Approche pour la génération des Diagrammes UML à partir de l'Univers de Discours et l'ontologie de domaine

Khadir Bekki[1],Ghalem Belalem[2]

[1] Département Informatique, Faculté des Sciences et Sciences de l'Ingénieur, Université de Tiaret
Bekki_kh@Yahoo.fr

[2] Département Informatique, Faculté des Sciences, Université d'Oran (Es-Sénia)
BP 1524, El M'Naouer, Oran, Algérie
Ghalem1dz@Yahoo.fr

Résumé. Aujourd'hui, le langage de modélisation unifié (UML) a été largement accepté par l'industrie et s'est établi comme le langage commun pour l'analyse et la conception en génie logiciel orienté objet. Néanmoins, il souffre de manque d'outils et de syntaxe qui lui permettent de raffiner davantage la sémantique de données à partir de l'univers de discours. Ce raffinement est maîtrisé dans la méthode Niam (Nijssen Information Analysis Methodology), qui se base sur la modélisation des systèmes d'informations à partir des exemples simples du domaine en utilisant une procedure appelée CSDP (Conceptual Schema and relational database Design Procedure). Afin de bénéficier de la puissance de cette procédure pour l'élaboration de diagrammes UML sémantiquement plus expressifs, on propose dans cet article, une adaptation de CSDP à la notation UML. L'utilisation des ontologies dans le développement des systèmes d'informations peut servir divers aspects. Elle permet d'optimiser l`analyse conceptuelle, de minimiser l'intervention de l'expert de domaine, d'enrichir la conception, d'éviter l'ambiguïté dans la spécification et de soutenir l'automatisation du processus de vérification de la fiabilité des systèmes. Notre approche est consolidée par l'utilisation de l'ontologie de domaine, afin de générer des diagrammes UML sémantiquement riches.

Mots clés : CSDP, NIAM, Système d'information, UML, ORM, Ontologie.

1 Introduction

La capture des informations pose un problème dû essentiellement à l'hétérogénéité des acteurs qui vont créer le système. Il fallait donc proposer une méthodologie qui permette une communicabilité importante et des facilités d'utilisation telles que toutes les populations d'intervenants puissent se retrouver face à des mêmes modèles dans un langage universel. Le seul moyen que tous les hommes aient en commun c'est le langage naturel. Toute phrase peut se ramener à la forme sujet-verbe-complement. Cette forme peut être facilement formulée par des prédicats binai-

res. Pour cela, Nijssen [3,13] a mis au point une méthode appelée Niam pour la construction de schémas conceptuels de base de données en utilisant le modèle relationnel binaire. Son principe est d'exprimer ce que l'on veut à l`aide de phrases simples (indécomposables) sans perte d'informations. Cette méthode a prouvé une puissance sémantique dans la capture des données nécessaires à partir du domaine de l'U°D (Univers de Discours). L'un des facteurs de sa puissance est la possession d'une procédure nommé CSDP permettant de passer des rapports de sortie et/ou formulaires d'entrée en un schéma conceptuel [15]. Ce travail a pour objectif de proposer une démarche nommée UDDPO (Uml Diagrams Design Procedure based Ontology) similaire à celle de la procédure CSDP de Niam pour obtenir des modèles orientés objets d'UML (diagramme de classes et diagramme d'objets) [1,2,11,14,16] à partir d'une partie de l'U°D (tableaux, rapports,…), tout en bénéficiant de la puissance de cette procédure dans le raffinement des modèles de structures d'UML.

2 Présentation de la CSDP

L'idée de base de la CSDP [15] est d'exprimer les concepts du schéma conceptuel (SC) en phrases élémentaires du langage naturel. L'approche fondamentale est de construire une conception incrémentale en commençant avec des exemples spécifiques et puis suivre une procédure bien définie utilisant des diagrammes visuels significatifs facilement peuplés pour des buts de validation. On obtient en dernier un schéma conceptuel final de Niam. La CSDP est présentée comme une séquence de neuf étapes:

1. Décomposer des exemples d'informations familiers en idées élémentaires et appliquer des versifications de qualité.
2. Dessiner un premier brouillon du SC avec une vérification de population.
3. Eliminer les types d'entités en surplus et les rôles communs et identifier le types d'idées dérivés.
4. Ajouter les contraintes d'unicité pour chaque type d'idée.
5. Vérifier que les types d'idées sont de (l'ordre droit).
6. Ajouter les contraintes de type d'entité, de totalité, de sous type et de fréquence d'occurrence.
7. Vérifier que chaque entité peut être identifiée.
8. Ajouter des contraintes d'égalité, d'exclusion, de sous ensemble et d'autres contraintes.
9. Vérifier que le schéma conceptuel est cohérent avec les exemples originaux, n'a pas de redondance et complet.

La procédure débute avec l'analyse des exemples d'informations qui vont constituer les entrée au système d`information. Fondamentalement, les trois premières étapes sont concernées par l`identification des types idées (stockées ou dérivées). Dans les autres étapes, on ajoute les contraintes aux types d`idées stockées. Tout au long de cette procédure des vérifications sont accomplies pour s'assurer qu'aucune erreur n'a été faite.

3 Démarche UDDPO (Uml Diagram Design Procedure based ontology)

La démarche proposée est inspirée de la procédure CSDP et basée sur un ensemble de règles proposées dans les travaux de Halpin [4][5][6][7][8] et enrichie par l'utilisation de l'ontologie de domaine.

3.1 Des exemples aux concepts élémentaires

Les Uses cases de la notation UML ont pour objectifs de modéliser un processus, capter les différentes spécifications. D'une façon analogue à la CSDP, on commence par les exemples familiers de l'univers de discours. Ces Data Uses cases peuvent être souvent augmentés par des informations de différents types (tables, formes, graphiques, diagrammes). Donc, cette étape consiste à la verbalisation de ces exemples en phrases simples (idées élémentaires).

Tableau 1. Matrice de données brutes d'un exemple de listing d'un système d'information

Em-ployé	Dept	Super-viseur	Fonction	Situation familiale (Sit fam)	N° Tel	Année Ma-riage	Nbre En-fants	Allocat ($)
Jones	D1	Fegan	Chef. Serv. Mouvem. Prd	M	3715821	1964	2	1000
Edwards	D1	Jones	Acheteur	C	-	-	-	-
Giles	D2	Jones	Vendeur	M	-	1975	-	-
Fegan	D2	*	Chef Dept. Commercial	M	3783437	1973	0	0

N.B : « * » = 'non-exitant' « - » = 'non enregsitré'

i) Résultats

Employé avec le nom '*Jones*' travaille dans le département de code '*D1*' ;
Employé avec le nom '*Jones*' a une situation familiale de code '*M*';
Employé avec le nom '*Fegan*' est un superviseur de l'employé avec le nom '*Jones*'
Employé avec le nom '*Edwards*' occupe la fonction *'acheteur'*.
Employé avec le nom '*Giles*' occupe la fonction *'vendeur'*.
Employé avec le nom '*Jones*'occupe la fonction *'Chef de serv. mouvements produits'*.
Employé avec le nom '*Fegan*' occupe la fonction *'Chef de département commercial'*
Le mariage de l'employé avec le nom '*Giles*' était en '*1975'*.
Le superviseur avec le no*m 'Jones'* a le numéro de téléphone '3715821'
Le superviseur avec le nom *'Jones'* a un nombre d'enfants égal à *'2'*
Chaque superviseur supervise au maximum deux employés.

ii) Vérification de qualité sur les idées élémentaires

La vérification consiste à se poser les questions suivantes :

Est-ce que les relations sont bien identifiées ? Est-ce que les relations peuvent être divisées en idées plus petites sans perte d'informations ?

3.2 Premier brouillon élémentaire des diagrammes de classes et d'objets et de cas d'utilisation

Cette étape consiste à dessiner un diagramme de classes à partir des relations élémentaires exprimées. Tout d'abord, on présente un diagramme d'occurrences à partir des relations élémentaires (idée élémentaire) au sens de Niam [12,13,15], on aura le diagramme d'occurrences suivant:

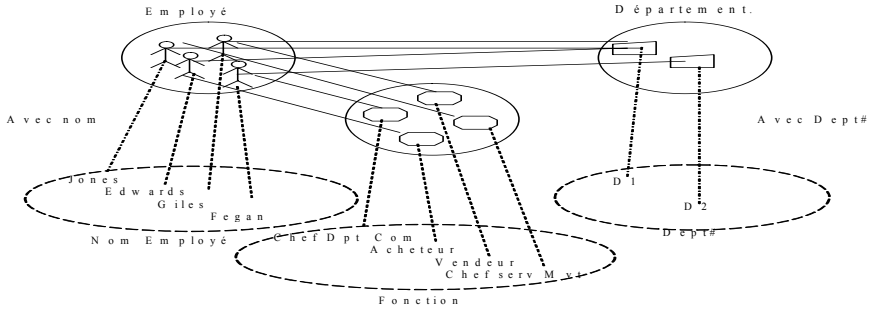

Fig. 1. Diagramme d'occurrences

Pour notre univers de discours, chaque employé a un et un seul nom, et chaque nom de personne se réfère exactement un employé. Ce type de référence est représenté dans le système de référence 1:1. Quand ce cas existe, on doit mettre le mode de référence entre parenthèses à coté du nom du type d'entité. De la même façon avec les autres idées , on aura les concepts suivants :

On reconnaît trois types d'objets. les entités sont les objets simples dans l'U°D. Les étiquettes sont les objets lexicaux qui sont utilisées pur se référer aux entités. Dans certains cas, on considère les objets et leurs relations comme des objets et on les appellera des objets composés.

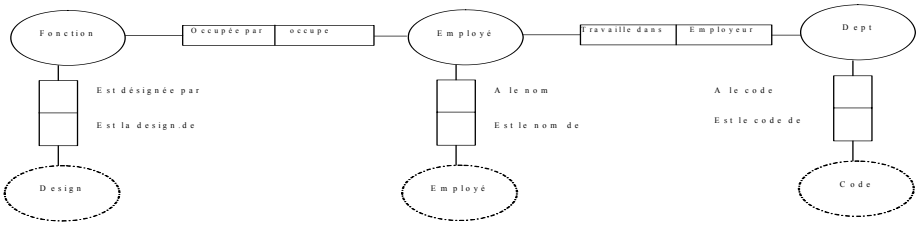

Fig. 2. Diagramme de schéma conceptuel correspondent en Niam

Pour représenter les diagrammes de classes et d'objets, on s'est basé sur un ensemble de règles proposées dans [7,8]:

- Les classes utilisées pour illustrer les types d'entités (NO Lexical Object Type);
- Les objets pour illustrer les entités;
- Les associations pour illustrer les types d'idées;
- Les liens pour illustrer les idées;
- Les rôles en UML pour illustrer les rôles de point de vue Niam;
- Les attributs pour illustrer les types d'étiquettes (Lexical Object Type) en Niam;

Concernant le premier diagramme d'utilisation, on transforme les associations en cas d'utilisation et les objets en acteurs.
On commence par présenter un diagramme d'objets pour les idées exprimées :

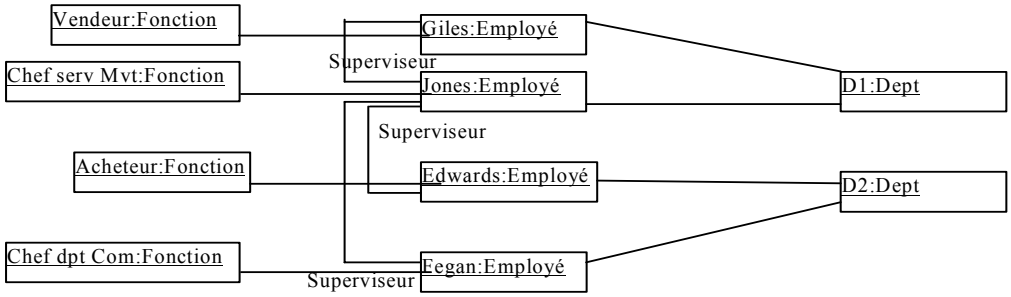

Fig. 3. Diagramme d'objets d'UML

Le diagramme de classes déduit à partir du diagramme d'objets est représenté par la figure suivante :

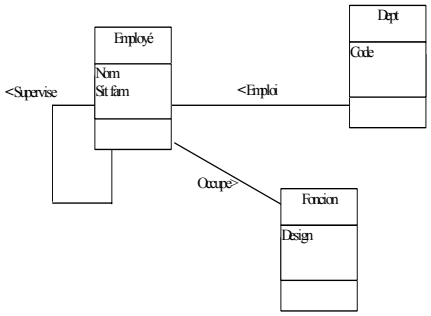

Fig. 4. Diagramme de classes d'UML

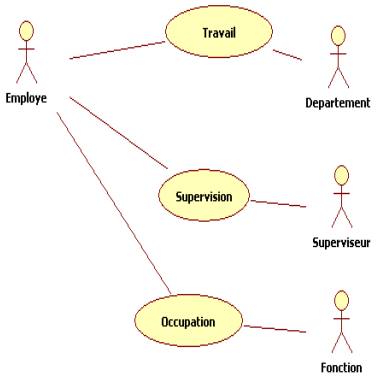

Fig. 5. Diagramme de cas d'utilisation

3.3 Arrangement des diagrammes et identification des attributs dérivés

Ayant dessiné notre premier brouillon du diagramme du schéma conceptuel et accompli une vérification de population. Cette étape consiste à éliminer les associations

en surplus ou les rôles communs et identifier n'importe quelles attributs dérivés. Dans cette étape, les questions pertinentes que nous nous posons, peuvent être résumées ainsi :

1. Est ce qu'un objet peut être instance de deux classes? Si oui combiner les classes en une seule.
2. Est ce que des objets de classes différentes peuvent être comparées d'une façon significative? Si oui combiner les classes en une seule.
3. Est ce que toutes les instances de deux classes différentes peuvent jouer le même rôle? Si oui combiner les classes en une seule et si nécessaire ajouter une nouvelle classe aux deux pour préserver la distinction originale.
4. Est ce qu'un attribut est dérivable à partir d'autres attributs ? Si c'est le cas, marquer le début de l'attribut dérivé par un *slach* "/" (selon la documentation d'Uml) ou le transformer en une opération qui encapsule le calcul de l'attribut.
5. Est-ce qu'il y a des cas d'utilisation ou acteurs en surplus? Si oui les supprimer.
6. Est ce que des cas d'utilisation différents peuvent être comparées d'une façon significative? Si oui combiner les cas d'utilisation en un seul.

En vérifiant notre brouillon de schéma conceptuel, on trouve que la valeur de Allocation enfants peut être dérivée de la valeur de Nombre d'enfants par la règle suivante : *Allocation enfants=500*nombre enfants*, alors on applique la règle 4 sur le schéma conceptuel de *Superviseur marié*, on obtient :

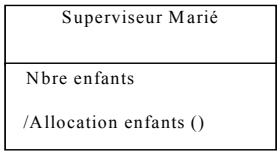

Fig. 6. Classe de *Superviseur Marié* du Diagramme de classes d'UML

Quant au cas d'utilisation, on constate que les deux cas d'utilisation travail et occupation peuvent être combinés en un seul par exemple tache. Les acteurs département et fonction sont des acteurs en surplus, on les supprime.

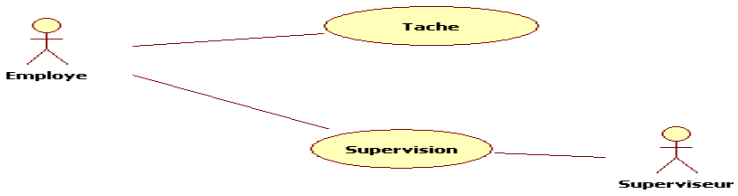

Fig.7. Diagramme de cas d'utilisation d'UML

3.4 Contraintes de multiplicité

Dans cette étape, on ajoute les contraintes de multiplicité dans le diagramme de classe obtenu, à partir des règles de passage présentées dans [8,9] (voir Fig. 6) et de la population de départ.

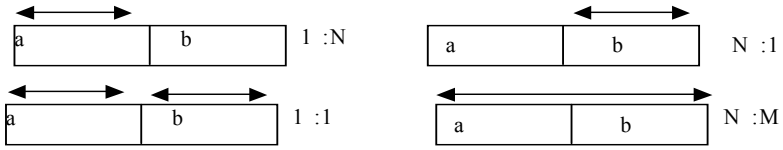

Fig. 8. Règles proposées de Niam pour la multiplicité en UML

Pour la population des classes dans les associations, on applique la multiplicité sur les associations, et pour la population des attributs, on applique la multiplicité sur les attributs. Quant aux contraintes sur la population qu'on ne peut pas exprimer à l'aide de contraintes dans Uml, on utilise les notes textuelles.

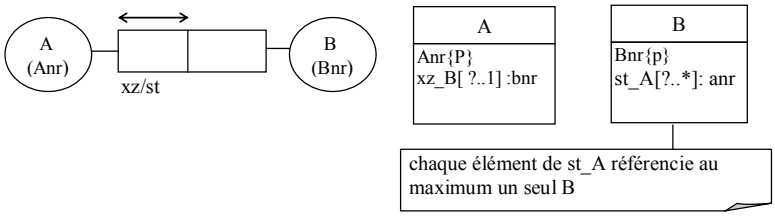

Fig. 9. Exemple de passage des multiplicités en UML

On peut également proposer d'autres types de contraintes si elles sont présentées en Niam ou déduites à partir de l'univers de discours, à l'aide des contraintes d'UML {Complet, Incomplet, Disjoint, …}.

3.5 Vérification de l'ordre

Cette étape vérifie que les relations ont un ordre exact. Si nous sommes plus précis dans les étapes précédentes, alors cette étape n'est pas exigée.
On vérifie l'ordre d'une association (type idée au sens Niam) de telle sorte qu'elle ne soit ni trop longue, ni trop courte par rapport à ce qu'elle doit être. Trois démarches sont possibles:

1. Utiliser le bon sens ou la connaissance antécédente de l'U°D pour décider si l'information sera perdue par une division d'association.
2. Utiliser les règles de division de la petite clé.
3. Fournir une table d'idée significative pour l'association, diviser ceci par projection et ensuite re-combiner par jointure naturelle. Si de nouvelles instances apparaissent alors l'association est indivisible.
4. Pour les cas d'utilisation, on substitue les cas d'utilisation génériques par des cas d'utilisation spécifiques.

Dans notre exemple, on substitue le cas d'utilisation "tache" par les activités spécifiques pour chaque fonction.

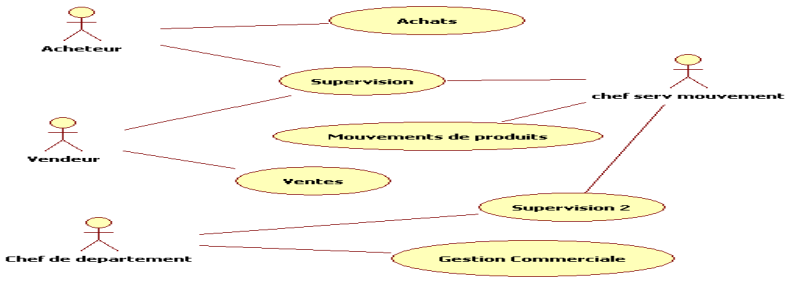

Fig.10. Diagramme de cas d'utilisation d'UML

3.6 Adjonction des contraintes de multiplicité, d'agrégation, de généralisation et de fréquence et les liens d'extension et d'utilisation.

Elles spécifient une restriction sur les valeurs possibles des instances d'un type d'entité données.

i) Les rôles impératifs et optionnels

Dans la notation UML, il n'existe pas une syntaxe pour exprimer qu'un rôle est impératif. Alors, on exprime ça par une contrainte textuelle ajoutée comme note.

ii) L'agrégation

Une agrégation représente une association non symétrique dans laquelle une des extrémités joue un rôle prédominant par rapport à l'autre extrémité. La contribution de l'expert de l'U°D est importante pour déterminer les relations d'agrégations entre les classes. La composition est un cas particulier de l'agrégation. Elle implique une contrainte sur la valeur de la multiplicité du coté de l'agrégat. : elle ne peut prendre que les valeurs 0 ou 1.

La composition et les attributs sont sémantiquement équivalents. La notation par composition s'emploie dans un diagramme de classes lorsqu'un attribut participe à d'autres relations dans le modèle [2,11,14]. A partir de cette définition, on peut noter que, si un attribut participe dans une relation dans le modèle et que sa participation est validée par une population du domaine, alors il devient une classe composite.

iii) La généralisation

Pour identifier les relations de généralisation entre les différentes classes, on utilise soit :

- Notre intuition et l'assistance de l'expert de l'U°D,
- Une technique connue par l'analyse de la matrice de sous typage [15]. Supposons qu'on a une entité A et qu'on veut lui appliquer le sous typage, on commence par diviser A en groupes où chaque membre d'un groupe a exactement le même rôle enregistré. Les détails enregistrés pour chaque membre de A forment l'enregistrement du modèle. Plusieurs groupes doivent avoir des modèles différents. Ayant partitionné A en groupes sur des modèles de base, nous les affichons à l'aide d'une matrice Partitions/Détails.

Tableau 2. Matrice Partitions/Détails du tableau 1.

Employé	Dept	Fonction	Supervis.	Sit fam	N° Tel	Année Mariage	Nbre Enf	Allocat ($)
(1)	1	1	1	1	1	1	1	1
(2)	1	1	1	1	0	0	0	0
(3)	1	1	1	1	0	1	0	0
(4)	1	1	1	1	1	0	0	0
Groupes	A	A	A	A	B	C	D	D

Le groupe A a les sous types B, C et D , le groupe B a le sous type D,
Le groupe C a le sous type D
Donc, le graphe de sous typage du tableau 2 est représenté par la **Fig. 8.**

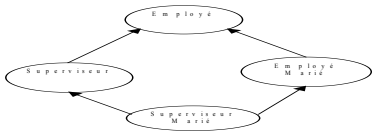

Fig. 11. Graphe de sous typage

iv) Fréquences d'occurrences

La fréquence d'occurrences représente soit :
- le nombre de fois qu'une entité d'un type donné peut être enregistrée pour jouer un rôle donné ;
- le nombre de fois que les étiquettes peuvent apparaître dans une colonne d'une relation donnée.

En UML, cette fréquence n'est pas toujours exprimée par les contraintes de multiplicité, dans le cas contraire, on utilise des contraintes textuelles.

v) Liens d'extension et d'utilisation:

On ajoute les liens d'extension et d'utilisation entre les cas d'utilisation. Généralement, pour y arriver, on fait recours à l'expert de domaine.

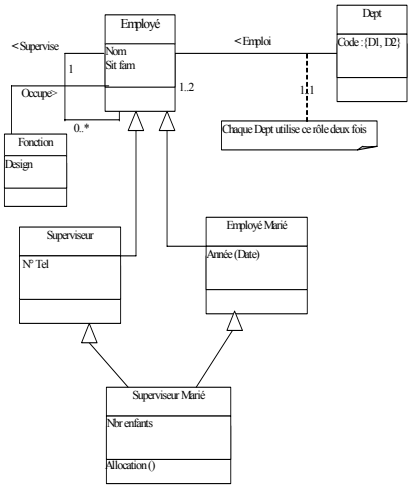

Fig. 12. Diagramme de classes d'UML

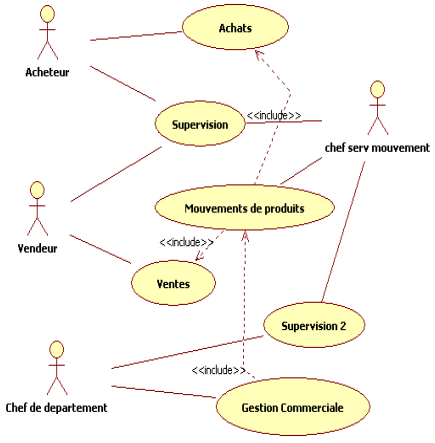

Fig.13.Diagramme de cas d'utilisation d'UML

3.7 Alignement avec l'ontologie de domaine.

Pour enrichir la sémantique de la conception obtenue, minimiser l'intervention de l'expert de domaine, nous utilisons l'ontologie de domaine.
L'identification des relations de l'univers de discours à travers de l'ontologie de domaine se base sur la similarité des classes de modèle et les concepts (ou classes) de l'ontologie. Ainsi, une classe de modèle est similaire à une classe de l'ontologie [13], si :

- Les deux entités (classe et concept) possèdent le même nom.
- Les deux entités (classe et concept) possèdent les mêmes attributs.
- Ils ont en commun un certain nombre d'attributs.

Notons ici que l'enrichissement du schéma conceptuel par les concepts de l'ontologie est relatif à l'intervention du concepteur validant les résultats.
Dans notre exemple, nous utilisons une ontologie de domaine d'une organisation qui décrit le domaine d'une personne et ses intérêts [14].
Les classes et les propriétés de cette ontologie sont :

Personne		
Employé		Groupe social
	-Nom	Fonction
	-Date de naissance	
	-Sexe	
	-Age	
	-Photo	Les relations sont:
	-Adresse	
Organisation		Membre de (Personne, Groupe social)
Repos		Travaille a (Employé, Organisation)
		Assurer (Employé, Fonction).

Avec l'utilisation de l'ontologie du domaine, la description de la classe employé est plus détaillée et le diagramme est enrichi avec une nouvelle classe 'groupe social' et une nouvelle association 'est membre de'.

3.8 Contraintes lexicales et qualifications

Dans cette étape on spécifie les contraintes lexicales sur les attributs, sur la base de la population de domaine ou on utilise l'avis de l'expert de l'U°D.
Dans notre exemple, l'attribut nom de la classe Employé, par exemple, est une chaîne de taille de 20 caractères. Alors, on ajoute le type chaîne à coté de cet attribut.
Les restrictions sur les associations (clés) peuvent être spécifiées par des qualifications.

3.9 Contraintes supplémentaires

Les contraintes d'égalité, d'exclusion et de sous ensemble peuvent être appliquées sur les associations. La syntaxe d'UML permet de représenter les contraintes ou-exclusif, disjoint, incomplet, complet, …, sur les associations.

3.10 Vérifications finales

Cette étape consiste à vérifier que le schéma conceptuel est consistant avec les exemples originaux , qu' il n'est pas redondant et qu'il est complet. On distingue deux étapes :- On vérifie si les contraintes exprimées sont validées par la population initiale, sinon on effectue des modifications nécessaires.
- On élimine la redondance des classes, des attributs et associations.

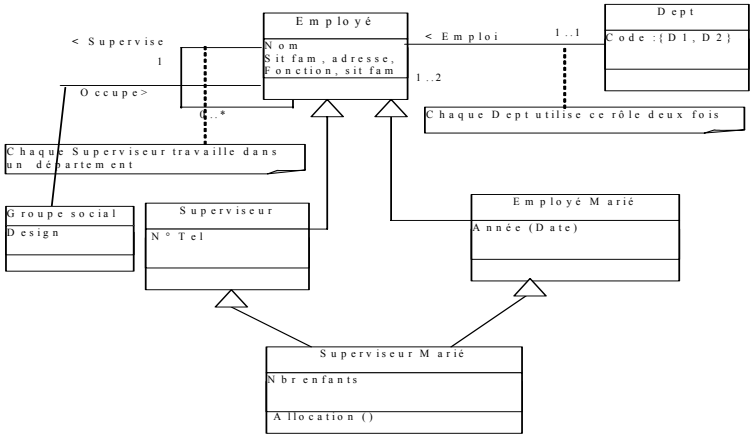

Fig. 14. Diagramme de classes d'UML enrichi par les contraintes textuelles

4 Conclusion

Pour élaborer des diagrammes UML sémantiquement riche et de qualité meilleure, nous avons proposée dans cet article une approche hybride (ascendante et descendante) faisant intervenir des sources sémantiques : de bas niveau d'abstraction (i.e. des exemples d'un système existant : tables, états…) et de haut niveau d'abstraction (i.e. l'ontologie de domaine).
La capture de la sémantique dans notre démarche s'est matérialisée par plusieurs façons, à savoir :La reprise de l'existant par des exemples significatifs de l'univers de discours, les connaissances de l'expert de domaine, l'utilisation d'un langage pseudo naturel pour assurer une meilleure communication entre les intervenants (l'expert, l'utilisateur et le système),utilisation de l'ontologie de domaine.
L'usage des ontologies dans le processus de spécification des systèmes d'information est une discipline récente. Il permet de faciliter le processus d'identification des besoins du système et de comprendre les liens sémantiques de ses composants et de réduire considérablement le temps nécessaire pour l'analyse conceptuelle.
Le processus de notre démarche appelée UDDPO (UML Diagram Design Procedure based on Ontology) se déroule en dix étapes successives, il commence par la verbalisation des exemples de l'univers de discours et se termine par la génération du schéma conceptuel en UML.

L'approche actuelle permet d'élaborer les digrammes de classes, d'objets et de cas d'utilisation UML. Elle pourrait être étendue pour élaborer les autres diagrammes en utilisant les ontologies de tâches ou d'autres techniques.
Cette proposition pourrait être adaptée à la re-ingénierie des ontologies et la re-ingénierie des applications web en des applications orientées services web en exploitant l'ontologie de tache OWL-S.

References

1. Andre P. Vailly A. « Spécification des logiciels : Deux exemples de pratiques récentes (Z et UML) », Ellipses, 2001.
2. Booch G. Rumbaugh J. Jacobson I. «The Unified Modeling Language User Guide», Addison-Wesley, Reading MA, USA, 1999.
3. Habrias H. «Le modèle relationnel binaire : Méthode IA (NIAM)», Eyrolles, 1988.
4. Halpin T. A. «Object Role Modeling (ORM/NIAM) », Handbook on Architectures of Information Systems, P. Bernus, K. Mertins & G. Schmidt eds, Springer-Verlag, Berlin, pp. 81-101, 1998.
5. Halpin T. A. «Object Role Modeling: an overview», 1998, available online at http://www.orm.net/overview.html.
6. Halpin T. A. «UML data models from an ORM perspective: Parts 1-10», Journal of Conceptual Modeling, InConcept, Minneapolis USA, 1998, available online from: www.orm.net/uml_orm.html.
7. Halpin, T.A. Bloesch, A.C. 1998, 'A comparison of UML and ORM for data modeling', Proc. EMMSAD'98: 3rd IFIP WG8.1 Int. Workshop on Evaluation of Modeling Methods in Systems Analysis and Design, Pisa, Italy (June).
8. Halpin T. A. «Data modeling in UML and ORM revisited», Proc. EMMSAD'99: 4th IFIP WG8.1 Int. Workshop on Evaluation of Modeling Methods in Systems Analysis and Design, Heidelberg, Germany (June), 1999.
9. Halpin, T.A. «Integrating fact-oriented modeling with object-oriented modeling», Information Modeling in the New Millenium, eds M. Rossi & K. Siau, Idea Group Publishing Company, Hershey, USA, 2000.
10. Halpin, T.A. «Augmenting UML with Fact-orientation», In workshop proceedings: UML: a critical evaluation and suggested future, HICCS-34 conference, Maui, January 2001.
11. Jacobson, I., Booch, G. & Rumbaugh, J. «The Unified Software Development Process», Addison-Wesley, Reading MA, USA, 1999.
12. Leung C. M. R., Nijssen G. M. «Relational database design using the NIAM conceptual schema», Information Systems, Volume 13, Number 2, 219-227, 1988.
13. Mitilian G. Rampnoux R. «Informatique: Méthoded d'Analyse pour la Gestion et l'Informatique», Ellipses, 1991.
14. Muller P. A. « Modélisation objet avec UML », Edition 2, Eyrolles, 2000.
15. Nijssen G. M. Halpin T. A. «Conceptual schema and relational database Design Procedure: a fact oriented approach», Prentice Hall, 1989.
16. Rumbaugh J. Jacobson I. Booch G. «The Unified Modeling Language Reference Manual», Addison-Wesley, Reading MA, USA, 1999.
17. Heflin J., Personal ontology, Univ. of Maryland, 2001. http://www.cs.umd.edu/projects/plus/DAML/onts/personal1.0