Recherche des composants logiciels référencés par un modèle ontologique

Anis Masmoudi*, **— Roger Champagne **— Gilbert Paquette *

**LICEF : Laboratoire Informatique Cognitive et Environnements de Formation, Montréal, Québec, Canada.*
*** ÉTS : École de technologie supérieure, Montréal, Québec, Canada.*
{anis.masmoudi, gilbert.paquette}@licef.teluq.uqam.ca,
roger.champagne@etsmtl.ca

RÉSUMÉ. *Nos travaux visent à développer les systèmes sous la forme de composants en les agrégeant à l'aide d'un modèle décrivant leurs principales caractéristiques. Nous avons récemment proposé une structure de métadonnées des composants logiciels intitulée SOCOM (SOftware COmponent Metadata). Nous l'utilisons pour construire une banque de composants qui servira de support pour les futurs développements logiciels. Le présent article décrit une implémentation de SOCOM à l'aide d'un modèle ontologique. Nous présentons d'abord une partie de ce modèle sous forme graphique dans l'éditeur d'ontologie MOT+OWL et sa visualisation sous forme textuelle dans l'outil Protégé. Ensuite, nous décrivons une architecture distribuée qui utilise cette ontologie SOCOM pour référencer et rechercher des composants dans le système TELOS*[1] *en cours de construction au sein du réseau LORNET (Learning Object Repositories' NETworks). À la fin, nous montrons comment se fait l'interrogation d'une ontologie de composants utilisant les spécifications DIG et les moteurs d'inférences qui s'appuient sur les logiques descriptives.*

ABSTRACT. *Our work aim to characterize software components using a set of metadata that covers their main specifications. In an earlier work, we have suggested a platform independent set of metadata, called SOCOM (SOftware COmponent Metadata), to build a components' warehouse. We now implement SOCOM using ontology to represent and structure the metadata. We designed graphically the SOCOM ontology using the MOT+ OWL editor and we manage it with the Protégé editor. Afterwards, we describe our distributed architecture that uses the SOCOM ontology to reference and search components within the TELOS system being built into the LORNET Project (Learning Object Repositories' NETworks). Finally, we explain request and response operations from distributed SOCOM ontologies using DIG specifications and DL (Description Logics) reasoners DIG compliant.*

MOT-CLÉS : *Composant, Ontologie, Développement basé sur les composants, Modélisation, Métadonnées de composants logiciels, moteur d'inférence, les spécifications DIG.*

KE YWORDS: *Component, Ontology, Component based development, Modeling, Software component metadata, DL reasoners, DIG specifications.*

Catégorie de la soumission : Chercheur

[1] Voir (Paquette, Rosca et al, 2006) et (Magnan et Paquette, 2006)

1. Introduction

La construction des systèmes informatiques et les développements logiciels se basent de plus en plus sur le composant comme unité de base. Ces composants doivent être référencés pour qu'ils puissent être réutilisés. Ce référencement doit couvrir plus d'un aspect afin d'obtenir une description globale du composant et pouvoir décider des différents scénarios de son agrégation avec d'autres composants. Plusieurs plateformes industrielles référencent les composants en tenant compte essentiellement de leurs caractéristiques techniques. Cependant, d'autres caractéristiques non fonctionnelles et métiers du composant peuvent informer des possibilités de sa réutilisation. Ainsi, nous retrouvons dans ces concepts le domaine du développement à base de composants (Allen & al, 1998). C'est dans ce même contexte que s'inscrit l'initiative du projet LORNET[2]. Ce dernier vise plus loin que le développement d'une simple application d'apprentissage. Un de ses objectifs principaux est la mise en place d'un système générateur de systèmes d'apprentissage. Ces systèmes à leur tour génèreront des applications spécifiques conformément à l'une des disciplines scientifiques. Ces applications sont accessibles par les tuteurs (chargés de cours) pour assister les apprenants lors du suivi d'un cours à distance. Chaque apprenant utilise des ressources pédagogiques et en fournit d'autres. Cette cascade, supportée par le système TELOS (*Tele-Learning Operating System*) du projet LORNET, est conçue par agrégation de composants logiciels et de ressources pédagogiques existantes.

Dans cet article, nous rappelons la structure des métadonnées des composants logiciels SOCOM (SOftware COmponent Metadata) que nous avons présentée ailleurs (Masmoudi et al., 2005). Nous présentons brièvement la nouvelle structure étendue de SOCOM (Section 2.). Puis nous expliquons la modélisation graphique du modèle ontologique de SOCOM à la section 3. La section 4. décrit les outils et les langages nécessaires pour la modélisation et l'implémentation OWL de ce modèle ontologique. Ensuite, nous introduisons au début de la section 5. les logiques descriptives et les spécifications DIG (Description Logics Implementation Group) qui permettent la représentation en XML de cette ontologie. Dans la sous section 5.3, nous présentons un exemple concret qui détaille une requête et une réponse décrites en DIG pour interroger notre ontologie de SOCOM. Après, nous utilisons la sous section 5.4 pour décrire notre architecture distribuée des banques ontologiques des composants qui s'appuie sur les paradigmes précédemment définis. À la fin, nous rappelons les idées maîtresses de cet article et nos perspectives de développement.

Les travaux décrits dans cet article ont été réalisés avec l'outil MOT+[3]OWL[4] pour la modélisation graphique des ontologies et l'outil Protégé[5] pour la gestion des

---

[2] LORNET: www.lornet.org
[3] MOT+ (Modélisation par Objets Typés): http://www.licef.teluq.uquebec.ca/realisations/

fichiers OWL. Deux moteurs d'inférence compatible avec les spécifications DIG qui se basent sur la logiques descriptive sont utilisés; Pellet[6] et Fact++[7].

2. La structure SOCOM des composants logiciels

Nous avons développé antérieurement la structure de métadonnées des composants logiciels SOCOM. Comme nous l'avons définit auparavant, cette structure couvre trois aspects d'un composant logiciel. Elle identifie des attributs techniques, des attributs non fonctionnels et des attributs métiers. Par cette caractérisation, nous avons voulu rendre accessible la description d'un composant logiciel par différents types d'intervenants tels qu'analyste, concepteur, agrégateur (intégrateur) et programmeur.

Nous avons raffiné cette structure par l'ajout d'un autre type d'attributs avec quelques ajustements sur leur nom et leur classification. Nous voulons par ces changements obtenir un modèle abstrait indépendant de la plateforme des composants. Autrement dit, à partir de notre modèle, nous voulons être capables de décrire plusieurs types de composants. Aussi nous espérons, générer les spécifications fonctionnelles et techniques de ce même composant dans son langage d'implémentation. Par exemple, les services d'un composant développé en Java seront décrits par instanciation de notre modèle. Lors d'un processus d'agrégation de ce composant avec un autre développé aussi en Java, nous voulons générer automatiquement les APIs[8] des deux composants et les réutiliser dans un scénario d'agrégation spécifique. En résumé, pour chaque composant, on distingue :

Les ***attributs fonctionnels*** décrivent le comportement interne et externe du composant. En effet, d'une part ils décrivent les services offerts par les composants pour servir son environnement externe, soit le contexte où il est développé. D'autre part, ils décrivent les services requis par le composant pour assurer son fonctionnement interne et délivrer ses responsabilités. D'autres attributs du composant sont couverts par notre structure comme l'identificateur local et global du composant, le nom du composant, l'endroit de son paquetage technique, le degré d'accessibilité, le niveau de couplage, la couche logicielle à laquelle il appartient, son modèle de composant UML et la catégorie de l'agrégation du composant.

Les ***attributs non fonctionnels*** sont des aspects dits non techniques (aussi nommés attributs de qualité dans la communauté d'architecture logicielle) du composant mais qui ont un impact sur une réutilisation potentielle de ce composant

---

[4] OWL : Un langage d'ontologie web, http://www.w3.org/2004/OWL/
[5] Protege : Un éditeur d'ontologie en OWL, http://protege.stanford.edu/
[6] Pellet : Un moteur d'inférence en Java, http://www.mindswap.org/2003/pellet/
[7] Fact++ : Un moteur d'inférence basé en C++, http://owl.man.ac.uk/factplusplus/
[8] API (Application Programming Interface): Interface des programmes en Java

dans un domaine autre que celui de sa création. Nous parlons des aspects tels la fiabilité, la sécurité, l'extensibilité, le ou les droits d'utilisation, la version, la licence, la performance, l'intégrité, etc.

Les ***attributs techniques*** couvrent certains aspects de l'environnement du composant au niveau des plates-formes de développement, de configuration, de modélisation, de la persistance (données) et des langages de programmation (OMG, 2003). Nous avons ajouté aussi des attributs de déploiement afin d'automatiser ultérieurement le déploiement du composant une fois le processus d'agrégation stabilisé. Ce volet ne sera pas discuté ici car il dépasse le cadre de cet article, demandant trop de développement vu son importance dans la faisabilité des agrégations et le déploiement des agrégats.

Les ***attributs métiers***, aussi nommés attributs du domaine par plusieurs, visent à capter des informations sur le contexte du composant. Dans notre cas, nous décrivons des composants logiciels à vocation pédagogique. Par la suite, nos attributs métiers sauvegardent les métadonnées pédagogiques sous un format standard IEEE LOM (L.T.S.C., 2002).

Dans ce qui suit, nous exposons une partie du modèle ontologique de SOCOM vu que nous n'avons pas assez d'espace pour le modèle au complet.

3. Le modèle ontologique dans MOT+ OWL

Avant de parler du modèle ontologie de SOCOM, nous rappelons la définition d'une ontologie. Une ontologie est un vocabulaire commun que se donnent des usagers ayant besoin de partager des informations dans un domaine. L'ontologie regroupe des définitions lisibles en machine, des concepts de base de ce domaine ou classes, des relations entre ces concepts et des axiomes énonçant des propriétés de ces classes et de ces relations (Smith, M. K et al, 2004). Dans notre cas le domaine étudié est le développement des logiciels à base de composants et les informations que nous voudrions partager sont les classes de métadonnées des composants et leurs relations.

Pour comprendre le modèle OWL de SOCOM, nous expliquons rapidement quelques formes et notations graphiques des modèles ontologiques de l'outil en question. À l'origine, l'éditeur graphique MOT, développé au LICEF (LICEF, 1992) est conçu pour concevoir des modèles pédagogiques et des modèles de connaissances de toutes sortes (Paquette, Léonard et al 2006). Récemment, l'éditeur a été spécialisé en un outil MOT+OWL permettant de construire graphiquement des ontologies selon le standard OWL (Ontology Web Language). Cet éditeur utilise trois formes graphiques mises en relation par des liens typés. Les rectangles représentent les concepts ou classes, les hexagones représentent les propriétés entre les relations, les rectangles aux coins coupés représentent les individus ou instances

Divers types de liens sont utilisés entre ces trois types d'entité et des étiquettes sur les objets représentent les axiomes.

La Figure 1 en présentent quelques uns : le lien R entre une propriété et sa classe d'origine (domaine) ou cible (co-domaine), le lien S entre une classe et ses sous-classes, le lien DISJ indiquant que deux classes sont disjointes, le lien I entre une classe une instance qui en fait partie, et un axiome indiquant qu'au moins deux individus du co-domaine sont associés par la propriété à chaque individu du domaine.

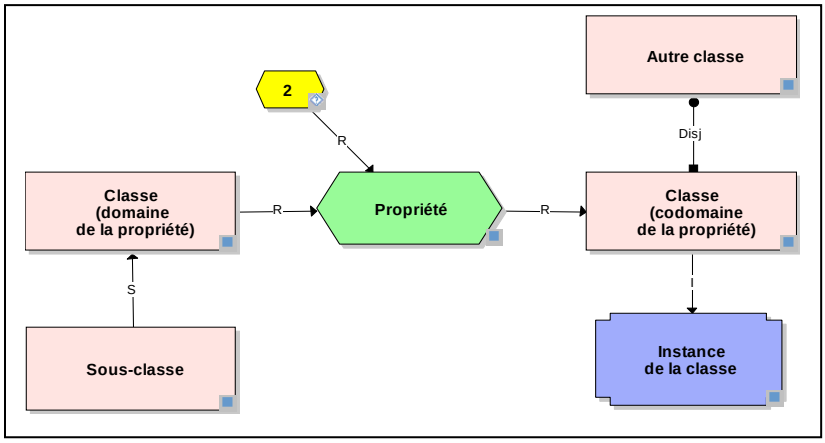

Figure 1. Les formes graphiques des objets et de certains lien dans MOT+OWL

4. Le modèle ontologique de la structure SOCOM

Dans ce qui suit, nous présentons la structure générale de SOCOM sous la forme d'un modèle ontologique en MOT+OWL (sous section 4.1). Ensuite, nous exportons ce modèle en un fichier XML obéissant au standard OWL. Celui-ci peut être lu, manipulé et instancié dans l'outil Protégé (sous section 4.2).

4.1. Le modèle MOT de l'ontologie SOCOM

La Figure 2 montre un modèle de premier niveau de l'ontologie SOCOM. En fait, ce modèle reprend la description textuelle de la section 2. Il ajoute le concept (aussi appelé Classe) agrégat qui se compose, au minimum, de deux composants logiciels. Pour élaborer en MOT+ le reste du modèle, nous explorons l'intérieur de chaque concept et nous détaillons le contenu par d'autres sous-modèles. Vue que ces sous-modèles de deuxième et de troisième niveau sont complexes et qu'ils ne seront pas

lisibles dans les figures, nous les représentons en partie dans la sous-section suivante par les interfaces usager de l'outil Protégé.

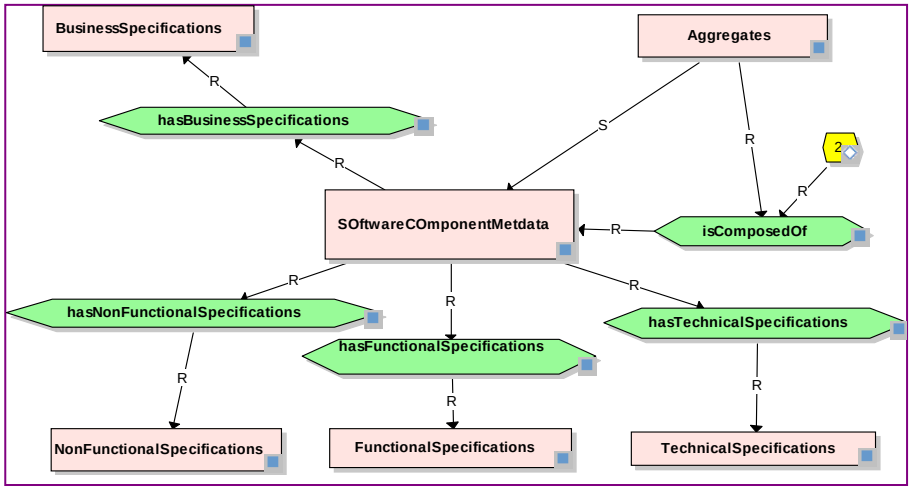

Figure 2. Le modèle ontologique de SOCOM (premier niveau)

4.2. SOCOM OWL sous Protégé

La Figure 3 montre un aperçu sur des instances des objets graphiques décrit dans la section 3. En effet, les classes de l'ontologie SOCOM occupent le coté gauche de la figure. Ces classes sont reliées par des objets propriétés représentés en partie au milieu. À droite, nous retrouvons quelques instances (individus) qui représentent quelques composants de la banque en construction qui sera utilisée dans TELOS.

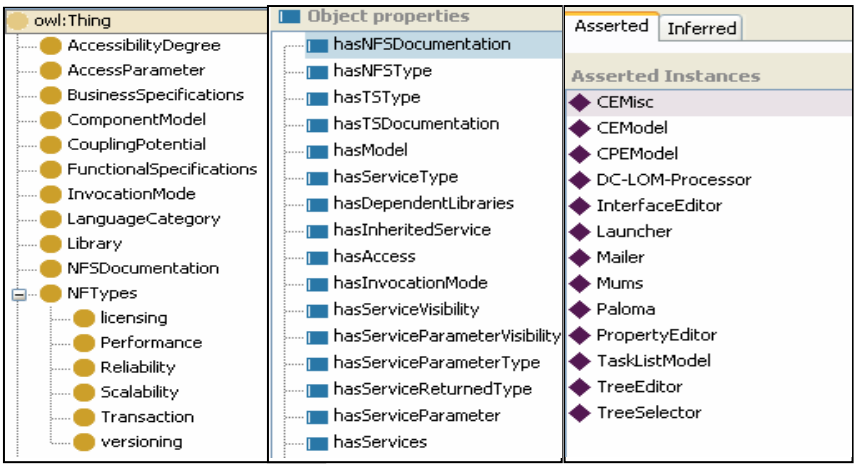

Figure 3. Une partie de l'ontologie SOCOM sous Protégé OWL

5. Les spécifications de DIG pour interroger l'ontologie de SOCOM

5.1. Les logiques descriptives

Les logiques descriptives (DL) sont une famille de langages de représentation des connaissances. Ces logiques peuvent être utilisées pour représenter, à l'aide d'un formalisme structuré et bien formé, la terminologie des connaissances d'une application d'un domaine spécifique. Les pierres d'assise de ce type de formalisme sont les concepts, les propriétés et les individus. Les concepts décrivent les propriétés communes d'une collection d'individus. Ils peuvent être considérés comme des prédicats de premier ordre qui sont interprétés comme un ensemble d'objets. Les propriétés sont des relations binaires entre les objets. Des axiomes tels que « à tout individu d'une classe correspond au moins un individu d'une autre » ou « aucun individu n'appartient à la fois à la classe A et à la classe B » peuvent être ajoutés pour décrire un domaine et façon précise et guider un moteur d'inférence dans les déductions qui lui permettent d'exécuter une requête d'un usage. Notre ontologie de SOCOM utilise le langage OWL DL. (Smith et al, 2004)

5.2. Les spécifications DIG

Les moteurs d'inférence qui s'appuient sur les logiques descriptives sont de plus en plus utilisés pour raisonner sur des sujets et des ressources du Web sémantique. Pour permettre à un client d'interagir avec différents moteurs d'une manière standard, une interface standard et commune est fortement souhaitable. Le groupe DIG[9] œuvre dans la conception des systèmes compatibles avec les logiques descriptives. Une de ses activités consiste à développer une interface normalisée en XML pour les systèmes qui utilisent ces logiques en leur fournissant une API de base. L'interface DIG est une nouvelle norme qui permet l'accès au moteur d'inférence via des interfaces web utilisant le protocole HTTP. Cette interface est supportée par la majorité des moteurs d'inférence et facilite la construction des composants logiciels réutilisables. Des moteurs d'inférences comme Pellet, Racer et Fact++ sont compatibles avec les spécifications DIG.

Ces spécifications se composent essentiellement d'un schéma XML décrivant le formalisme du langage descriptif. Deux opérations sont disponibles pour construire et questionner une ontologie. L'ontologie est construite par l'opération « tells» et interrogée par l'opération « asks ». Pour plus de détails, les spécifications DIG de version 1.1 sont disponibles dans (Bechhofer, 2003).

Dans la sous section suivante, nous décrivons une architecture distribuée des banques de composants qui utilise les paradigmes et les spécifications précédemment décrits : SOCOM (section 2.), les ontologies (section 4.), les

[9] DIG: http://dl.kr.org/dig/

spécifications DIG (sous section 5.2) et les moteurs d'inférences compatibles avec les logiques descriptives.

5.3. Un exemple de requête et de réponse DIG

Techniquement, notre ontologie de SOCOM renferme sa structure et ses données dans un fichier OWL. Plusieurs interfaces de programmation en Java et en C++ sont disponibles pour interroger des fichiers OWL (API OWL de Protégé[10], Jena API[11], etc.). Cependant, nous ne voulons pas nous restreindre à une implémentation spécifique pour interroger notre ontologie, ce qui nous a conduit à utiliser les spécifications DIG (voir section 5.2). Nous avons transformé à l'aide de Protégé notre fichier OWL en fichier XML qui est décrit par le concept « tells ». Puis nous avons préparé nos requêtes sous la forme de fichier XML construit avec l'opération « asks ». Pour l'exécution de nos requêtes, nous spécifions à n'importe lequel des moteurs d'inférences compatibles DIG ces deux fichiers XML. Ce dernier nous retourne un autre fichier XML réponse construit avec le concept « responses ». La Figure 6 décrit sommairement ce scénario.

Nous avons choisi un exemple simple et fréquemment utilisé dans notre contexte pour expliquer les structures « asks » et « responses ». La Figure 4 illustre une requête qui demande la liste des services des composants de l'ontologie. Dans la requête, il faut spécifier l'identificateur de la requête «qTelosComponentServices » et la propriété « hasServices » qui rattache les concepts « TechnicalSecifcation » et « Services ».

La réponse à cette requête est le fichier XML de la Figure 5. Ce dernier est retourné par le moteur d'inférence et montre les résultats trouvés sous la forme d'individus en langage ontologique ou bien les instances en langage objet. La liste retournée est un ensemble de paires. En effet, le premier individu de la paire est l'instance du concept source « TechnicalSpeicifcation » alors que le deuxième est l'instance du concept « Services ». Le tout est encapsulé dans la balise « responses ». Cette réponse doit rappeler l'identificateur de la requête.

```
<asks xmlns="http://dl.kr.org/dig/2003/02/lang">
   <relatedIndividuals id="qTelosComponentServices">
        <ratom name="hasServices"/>
   </relatedIndividuals>
</asks>
```

Figure 4. Exemple d'une requête DIG utilisant le concept « asks »

[10] API OWL de Protege : http://protege.stanford.edu/plugins/owl/api/
[11] API de Jena : http://jena.sourceforge.net/

```
<responses xmlns="http://dl.kr.org/dig/2003/02/lang">
  <individualPairSet id="qTelosComponentServices">
    <individualPair>
      <individual name="TechnicalSpecifications_CPEModel"/>
      <individual name="getCPEModelService"/>
    </individualPair>
    <individualPair>
      <individual name="TechnicalSpecifications_TreeEditor"/>
      <individual name="performTreeEditorService"/>
    </individualPair>
      …
  </individualPairSet>
</responses>
```

Figure 5. Exemple d'une réponse DIG utilisant le concept « responses »

5.4. Architecture distribuée des banques de composants logiciels

Les ontologies des composants logiciels qui implémentent la structure SOCOM peuvent être déployées sous différents serveurs web. Pour assurer leur gestion en termes d'exploration et d'instanciation des concepts, un administrateur peut utiliser un outil comme Protégé ou Swoop[12]. Le modèle de la Figure 6 montre un scénario de recherche sur les ontologies géographiquement distribuées qui référencent les composants logiciels. Autrement dit, nous pouvons distribuer sur différents serveurs les ontologies de SOCOM, les requêtes DIG, les moteurs d'inférences et le code de traitement qui implémente les interfaces DIG en plusieurs langages (Serveur de recherche DIG –SOCOM).

À cet effet, nous avons développé un client Java qui utilise une API Java des spécifications DIG (DIG I., 2003). Nous avons transformé l'ontologie du format OWL en format DIG basé sur l'opération « tells ». Notre client utilise le fichier XML des requêtes basées sur l'opération « asks » et l'ontologie traduite en un fichier XML construit avec le concept « tells ». Ces deux fichiers sont transmis par HTTP à un moteur d'inférence compatible DIG (Pellet dans notre cas). Ce dernier exécute la requête et retourne une réponse sous forme d'un fichier XML encapsulé dans le concept « responses ». Les libraires Java des spécifications DIG sérialisent la réponse XML en objet Java « DocumentResponse ». En utilisant cet objet, le client Java extrait les informations nécessaires pour les afficher à l'usager ou les utiliser dans un processus d'agrégation des composants logiciels.

Les avantages d'une telle architecture sont dues aux requêtes DIG et à la description DIG des ontologies de SOCOM (Par l'opération « tells »). Ces denières sont indépendantes des plateformes d'inférence et du langage d'implémentation du client. Aussi, nous ne sommes pas contraints d'utiliser un moteur d'inférence spécifique. Tous les moteurs qui sont compatibles DIG peuvent répondre à nos requêtes DIG (sous section 5.2). Nous ajoutons que même nos requêtes et nos

[12] Swoop: Editeur OWL, www.mindswap.org/2004/SWOOP/

réponses utilisent un format XML compatible DIG, par conséquent nous pouvons créer n'importe quel type de client dans n'importe quel langage de programmation dés qu'il permet la manipulation des requêtes et des réponses DIG dans son code natif en termes de sérialisation et de désérialisation.

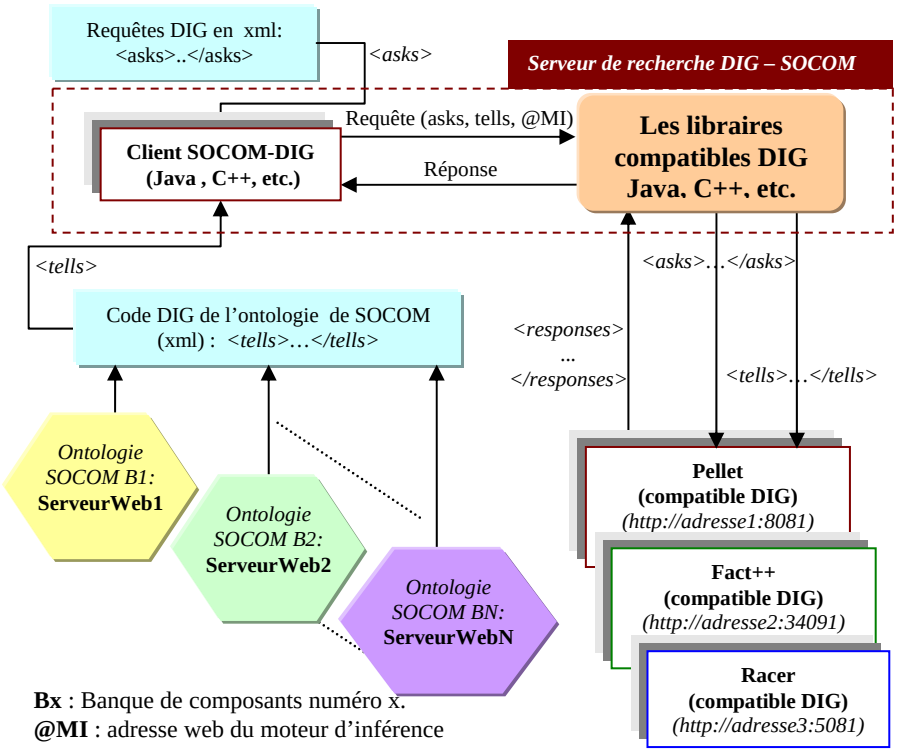

Figure 6. L'architecture distribuée des banques des composants(SOCOM, DIG)

6. Conclusion et perspective

Cet article présente l'évolution de nos travaux sur l'indexation des composants logiciels et leurs méthodes d'agrégation. Nous avons commencé par la recherche d'une structure commune pour décrire les composants logiciels en nous appuyant sur les spécifications fonctionnelles, non fonctionnelles, techniques et métiers des composants logiciels. Il est intéressant de rappeler que nous avons pensé à une implémentation relationnelle de cette structure SOCOM. Les tests avec première implémentation relationnelle nous a garanti une simplicité au niveau de

l'implémentation de SOCOM et du développement de l'application par Java et MySQL[13], mais nous fournit peu de sémantique sur les composants logiciels. Une représentation ontologique de SOCOM nous a permis de classifier les composants en se basant sur les couches logiciels, le degré d'accessibilité du composant (boite blanche, boite grise et boite noire), etc. (Masmoudi et al., 2005). Techniquement, ces caractéristiques ont été traduites en propriétés dans l'ontologie de SOCOM et nous permettent des déductions et du raisonnement via les moteurs d'inférence. En fait, tous ces résultats auraient pu être obtenus par notre première implémentation relationnelle de SOCOM sauf qu'il nous aurait fallu développer toute la logique descriptive que les moteurs d'inférences nous offrent. C'est pour cela que nous adoptons le modèle ontologique de SOCOM. Nous profitons aussi de l'apport des spécifications DIG vu qu'elles nous permettent de formuler des requêtes en XML indépendantes des moteurs d'inférences compatibles DIG.

Nous avons manipulé les ontologies à l'aide des éditeurs MOT+ OWL et Protégé. En effet, nous utilisons la simplicité des formes graphiques des modèles MOT+ OWL en début de conception pour construire l'ontologie de SOCOM. Ensuite, nous l'avons exportée en format OWL pour la peupler dans Protégé. Ce dernier nous génère dynamiquement des formulaires en se basant sur les propriétés des classes conçues dans MOT+OWL. En plus, nous pouvons vérifier avec Protégé aussi bien la consistance que la classification de nos classes en nous connectant à un des moteurs d'inférences disponibles via HTTP. L'utilisation de ses deux outils nous permet de gérer des ontologies de bout en bout, depuis la conception initiale jusqu'au raisonnement.

Dans nos travaux futurs, nous essayerons d'intégrer des règles d'agrégation par le développement d'une ontologie d'agrégation. En effet, nous avons catégorisé les agrégations des composants logiciels en nous appuyant sur trois caractéristiques des composants. Ultérieurement, nous ajusterons cette catégorisation et nous étudions l'apport des ontologies pour aider un ingénieur ou un technologue dans le processus de construction des systèmes d'apprentissages à distance par agrégation des composants intégrés dans la banque de ressource du système TELOS.

Remerciements

Nous tenons à remercier le CRSNG[14] d'avoir financé le projet LORNET et l'équipe du laboratoire LICEF qui développe actuellement le système TELOS.

7. Bibliographies

Allen, P., Frosts, S., Tackling the Key Issues in CBD. In Select Software Tools, March 1998.

[13] MySQL : SGBDR, www.mysql.org

[14] CRSNG : Conseil de recherches en Sciences Naturelles et en Génie du Canada

Bechhofer S., DIG Specifications, version 1.1, disponible sur l'adresse http://dl-web.man.ac.uk/dig/2003/02/interface.pdf , University of Manchester, 2003

Description Logics Implementation Group (DIG), DIG Interfaces, from http://sourceforge.net/projects/dig , 2005.

DIG Interfaces, disponible sur le site de sourceforge à l'adresse : http://dig.sourceforge.net/, 2003.

Learning Technologies Standards Committee of the IEEE, Draft Standard for Learning Object Metadata, IEEE, july 2002.

LICEF, Laboratoire en Informatique Cognitif et Environnements de Formation, http://www.licef.teluq.uquebec.ca/ , centre de recherche fondé en 1992 ((page consultée le 17 avril 2006).

Magnan, F. and G. Paquette TELOS: An ontology driven eLearning OS. Proceeding of the SOA-AIS-2006 Worskhop, Dublin, Ireland, June 2006

Masmoudi, A., Paquette, G., & Champagne, R. (2005). Agrégation de Composant logiciel Dirigée par les Métadonnées. Papier présenté à l'atelier OCM de la conférence INFORSID, Grenoble-France, mai 2005.

Object Management Group, *Deployment and Configuration of Component-based Distributed Applications Specification*, Draft Adopted Specification ptc/03-07-02, juin 2003, disponible http://www.omg.org/docs/ptc/03-07-02.pdf (page consultée le 24 janvier 2005).

Paquette, G., La modélisation par objets typés: une méthode de représentation pour les systèmes d'apprentissage et d'aide a la tâche. Sciences et techniques éducatives , France, avril 1996.

Paquette G., Léonard M., Lundgren-Cayrol K., Mihaila S. et Gareau D. Learning Design based on Graphical Knowledge-Modeling , Journal of Educational technology and Society ET&S , Special issue on Learning Design, January 2006 and Proceedings of the UNFOLD-PROLEARN Joint Workshop, Valkenburh, The Netherlands, September 2005 on Current Research on IMS Learning Design, 2006.

Paquette, G., Rosca. I, Mihaila S. and Masmoudi A. TELOS, a service-oriented framework to support learning and knowledge Management in S. Pierre (Ed) E-Learning Networked Environments and Architectures: a Knowledge Processing Perspective, Springer-Verlag, 2006 in press.

Selic, B., The pragmatics of model-driven development. Software, IEEE, 20(5), 19-25. 2003.

Smith, M. K., Welty, C., & McGuinness, D. L., OWL Web Ontology Language Guide (W3C Recommendation), 2004, disponible à l'adresse http://www.w3.org/TR/owl-guide/ .