

Construction d'une ontologie de descripteurs UCD en astronomie

Emmanuel Nauer^{1,2}, Alexandre Richard^{1,3}, Sébastien Derriere³, Françoise Genova³, Amedeo Napoli¹, Yannick Toussaint¹

¹ LORIA, campus scientifique, B.P. 239, 54506 Vandœuvre-les-Nancy CEDEX.
{nauer, napoli, yannick}@loria.fr

² Université Paul Verlaine, Île du Saulcy, 57000 Metz.

³ ULP/CNRS - CDS, 11 rue de l'Université, 67000 Strasbourg.
{richard, derriere, genova}@astro.u-strasbg.fr

Résumé Les descripteurs UCD (Unified Content Descriptors) permettent de décrire des données d'astronomie. Une première étape de standardisation est restée purement syntaxique. Nous proposons d'y associer une sémantique formelle par la construction d'une ontologie. Dans cet article, nous présentons d'une part, une méthodologie pour construire une ontologie des UCD normalisés en OWL, et d'autre part une méthode d'annotation dans laquelle l'ontologie est exploitée pour l'attribution semi-automatique d'UCD à des descriptions de propriétés d'objets célestes issues de catalogues d'astronomie. Cette procédure exploite une classification approchée et progressive s'appuyant sur des méta-données qui établissent des liens entre les éléments lexicaux employés dans les descriptions et les propriétés des objets représentant les UCD dans l'ontologie.

Mots-clés : construction d'ontologie, classification approchée, classification progressive, OWL, logiques de descriptions, UCD.

1 Introduction

Les descripteurs UCD (*Unified Content Descriptors*) permettent de décrire des corps célestes en astronomie. Cependant, aucune sémantique formelle ne leur est initialement associée. Par exemple, `pos.eq` est le descripteur UCD qui doit être utilisé pour décrire une position en coordonnées équatoriales, sans définir aucunement ce qu'est une position en coordonnées équatoriales. Notre objectif ici est de construire une ontologie, nommée \mathcal{O}_{UCD} , des descripteurs UCD, où chaque UCD se voit attribuer une définition non ambiguë. La construction d' \mathcal{O}_{UCD} a pour but d'attribuer une définition et une sémantique aux UCD. Rappelons d'ores et déjà qu'une définition se matérialise par un ensemble de propriétés jouant le rôle de conditions nécessaires et suffisantes pour déterminer qu'un individu satisfait la définition ; ce qui est précisé ci-après. Ce travail a nécessité de définir, en collaboration avec un expert du domaine, chacun des descripteurs UCD par un ensemble de *propriétés*. Une

des utilisations de l'ontologie \mathcal{O}_{UCD} concerne l'annotation automatique du contenu de tables de données d'astronomie.

Dans la suite, nous nous intéressons à trois types de données :

- des tables de données *objets x propriétés* où les objets sont principalement des corps célestes et les propriétés des mesures ou des informations sur ces mesures,
- des méta-données qui décrivent les propriétés des tables de données, et
- des UCD représentant de façon standardisée ces propriétés.

L'objectif de l'annotation est d'identifier dans l'ontologie \mathcal{O}_{UCD} le concept — représentant un descripteur UCD — qui correspond au mieux à chacune des propriétés de la table. La méthode mise en œuvre pour l'annotation des propriétés consiste alors à définir un objet à classer en fonction d'un ensemble de propriétés, puis de rechercher par une classification *approchée* et *progressive* le ou les concept(s) qui partagent le maximum de propriétés avec l'objet à classer.

En outre, cette expérience de conception a permis d'établir un ensemble de règles générales pour :

- la construction d'ontologies d'unités de mesures comme les UCD, sachant qu'il existe plutôt des thésaurus d'unités de mesure mais peu voire pas d'ontologies du même genre,
- l'annotation de documents par l'intermédiaire d'un processus de classification.

Cet article se décompose en quatre parties. Nous exposons tout d'abord la problématique générale de notre travail, en parallèle avec une présentation des données d'astronomie. Nous exposons ensuite comment nous avons construit l'ontologie \mathcal{O}_{UCD} en OWL. Nous détaillons enfin comment nous mettons en œuvre une classification approchée et progressive de concepts dans \mathcal{O}_{UCD} et discutons nos résultats. Nous concluons par les perspectives de notre travail.

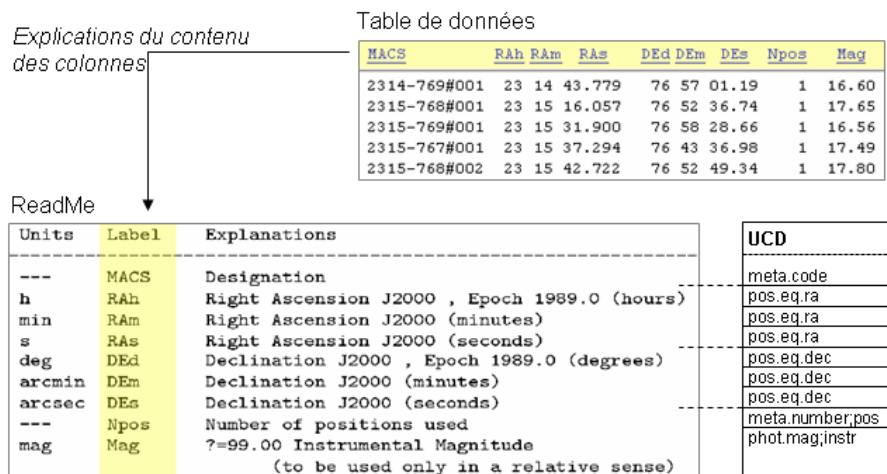


FIG. 1 – Exemple d’une table de données, d’un fichier ReadMe et des UCD associés. Chaque ligne de la table de données correspond à la description d’un objet céleste, chaque colonne représente une propriété de cet objet. Le contenu de chaque colonne de la table est explicité par une ligne dans le fichier ReadMe associé.

2 Problématique

Ce travail s’inscrit dans le cadre de l’ACI “*Masse de données en Astronomie*” qui se fait en collaboration avec les chercheurs du CDS, Centre de Données en astronomie de Strasbourg¹. Ce projet inclut des travaux sur les descripteurs UCD qui sont appelés à devenir les éléments du vocabulaire standardisé et universel de description des corps célestes en astronomie, et dont l’utilisation a pour but d’offrir une plus grande interopérabilité entre les sources de données astronomiques. Parmi ces sources, nous nous intéressons plus particulièrement dans notre travail à l’exploitation de catalogues d’astronomie pour lesquels il existe des propositions de standards de description (17). Un catalogue contient en particulier des tables de données astronomiques et un fichier nommé ReadMe qui fournit des méta-données relatives aux tables de données, permettant ainsi l’interprétation de ces tables par une personne. La figure 1 illustre ces deux types de données. Chaque colonne de la table de données correspond à un type de mesure ou d’observation qui est décrite par un Label composé d’un seul mot (exemple : RAh). Ce label est explicité dans le fichier ReadMe par un élément Units qui indique l’unité de mesure (dans le cas où la colonne décrite contient des mesures) et par un élément Explanations qui donne une courte explication textuelle associée au label de la colonne. Il y a ainsi une correspondance explicite entre une colonne de table de données et une ligne de ReadMe.

Le problème d’association d’un descripteur UCD à une colonne de la table de données est traité à partir du contenu des trois colonnes du fichier ReadMe, c’est-à-dire Units, Label et Explanations. Associer un UCD à une co-

lonne de la table de données revient à associer un UCD à la ligne du ReadMe qui décrit la colonne. Le résultat escompté est donné en illustration sur la figure 1 par les UCD correspondant effectivement à chacune des lignes du ReadMe. Par exemple, pos.eq.ra est le descripteur UCD représentant une ascension droite (*right ascension*) ; il doit être associé aux colonnes RAh, RAm et RA_s de la table de données. En réalité, cela correspond à la mesure d’une ascension droite en heures, minutes et secondes.

2.1 Présentation des UCD

Les UCD sont des descripteurs formels de données astronomiques sur les corps célestes (8) contrôlés par l’Alliance de l’Observatoire Virtuel International (IVOA²). La création des UCD répond à plusieurs besoins :

- éviter les ambiguïtés (un descripteur pour un type de données précis) et
- éviter les redondances (un descripteur et un seul pour un même type de données).

Les UCD servent à représenter des mesures ou des informations sur ces mesures, notamment le type des mesures. Par exemple, phys.temperature est un UCD qui renvoie à une température. Les UCD ont été créés il y a une dizaine d’années et ont connu une première révision avec le passage de la version UCD1 à la version UCD1+ (8; 15). Cette révision s’est en particulier traduite par l’adoption d’une nouvelle syntaxe. Nos travaux concernent les UCD1+ et toutes les références aux UCD dans ce papier renvoient aux UCD1+.

Dans le cadre du partage de données provenant des différents observatoires internationaux, l’objectif principal des

¹<http://cdsweb.u-strasbg.fr/>

²<http://www.ivoa.net/>

UCD est de permettre l'interopérabilité entre sources de données hétérogènes, pour résoudre des problèmes d'hétérogénéités telles que :

- des descriptions dans des langues différentes : par exemple, "*Motion in Right Ascension*" et "*Mouvement mesuré en ascension droite*" font, tous les deux, référence au concept du mouvement d'un corps céleste décrit par l'évolution de son ascension droite ;
- des raccourcis d'écriture différents pour une même description : par exemple, *posang*, *pa* et *apa* sont trois raccourcis différents de l'expression "*position angle*" qui exprime une orientation définie par un angle.

2.2 Syntaxe des UCD

L'ensemble des règles assurant la validité d'un UCD est détaillé dans (8) et (15) et une synthèse peut être consultée dans (19). Nous nous limitons ici aux points essentiels à la compréhension de notre travail.

Format d'un UCD : un UCD est une chaîne de caractères constituée de mots, eux-mêmes constitués d'atomes. Les mots sont séparés par des points-virgules et les atomes par des points (8).

Ainsi, par exemple :

- *pos*, *eq*, *ra*, *meta* et *main* sont des *atomes*
- *pos.eq.ra* et *meta.main* sont des *mots*
- *pos.eq.ra;meta.main* est un UCD

Un UCD est composé au minimum d'un mot, et chaque mot est composé d'au moins un atome. Un comité scientifique mis en place par l'IVOA contrôle l'ensemble des UCD en définissant l'ensemble des mots valides. Par conséquent, les briques de base des UCD sont les mots autorisés et non les atomes. Les mots sont regroupés dans une *hiérarchie* (en fonction des atomes par lesquels ils commencent) comprenant 12 catégories principales. Par exemple, les mots commençant par *pos* constituent la catégorie des positions.

Combinaison de mots pour la construction d'UCD

composés : chaque mot valide se voit associer un code qui indique quel rôle il peut jouer dans un UCD composé. Il existe actuellement 6 codes, parmi lesquels :

- code P : mot se trouvant nécessairement en première position dans un UCD
- code S : mot ne pouvant être en première position dans un UCD
- code Q : mot pouvant être à une position quelconque dans un UCD
- code E : mot nécessairement suivi de 1 mot de la catégorie *em* (mot commençant par l'atome *em*, représentant des concepts liés aux rayonnements électromagnétiques)
- etc.

Pour être valide, un UCD doit être composé uniquement de mots valides ; les mots qui le composent

doivent tous respecter la règle correspondant au code qui leur est associé. L'ensemble des codes joue ainsi le rôle d'une grammaire (19). Dans la suite, nous utilisons "*UCD simple*" pour désigner un UCD formé d'un seul mot et "*UCD composé*" pour un UCD formé de plusieurs mots.

Interprétation : une interprétation concrète en langage naturel est associée à chaque UCD dans (15). Par exemple, *pos* représente le concept de position, *stat.error;pos.eq.ra* représente une erreur sur la mesure d'une ascension droite, etc. De façon générale, le premier mot d'un UCD composé représente le concept principal, les mots suivants sont destinés à préciser ce concept principal.

2.3 Pourquoi une ontologie des UCD ?

Une ontologie permet de représenter formellement un ensemble de concepts en leur attribuant une sémantique. Les avantages qui découlent de l'existence d'une ontologie peuvent être classés en deux grandes catégories (21; 23). Du point de vue de la "*communication et interopérabilité*", une ontologie facilite l'échange d'information ou de connaissances venant de sources hétérogènes entre des êtres humains et/ou des machines, car elle permet la désambiguïsation et la vérification de cohérence des éléments représentés. Du point de vue de la "*spécification, intégration et ré-utilisation*", une ontologie peut aider à la spécification de problèmes liés au domaine qu'elle représente et servir de passerelle pour faciliter l'intégration d'une application à une plate-forme. Enfin, même si la construction de l'ontologie est influencée par les applications, tant que la structure de l'ontologie n'est pas remise en cause, l'ontologie et les applications peuvent évoluer séparément. Concrètement, disposer d'une ontologie permet de résoudre certains problèmes comme guider l'utilisateur lors d'un accès à des données (9; 20) ou encore organiser et/ou comparer des documents par rapport à leur contenu (1).

\mathcal{O}_{UCD} permet, du point de vue de la représentation, de garantir la cohérence et la non-ambiguïté des UCD, ainsi que de les organiser/classifier les uns par rapports aux autres. \mathcal{O}_{UCD} est utilisée pour la recherche d'information et pour détecter des erreurs ou omissions dans des descriptions, par exemple, pour vérifier la correction de lignes de *ReadMe*. Pour illustration, dans la ligne de *ReadMe* donnée en figure 2, les unités ont été mentionnées dans la colonne *Label*. Or, le concept définissant l'UCD *ascension droite* de cette ligne de *ReadMe* dans \mathcal{O}_{UCD} possède une propriété *hasAngleUnit*, ce qui signifie que la colonne *Units* doit être renseignée. Ceci permet donc de détecter une omission sur cette ligne et, éventuellement, de la corriger.

Units	Label	Explanations
---	RAhms	Right ascension in h m s, ICRS (J1991.25) (T3)

FIG. 2 – Exemple d’une ligne de ReadMe non complète.

3 Construction d’une ontologie des UCD en OWL

Beaucoup de propositions ont été faites pour définir ce qu’est une ontologie. La plus connue en informatique est probablement celle de Gruber (12), à savoir qu’*une ontologie est la spécification explicite d’une conceptualisation*. Dans l’acceptation standard de cette définition informelle, le terme *conceptualisation* renvoie à un modèle abstrait prenant la forme d’un ensemble de définitions de concepts et de propriétés de concepts organisés en hiérarchie. Par ailleurs, l’expression *spécification explicite* renvoie au fait que le modèle doit être représenté dans un langage de représentation des connaissances (muni d’une syntaxe et d’une sémantique) pour que le modèle soit utilisable aussi bien par des machines que des êtres humains (11).

3.1 Formalisme et composants des ontologies

\mathcal{O}_{UCD} est codée en OWL-DL qui repose sur l’utilisation du formalisme de représentation des logiques de descriptions (4; 2). Ce formalisme s’appuie sur trois types d’entités :

- les *concepts* (ou *classes*) qui représentent des ensembles d’individus ayant des propriétés communes. L’ensemble des individus correspond à l’*extension* du concept. Par exemple, le concept `Measure` représente un ensemble de mesures ; cet ensemble de mesures est l’extension du concept `Measure`.
- des *individus* (ou *instances* des concepts). Par exemple, les individus `100 mètres` et `45 degrés` sont des instances du concept `Measure`.
- les *rôles* (ou *propriétés*) qui représentent des relations binaires entre les concepts. Par exemple, le rôle `hasUnit` entre le concept `Measure` et le concept `Unit` exprime qu’une mesure (instance du concept `Measure`) est en relation avec une unité (instance du concept `Unit`). Chaque rôle possède un *domaine* et un *co-domaine* : le domaine est le concept où est déclaré le rôle et le co-domaine est le concept avec lequel le rôle établit une relation. Ainsi, `hasUnit` a pour domaine le concept `Measure` et pour co-domaine le concept `Unit`.

Les concepts (et les rôles le cas échéant) sont organisés en une hiérarchie par la relation de *subsumption*, notée \sqsubseteq , où $D \sqsubseteq C$ se lit “ C subsume D ” ou “ D est subsumé par C ”. Un concept C subsume un concept D si et seulement

si C est plus général que D . On note \top le concept le plus général, qui est à la racine de la hiérarchie de concepts et subsume tous les autres concepts.

Un concept définit des conditions d’appartenance d’un individu à l’extension de ce concept. Suivant ces conditions, le concept est *primitif* ou *défini* :

- s’il ne s’agit que de conditions nécessaires, alors le concept est *primitif*. Par exemple, le concept `Unit` introduit par `Unit` \sqsubseteq \top est primitif.
- s’il s’agit d’un ensemble de conditions nécessaires et suffisantes (une instance X fait partie de l’extension d’un concept C si et seulement si X a les mêmes propriétés que C), alors le concept est *défini* et l’ensemble de conditions constitue la *définition* de ce concept. Les concepts définis sont introduits par une équivalence notée \equiv , où $C \equiv D$ signifie $C \sqsubseteq D$ et $D \sqsubseteq C$. Par exemple, le concept `Measure` introduit par `Measure` \equiv (`hasUnit=1`) \sqcap (\forall `hasUnit.Unit`) \sqcap (`hasMeasuredValue=1`) \sqcap (\forall `hasMeasuredValue.MeasuredValue`) est défini. Si X a une valeur mesurée (instance du concept `MeasuredValue`) et une unité (instance du concept `Unit`) alors X est une mesure (instance du concept `Measure`) ; à l’inverse, une mesure possède exactement une unité et une valeur mesurée³.

Les définitions de concepts font essentiellement intervenir les constructeurs suivants :

- la *conjonction de concepts*, notée $C \sqcap D$,
- la *quantification universelle*, notée $\forall r.C$, qui exprime que le co-domaine du rôle r est restreint au concept C ,
- la *cardinalité* qui fixe le nombre minimal et maximal de valeurs élémentaires que peut prendre un rôle r . Elle est notée $(r \leq n)$, $(r \geq n)$ ou $(r = n)$ suivant que n est une cardinalité maximale, minimale ou exacte.

\mathcal{O}_{UCD} a été codée en OWL par l’intermédiaire de l’éditeur d’ontologies PROTÉGÉ⁴ (14) et a été stockée en XML. Le système RACER a été utilisé comme moteur de subsumption et de classification (13).

Il n’existe pas de méthode unifiée de création d’une ontologie, mais un processus itératif, introduit dans (21; 24), donne les grandes lignes de la conception d’une ontologie (un guide détaillé est proposé dans (16)) :

- la phase d’analyse nécessite de déterminer ce que conceptualise l’ontologie et quelles sont les utilisations qui vont en être faites.
- la phase de construction s’attache à la mise en place

³la valeur est la plupart du temps un nombre, mais peut aussi être d’une autre forme, comme une durée exprimée en *hh:mm:ss*, par exemple.

⁴<http://protege.stanford.edu>

des hiérarchies de concepts et de rôles ainsi qu'aux définitions de concepts.

- la phase d'évaluation a pour but de tester la cohérence de l'ontologie et d'effectuer des tests d'utilisation dans le but de corriger ou d'ajuster les éléments nécessaires (par un retour à la phase de construction).
- la phase de maintenance concerne l'utilisation effective de l'ontologie ; un retour à la phase de construction permet de prendre en compte les évolutions nécessaires.

Lors de la création d'une ontologie, un soin particulier doit être apporté au respect des critères suivants (12) :

Adéquation : une ontologie doit être adaptée à l'usage qui en est fait. Dans une ontologie trop générale ou trop spécifique, les définitions sont trop vagues ou trop complexes (car mettant en jeu des paramètres n'ayant aucun rapport avec le cadre d'exploitation de l'ontologie).

Clarté : les concepts définis sont toujours préférables aux concepts primitifs.

Cohérence : l'ontologie ne doit pas être incohérente, c'est-à-dire qu'on ne doit pas pouvoir dériver à la fois les concepts A et $\neg A$.

Évolution : une ontologie doit pouvoir être étendue sans remettre en cause sa cohérence. Ceci implique en particulier d'avoir des règles de construction précises.

La construction de l'ontologie nécessite une connaissance du domaine d'application et donc l'intervention d'experts du domaine tant pour la construction elle-même que pour la vérification et la validation de l'ontologie (seule la cohérence peut être vérifiée par un non-expert).

Afin de restreindre l'ampleur de la construction d'une ontologie des UCD, nous avons limité dans un premier temps le domaine d'étude aux UCD représentant des positions. Les UCD décrivant des positions font principalement intervenir des mots commençant par l'atome `pos`. Ces mots sont désignés par l'expression "mot `pos`" dans ce qui suit. Parmi les mots `pos`, on trouve par exemple :

- `pos` : qui représente le concept de position,
- `pos.eq` : qui représente des coordonnées équatoriales,
- `pos.eq.ra` : qui représente une ascension droite en coordonnées équatoriales.

En réponse aux deux grandes questions de la phase d'analyse, il faut construire une ontologie des UCD, \mathcal{O}_{UCD} , incluant au moins un mot `pos`, et pouvoir exploiter \mathcal{O}_{UCD} pour mettre en correspondance une ligne de `ReadMe` avec un UCD. Cependant, la construction de \mathcal{O}_{UCD} se heurte à deux problèmes majeurs. Primo, il n'existe pas de liste exhaustive des UCD composés incluant des mots `pos`. En effet, seuls les UCD simples sont répertoriés de façon exhaustive, les UCD composés étant uniquement contrôlés par les règles de composition énoncées précédemment. Secundo, le nombre d'UCD composés incluant des mots `pos` construits à partir de ces mêmes règles dépasse potentiellement 10^6 .

Au delà des problèmes de faisabilité, une telle taille rendrait l'ontologie difficilement lisible et utilisable. Mais s'il n'est pas possible de construire une ontologie de tous les UCD incluant un mot `pos`, il est possible de le faire pour les UCD simples, à partir de la liste des mots `pos` qui contient actuellement 58 mots `pos`. Le nombre réduit de mots rend possible la création manuelle d'une telle ontologie. Nous avons donc décidé de construire une ontologie de concepts définissant ces mots. Les UCD composés ne sont, par conséquent, pas représentés dans l'ontologie ; leur participation dans l'annotation des tables étant prise en compte dans la phase d'exploitation de l'ontologie à travers une procédure spécifique.

3.2 Processus de construction de l'ontologie

Des méthodes semi-automatiques, reposant sur l'utilisation de techniques d'apprentissage, peuvent être mises en œuvre dans la construction d'ontologies. Certaines de ces méthodes reposent sur des techniques de classification (2) ou d'extraction de règles d'associations (18). D'autres méthodes de construction d'ontologies à partir de textes reposent sur une phase d'analyse syntaxique, puis d'analyse distributionnelle telles que mises en œuvre par exemple dans (6; 3; 22). Nous n'avons pas retenu ces approches car nous ne disposons pas de données (tables ou textes) décrivant les UCD. En effet, nous ne disposons que d'une seule définition textuelle pour chacun des UCD (15). C'est pourquoi nous avons opté pour une construction manuelle de l'ontologie en privilégiant la discussion avec les experts et en suivant les principes énoncés dans (16; 2), le problème principal étant d'identifier les propriétés qui permettent de définir les concepts du domaine.

Nous détaillons maintenant les 5 grandes étapes que nous avons identifiées dans cette expérience de création de \mathcal{O}_{UCD} :

1. Identification des concepts

Règle : un concept représentant un *mot* est nécessairement défini et ne peut représenter qu'un seul mot. Pour nous qui cherchons à représenter les mots `pos`, de tels mots sont par exemple : `pos.eq`, `pos.eq.ra`, `pos.pm.ra`. Le nom choisi pour identifier chaque concept est le mot qu'il représente.

2. Écriture des définitions de concepts et identification des rôles

Les définitions se ramenant à des ensembles de rôles, établir les définitions des concepts passe par l'identification des rôles que possède un concept. Par exemple : pour établir la définition de `pos.eq.ra`, nous devons savoir ce qu'est une ascension droite dans notre contexte. L'expert du domaine, en charge d'explicitier les concepts, a établi que :

- *une ascension droite est une mesure⁵ qui*

⁵ici, le concept *Measure* n'est pas utilisé pour définir `pos.eq.ra`

a nécessairement une valeur mesurée; nous obtenons donc la partie de définition suivante : $(\text{hasMeasuredValue}=1) \sqcap (\forall \text{hasMeasuredValue.MeasuredValue})$
- *cette valeur mesurée est exprimée avec une unité d'angle*, soit : $(\text{hasAngleUnit}=1) \sqcap (\forall \text{hasAngleUnit.AngleUnit})$
- *cette mesure est faite dans un repère équatorial*, soit : $(\text{hasFrameTypeEq}=1) \sqcap (\forall \text{hasFrameTypeEq.FrameTypeEq})$
- *l'ascension droite est mesurée à partir de l'origine des ascensions droites du repère équatorial*, soit : $(\text{hasCcOriginEqRa}=1) \sqcap (\forall \text{hasCcOriginEqRa.CcOriginEqRa})$.

Introduire les rôles implique d'inclure les concepts co-domaines de ces rôles s'ils ne sont pas déjà présents dans \mathcal{O}_{UCD} . Le cadre d'exploitation de \mathcal{O}_{UCD} , à savoir l'identification d'un concept à partir de ses propriétés, nous a amené à définir chaque concept de la façon la plus précise possible. Par conséquent, les concepts utilisés en tant que co-domaines sont nombreux et doivent être différenciés pour associer *un co-domaine unique et différent à chaque rôle*. Cette règle de construction a pour objectif de faire correspondre un rôle avec un co-domaine (et inversement) afin de pouvoir identifier un rôle de manière non ambiguë à partir de son co-domaine. Par exemple, le co-domaine `AngleUnit` ne peut être associé à une définition de concept que par le rôle `hasAngleUnit`. Ce choix, peu contraignant lors de la construction, facilite grandement l'exploitation de l'ontologie car il permet de retrouver les rôles à partir des co-domaines (et réciproquement).

3. Hiérarchisation des concepts

La hiérarchie des concepts est organisée par la relation de subsomption. Pour les concepts définis, la hiérarchie découle des relations de subsomption entre définitions : si la définition d'un concept `C1` est plus spécialisée que la définition d'un concept `C2`, alors $C1 \sqsubseteq C2$. Ainsi, `pos.barycenter` \sqsubseteq `pos` car :

$$\text{pos} \equiv (\text{hasFrameType}=1) \sqcap (\forall \text{hasFrameType.FrameType})$$

et

$$\begin{aligned} \text{pos.barycenter} &\equiv (\text{hasFrameType}=1) \\ &\sqcap (\forall \text{hasFrameType.FrameType}) \\ &\sqcap (\text{hasBarycentricCoordinates} \geq 2) \\ &\sqcap (\forall \text{hasBarycentricCoordinates.BarycentricCoordinates}). \end{aligned}$$

Pour les concepts primitifs, la hiérarchisation est don-

car chaque concept a été construit par énumération de ses propriétés propres. De plus, le concept `Measure` n'a été créé qu'ultérieurement en vue d'améliorer la lisibilité de l'ontologie (voir plus loin). Ceci n'a cependant pas de conséquence sur \mathcal{O}_{UCD} puisque, d'après les définitions de `pos.eq.ra` et `Measure`, on a bien `pos.eq.ra` \sqsubseteq `Measure`; d'où `pos.eq.ra` est bien représentée dans \mathcal{O}_{UCD} comme étant une *mesure*.

née par l'expert qui aide à construire l'ontologie du domaine. Ainsi, l'expert a par exemple établi pour les deux concepts primitifs `Unit` et `AngleUnit`, la relation suivante : `AngleUnit` \sqsubseteq `Unit`.

4. Hiérarchisation des rôles

La hiérarchie des rôles est également organisée par la relation de subsomption, selon la règle suivante : $r1 \sqsubseteq r2$ si et seulement si $\text{domaine}(r1) \sqsubseteq \text{domaine}(r2)$ et $\text{co-domaine}(r1) \sqsubseteq \text{co-domaine}(r2)$. Par exemple :

- `hasUnit` a pour domaine `Measure` et pour co-domaine `Unit`

- `hasAngleUnit` a pour domaine `pos.angDistance` et pour co-domaine `AngleUnit`

- `hasAngleUnit` \sqsubseteq `hasUnit` si et seulement si `pos.angDistance` \sqsubseteq `Measure` et `AngleUnit` \sqsubseteq `Unit`.

5. Amélioration éventuelle de la structuration.

Enfin, on peut souhaiter améliorer la structuration de l'ontologie en créant des concepts *abstrait*s, subsumants d'un ensemble de concepts plus spécifiques, à la façon des classes abstraites en programmation par objets (10). Toujours dans un but de structuration, nous avons souhaité que ces concepts abstraits respectent la condition suivante : l'expert doit juger que le concept abstrait a un sens dans le domaine traité. Par exemple :

$$\begin{aligned} \text{Measure} &\equiv (\text{hasMeasuredValue}=1) \\ &\sqcap (\forall \text{hasMeasuredValue.MeasuredValue}) \\ &\sqcap (\text{hasAngleUnit}=1) \\ &\sqcap (\forall \text{hasAngleUnit.AngleUnit}) \end{aligned}$$

subsume tous les concepts renvoyant à des mesures (les concepts des mesures spécifiques ont tous une valeur mesurée et une unité) et représente la mesure en général (ce qui a un sens en astronomie). Il peut donc être intégré à \mathcal{O}_{UCD} .

Après avoir décrit notre approche pour la conception de \mathcal{O}_{UCD} , nous montrons dans la partie suivante comment nous avons exploité cette ontologie pour associer un UCD à une ligne de `ReadMe`.

4 Attribution d'un UCD à une ligne de ReadMe

L'objectif est ici d'attribuer un UCD à une ligne d'un fichier `ReadMe`, à partir du contenu des colonnes `Units`, `Label` et `Explanations`. À l'heure actuelle, le système développé n'associe que des UCD simples ; nous discutons de l'association d'UCD composés en paragraphe 4.4.

4.1 Processus d'annotation

Le fonctionnement global du système est présenté en figure 3. Nous partons d'une ligne de `ReadMe` pour arriver à une liste triée de concepts pertinents de \mathcal{O}_{UCD} . Dans

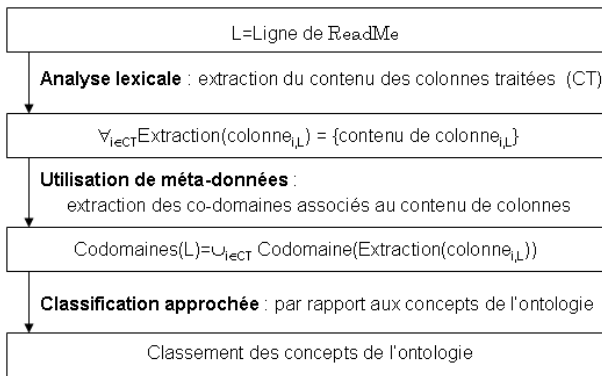


FIG. 3 – Schéma global de fonctionnement du système d'attribution d'UCD.

cette liste, les concepts les mieux classés sont ceux qui représentent les mots `pos` dont la description correspond le mieux à la ligne de `ReadMe`. Dans un premier temps, nous n'exploitons que les colonnes `Units` et `Explanations` car le contenu de la colonne `Label` est souvent ambigu (voir plus loin); `CT`, qui représente l'ensemble des colonnes traitées, vaut $\{\text{Units}, \text{Explanations}\}$. Nous décrivons à présent en détail le fonctionnement du système avec l'exemple de la ligne `L` suivante :

Units	Label	Explanations
s	RAS	Right Ascension J2000 (seconds)

1. Analyse lexicale

La première étape nécessite une analyse lexicale de la ligne du `ReadMe` pour extraire le contenu des colonnes `Units` et `Explanations`. En pratique, à partir de la ligne `L` précédente, le système renvoie :

```
Extraction(Units, L) = {s}, et
Extraction(Explanations, L) =
    {right, ascension, J2000, seconds}.
```

2. Utilisation de méta-données établies par un expert du domaine

Le but de cette seconde étape est d'obtenir un ensemble de co-domaines de rôles à partir des ensembles de termes obtenus à l'étape 1. Des fichiers de méta-données ont été construits avec l'expert pour chaque colonne exploitée, dans le but d'associer à chaque élément lexical de la colonne un ensemble de co-domaines auxquels l'élément lexical fait référence. Dans le cadre de ce travail, les fichiers de méta-données ont été créés manuellement par l'expert. Cette tâche manuelle — et fastidieuse — ne permet pas de garantir que l'ensemble des entrées lexicales associées à un co-domaine soit exhaustif. Une façon de pallier ce problème d'exhaustivité serait la construction automatique ou semi-automatique de ces fichiers. Par exemple, à partir de lignes de `ReadMe` dont l'UCD est connu, l'utilisation de méthodes d'extraction de connaissances pourrait permettre d'extraire les corrélations entre entrées lexicales et co-domaines. Ce point

constitue une perspective de ce travail.

En pratique, nous fournissons en entrée les ensembles $\{s\}$ et $\{\text{right}, \text{ascension}, \text{J2000}, \text{seconds}\}$, obtenus à l'étape 1. Les fichiers de méta-données, de la forme : $\{(\text{contenu de la colonne} \Rightarrow \{\text{co-domaines}\})\}$ et construits a priori par l'expert permettent d'associer des co-domaines potentiels aux entrées lexicales présentes dans les colonnes :

```
Méta(Units) =
    {(s => {AngleUnit, MeasuredValue},
      (mas/yr => {AngularSpeedUnit, Measured-
        Value })),
     ...}
Méta(Explanations) =
    {(right => {CcOriginEqRa, MeasuredValue}),
     (ascension => {MeasuredValue}),
     ...}
```

Le système retourne en sortie l'ensemble des co-domaines associés à la ligne de `ReadMe` traitée, à savoir (en ignorant les doublons) : $\text{Codomaines}(L) = \{\text{AngleUnit}, \text{CcOriginEqRa}, \text{MeasuredValue}\}$.

3. Classification approchée et progressive

Dans cette dernière étape, on recherche les concepts de \mathcal{O}_{UCD} possédant les rôles de $\text{Codomaines}(L)$ via un parcours de \mathcal{O}_{UCD} , puis on classe ces concepts en fonction d'un score égal au nombre de rôles qu'ils partagent avec les rôles de $\text{Codomaines}(L)$. C'est ici qu'est exploitée l'hypothèse que le co-domaine d'un rôle identifie le rôle de manière unique (hypothèse découlant de la règle suivie lors de la construction de l'ontologie précisée au paragraphe 3.2 : chaque rôle a un co-domaine différent et unique). Dans notre exemple, l'ensemble des rôles de $\text{Codomaines}(L)$ est : $\{\text{hasAngleUnit}, \text{hasCcOriginEqRa}, \text{hasMeasuredValue}\}$.

Le classement des concepts de \mathcal{O}_{UCD} qui partagent des rôles avec $\text{Codomaines}(L)$ est donné en figure 4 (les rôles qui doivent être partagés ont été écrits en italiques). Dans cet exemple, comme il existe un unique meilleur classé, c'est lui que le système désigne comme l'UCD à associer à la ligne de `ReadMe`, à savoir `pos.eq.ra`.

En cas d'égalité au premier rang du classement, le processus de classification est repris, dans un second temps — d'où l'appellation de classification *progressive* — avec $\text{CT} = \{\text{Units}, \text{Explanations}, \text{Label}\}$ pour exploiter la colonne `Label` du `ReadMe`. Les labels sont, en effet, porteurs d'information importante, mais leur exploitation ne peut avoir lieu que dans une seconde phase car ils sont souvent ambigus. Par exemple, le même label `alpha` peut désigner à la fois (dans deux lignes de `ReadMe`) un angle de phase

CONCEPT	SCORES	RÔLES PARTAGÉS
pos.eq.ra	3	<i>hasAngleUnit, hasMeasuredValue, hasCcOriginEqRa, hasFrameTypeEq</i>
pos.angDistance	2	<i>hasAngleUnit, hasMeasuredValue</i>
pos.az.alt	2	<i>hasAngleUnit, hasMeasuredValue, hasCcOriginAzAlt, hasFrameTypeAz</i>
pos.eq.dec	2	<i>hasAngleUnit, hasMeasuredValue, hasCcOriginEqDec, hasFrameTypeEq</i>
...

FIG. 4 – Classement des concepts de \mathcal{O}_{UCD} qui partagent les rôles $\{hasAngleUnit, hasCcOriginEqRa, hasMeasuredValue\}$; les autres rôles n’interviennent pas dans le classement.

d’une orbite d’un corps céleste et une ascension droite d’un corps céleste. Il est donc difficile pour l’expert d’associer des co-domaines à un label de manière sûre lors de la construction du fichier de méta-données `MÉta(Label)`. De fait, une recherche de concept effectuée avec l’exploitation de la colonne `Label` peut conduire à un résultat erroné; c’est pourquoi nous n’exploitons cette colonne qu’en cas de nécessité, c’est-à-dire si la première passe débouche sur une égalité au premier rang.

Enfin, s’il y a toujours égalité au premier rang après la seconde passe, c’est à l’expert de choisir l’UCD correspondant au mieux à une ligne de `ReadMe` parmi les concepts à égalité au premier rang.

4.2 Évaluation de la méthode

Pour nos tests, nous avons utilisé des `ReadMe` des catalogues d’astronomie les plus employés et nous avons confronté nos résultats à des attributions d’UCD faites par un expert humain. Dans une première expérience, pour 75 lignes de `ReadMe` traitées, le bilan est le suivant :

- 4 lignes ont un UCD attribué dès la première passe,
- 42 lignes supplémentaires ont un UCD attribué après la deuxième passe,
- 26 lignes relèvent du choix d’un expert (égalité au premier rang du tableau) après la deuxième passe,
- 3 lignes ont conduit à un échec (l’UCD attendu n’apparaît pas au premier rang du tableau de concepts renvoyé en fin de traitement).

Potentiellement, une ligne de `ReadMe` fait émerger un certain nombre de co-domaines possibles, ce qui conduit à un nombre élevé de concepts possibles (partageant les rôles associés à ces co-domaines) comme UCD. Les trois échecs sont dus plus particulièrement à des erreurs d’attribution entre les éléments de la ligne de `ReadMe` et les co-domaines possibles.

4.3 Comparaison à l’existant

Le système mis en œuvre jusqu’alors au Centre de Données Astronomiques de Strasbourg pour l’association

d’UCD à des lignes de `ReadMe` est un logiciel spécifique nommé *UCD Builder*⁶. Ce logiciel utilise un ensemble de règles lexicales et syntaxiques qui associent chacune des scores à des UCD potentiels; une liste triée d’UCD est proposée en sortie en fonction d’un score global. Cette application, qui n’exploite que la partie `Explanations`, donne de bons résultats, aussi bien sur des UCD composés que des UCD simples. Par contre, une limitation majeure d’un tel système est la dépendance des règles lexicales et syntaxiques aux évolutions des UCD et aux langues naturelles utilisées dans les descriptions textuelles : les règles doivent être réécrites pour tenir compte de chaque changement.

L’approche proposée dans ce papier rend le processus d’identification relativement indépendant de la langue utilisée dans les `ReadMe` et des évolutions des UCD. En effet, dans le premier cas, il suffit de rajouter des entrées lexicales dans les fichiers de méta-données et dans le deuxième cas de s’assurer que l’ontologie est toujours adaptée et de la faire évoluer le cas échéant, sans avoir à changer la procédure d’association d’un concept à une ligne de `ReadMe`. Par conséquent, cette approche a pour avantage d’être moins sensible aux évolutions que la méthode employée dans *UCD Builder*.

Il reste encore à prendre en compte les UCD composés avant de pouvoir faire une comparaison complète entre les performances de notre système et celles d’*UCD Builder*.

4.4 Perspectives

La stratégie de traitement proposée pour traiter le cas des UCD simples donne des résultats encourageants. Nous travaillons actuellement à la prise en compte des UCD composés. L’objectif est ici d’exploiter les règles de compositions. Ainsi, à partir des mots obtenus par la procédure d’attribution d’UCD simples et à partir des règles de composition précisant les positions potentielles que peuvent occuper les mots dans un UCD composé se calcule un score global pour chaque UCD valide. Par exemple, pour la ligne de `ReadMe` donnée en figure 5, correspondant à l’UCD composé `pos.pm;pos.eq.ra`, la procédure d’attribution UCD simples produit le résultat donné en figure 6

⁶<http://vizier.u-strasbg.fr/UCD/cgi-bin/descr2ucd>

Units	Label	Explanations
Mas/yr	pmRA	[-4418.0, 6544.2]? Prop.mot in RA*cos(dec)

FIG. 5 – Exemple d'une ligne de ReadMe correspondant à l'UCD composé `pos.pm;pos.eq.ra`.

CONCEPT	SCORES	RÔLES PARTAGÉS
<code>pos.eq.ra</code>	3	<i>hasMeasuredValue, hasAngleUnit, hasCcOriginEqRa, hasFrameTypeEq</i>
<code>pos.eq.dec</code>	3	<i>hasMeasuredValue, hasAngleUnit, hasCcOriginEqDec, hasFrameTypeEq</i>
<code>pos.pm</code>	2	<i>hasMeasuredValue, hasAngularSpeedUnit, hasMotionTypePm</i>
<code>pos.angDistance</code>	2	<i>hasMeasuredValue, hasAngleUnit</i>
...

FIG. 6 – Classement des concepts de \mathcal{O}_{UCD} obtenu par la procédure d'attribution d'UCD simples sur la ligne de ReadMe donnée en figure 5.

(`pos.eq.ra` étant mieux classé que `pos.eq.dec` par l'exploitation de la colonne Label).

L'utilisation des règles de compositions indique que les quatre mots présentés ci-dessus possèdent un code Q et peuvent donc tous apparaître en première ou seconde place dans une composition. Ainsi tous les couples composés de deux de ces termes, à savoir `pos.eq.ra;pos.eq.dec`, `pos.eq.dec;pos.eq.ra`, `pos.eq.dec;pos.pm`, `pos.pm;pos.eq.ra`, etc., sont valides. Cet exemple présente le cas le moins favorable d'utilisation de la grammaire des UCD puisque toutes les combinaisons de termes sont possibles. Cependant, dans un cas plus favorable, l'utilisation de la grammaire doit permettre d'éliminer les combinaisons impossibles. Par exemple, un mot associé à un code P (resp. S) ne peut se trouver qu'en première (resp. seconde) position dans une composition.

À partir des combinaisons potentiellement valides par rapport à la grammaire de composition, l'idée dans le calcul du score global est de trouver le couplage qui maximise le nombre de rôles différents recherchés. Ainsi, on peut noter que seul `pos.pm` possède le rôle *hasAngularSpeedUnit*. Le poids de ce mot dans la recherche d'un UCD composé est donc à majorer. Une telle construction peut être mise en lien avec la recherche de classes polythétiques, classes composées en partie de disjonction d'attributs introduits par le type UNION dans (7; 5).

Pour finir, il faut souligner qu'un point à résoudre est de pouvoir déterminer, pour une ligne de ReadMe donnée, si la recherche d'UCD concerne un UCD simple ou un UCD composé.

5 Conclusion

Cet article présente une méthodologie pour la construction manuelle d'ontologie, à partir de connaissances collectées auprès d'experts du domaine. Cette méthodologie en 5

points, appliquée ici au domaine de l'astronomie, est suffisamment générique pour être applicable à la construction d'ontologies dans d'autres domaines. La méthode d'annotation proposée identifie dans l'ontologie le concept qui correspond *au mieux* à une courte description textuelle. L'originalité de cette méthode consiste tout d'abord à associer un ensemble de rôles potentiels à la description textuelle à annoter, ceci, en fonction des éléments textuels qui la composent et de fichiers de méta-données. Cet ensemble de rôles est ensuite exploité pour la recherche du ou des concept(s) de l'ontologie dont la définition partage un maximum de ces rôles. Cette idée semble également applicable à un contexte d'annotation plus général, comme l'annotation de documents textuels par exemple.

Références

- [1] R. al Hulou, A. Napoli et E. Nauer. Une mesure de similarité sémantique pour la classification de documents par le contenu. In J. Euzenat et B. Carré, éditeurs, *Langages et Modèles à Objets (LMO'04)*, volume 10 of *RSTI - L'objet*, pages 217–230. Hermes, Paris, 2004.
- [2] G. Antoniou et F. van Harmelen. *A Semantic Web Primer*. The MIT Press, Cambridge, Massachusetts, 2004.
- [3] N. Aussenac-Gilles, B. Biébow B. et S. Szulman. D'une méthode à un guide pratique de modélisation des connaissances à partir de textes. In F. Rousselot, éditeur, *Actes des 5èmes journées Terminologie et Intelligence Artificielle (TIA 2003)*, pages 41–53. ENSAIS, 2003.
- [4] F. Baader, D. Calvanese, D. McGuinness, D. Nardi et P. Patel-Schneider, éditeurs. *The Description Logic*

- Handbook*. Cambridge University Press, Cambridge, UK, 2003.
- [5] E. Bertino, G. Guerrini, I. Merlo et M. Mesiti. An Approach to Classify Semi-Structured Objects. In R. Guerraoui, éditeur, *Proceedings of ECOOP'99, Lisboa (Portugal)*, volume 1628 of *Lecture Notes in Computer Science*, pages 416–440. Springer, 1999.
- [6] D. Bourigault, N. Aussenac-Gilles et J. Charlet. Construction de ressources terminologiques ou ontologiques à partir de textes : un cadre unificateur pour trois études de cas. *Revue d'Intelligence Artificielle (RIA)*, " *Techniques Informatiques et structuration de terminologies*, 18(1/2004) :87–110, 2004.
- [7] V. Christophides, S. Abiteboul, S. Cluet et M. Scholl. From structured documents to novel query facilities. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1994.
- [8] S. Derriere, N. Gray, R. Mann, A. Preite Martinez, J. McDowell, T. McGlynn, F. Oschenbein, P. Osuna, G. Rixon et R. Williams. Ucd (unified content descriptor) - moving to ucd1+. version 1.06, ivoa proposed recommendation 2004-10-26. <http://www.ivoa.net/documents/latest/ucd.html/>, 2004.
- [9] M. d'Aquin, C. Bouthier, S. Brachais, J. Lieber et A. Napoli. Knowledge editing and maintenance tools for a semantic portal in oncology. *International Journal of Computer Human Studies*, 62(5) :619–638, 2005.
- [10] R. Ducournau, J. Euzenat, G. Masini et A Napoli. *Languages et modèles à objets – Etat des recherches et perspectives*, volume 19 of *Didactique*. INRIA Rocquencourt, 78153 Le Chesnay, 1998.
- [11] D. Fensel, J. Hendler, H. Lieberman et W. Wahlster, éditeurs. *Spinning the Semantic Web*. The MIT Press, Cambridge, Massachusetts, 2003.
- [12] T. R. Gruber. *Formal Ontology in Conceptual Analysis and Knowledge Representation*, chapitre Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Kluwer Academic Publishers, 1993.
- [13] V. Haarslev et R. Möller. RACER System Description. In *First International Joint Conference on Automated Reasoning, IJCAR'2001*, volume 2083 of *Lecture Notes in Computer Science*, pages 701–706. Springer-Verlag, 2001. <http://www.racer-systems.com/>.
- [14] M. Horridge, H. Knublauch, A. Rector, R. Stevens et C. Wroe. *A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools. Edition 1.0*. University Of Manchester, 2004.
- [15] A. Preite Martinez, S. Derriere, N. Gray, R. Mann, J. McDowell, T. Mc Glynn, F. Ochsenbein, P. Osuna, G. Rixon et R. Williams. The ucd1+ controlled vocabulary, version 1.11, ivoa recommendation 2005-12-31. <http://www.ivoa.net/Documents/latest/UCDlist.html/>, 2005.
- [16] Natalya F. Noy et Deborah L. McGuinness. Ontology development 101 : A guide to creating your first ontology. Technical report, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, 2001.
- [17] F. Ochsenbein. Astronomical Catalogues and Tables Adopted Standards version 2.0, 2000. <http://vizier.u-strasbg.fr/doc/catstd.htx/>.
- [18] B. Omelayenko. Learning of Ontologies for the Web : the Analysis of Existent Approaches. In *Proceedings of the International Workshop on Web Dynamics, 8th International Conference on Database Theory, ICDT'01*, 2001.
- [19] A. Richard. Construction d'une ontologie de descripteurs en astronomie à partir de tables de données. Mémoire de DEA, LORIA, Juin 2005.
- [20] B. Safar, H. Kefi et C. Reynaud. Ontorefiner, a user query refinement interface usable for semantic web portals. In *Application of Semantic Web Technologies to Web Communities Workshop, 16th European Conference on Artificial Intelligence*, 2004.
- [21] S. Staab et R. Studer, éditeurs. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2004.
- [22] S. Szulman, B. Biébow B. et N. Aussenac-Gilles. Structuration de terminologies à l'aide d'outils d'analyse de textes avec terminae. In *TAL*, volume 43, pages 103–128, 2002.
- [23] M. Uschold et M. Gruninger. Ontologies : Principles, methods and applications. *Knowledge Engineering Review*, 11(2) :93–155, 1996.
- [24] M. Uschold et M. King. Towards a Methodology for Building Ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*, 1995.