Institut Francophone International

Année académique 2016-2017

MODULE : WEB SEMANTIQUE

RAPPORT PROJET :

Ontologie d'un réseau social en milieu universitaire

PRESENTER PAR :

KAFANDO Rodriqueuu

KOUADIO Kouamé Olivier

MILORME Pierre Rubens

Étudiants en master 1 – promotion 21

Enseignant : **M. Ta Tuan Anh**

Hanoï, Novembre 2017

SOMMAIRE

INTRODUCTION

Ce rapport présente les travaux que nous avons réalisés au cours de notre projet. En effet dans le cadre du module de web sémantique, nous avons eu pour tâche de réaliser une ontologie d'un réseau social. L'objectif de ce travail est de construire une ontologie en langage XML afin d'illustrer les différents types d'interactions qui puissent exister entre les entités en milieu scolaire ou universitaire. Nous présentons dans ce rapport toutes les étapes de conception que nous avons abordés pour la réalisation de ce projet. Elles partent de la définition des notions de l'ontologie à la conception de cette ontologie sur l'outil protégé.

CHAPITRE 1 : NOTIONS SUR LES ONTOLOGIES

I-Définition

Terme d'origine philosophique selon la littérature, ontologie désigne la théorie d'étude de la « nature de l'existant ». Cette notion a été reprise par les chercheurs dans le domaine de l'intelligence artificielle et utilisée dans le cadre de construction des systèmes à base de connaissances. L'idée était de séparer, d'un côté, la modélisation des connaissances d'un domaine, et d'un autre côté, l'utilisation de ces connaissances (i.e. le raisonnement). Dans ce contexte, plusieurs définitions des ontologies ont été proposées. La première a été proposée dans [2] « Une ontologie définit les termes et les relations de base du vocabulaire d'un domaine ainsi que les règles qui permettent de combiner les termes et les relations afin de pouvoir étendre le vocabulaire ». Cette définition descriptive donne un premier aperçu sur la manière de construire une ontologie, à savoir l'identification des termes et des relations d'un domaine ainsi que les règles pouvant s'appliquer sur ces derniers. Deux années plus tard, Gruber, dans [1] donne la définition qui est devenue la plus utilisée dans la littérature : « Une ontologie est une spécification explicite d'une conceptualisation ». La conceptualisation se réfère ici à l'élaboration d'un modèle abstrait d'un domaine du monde réel en identifiant et en classant les concepts pertinents décrivant ce domaine. La formalisation consiste à rendre cette conceptualisation exploitable par des machines. Dans cette même logique une autre définition dans [3] a été proposé qui stipule qu'« une ontologie est une théorie logique proposant une vue explicite et partielle d'une conceptualisation » . Depuis, de nombreuses définitions, à la fois complémentaires et précises, ont vu le jour. Dans [4] les

auteurs soulignent la dépendance entre la formalisation de l'ontologie et l'application dans laquelle elle va être utilisée : Une ontologie organisée dans un réseau des concepts représentant un domaine. Son contenu et son degré de formalisation sont choisis en fonction d'une application ». Nous retiendrons donc qu'une ontologie traduit un consensus explicite sur la formalisation des connaissances d'un domaine afin de faciliter le partage et la réutilisation de ces connaissances par les membres d'une communauté ou par des agents logiciels. C'est dans cette optique que les ontologies se présentent comme un pilier du web sémantique, car elles permettent de faire communiquer les hommes et les machines en utilisant la sémantique partagée par les différents acteurs du web et en décrivant ses ressources.

II- Approche opérationnelle

Parallèlement à ces précédentes définitions de ce que représente une ontologie, une autre définition, plus opérationnelle, peut être formulée ainsi :
Une ontologie est un réseau sémantique qui regroupe un ensemble de concept décrivant complétement un domaine. Ces concepts sont liés les uns aux autres par des relations taxonomiques (hiérarchisation des concepts) d'une part, et sémantiques d'autre part. Cette définition rend possible l'écriture de langages destinés à implémenter des ontologies. Pour construire une ontologie, on dispose d'au moins trois de ces notions :
- détermination des agents passifs ou actifs.
- leurs conditions fonctionnelles et contextuelles.
- leurs transformations possibles vers des objectifs limités.

Pour modéliser une ontologie, on utilisera ces outils (mécanismes) :
- Raffiner les vocabulaires et notions adjacentes.
- Décomposer en catégories et autres sujets.
- Prédiquer afin de connaître les transformations adjacentes et d'orienter vers les objectifs internes.
- Relativiser afin d'englober des concepts.
- Similiser afin de réduire à des bases totalement distinctes.
- Instancier afin de reproduire l'ensemble d'une « branche » vers une autre ontologie.

III- Les composants d'une ontologie et Utilisation des ontologies.

III.1- Les composants d'une ontologie

Une ontologie peut être vue comme un ensemble structuré de concepts et de relations entre ces concepts destinés à représenter les objets du monde sous une forme compréhensible aussi bien par les hommes que par les machines. Les composants d'une ontologie sont :

- **Concept** : ou classe, définissant un ensemble d'objet, abstrait ou concret, que l'on souhaite modéliser pour un domaine donné. Les connaissances portent sur des objets auxquels on se réfère à travers des concepts. Un concept peut représenter un objet matériel, une notion, une idée. Les concepts dans l'ontologie sont habituellement organisés dans des taxonomies.
- **Les instances** : ou individus, constituent la définition extensionnelle de l'ontologie (pour représenter les éléments spécifiques) les relations :
- **Une relation** : permet de lier des instances de concepts ou des concepts génériques. Elles sont caractérisées par un terme ou plusieurs, et une signature qui précise le nombre d'instances de concepts que la relation lie, leurs types et l'ordre des concepts, c'est – à – dire la façon dont la relation doit être lue.
- **Les axiomes** : Une ontologie est en outre composée d'axiomes qui forment des contraintes sémantiques pour le raisonnement et donnent un acompte d'une conceptualisation. Ils prennent la forme d'une théorie logique.

III.2- Utilisation des ontologies

Même si le besoin de développer une ontologie est très varié et dépend du domaine d'application, nous pouvons facilement énumérer un certain nombre d'utilités, notamment :

- **La connaissance du domaine** : Les ontologies permettent la modélisation des connaissances dans un domaine particulier, dans lequel opère le système à développer.
- **La communication** : les ontologies assurent une communication fiable et hétérogène entre personnes et machines (agents logiciels ou organisations) du fait qu'elle permet de mettre en place un langage ou un vocabulaire conceptuel commun.
- **L'interopérabilité** : La représentation explicite des connaissances dans un domaine donné sous forme d'une ontologie, permet à son tour une

plus grande réutilisation, un partage plus large et une interopérabilité plus étendue.

- **L'aide à la spécification des systèmes** : La représentation conceptuelle des éléments du domaine, permet aux systèmes de réaliser des raisonnements logiques qu'on appelle inférences, et de sortir avec des conclusions capables d'aider l'utilisateur ou le gestionnaire dans ses décisions.
- **L'indexation et la recherche d'information** : Dans le web sémantique, d'une façon générale, et dans notre application en particulier, les ontologies sont utilisées pour indexer et décrire les ressources utilisées. Cela permet une plus grande précision dans les résultats des recherches ou d'assignation des ressources.

IV- Les types et avantages d'ontologie

IV.1- Les types d'ontologie

Nous listons ci-dessous les différents types d'ontologies les plus utilisées :

- **Les ontologies de représentation** : n'appartiennent à aucun domaine, mais définissent et organisent les primitives de la théorie logique pour permettre la représentation des ontologies. Qui définit d'une manière formelle, les primitives de représentation (classes, sous classes, attributs, valeurs, relations et axiomes) dans un environnement implémentant les langages de Frame.
- **Les ontologies génériques** : sont aussi appelée Ontologie de haut niveau ou ontologie Top, elles décrivent des concepts généraux, indépendants d'un domaine ou d'un problème particulier. Elles permettent par exemple de formaliser les aspects temporels ou spatiaux des objets du monde réel.
- **Les ontologies de domaine** : elles sont construites sur un domaine particulier de la connaissance. Les ontologies de domaine fournissent des vocabulaires au sujet des concepts dans un domaine et leurs relations au sujet des activités qui ont lieu dans ce domaine, et au sujet des théories et des principes élémentaires régissant ce domaine.
- **L'ontologie de tâche** : décrit les connaissances portant sur tâches et/ou des activités particulières. Ces ontologies fournissent un ensemble de termes au moyen desquels on peut décrire au niveau générique comment résoudre un type de problème. Elles incluent des noms génériques (objectif, contrainte…), des verbes génériques (classer, sélectionner), des

adjectifs génériques (assigné) et autres dans les descriptions de tâches.

- **Ontologies d'application** : aussi appelée ontologie de domaine-tâche : Ce sont les ontologies les plus spécifiques, elles contiennent les connaissances requises pour une application particulière permettant ainsi de modéliser une activité spécifique dans un domaine donné.

IV.2- Les avantages d'une ontologie

Parmi les avantages de la méthode des ontologies la compréhension commune de la structure de l'information entre les personnes ou les fabricants de logiciels, assurer l'interopérabilité entre systèmes, permettre l'échange de connaissances entre Systèmes, Permettre la réutilisation du savoir sur un domaine: créer et conserver des bases de connaissances réutilisables, Expliciter ce qui est considéré comme implicite sur un domaine, Distinguer le savoir sur un domaine du savoir opérationnel et Analyser le savoir sur un domaine.

V- Les étapes de conceptualisation et critères d'évaluation d'une ontologie

❖ Les étapes de conceptualisation

La conceptualisation consiste à identifier et à structurer, à partir des sources d'informations, les connaissances du domaine. La découverte de ces connaissances peut s'appuyer à la fois sur l'analyse de documents et sur l'interview d'experts du domaine. Dans cette étape, les termes de l'ontologie sont choisis, ainsi que leurs propriétés, les relations qu'ils entretiennent entre eux, les contraintes qui s'appliquent sur eux, etc. Pour cela on distingue les principales activités suivantes :

1. **Construction d'un glossaire de termes** : La première chose à faire quand le constructeur de l'ontologie essaye de capturer la connaissance d'un domaine donné est de construire un glossaire de terme. Le glossaire recueille et décrit tous les termes qui sont utiles et potentiellement utilisables dans le domaine que l'on investit. Pour chaque terme, il faut identifier sa description et sa référence (source d'information), sans s'inquiéter à propos de son type (concept, relation ou propriété).

2. **Classifier les concepts dans des hiérarchies de concepts** : Une hiérarchie de concepts organise un groupe de concepts entre eux sous forme d'une taxonomie (basée sur la relation Is-a). Sa construction peut se faire selon une approche ascendante (commencer par les concepts les plus détaillés), descendante (commencer par les concepts les plus généraux), ou combinée (Combinaison des deux approches précédentes et utilisation d'un concept de

niveau milieu). Il faut vérifier que tous les concepts qui sont représentés dans les différentes hiérarchies de concepts, sont dans le glossaire. Dans le cas contraire ces derniers devraient être rajoutés au glossaire.

3. **Représenter les relations qui existent entre les différents concepts par un diagramme de relations binaires** : la relation binaire lie un concept source à un concept cible, qu'ils appartiennent ou non à la même hiérarchie. Il faut vérifier que chaque nom d'un concept source représenté dans le diagramme doit être défini dans le glossaire, sinon il doit être rajouté.

4. **Identifier les concepts par leurs instances, leurs attributs ainsi que leurs concepts synonymes dans un dictionnaire de concepts (DC)**.

5. **Décrire les relations dans une table de relations binaires** : La table des relations binaires définie pour chaque relation utilisée dans le diagramme, le nom de la relation, le nom des concepts sources et cibles, le nom de la relation inverse et les cardinalité source et cible.

6. **Spécifier des contraintes sur les attributs dans une table d'attributs** : Pour chaque attribut d'un concept inclut dans le DC, la table des attributs spécifie le nom de cet attribut, son concept, sa description, son type.

7. **Spécifier des axiomes sur les concepts dans une table d'axiomes logiques** : un axiome permet de définir certains concepts au moyen des expressions logiques. Sa description doit comporter le nom du concept sur lequel il porte, une définition en langage naturel et son expression logique.

8. **Décrire les instances des concepts dans une table d'instances** : La table des instances décrit toutes les instances (incluses dans le champ instances de dictionnaire de concepts) avec leurs attributs et valeurs.

❖ Critères d'évaluation d'une ontologie

L'objectif premier d'une ontologie est de modéliser un ensemble de connaissances dans un domaine donné, qui peut être réel ou imaginaire. Les ontologies sont employées dans l'intelligence artificielle, le Web sémantique, le génie logiciel, l'informatique biomédicale ou encore l'architecture de l'information comme une forme de représentation de la connaissance au sujet d'un monde ou d'une certaine partie de ce monde. D'après Gruber dans [1], cinq critères permettent de mettre en évidence des aspects importants d'une ontologie :

• **La clarté** : la définition d'un concept doit faire passer le sens voulu du terme, de manière aussi objective que possible (indépendante du contexte). Une définition doit de plus être complète (c'est- dire définie par des conditions à la

fois nécessaires et suffisantes) et documentée en langage naturel.

• **La cohérence** : rien qui ne puisse être inféré de l'ontologie ne doit entrer en contradiction avec les définitions des concepts (y compris celles qui sont exprimées en langage naturel).

• **L'extensibilité :** les extensions qui pourront être ajoutées à l'ontologie doivent être anticipées. Il doit être possible d'ajouter de nouveaux concepts sans avoir à toucher aux fondations de l'ontologie.

• **Une déformation d'encodage minimale** : une déformation d'encodage a lieu lorsque la spécification influe la conceptualisation (un concept donné peut être plus simple à définir d'une certaine façon pour un langage d'ontologie donné, bien que cette définition ne corresponde pas exactement au sens initial). Ces déformations doivent être évitées autant que possible.

• **Un engagement ontologique minimal** : le but d'une ontologie est de définir un vocabulaire pour décrire un domaine, si possible de manière complète ; ni plus, ni moins. Contrairement aux bases de connaissances par exemple, on n'attend pas d'une ontologie qu'elle soit en mesure de fournir systématiquement une réponse à une question arbitraire sur le domaine. Toujours selon Gruber [1], l'engagement ontologique peut être minimisé en spécifiant la théorie la plus faible (celle permettant le plus de modèles) couvrant un domaine ; elle ne définit que les termes nécessaires pour partager les connaissances consistantes avec cette théorie.

CHAPITRE 2 : LANGAGE ET OUTILS DE CONCEPTION DES ONTOLOGIES

I- LES LANGAGES DES ONTOLOGIES

Il existe plusieurs langages pour la définition des ontologies, La plupart de ces langages se basent ou sont proches sur la logique du premier ordre, et représentent donc les connaissances sous forme d'assertion (sujet, prédicat, objet). Ces langages sont typiquement conçus pour s'abstraire des structures de données et se concentrer sur la sémantique. Ce sont :

- **RDF** (Resource Description Framework)
- **RDFS** (RDF Schema)
- **OWL** (Web Ontology Language)
- **XML** (eXtensible Markup Language)
- **SPARQL**(***S****PARQL* ***P****rotocol* ***a****nd* ***R****DF* ***Q****uery* ***L****anguage)*

Dans le cadre de ce travail notre intérêt s'est porté sur le langage XML.

❖ XML

Le langage eXtensible Markup Language (XML) est un méta-langage permettant de représenter un document texte de manière arborescente en utilisant un système de balisage. XML fournit ainsi un cadre de structuration des données qui peut être employé pour la définition rapide et sans ambiguïté syntaxique d'un format de document. Ce langage a été élaboré pour faciliter l'échange, le partage et la publication des données à travers le web. Ainsi, la majorité des langages/modèles proposés pour le web sémantique sont exprimés en XML. XML permet de structurer un document en définissant ses propres balises en fonction des besoins et sans tenir compte ni de la signification de cette structure ni des systèmes informatiques qui vont l'exploiter. Des standards comme XPath et XQuery ont été développés afin de parcourir et d'interroger l'arborescence XML des documents.

❖ Les composants d'un schema XML

Un schéma XML comporte des composants suivants :

- ✓ **Éléments** : les éléments sont le bloc de construction principal de tout document XML, ils contiennent les données et déterminent la structure du document. Un élément peut être défini dans un schéma XML (XSD) comme suit : `<xs:element name="x" type="y"/>`
- ✓ **Cardinalité** : la spécification du nombre de fois qu'un élément peut apparaître est appelée cardinalité et est spécifiée à l'aide des attributs minOccurs et maxOccurs. De cette manière, un élément peut être obligatoire, facultatif ou apparaître plusieurs fois. minOccurs peut être affecté d'une valeur entière non négative (par exemple 0, 1, 2, 3 ... etc.), et maxOccurs peut recevoir n'importe quelle valeur entière non négative.
- ✓ **Types :** nous avons deux types dans un schéma xml à savoir : le type simple et le type complexe.
- ✓ **Compositeurs** :
  Il y a 3 types compositeurs <xs:sequence>, <xs:choice>et <xs:all>. Ces compositeurs nous permettent de déterminer la manière dont les éléments enfants qui s'y trouvent apparaissent dans le document XML.
- ✓ **Les attributs :** Un attribut fournit des informations supplémentaires dans un élément. Les Attributs sont définis dans un XSD comme suit, ayant nom et type propriétés.

II- COMPARAISON DES LANGAGES DE DÉFINITION DE SCHÉMA

Caractéristiques	Langages de définition de schéma			
	XML Schema	RDF/S	OWL 1	OWL 2
Type de modèle	Hiérarchique	Graphe direct	Graphe direct	Graphe direct
Syntaxe concrète	XML	RDF/XML, N3, N-Triples, Turtle	RDF/XML, N3, N-Triples, Turtle	OWL/XML, Functional, Manchester, RDF/XML, N3, N-Triples, Turt
Constructions de base	Simple type, Complex type, Attribute, Element, Attribute group, Sequence Choice Annotation Extension, Restriction, Unique, Key, Keyref, Substitution Group, Alternative+ , Assert+ , Override+ , Redefine+ , etc	Statement, Class, Property, Resource, type, subject, predicate, object, subClassOf, subPropertyOf, domain, range, Datatype, Literal, Bag, Seq, List, Alt, BlankNode	Class, Datatype Property, Object Property, Individual, Thing, Nothing, equivalentClass, intersectionOf, unionOf, complementOf, disjointWith, minCardinality, sameAs, oneof, hasValue, TransitiveProperty, FunctionalProperty, etc	InverseOf, NegativePropertyAssert ion, propertyChainAxiom, minQualifiedCardinality , hasSelf, AsymmetricProperty, AllDisjointClasses, disjoint, ReflexiveProperty, maxQualifiedCardinalit y, UnionOf, etc.
Sémantique	Informel	Model Theory	Model Theory, RDF Graphs	Model Theory, Extension of SROIQ DL
Identité Contraintes	Unique, Key, Keyre	-	-	hasKey
Type de données	minInclusive, minExclusive, maxLength, length, totalDigits, etc.	-	-	xsd:minInclusive, xsd:minExclusive, owl:onDatatype, owl:withRestrictions, etc.

III- LES OUTILS

Dans cette partie nous énumérons quelques outils pour le web sémantique.

❖ Annotea

Annotea est un projet du W3C pour faire de l'annotation web. Les annotations sont des commentaires, des notes, des explications ou d'autres types de remarques externes qui peuvent être associés à tout document web ou une partie sans que cela ne soit nécessaire pour modifier le document. Une fois qu'on a un document on peut charger les annotations le concernant depuis un serveur afin de les exploiter. On utilise un schéma basé sur RDF pour décrire les annotations comme des méta-données. Xpointer est utilisé pour localiser des annotations dans un document annoté. Les annotations sont stockées dans des serveurs qui peuvent être interrogés. IL existe des clients d'annotation (editor/browser) comme Amaya.

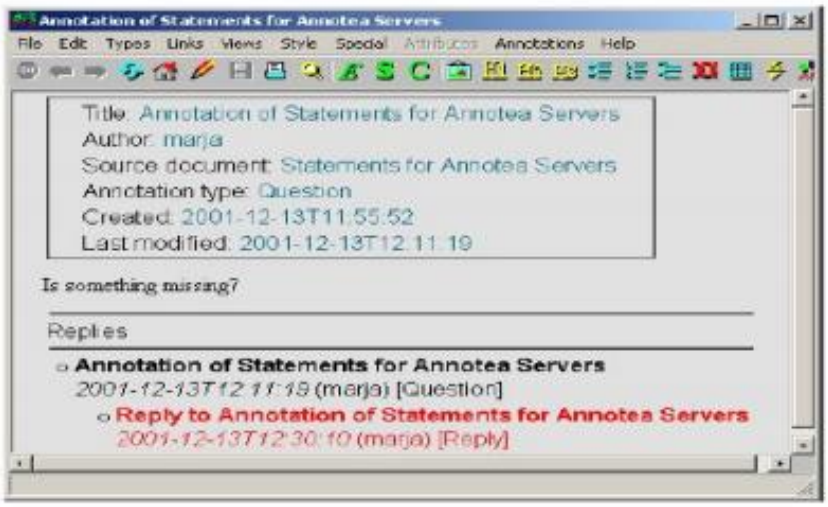

Figure 2 : l'éditeur d'annotation Annotea

❖ Jena

Jena est un framework en Java qui permet de construire des applications Web Sémantique. Il fournit un environnement de programmation pour RDF, RDFS et OWL (voir la Figure). Il inclut également un moteur de règles d'inférence. Le framework contient : Un api RDF. Un api pour la lecture et l'écriture de RDF en RDF/XML, N3, et N-Triples. Un api OWL. Un système de persistance. RDQL, un langage de requête pour RDF.

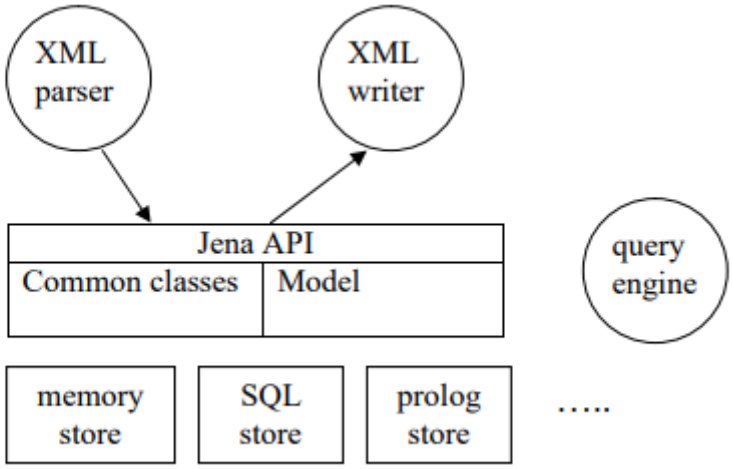

Figure 3 : Architecture de Jena

❖ L'Éditeur d'Ontologies Protégé

Protégé est un éditeur d'ontologies distribué en open source par l'université en informatique médicale de Stanford. Il permet de construire une ontologie pour un domaine donné, de définir des formulaires d'entrée de données, et d'acquérir des données à l'aide de ces formulaires sous forme d'instances de cette ontologie. Protégé est aussi une plate-forme extensible, grâce au système de plug-ins, qui permet de gérer des contenus multimédias, interroger, évaluer et fusionner des ontologies, etc. Protégé possède une interface utilisateur graphique (GUI) lui permettant de manipuler aisément tous les éléments d'une ontologie : classe, propriété, instance, etc. Protégé peut être utilisé dans n'importe quel domaine où les concepts peuvent être modélisés en une hiérarchie des classes. Protégé permet aussi de créer ou d'importer des ontologies écrites dans les différents langages d'ontologies tel que : RDF-Schema, OWL, DAML, OIL, …etc. Cela est rendu possible grâce à l'utilisation de plugins qui sont disponibles en téléchargement pour la plupart de ces langages.

CHAPITRE 3 : CAS PRATIQUE : **CONCEPTION D'UN RÉSEAU SOCIAL UNIVERSITAIRE**

I- CONCEPTION

I.2- Présentation du projet

L’objectif de ce travail est de construire une ontologie en langage XML afin d’illustrer les différents types d’interactions qui puissent exister entre les entités en milieu scolaire ou universitaire.

❖ Le diagramme de l'ontologie de notre réseau social

Le diagramme ci-après permet de voir de façon représentative, les relations qui existent entre différentes classes et leurs propriétés.

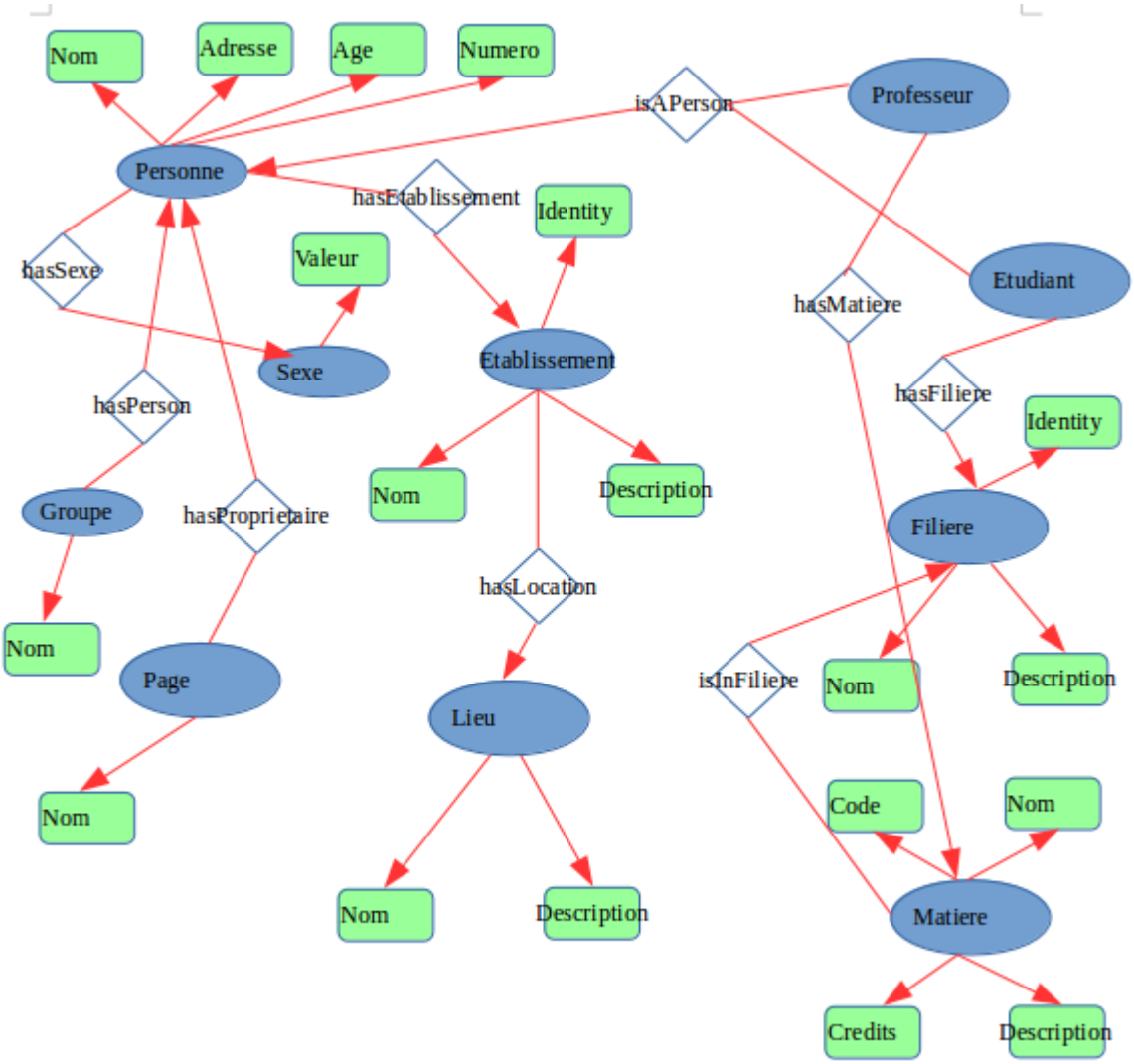

Figure 4 : Diagramme représentative

I.2- Description du schéma XSD généré pour notre ontologie .

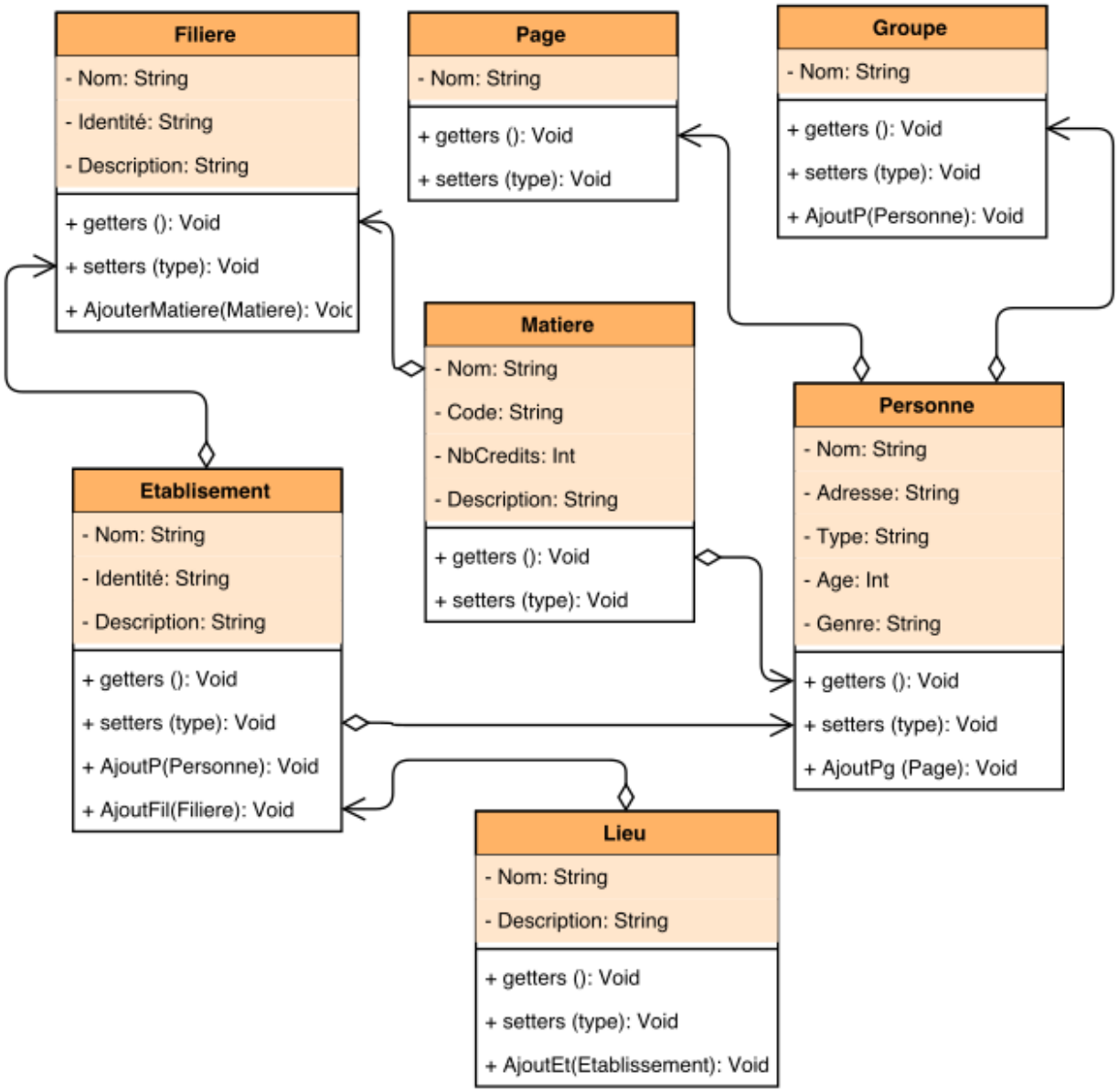

Figure 5 : **Diagramme de classes UML du réseau social**

Explication et Definition des elements, des types et des attributs

- type

```
<xs:simpleType name="type">
    <xs:restriction base="xs:string">
        <xs:enumeration value="etudiant"></xs:enumeration>
        <xs:enumeration value="proffesseur"></xs:enumeration>
        <xs:enumeration value="surveillant"></xs:enumeration>
        <xs:enumeration value="directeur"></xs:enumeration>
        <xs:enumeration value="secretaire"></xs:enumeration>
```

```
                    <xs:enumeration value="comptable"></xs:enumeration>
                </xs:restriction>
            </xs:simpleType>
```

- etablissementType

```
            <xs:complexType name="etablissementType">
                <xs:sequence>
                    <xs:element name="nom" type="xs:string" />
                    <xs:element name="identite" type="xs:integer" />
                    <xs:element name="description" type="xs:string" />
                </xs:sequence>
            </xs:complexType>
```

- filiereType

```
            <xs:complexType name="filiereType">
                <xs:sequence>
                    <xs:element name="nom" type="xs:string" />
                    <xs:element name="identite" type="xs:integer" />
                    <xs:element name="description" type="xs:string" />

                    <xs:element name="matiere" type="tns:matiereType"
minOccurs="1" maxOccurs="unbounded" />
                </xs:sequence>
            </xs:complexType>
```

- groupeType

```
            <xs:complexType name="groupeType">
                <xs:sequence>
                    <xs:element name="nom" type="xs:string" />
                    <xs:element name="identifiant" type="xs:integer" />
                    <xs:element name="description" type="xs:string" />
                    <xs:element name="personne" type="tns:personneType"
minOccurs="1"       maxOccurs="unbounded" />
                </xs:sequence>
            </xs:complexType>
```

- lieuType

```
            <xs:complexType name="lieuType">
                <xs:sequence>
                    <xs:element name="nom" type="xs:string" />
                    <xs:element name="description" type="xs:string" />
                </xs:sequence>
            </xs:complexType>
```

- matiereType

```
<xs:complexType name="matiereType">
    <xs:sequence>
        <xs:element name="nom" type="xs:string" />
        <xs:element name="code" type="xs:int" />
        <xs:element name="nbCredits" type="xs:int" />
        <xs:element name="description" type="xs:string" />
    </xs:sequence>
</xs:complexType>
```

- pageType

```
<xs:complexType name="pageType">
    <xs:sequence>
        <xs:element name="nom" type="xs:string" />
    </xs:sequence>
</xs:complexType>
```

- personneType

```
<xs:complexType name="personneType">
    <xs:sequence>
        <xs:element name="nom" type="xs:string" />
        <xs:element name="addresse" type="xs:string" />
        <xs:element name="type" type="xs:string" />
        <xs:element name="age" type="xs:int" />
        <xs:element name="page" type="tns:pageType"
minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>

    <xs:attribute name="genre" type="tns:genre" />
    <xs:attribute name="type" type="tns:type" />
</xs:complexType>
```

Attributs

- genre

```
<xs:simpleType name="genre">
    <xs:restriction base="xs:string">
        <xs:enumeration value="M"></xs:enumeration>
        <xs:enumeration value="F"></xs:enumeration>
    </xs:restriction>
</xs:simpleType>
```

- type

```
<xs:simpleType name="type">
    <xs:restriction base="xs:string">
        <xs:enumeration value="etudiant"></xs:enumeration>
        <xs:enumeration value="proffesseur"></xs:enumeration>
        <xs:enumeration value="surveillant"></xs:enumeration>
        <xs:enumeration value="directeur"></xs:enumeration>
        <xs:enumeration value="secretaire"></xs:enumeration>
        <xs:enumeration value="comptable"></xs:enumeration>
    </xs:restriction>
</xs:simpleType>
```

Elements

- filiere

```
<xs:element name="filiere"  type="tns:filiereType" minOccurs="1"
    maxOccurs="unbounded"/>
```

- page

```
<xs:element name="page" type="tns:pageType" minOccurs="1"
    maxOccurs="unbounded" />
```

- groupe

```
<xs:element name="groupe" type="tns:groupeType" minOccurs="1"
    maxOccurs="unbounded" />
```

- matiere

```
<xs:element name="matiere" type="tns:matiereType" minOccurs="1"
    maxOccurs="unbounded" />
```

- etablissement

```
<xs:element name="etablissement" type="tns:etablissementType" minOccurs="1"
    maxOccurs="unbounded"/>
```

- personne

```
<xs:element name="personne" type="tns:personneType" minOccurs="1"
    maxOccurs="unbounded"/>
```

lieu

```
<xs:element name="lieu" type="tns:lieuType" minOccurs="1"
        maxOccurs="unbounded"/>
```

Element racine

- reseau_social_universitaire

```
<xs:element name="reseau_social_universitaire">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="filiere"  type="tns:filiereType"
minOccurs="1"       maxOccurs="unbounded"/>
                        <xs:element name="page" type="tns:pageType"
minOccurs="1"       maxOccurs="unbounded" />
                        <xs:element name="groupe" type="tns:groupeType"
minOccurs="1"       maxOccurs="unbounded" />
                        <xs:element name="matiere" type="tns:matiereType"
minOccurs="1"       maxOccurs="unbounded" />
                        <xs:element name="etablissement"
type="tns:etablissementType" minOccurs="1"      maxOccurs="unbounded"/>
                <xs:element name="personne" type="tns:personneType"
minOccurs="1"       maxOccurs="unbounded"/>
                        <xs:element name="lieu" type="tns:lieuType"
minOccurs="1"       maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
```

I.3- Présentation du schéma XSD final défini pour l'ontologie de notre réseau social universitaire

Ci-dessous le schéma XSD final défini pour l'ontologie de réseau social universitaire que nous voulons mettre en place.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.org/reseau_social_universitaire"
        xmlns:tns="http://www.example.org/reseau_social_universitaire"
        elementFormDefault="qualified">

        <xs:annotation>
```

```
                <xs:documentation>
```

Ce schema XML decrit une ontologie dans le cadre d'un reaseau social universitaire

```
                </xs:documentation>
            </xs:annotation>

            <!-- Types -->
            <xs:complexType name="filiereType">
                <xs:sequence>
                    <xs:element name="nom" type="xs:string" />
                    <xs:element name="identite" type="xs:integer" />
                    <xs:element name="description" type="xs:string" />

                    <xs:element name="matiere" type="tns:matiereType"
minOccurs="1" maxOccurs="unbounded" />
                </xs:sequence>
            </xs:complexType>


            <xs:complexType name="matiereType">
                <xs:sequence>
                    <xs:element name="nom" type="xs:string" />
                    <xs:element name="code" type="xs:int" />
                    <xs:element name="nbCredits" type="xs:int" />
                    <xs:element name="description" type="xs:string" />
                </xs:sequence>
            </xs:complexType>


            <xs:complexType name="pageType">
                <xs:sequence>
                    <xs:element name="nom" type="xs:string" />
                </xs:sequence>
            </xs:complexType>


            <xs:complexType name="personneType">
                <xs:sequence>
                    <xs:element name="nom" type="xs:string" />
                    <xs:element name="addresse" type="xs:string" />
                    <xs:element name="type" type="xs:string" />
                    <xs:element name="age" type="xs:int" />
                    <xs:element name="page" type="tns:pageType"
minOccurs="1" maxOccurs="unbounded" />
                </xs:sequence>

                <xs:attribute name="genre" type="tns:genre" />
```

```
            <xs:attribute name="type" type="tns:type" />
        </xs:complexType>

        <xs:simpleType name="genre">
            <xs:restriction base="xs:string">
                <xs:enumeration value="M"></xs:enumeration>
                <xs:enumeration value="F"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>

        <xs:simpleType name="type">
            <xs:restriction base="xs:string">
                <xs:enumeration value="etudiant"></xs:enumeration>
                <xs:enumeration value="proffesseur"></xs:enumeration>
                <xs:enumeration value="surveillant"></xs:enumeration>
                <xs:enumeration value="directeur"></xs:enumeration>
                <xs:enumeration value="secretaire"></xs:enumeration>
                <xs:enumeration value="comptable"></xs:enumeration>
            </xs:restriction>
        </xs:simpleType>

        <xs:complexType name="groupeType">
            <xs:sequence>
                <xs:element name="nom" type="xs:string" />
                <xs:element name="identifiant" type="xs:integer" />
                <xs:element name="description" type="xs:string" />
                <xs:element name="personne" type="tns:personneType"
minOccurs="1"       maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>


        <xs:complexType name="etablissementType">
            <xs:sequence>
                <xs:element name="nom" type="xs:string" />
                <xs:element name="identite" type="xs:integer" />
                <xs:element name="description" type="xs:string" />
            </xs:sequence>
        </xs:complexType>

        <xs:complexType name="lieuType">
            <xs:sequence>
                <xs:element name="nom" type="xs:string" />
                <xs:element name="description" type="xs:string" />
            </xs:sequence>
        </xs:complexType>


        <!-- Elements -->
```

```
                <xs:element name="reseau_social_universitaire">
                        <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="filiere"
type="tns:filiereType" minOccurs="1"        maxOccurs="unbounded"/>

                                                <xs:element name="page"
type="tns:pageType" minOccurs="1"        maxOccurs="unbounded" />

                                                <xs:element name="groupe"
type="tns:groupeType" minOccurs="1"      maxOccurs="unbounded" />

                                                <xs:element name="matiere"
type="tns:matiereType" minOccurs="1"     maxOccurs="unbounded" />

                                                <xs:element name="etablissement"
type="tns:etablissementType" minOccurs="1"       maxOccurs="unbounded"/>

                                                <xs:element name="personne"
type="tns:personneType" minOccurs="1"   maxOccurs="unbounded"/>

                                                <xs:element name="lieu" type="tns:lieuType"
minOccurs="1"        maxOccurs="unbounded"/>
                                        </xs:sequence>
                        </xs:complexType>
                </xs:element>

</xs:schema>
```

Exemple de d'instanciation et de génération d'un fichier XML

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:Etablissement xmlns:tns="http://www.example.org/File1"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.example.org/File1 XSDXML.xsd ">
 <tns:personne genre="M" type="Etudiant">
  <tns:name>KAFANDO</tns:name>
  <tns:address>01262130563</tns:address>
  <tns:type>tempsPlein</tns:type>
  <tns:age>25</tns:age>
  <tns:page>
   <tns:name>mystories</tns:name>
  </tns:page>
 </tns:personne>

 <tns:filiere>
  <tns:name>Physique</tns:name>
```

```
   <tns:identite>05</tns:identite>
   <tns:description>Cours de physique</tns:description>
   <tns:Matiere>
    <tns:name>Thermodynaamique</tns:name>
    <tns:code>12</tns:code>
    <tns:nbCredit>5</tns:nbCredit>
    <tns:description>Cours de thermodynamique</tns:description>
   </tns:Matiere>
  </tns:filiere>
</tns:Etablissement>
```

CONCLUSION

Au cours de ce travail nous avons à travers l'état de l'art définit l'ontologie. En nous basant sur les étapes de conceptualisations nous avons modélisé notre réseau social à l'aide de UML. Puis nous avons généré le code source XML de notre conception. Ce projet a été un succès et nous a permis de mettre en pratique nos connaissances théoriques acquises au cours de ce module.

REFRENCES

[1]: Gruber T. (1993): A translation approach to portable ontology specifications. Knowledge Acquisition. 5(2):199–220, 1993.

[2]: Neches R., Fikes R.E., Finin T., Gruber T.R., Patil R., Senator T., et Swartout W. R. (1991): Enabling technology for knowledge sharing. AI Magazine, 12(3), 16- 36.

[3]: Guarino N. and Giaretti P. (1995): Ontologies and knowledge bases: Towards a terminological clarific ation. In Towards Very Large Knowledge Bases. N. J. I. Mars, Ed., IOS Press: 25-32.

[4]: Aussenac-Gilles N., Biébow B., Szulman N. (2000): Revisiting Ontology Design: a method based on corpus analysis. Proc of EKAW'2000. Juan-Les-Pins (F). Oct 2000. Lecture Notes in Artificial Intelligence Vol 1937. Springer

Verlag. pp. 172-188.