste - Java.util.List) de ce parent;
- **getContent**() : retourne le contenu de ce parent sous forme d'une liste (Java.util.List) contenant des objets de type « contenu » (Content) ;
- **getContent**(Filter filter) : retourne le contenu de ce parent sous forme d'une liste (Java.util.List) filtrée par le filtre filter ;
- **getContent**(int index) : retourne dans un objet de type "contenu" (Content) l'enfant positionné à l'index fourni en paramètre ;

- **getContentSize**() : retourne le nombre d'enfants (int) présents dans la liste "contenu" (Content) de ce parent ;
- **getDescendants**() : retourne un Iterator (Java.util.Iterator) qui parcourt tous les descendants dans l'ordre du document ;
- **getDescendants**(Filter filter) : retourne un Iterator qui parcourt tous les descendants dans l'ordre du document en appliquant un filtre qui ne retourne que les éléments désignés par le filtre ;
- **getDocument**() : retourne le document (Document) auquel est rattaché ce parent ou "null" si la branche contenant ce parent est momentanément non attachée à un document ;
- **getParent**() : retourne le Parent de ce parent, ou null if ce parent est non attaché à un autre parent ;
- **indexOf**(Content child) : retourne un int représentant l'index de l'enfant fourni dans la liste contenu, ou -1 si l'enfant n'est pas enfant de ce parent ;
- **removeContent**() : retire tous les contenus de ce parent et retourne la List des enfants détachés ;
- **removeContent**(Content child) : retourne un Boolean, retire un noeud enfant unique de la liste Content ;
- **removeContent**(Filter filter) : retire de ce parent tous les enfants correspondant au critère du filter et retourne une List de ces enfants retirés ;
- **removeContent**(int index) : retire et retourne l'enfant positionné à "index", ou null si il n'y a pas d'enfant à cet index.

La classe *Document*

La classe "Document" de JDOM représente un document XML bien-formé complet. La classe « Document » implémente l'interface « Parent ». Un document a trois propriétés :

- Un Element root : peut être nul temporairement ;
- Un « DocType object » (DTD) : peut être nul ;
- Une liste contenant l'élément root et, éventuellement, des « processing instructions » ou des « comments » dans le prologue et l'épilogue du document XML.

Les méthodes relatives au « root » et à la liste de contenu :

- public boolean hasRootElement() ;
- public Element getRootElement() ;
- public Document setRootElement(Element rootElement) ;

- public Element detachRootElement() ;
- public List getContent() : normal car implémente Parent ;
- public List getContent(Filter filter) : normal car implémente Parent ;
- public Document setContent(List newContent).

La classe *Element*

La structure d'un document XML est basée sur ses éléments. La classe *Element* est donc la classe la plus importante et la plus volumineuse de l'API JDOM. La classe *Element* est le moyen privilégié par lequel un programme parcourt l'arbre pour trouver un contenu « *Content* » particulier.

Un objet *Element* a sept propriétés de base :

- **Local name** : un String initialisé à la construction de l'Element ; ne peut jamais être null ou vide ; est accessible via les méthodes setName() et getName() ;
- **Namespace** : un objet "espace de nommage" ; peut prendre la valeur Namespace.NO_NAMESPACE si l'élément n'appartient pas à un espace de nommage. Il est accessible via les méthodes setNamespace() et getNamespace() ;
- **Content** : une List sans doublon contenant tous les enfants dans l'ordre ;
- **Parent** : Element parent contenant cet Element ou null si cet Element est le root ou si cet Element n'est pas attaché à un Document ;
- **Owner document** : le Document qui contient cet Element ou null si cet Element n'est pas rattaché à un Document ; accessible par la méthode getDocument(), il peut être changé par l'ajout d'un Element à un nouveau Document après que cet Element ait été détaché (detach()) de son parent précédent ;
- **Attributes** : une List contenant des objets Attribute, un pour chacun des attributes de l'élément. Bien que JDOM stock les attributs dans une List, l'ordre n'est pas significatif et peut être different de l'ordre d'apparition dans le document XML. La liste est accessible via les méthodes getAttributes() et setAttributes(). Les articles de la liste peuvent être lus et/ou modifiés par les méthodes getAttribute(), getAttributeValue(), and setAttribute() ;
- **Additional namespaces** : une List contenant les objets "espace de nommage".

Ici, JDOM ne suit pas les conventions Java. Plutôt que de retourner void, ces méthodes retournent toutes l'objet Element invoqué par la méthode. L'objectif est d'autoriser le chaînage (l'imbrication) des « setters ».

Parcours d'arbre total

La méthode getContent() est le moyen de naviguer au sein d'un arbre JDOM. Cette méthode retourne une liste incluant tous les enfants d'un Element (comments, processing instructions, text nodes, elements). Le parcours total de l'arbre se fait par appel récursif de getContent().

Parcours d'arbre en ne s'intéressant qu'aux Elements – Tous les enfants

La class Element dispose de cinq méthodes permettant d'opérer uniquement sur les enfants de type Element :

public List getChildren();

public List getChildren(String *name*);

public List getChildren(String *name*, Namespace *namespace*);

public List removeChildren(String *name*);

public List removeChildren(String *name*, Namespace *namespace*);

Ces méthodes sont similaires à getContent() et removeContent() mais elles ne s'intéressent qu'aux Element. Il est important de se rappeler que lorsqu'un Element est supprimé, la totalité de l'Element est supprimé : ses enfants et autres descendants sont supprimés en même temps que lui.

Parcours d'arbre en ne s'intéressant qu'aux Elements – Un seul enfant à la fois

public List getChild(String *name*);

public List getChild(String *name*, Namespace *namespace*);

Bien qu'elles permettent d'accéder directement à un enfant lorsqu'il est unique (plutôt que d'itérer dans une liste), l'utilisation de ces méthodes présente deux problèmes. Le premier est qu'elles ne retournent que le premier enfant : s'il existe plus d'un enfant possédant le même nom, seul le premier est retourné. Le second problème est que, s'il existe pas d'enfant, alors getChild() retourne null, menant ainsi à une NullPointerException. En l'absence d'utilisation de schéma ou de DTD pour vérifier ce qu'est exactement un enfant auquel vous adressez ces méthodes, il faut préférer les méthodes getChildren(). Celles-ci retournent toujours une liste non-null.

public Element removeChild(String *name*);

public Element removeChild(String *name*, Namespace *namespace*);

Les méthodes removeChild() partagent avec getChild() le même problème : elles ne s'intéressent qu'au premier enfant. Cependant, après avoir supprimé le premier enfant, le second est maintenant le premier… Après avoir supprimé celui-ci, le troisième déviant alors à son tour le premier. Il existe donc une option qui ne fonctionne pas avec getChild() : vous pouvez simplement appeler removeChild() jusqu'à ce que null soit retourné, indiquant qu'il n'y a plus d'enfants.

Lire et écrire le texte d'un élément

Les méthodes permettant de lire le texte d'un élément sont :

public String getText();

public String getTextTrim();

public String getTextNormalize();

public Element setText(String *text*);

La méthode getText() retourne le contenu de type PCDATA d'un element. La méthode getTextTrim() retourne la même chose en ayant supprimé les espaces précédents et suivants le texte. La méthode getTextNormalize() fait de même et supprime également, au sein du texte, tous les espaces consécutifs de façon à assurer que les mots ne sont séparés que par un espace unique.

Lire le texte d'un enfant

Un cas fréquent au sein des documents XML est celui d'un élément qui ne contient que des éléments, ces derniers ne contenant eux-mêmes que des PCDATA. Par exemple:

<channel>

<title>Slashdot: News for nerds, stuff that matters</title>

<link>http://slashdot.org/</link>

<description>News for nerds, stuff that matters</description>

</channel>

Considérant un tel élément, JDOM fournit six méthodes d'extraction du texte de ces éléments enfants :

public String getChildText(String *name*);

public String getChildText(String *name*, Namespace *namespace*);

public String getChildTextTrim(String *name*);

public String getChildTextTrim(String *name*, Namespace *namespace*);

public String getChildTextNormalize(String *name*);

public String getChildTextNormalize(String *name*, Namespace *namespace*);

Si nous reprenons l'exemple :

String title = channel.getChildText("title");

String description = channel.getChildText("description");

String link = channel.getChildText("link");

Ces méthodes posent toutefois problème. Tout d'abord, tout comme les méthodes getText/getTextTrim/getTextNormalize(), elles faillent silencieusement lorsque l'un d'un des éléments enfant contient des éléments enfants. En l'absence d'un schéma ou d'une DTD, il est donc, comme les méthodes getText/getTextTrim/getTextNormalize(), déconseillé de les utiliser.

Filtres

Il est possible de paramètrer la méthode getContent() en lui passant un objet org.jdom.filter.Filter de façon à limiter le contenu retourné par la méthode. Cette interface determine si un objet peut être ajouté, supprimé ou inclus dans une liste particulière. Pour le parcours et la recherche, seule la méthode matches() importe. Elle détermine si un objet particulier est inclus ou non dans la List retournée par getContent(). Les méthodes canAdd() et canRemove() teste si un objet particulier peut être ajouté ou supprimé. Le package org.jdom.filter inclus deux implementations de cette interface : ContentFilter and ElementFilter.

Ajouter ou supprimer des enfants

L'ajout de noeud à un Element se fait par l'appel aux méthodes :

public Element addContent(String *s*);

public Element addContent(Text *text*) throws IllegalAddException;

public Element addContent(Element *element*) throws IllegalAddException;

public Element addContent(ProcessingInstruction *instruction*) throws IllegalAddException;

public Element addContent(EntityRef *ref*) throws IllegalAddException;

public Element addContent(Comment *comment*) throws IllegalAddException;

Ces méthodes ajoutent leur argument à la liste d'enfants de l'Element. Sauf pour addContent(String), elles traitent toutes une IllegalAddException si l'argument a déjà un parent. La méthode addContent(String) crée juste un noeud Text ; elle n'ajoute pas un objet String à la liste Content. Toutes ces méthodes retournent l'objet Element invoqué. Ces méthodes ajoutent le nouveau nœud à la fin de la liste de l'élément. Pour insérer un nœud à une position différente, il faut récupérer l'objet List lui-même.

Il existe six méthodes de suppression d'un noeud de la liste :

public Element removeContent(Text *text*); public Element removeContent(CDATA *cdata*);
public Element removeContent(Element *element*);
public Element removeContent(ProcessingInstruction *instruction*);
public Element removeContent(EntityRef *ref*);
public Element removeContent(Comment *comment*);
Il est également possible de retrouver la List par la méthode getContent() et de supprimer les éléments en utilisant les méthodes de List remove() et removeAll().
Il n'existe actuellement pas de méthode pour supprimer tous les enfants d'un Element. Cependant, passer null en paramètre de elt.setContent() supprime les enfants de elt.
Parents et ancêtres
Le parcours de l'arbre peut aussi se faire en remontant. Il suffit pour cela de demander le parent du parent … jusqu'à ce que la méthode getParent() retourne null. Ceci indique qu'on est face à la racine de l'arbre, ou face à son noeud le moins profond.
public Document getDocument();
public boolean isRootElement();
public Element getParent();
Contrairement aux éléments DOM, les éléments JDOM ne sont pas irrévocablement attachés à un même Document. Un Element peut n'appartenir à aucun Document (dans ce cas, getDocument() retourne null) ; il peut également changer de Document. Cependant, les Elements ne peuvent avoir plus d'un parent à la fois. Donc, avant de bouger un élément vers un Document différent ou vers une position différente dans le même Document, il faut au préalable détacher l'élément de son parent courant en invoquant la méthode :
public Element detach();
Après avoir appelé detach(), il est possible d'ajouter l'Element à un autre Element ou Document. Sans l'appel à la méthode detach(), une IllegalAddException intervient. Une autre limitation est qu'un Element ne peut pas être son propre parent ou ancêtre, directement ou indirectement. Tenter l'ajout d'un élément en violant cette restriction provoque ici encore une IllegalAddException. Il est possible de tester si un élément est l'ancêtre d'un autre en utilisant la méthode :
public boolean isAncestor(Element *element*);

Attributs

La classe Element est dotée de treize méthodes qui lisent et écrivent les valeurs des différents attributs de l'élément. Hormis cas particulier, ces treize méthodes suffisent à traiter les attributs et il n'est pas nécessaire d'utiliser directement la classe Attribute.

public Attribute getAttribute(String *name*);

public Attribute getAttribute(String *name*, Namespace *namespace*);

public String getAttributeValue(String *name*);

public String getAttributeValue(String *name*, Namespace *namespace*);

public String getAttributeValue(String *name*, String *default*);

public String getAttributeValue(String *name*, Namespace *namespace*, String *default*);

public Element setAttributes(List *attributes*) throws IllegalAddException;

public Element setAttribute(String *name*, String *value*) throws IllegalNameException, IllegalDataException;

public Element setAttribute(String *name*, String *value*, Namespace *namespace*) throws IllegalNameException, IllegalDataException;

public Element setAttribute(Attribute *attribute*) throws IllegalAddException;

public boolean removeAttribute(String *name*, String *value*);

public boolean removeAttribute(String *name*, Namespace *namespace*);

public boolean removeAttribute(Attribute *attribute*);

Ces méthodes suivent toutes la même règle basique. Si un attribut est dans un espace de nommage, il faut spécifier le nom local et l'espace de nom pour y accéder. Si un attribut n'est pas dans un espace de nommage, il suffit d'utiliser son nom pour y accéder. Les méthodes de lecture peuvent spécifier une valeur par défaut utilisée lorsque l'attribut n'est pas trouvé. Les méthodes getAttributeValue() retournent toutes la valeur de l'attribut sous forme de String. Si l'attribut est lu par un parseur, la valeur sera normalisée conformément à son type. Cependant, les attributs ajoutés en mémoire avec la méthode setAttribute() ne sont pas normalises. Les méthodes d'écriture retournent toutes l'objet Element lui-même. Les méthodes de suppression retournent toutes un boolean : *true* si l'attribut a été supprimé, *false* s'il n'a pas été supprimé.

La classe *Attribute*

Méthodes de la classe Attribute :

- **clone**() : retourne un clone de cet Attribut ;

- **detach**() : détache l'attribut de son parent et le retourne ou ne fait rien si l'attribut n'a pas de parent ;
- **equals**(Java.lang.Object ob) : retourne un Boolean indiquant l'égalité ou non entre l'Attribut et Object ob ;
- **getAttributeType**() : retourne un int correspondant à type déclaré de l'attribut ;
- **getBooleanValue**() : renvoie la valeur de l'attribut sous forme d'un boolean ou une DataConversionException si la conversion ne peut être réalisée ;
- **getDocument**() : renvoie le Document auquel se rattache l'attribut, null si l'attribut ne se rattache à aucun Document ;
- **getDoubleValue**() : renvoie la valeur de l'attribut sous forme d'un double ou une DataConversionException si la conversion ne peut être réalisée ;
- **getFloatValue**() : renvoie la valeur de l'attribut sous forme d'un float ou une DataConversionException si la conversion ne peut être réalisée ;
- **getIntValue**() : renvoie la valeur de l'attribut sous forme d'un int ou une DataConversionException si la conversion ne peut être réalisée ;
- **getLongValue**() : renvoie la valeur de l'attribut sous forme d'un long ou une DataConversionException si la conversion ne peut être réalisée ;
- **getName**() : renvoie le nom local de l'attribut sous forme d'un String ;
- **getNamespace**() : renvoie le Namespace de l'attribut :
- **getNamespacePrefix**() : renvoie sous forme de String le "namespace prefix » de l'attribut.
- **getNamespaceURI**() : renvoie le URI sous forme de String ;
- **getParent**() : retourne l'élément Parent de cet attribute ;
- **getQualifiedName**() : renvoie sous forme de String le nom qualifié de l'attribut ;
- **getValue**() : retourne sous forme de String la valeur de l'attribut ;
- **hashCode**() : retourne le hash code (int) de l'attribut ;
- **setAttributeType**(int type) : initialise le type de l'attribut ;
- **setName**(Java.lang.String name) : initialise le nom local de l'attribut ;
- **setNamespace**(Namespace namespace) : initialise l'espace de nommage de l'attribut ;
- **setParent**(Element parent) : initialise le parent de l'attribut ;
- **setValue**(Java.lang.String value) : initialise la valeur de l'attribut ;
- **toString**() : retourne une représentation sous forme de String de l'attribut (pratique pour le débogage).

Chaque Attribut a cinq propriétés de base :

Nom local - Un string accessible par les méthodes setName() et getName().

Namespace (espace de nommage) - Un objet “espace de nommage” accessible via setNamespace() et getNamespace(). Tout attribut non préfixé est initialisé par Namespace.NO_NAMESPACE.

Valeur - Un String contenant la valeur normalisée de l’Attribut. Les méthodes d’accès sont getValue() et setValue(). Il existe également des méthodes permettant de lire la valeur de l’attribut sous forme de double, float, long, int, ou boolean.

Parent - L’Elément qui possède l’Attribut, accessible via les méthodes getParent() et setParent(). Un attribut ne peut avoir plus d’un parent. Avant d’attacher un Attribut à un nouveau parent, vous devez invoquer detach() pour séparer l’Attribut de son parent en cours.

Type - Le type de l’Attribut tel que spécifié par la DTD , correspond à une des dix constantes : Attribute.CDATA_ATTRIBUTE, Attribute.ID_ATTRIBUTE, Attribute.IDREFS_ATTRIBUTE, Attribute.ENUMERATED_ATTRIBUTE, … Si la DTD ne spécifie aucun type d’attribut, le type est alors par défaut Attribute.UNDECLARED_ATTRIBUTE. Le type est accessible via les méthodes getAttributeType() et setAttributeType().

Il est également possible en appelant getDocument() de connaître le Document auquel appartient l’attribut mais ceci n’est pas indépendant de l’Element auquel appartient l’attribut.

Annexe 4

SAX

SAX est une API dédiée à l'analyse syntaxique. Elle est basée sur un modèle événementiel, cela signifie que SAX permet de déclencher des événements au cours de l'analyse du document XML : une balise ouvrante est un événement, une balise fermante, un autre événement... Soit le document XML suivant :

<personne>
<nom>Pillou</nom>
<prenom>Jean-François</prenom>
</personne>

Une interface événementielle telle que l'API SAX permet de créer des événements à partir de la lecture du document ci-dessus. Les événements générés seront :

start document
start element: personne
start element: nom
characters: Pillou
end element: nom
start element: prenom
characters: Jean-François
end element: prenom
end element: personne
end document

Ainsi une application basée sur SAX peut gérer uniquement les éléments dont elle a besoin sans avoir à construire en mémoire une structure contenant l'intégralité du document. L'API SAX définit les quatre interfaces suivantes :

DocumentHandler possédant des méthodes renvoyant des événements relatifs au document:

- *startDocument()* renvoyant un événement lié à l'ouverture du document ;
- *startElement()* renvoyant un événement lié à la rencontre d'un nouvel élément ;
- *characters()* renvoyant les caractères rencontrés ;
- *endElement()* renvoyant un événement lié à la fin d'un élément ;
- *endDocument()* renvoyant un événement lié à la fermeture du document ;

ErrorHandler possèdant des méthodes renvoyant des événements relatifs aux erreurs ou aux avertissements ;
DTDHandler renvoie des événements relatifs à la lecture de la DTD du document XML ;
EntityResolver permet de renvoyer une URL lorsqu'une URI est rencontrée.

SAX est une API générale pour la lecture d'un flux XML. Il existe des implémentations de cette API dans la plupart des langages (C++, C#, Java, Pascal, PHP, ...)

Annexe 5

La mesure *cf.idf* [35]

$$cf.idf(d,k) = cf(d,k) * \log\left(\frac{N}{df(k)}\right)$$

$$cf(d,k) = \sum_{i=1}^{p} freq(T_i)$$

- ✓ ***cf(d,k)*** est la fréquence du concept ***k*** dans le document ***d*** (la mesure locale)
- ✓ ***freq(Ti)*** est la fréquence du terme ***i*** du concept ***k*** dans le document ***d***
- ✓ ***N*** est le nombre de documents de la collection
- ✓ ***df(k)*** est le nombre de documents ou le concept ***k*** apparaît

$$\log\left(\frac{N}{df(k)}\right)$$

Est la mesure globale qui nous montre la relation du concept dans l'ensemble de documents.

Références

[1] El Hassan Abdelwahed — Azzedinne Lazrek ,Des ontologies pour la description des ressources pédagogiques et des profils des apprenants dans l'e-learning *Laboratoire d'Ingénierie des Systèmes Informatiques (LISI) Département d'Informatique, Faculté des Sciences Semlalia de Marrakech B.P. 2390, Boulevard Prince My Abdellah, 40000, Marrakech.*

[2] R. Studer, R. Benjamins, D. Fensel, Knowledge Engineering: Principles and Methods, Data and Knowledge Engineering, 25(1-2) pp 161-197, 1998.

[3] Rapport de Master 2ème année. Environnements Informatiques pour l'Apprentissage umain et Didactique. Structuration et analyse de trac eshybrides issues de situation d'apprentissage Présenté par : Gwenegan Rzédécembre-2005.

[4] Jermann P., Soller A., Muehlenbrock M., From Mirroring to Guiding : A Review State of the Art Technologyfor supporting Collaborative Learning. Proceedings of the First European Conference on Computer-Supported Collaborative Learning, 2001.

[5] CSE, un modèle de traitement de traces. Rapport interne de recherche CLIPS-IMAG 2005.

[6] Champin P-A, Prié Y., Mille A., MUSETTE : a Framework for Knowledge Capture from Experience. EGC'04, Clermont Ferrand, 2004.

[7] Pernin J.P., "CSE, un modèle de traitement de traces. Rapport interne de recherche" CLIPS-IMAG 2005.

[8] THÈSE Pour l'obtention du grade de Docteur de l'Université Louis Lumière Lyon 2 Traces d'interactions et processus cognitifs en activité conjointe : Le cas d'une co-rédaction médiée par un artefact numérique Présentée par Magali OLLAGNIER-BELDAME.

[9] Szilas, N., Kavakli, M. (2006). PastMaster@Strorytelling: A Controlled Interface for Interactive Drama, In *Proceedings of IUI 2006: International Conference on Intelligent user Interfaces*, CSIRO ICT Centre, Macquarie University, Sydney, Australia, 29 January to 1 February, pp288-290.

[10] Lieberman, H. (2001). Interfaces that Give and Take Advice, in Carroll, J. (Ed). Human-Computer Interaction for the New Millenium, ACM Press/Addison-Wesley, pp. 475-485.

[11] Després C. (2001). Modélisation et Conception d'un Environnement de Suivi Pédagogique Synchrone d'Activités d'Apprentissage à Distance, Thèse de Doctorat. Université du Maine, Le Mans.

[12] Neal, A. S. & Simons, R. M. (1983). Playback: A method for evaluating the usability of software and its documentation. In Proceedings of CHI '83, pp. 78-82.

[13] Dubois, J.-M., Dao-Duy, J.-M., & Eldika, S. (2000). L'analyse des traces informatiques des usages : un outil pour valider la conception d'un site web. Dans Actes des rencontres jeunes chercheurs en Interaction Homme-Machine 2000, pp. 85-89.

[14] Greenberg, S. & Witten, I.H. (1988). How Users Repeat Their Actions on Computers: Principles for Design of History Mechanisms. In: Soloway, E., Frye, D., Sheppard, S. B. (Eds) Proceedings of the ACM CHI 88 Human Factors in Computing Systems Conference. June 15-19, 1988, Washington, DC, USA. pp.171-178.

[15] Allal L., & Saada-Robert M. (1992). La métacognition : cadre conceptuel pour l'étude des régulations en situation scolaire, Archives de Psychologie, 60, pp. 265-296.

[16] Flavell J.H. (1977). Cognitive development, Englewood Cliffs, N.J. : Prentice-Hall Inc, 1977.

[17] Schraw G. & Moshman D. (1995). Metacognitive Theories, Educational Psychology Review, vol. 7., N° 4, pp. 351-371.

[18] Mollo, V. (2002) La construction des procédures par la pratique : le rôle des outils ergonomiques. Dans Actes du XXXVIIIème Congrès de la SELF, "Les évolutions de la prescription" Aix-en-Provence, 25-26-27 septembre, pp. 201-208.

[19] Mille, A. (2005). Raisonner à partir de l'expérience tracée (RàPET). Définition, illustration et résonances avec le « storytelling ». Synthèse des travaux de l'équipe « Cognition, Expérience et Agents situés » du laboratoire LIRIS UMR CNRS 5205.

[20] Kolodner, J. L. (1993). Case-Based Reasoning, MorganKaufman, San Mateo, CA.

[21] Calcul des indicateurs collaboratifs à partir des transformations spécialisées dans un SBT, Tarek DJOUAD, Alain MILLE (LIRIS, Lyon), Christophe EFFAY(LIFC,Besançon), Mohamed BENMOHAMED (LIRE, Algérie).

[22] Dimitrakopoulou A. Bruillard E. (2006). Enrichir les interfaces de forums par la visualisa-tion d'analyses automatiques des interactions et du contenu. *STICEF.*

[23] OSSIR Groupe de travail blanc sur les logs

[24] thème : mesure d'audience d'un site web. Mémoire de fin d'étude « ingénieur », université Ahmed Bouguera Boumerdes Algérie.

[25] Le rapport final du TPE Classification de documents à l'aide de cartes auto-organisatrices (SOM) basée sur une ontologie

[26**]** Thèse Spécialité Informatique Présentée devant L'Institut National des Sciences Appliquées de Lyon Pour obtenir Le grade de docteur Formation doctorale : Documents multimedia, Images et Systèmes d'Information Communicants (DISIC) École doctorale : Ecole Doctorale Informatique et Information pour la Société (EDIIS) Par Catherine Roussey (Ingénieur) Une méthode d'indexation sémantique adaptée aux corpus multilingues Soutenue le 10 décembre 2001.

[27] THESE Présentée devant L'Université Paul Sabatier de Toulouse en vue de l'obtention du Doctorat de l'Université Paul Sabatier Spécialité Informatique Par Nathalie HERNANDEZ ONTOLOGIES DE DOMAINE POUR LA MODELISATION DU CONTEXTE EN RECHERCHE D'INFORMATION.

[28] J. Euzenat, Eight questions about semantic Web annotations, IEEE Intelligent systems 17(2), pp 55-62, 2002.

[29] [Denoue 2000] Denoue, L. De la création à la capitalisation des annotations dans un espace personnel d'informations. Informatique. Annecy, Université de Savoie.p. 159. Sous la direction de L. Vignollet. 2000.

[30] Marshall, C. "Toward an ecology of hypertext annotation." *Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space---*

*structure in hypermedia systems: links, objects, time and space---structure in hypermedia systems.*ACM Press New York, NY, USA. pp.40-49. 1998.

[31] Miller, T. H. Improving graduate education through digital library tools. Computer Science and Computer Engineering. Blacksburg, Virginia, the Faculty of the Virginia Polytechnic Institute and State University.p. 53. Sous la direction de E.A.Fox. 1999.

[32] Marshall, C. "Annotation: from paper books to the digital library." Proceedings of the second ACM international conference on Digital libraries.ACM Press New York, NY, USA. pp.131-140. 1997.

[33] Marshall, C. C. The Haunting Question of Intelligibility. First International Workshop on Spatial Hypertext, Aaruhs, Denmark. August 14, 2001 2001.

[34] O'Hara, K. et Sellen, A. "A Comparison of Reading Paper and On-Line Documents." Proceedings of the SIGCHI conference on Human factors in computingsystems.ACM Press New York, NY, USA. pp.335-342. 1997. *[35]* Commanditaire : Miled Rousset, Responsable technique : Yannick Prié, Responsable communication : Stéphanie Pouchot , Livre blanc (13/05/2012) *Indexation de corpus spécialisés avec ou sans référentiels.*

[36] H.M. Haav, T.L. Lubi, A Survey of Concept-based Information Retrieval Tools on the Web, In Proceedings of the 5th East-European Conference ADBIS, Vol 2, pp 29-41, 2001.

[37] N. Aussenac-Gilles, J. Mothe, Ontologies as Background Knowledge to Explore Document Collections, In Actes de la Conférence sur la Recherche d'Information Assistée par Ordinateur (RIAO), pp 129-142, 2004.

[38] R.V. Guha, R. McCool, E. Miller, Semantic search, In Proceedings of the 12th International World Wide Web Conference, pp 700-709, 2003.

[39] A. Kiryakov, B. Popov, I. Terziev, D. Manov, D. Ognyanoff, Semantic annotation, indexing, and retrieval, Journal of Web Semantics, 2(1), 2004.

[40] R. Mihalcea, D.I. Moldovan, Semantic Indexing using WordNet Senses, In Proceedings of ACL Workshop on IR & NLP, 2000

[41] theseBousbiaNabila, analyse des traces de navigation des apprenants dans un environnement de formation dans une perspective de détection automatique des styles d'apprentissage: 10/01/2011

[42] CHAUMIER, J., DEJEAN, M. L'indexation Documentaire, de l'analyse conceptuelle humaine à l'analyse automatique morphosyntaxique. In *Documentaliste*, novembre décembre 1990, Vol. 27, N°6. p 275-279.

[43] FLUHR C. Multilingual Information Retrieval. [On-line] In *Survey of the State of the Art in Human Language Technology*, Edited by Cole R. A., Mariani J., Uszkoreit H., Zaenen A., and Zue V. Center for Spoken Langage Understanding, Oregon : Graduate Institute, 1996. p305-391 disponible sur internet : <URL : http://www.cse.ogi.edu/CSLU/HLTsurvey/ch8node7.html.>

[44] Une extension de mesure de similarité entre les concepts d'une ontologie Thabet Slimani*, Boutheina Ben Yaghlane** & Khaled Mellouli* *LARODEC, ISG de tunis, 41 Bouchoucha, Bardo. Tunisia **IHEC Carthage, Carthage Présidence 2016. Tunisia.

[45] Indexation de documents pédagogiques : fusionné les approches du Web Sémantique et du Web Participatif THESE présentée et soutenue publiquement le 29 Octobre 2009 pour l'obtention du Doctorat de l'université Henri Poincaré - Nancy 1 (Spécialité informatique) Par Benjamin Huynh-Kim-Bang.

[46] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. Scienti_c American Magazine. 23, 187.

[47] Réalisation de l'interopérabilité sémantique des systèmes, basée sur les ontologies et les flux d'information : THÈSE pour obtenir le grade de DOCTEUR DE L'UNIVERSITÉ DE SAVOIE Discipline : Informatique, *présentée et soutenue publiquement par* : Naçima MELLAL *la Date de soutenance : 19 Décembre 2007 à Polytech'Savoie.*

[48] Christophe ROCHE. Terminologie et ontologie. *Revue Langages, numéro 157*,Mars 2005.

[49] *Dictionnaire de l'Académie française, neuvième édition*. Journal officiel, Paris, 2005.

[50] Thomas GRUBER. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[51] John SOWA. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks/Cole, August 2000.

[52] Nicola GUARINO et Pierdaniele GIARETTA. Ontologies and knowledge bases: Towards a terminological clarification. In N MARS, réd., *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 25–32. IOS Press, 1995.

[53] Mike USCHOLD et Michael GRÜNINGER. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[54] Frédéric FÜRST. L'ingénierie ontologique. Rapport technique, Institut de recherche en Informatique de Nantes, 2002.

[55] MottaENRIC, BUCKINGHAM et DOMINGUE. Ontology-driven document enrichment : principles, tools and applications. 52(6):1071–1109, June 2000.

[56] KLEIN et LOEBBECKE. The transformation of pricing models on the web: examples from the airline industry,. In *13th International Bled Electronic Commerce Conference*, pages 19–21, June 2000.

[57] Sean LUKE, Lee SPECTOR, David RAGER et James HANDLER. Ontology-based web agents. In W. Lewis JOHNSON et Barbara HAYES-ROTH, réds., *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 59–68, Marina del Rey, CA, USA, 1997. ACM Press.

[58] Valéry PSYCHÉ, Olavo MENDES et Jacqueline BOURDEAU. Apport de l'ingénierie ontologique aux environnements de formation à distance. *Revue STICEF*, 10, 2003.

[59] Riichiro MIZOGUCHI et Jacqueline BOURDEAU. Using ontological engineering to overcome common ai-ed problems. *IJAIED*, 2000.

[60] PSYCHE V., MENDES O., BOURDEAU J. (2003) Apport de l'ingénierie ontologique aux environnements de formation à distance, in *Sciences et Technologies de l'Information et de la* n*Communication pour l'Education et la Formation* (STICEF), Numéro spécial : Technologies et Formation à distance, Vol 10.

[61] Conception d'une ontologie pour une plate forme d'enseignement à distance *par* Saloua & Amina Chettibi & Rouibah université de jijel - ingénieur informatique 2005.

[62] Benayache Ahcene (2005) Construction d'une mémoire organisationnelle de formation et évaluation dans un contexte e-learning *: le projet MEMORAe.*

[63] Ouafia Ghebghoub, M .Djoudi, M,Ibn mouhammed (Congrès AIPU, Monastir Tunisie du 15 au 18 Mai 2006) Ontologies et web sémantique en enseignement à distance.

[64] Jean-Yves FORTIER (Octobre 2001) Construction d'une ontologie pour gérer la documentation d'un centre de recherche.

[65] Sabri Boutemedjet (2004) Web Sémantique et e-Learning.

[66] Sticef.org (2003) Apport de l'ingénierie ontologique aux environnements de formation à distance.

[67] Cyrille Desmoulins, Monique Grandbastien (2000) .Des ontologies pour indexer des documents techniques pour la formation professionnelle.

[68] David FLATER. Sumo2loom documentation.

[69] (Bousbia et Labat, 2007) Bousbia, N., et Labat, J.M. (2007). « Perception de l'activité de l'apprenant dans un environnement de formation ». Environnements Informatiques pour l'Apprentissage Humain, EIAH 2007, Lausanne, Suisse, In INRP, pp. 233-238.

[70] (Fisher, 2005). L'utilisation des traces numériques dans l'enseignement à distance. Rapport de recherche bibliographique.

[71] (Broisin et Vidal, 2007) Broisin, J., et Vidal, P. (2007). Une approche conduite par les modèles pour le traçage des activités des utilisateurs dans des EIAH hétérogènes, *Revue STICEF*, Volume 14.

[72] (Fansler et Riegel, 2004) Fansler, K., et Riegel, R. (2004). « A Model of Online Instructional Design Analytics ». 20th Annual Conference on Distance Teaching and Learning, Madison, Etats-Unis.

[73] (Beauvisage, 2004). Beauvisage, T. (2004). Sémantique des parcours des utilisateurs sur le Web. Thèse de doctorat, Université de Paris X : 361p.

[74] (Zhu et al., 2003) Zhu, T., Greiner, R., et Haeubl, G. (2003). « Learning a model of a web user's interests ». The 9th International Conference on User Modeling (UM'03).

[75] (Loghin, 2008) Loghin, G.-C. (2008). Observer un Environnement Numérique de Travail pour réguler les activités qui s‟y déroulent. Thèse de Doctorat. Cotutelle entre l'Université de Savoie et l'Université Technique de Cluj-Napoca.

[76] Michel, C., Prié, Y., Le Graet, L. (2005). « Construction d‟une base de connaissance pour l‟évaluation de l‟usage d‟un environnement STIC ». 17eme Conférence Internationale Francophone sur l‟Interaction Homme-Machine, Toulouse, France, pp. 199-202.

[77] *Indexation de corpus spécialisés avec ou sans référentiels.*
*Réalisé par***:** HADJEB Sonia, BEDOK Kévin, BERMOND Ameline, FARGE Yohan, RAZAFIMANANTSOA Princy, SIBEUD Nadège.
Livre blanc (13/05/2012).

[78] Gennari, J., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubezy, M., Eriksson, H., Noy, N. F., et Tu, S. W. (2003). The evolution of prot ?g ? : An environment for knowledge-based systems development. International Journal of Human- Computer Studies, 58 :1:89_123.

[79] Minsky, M. (1975). A framework for representing knowledge. The Psychology of Computer Vision, Winston, Patrick, ed. New York : Mc Graw Hill, pages 211_77.

[80] Chaudhri, V. K., Farquhar, A., Fikes, R., Karp, P. D. et Rice, J. (1998). Okbc : A programmatic foundation for knowledge base interoperability. pages 600_607.

[81] Haarslev, V. et Möller, R. (2001). RACER system description. Lecture Notes in Computer Science, 2083:701_ ? ?

[82] Sirin, E. et Parsia, B. (2004). Pellet : An owl dl reasoner.

Sources pour annexe3 (jdom):

* http://www-106.ibm.com/developerworks/fr_fr/Java/library/j-jdom/
* "Processing XML with Java", Elliotte Rusty Harold, chapitre 14 (JDOM), chapitre 15 (JDOM Model) disponible sous www.cafeconleche.org/books/XMLjava/
* javadoc de JDOM v1.0, (www.jdom.org/docs/apidocs/)

Sources pour annexe4 (SAX)

* www.commentcamarche.net/XML/XMLdomsax.php3 et
* smeric.developpez.com/Java/cours/XML/sax/