

Le modèle en Y pour la réutilisation dans les bases de connaissances

Mourad OUSSALAH, Karima MESSAADIA
LGI2P/EMA-EERIE

Parc scientifique Georges Besse - 30000 Nîmes, France

Email : oussalah@eerie.fr, messaadi@eerie.fr.

Résumé: Construire des applications à partir de composants préexistants, permet de minimiser le temps de développement, et de faciliter leur évolution et leur maintenance. Les approches actuelles dans le domaine du génie logiciel comme dans celui des systèmes d'information s'orientent vers la construction de bibliothèques de composants réutilisables. Les concepts communément utilisés pour décrire une bases de connaissances sont : la tâche définition la tâche à résoudre, le PSM (Problem Solving Method) représentant la méthode de résolution et le domaine décrivant les connaissances d'un domaine particulier. Dans cet article, nous présentons le modèle en Y pour la construction de bases de connaissances par réutilisation de composants. Ce modèle est basé sur la description des trois concepts de tâche, PSM et domaine, de manière indépendante les uns des autres en utilisant la notion d'ontologie, dans un souci de réutilisation et de partage. En plus des descriptions indépendantes des trois composants (concepts), nous explicitons leurs différentes interactions en utilisant la notion de lien sémantique et de lien de transfert. Ces liens sont décrits comme des entités indépendantes des composants qu'ils lient donc pouvant être eux même réutilisés.

Mots clés: base de connaissances, modélisation objet, réutilisation, lien sémantique, lien de transfert.

1 Introduction

Construire des applications à partir de composants préexistants, permet de minimiser le temps de développement, et de faciliter leur évolution et leur maintenance. Les approches actuelles dans le domaine du génie logiciel comme dans celui des systèmes d'information s'orientent vers la construction de bibliothèques de composants réutilisables. Les problèmes souvent rencontrés par ces approches sont : comment décrire un composant (langage de description), à quel niveau de granularité le décrire, et comment structurer et indexer ces composants dans une bibliothèque, afin que l'utilisateur (développeur) puisse facilement comprendre, trouver, configurer et intégrer ces composants pour construire son application. Devant un tel attrait, l'ingénierie de connaissances s'est orientée vers la construction de bibliothèques de composants réutilisables afin d'optimiser le développement des bases de connaissances. La plupart des méthodologies existantes (CommonKADS, Protégé II, ...), décrivent une base de connaissances en utilisant les trois concepts : Tâche, PSM et domaine. Une telle séparation entre le traitement à faire, le raisonnement utilisé pour réaliser un traitement donné, et les données du domaine d'application nous permet de voir la construction d'une base de connaissances comme un assemblage de ces trois composants préalablement décrits et testés. En recensant les approches orientées bibliothèque de composants qui ont été définies que ce soit en ingénierie de connaissances ou en génie logiciel, on découvre qu'elles décrivent souvent des composants de même type ou bien spécifiques à une application (bibliothèque de PSM de Benjamins, bibliothèque d'implantations de PSM dans ProtégéII ...), et peu d'entre elles décrivent la méthodologie qui permet de les utiliser/ réutiliser, et la manière de les assembler (approche utilisateur).

Nous pensons qu'une bibliothèque qui intégrerait des composants de différents types et décrits à différents niveaux de granularités tel que des composants de modélisations ou bien de code (*primitive d'implantation*), ainsi que les informations permettant de les assembler élargirait le champ de son utilisation/réutilisation. L'utilisateur pourra choisir le composant voulu, au degré de spécialisation voulu. Nous décrivons dans cet article une méthodologie de modélisation de bases de connaissances qui intègre les trois composants, PSM (méthode de résolution), tâche, et domaine. Nous avons décrits ces trois composants de manière indépendante les uns des autres en utilisant la notion d'ontologie, dans un souci de réutilisation et de partage. En plus des descriptions indépendantes des trois composants, nous avons explicité leurs différentes interactions en utilisant la notion de lien sémantique

et de lien de transfert. Ces liens sont décrits comme des entités indépendantes des composants qu'ils lient donc pouvant être eux même réutilisés.

Dans ce qui suit, nous allons présenter le modèle en Y puis nous nous focaliserons sur les liens sémantiques et les liens de transfert en explorant plus en détails leurs différents types, et enfin, nous aborderons la description et la gestion de la bibliothèque en définissant une méthodologie de réutilisation qui explicite aux différents utilisateurs comment décrire un composant, comment l'intégrer dans la bibliothèque et comment le réutiliser.

2 Description du modèle en Y

Les trois concepts-clés sur lesquels se basent la plupart des méthodologies sont la tâche, le PSM et le domaine. La tâche représente le quoi à savoir le but à résoudre. Une tâche peut être réalisée soit par un seul PSM, soit par un ensemble de PSM suivant un enchaînement bien défini appelé *contrôle*. Le PSM représente la *Méthode de résolution* d'une tâche donnée. Il peut être terminal représentant un code (procédure) et on le désigne par *primitive de résolution* ou bien composé de sous tâches. Un PSM composé peut avoir différentes décompositions possibles en sous tâches suivant un *contrôle*. Quant au domaine, il représente la description des connaissances d'un domaine particulier. Ces trois concepts sont liés pour construire une application. Afin d'avoir des spécifications indépendantes de chacun de ces trois concepts pour une plus grande réutilisation, nous proposons de définir leurs dépendances au niveau des liens *inter-concepts* et des liens *intra-concept*. Un lien *inter-concepts* est un lien entre deux concepts différents : tâche/PSM, PSM/domaine ou tâche/domaine. Un lien tâche/PSM permet par exemple de décrire les informations dont a besoin la tâche du PSM. Il peut aussi transférer des informations entre deux concepts telles que lier la sortie d'une tâche à l'entrée d'un PSM. Un lien *intra-concept* est un lien entre deux concepts de même type. Il peut être, un lien tâche/tâche, un lien PSM/PSM ou bien un lien domaine/domaine. Un lien tâche/tâche permet par exemple de décrire l'enchaînement des tâches. Il peut aussi transférer des informations entre les concepts qu'il lie. Ces liens seront plus détaillés dans le paragraphe suivant.

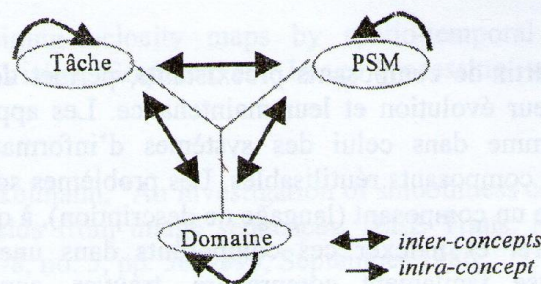


Fig 1. Le modèle en Y au niveau conceptuel

3 Le cycle de développement du modèle en Y

Dans le modèle en Y, une base de connaissances orientée tâche est représentée au niveau connaissances (knowledge level) suivant quatre niveaux de descriptions Fig. 2. Chaque niveau permet d'expliciter le niveau précédent par ajouts de spécifications.

3.1 Niveau conceptuel

Le niveau conceptuel permet d'identifier les concepts nécessaires et suffisants pour décrire un système donné. Dans le cas des bases de connaissance orientées tâche, ceci revient à définir généralement les trois concepts : tâche, PSM et domaine. Pour le modèle en Y, en plus de ces trois concepts, nous définissons deux nouveaux concepts que nous appelons lien *inter-concepts* et lien *intra-concept*. Le premier modélise les interactions qui peuvent exister entre des concepts de types différents tel que tâche/PSM et le second modélise les interactions entre des concepts de même type tel que tâche/tâche.

3.2 Niveau méta ontologique

Les cinq concepts du niveau conceptuel sont explicités à ce niveau. Ainsi, le niveau méta ontologique fournit des primitives de modélisations de ces cinq concepts.

- Méta ontologie de tâche

La tâche au niveau méta ontologique est spécifiée par son *nom* qui est un terme unique la désignant, son *entrée*, sa *sortie*, son *but* c'est à dire le problème à résoudre ainsi que les liens qui lui sont attachés.

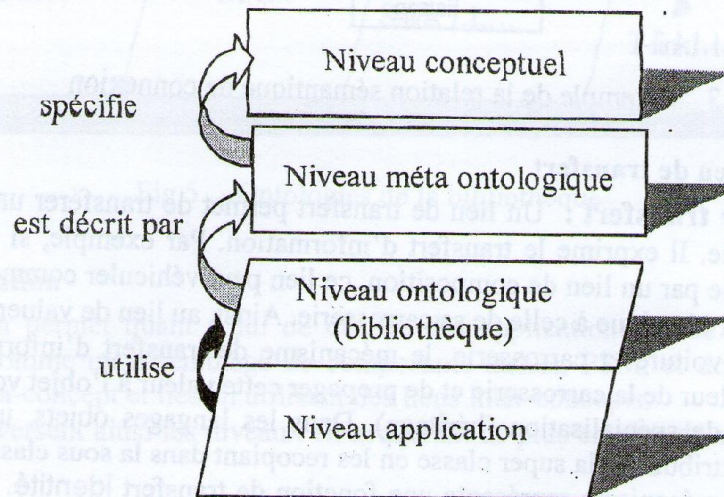


Fig 2. Les différents niveaux d'abstractions du modèle en Y

3.3 Niveau méta ontologique

Les cinq concepts du niveau conceptuel sont explicités à ce niveau. Ainsi, le niveau méta ontologique fournit des primitives de modélisations de ces cinq concepts.

- Méta ontologie de tâche

La tâche au niveau méta ontologique est spécifiée par son *nom* qui est un terme unique la désignant, son *entrée*, sa *sortie*, son *but* c'est à dire le problème à résoudre ainsi que les liens qui lui sont attachés.

- Méta ontologie de PSM

Le PSM au niveau méta ontologique est spécifié par un *nom*, terme unique le désignant, son *entrée*, sa *sortie*, sa *compétence* à résoudre un problème ainsi que les différents liens qui lui sont attachés.

- Méta ontologie de domaine

Afin de pouvoir réutiliser une description d'un domaine pour différentes tâches et différents PSM, une méta ontologie domaine doit être décrite suivant plusieurs niveaux de généralité. Généralement, l'ontologie de domaine peut être décrite par les *concepts* d'un domaine tel que *état*; Les *relations* inter-concepts telle que *relation causale*, les *caractéristiques de ces relations* telles que *transitive* ou *binaire* ainsi que les différents liens qui lui sont attachés.

- Méta Ontologie de lien sémantique

Définition d'un lien sémantique : Un lien (ou relation) sémantique décrit une relation entre deux entités(concepts) qu'elle lie. Elle est dotée d'une sémantique et d'un comportement propre permettant aux entités liées de communiquer et de collaborer. Sa sémantique pouvant exprimer une association (logique, physique, etc.) une composition, un héritage, etc. Les liens sémantiques, souvent utilisées dans la modélisation de données, par exemple dans les bases de données, permettent de garder les informations propres à la sémantique de la relation au niveau du lien et non distribués dans les concepts liés pour faciliter la réutilisation et l'évolution.

Prenant l'exemple d'un réseau d'accès. Le lien *connecté à* permet d'exprimer la liaison (connexion physique) d'un terminal à un concentrateur. Le coût de la liaison et le débit de transfert sont des caractéristiques propres à la connexion donc se trouvant au niveau du lien et non au niveau des deux concepts liés ce qui facilite leur manipulation. Ainsi, un changement du type de connexion ou des caractéristiques sera géré au niveau du lien et n'affectera pas les concepts liés.

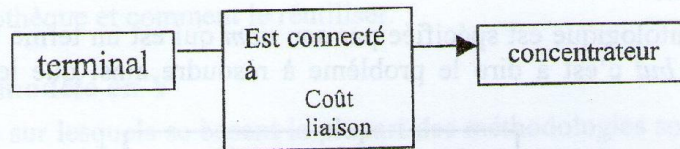


Fig 3. Exemple de la relation sémantique de connexion

• méta ontologie de lien de transfert

Définition d'un lien de transfert : Un lien de transfert permet de transférer un flux de données entre les concepts qu'il lie. Il exprime le transfert d'information. Par exemple, si on considère une voiture liée à sa carrosserie par un lien de composition, ce lien peut véhiculer comme information que la couleur de la voiture est identique à celle de sa carrosserie. Ainsi, au lieu de valuer l'attribut couleur à la fois dans les objets voiture et carrosserie, le mécanisme de transfert d'information permet de définir uniquement la couleur de la carrosserie et de propager cette valeur à l'objet voiture. Si on prend l'exemple de la relation de spécialisation (héritage). Dans les langages objets, un mécanisme qui permet de transférer les attributs de la super classe en les recopiant dans la sous classe de transfert, est défini implicitement. Ce mécanisme représente une fonction de transfert identité. Le mécanisme de transfert que nous proposons s'applique à n'importe quel type de lien sémantique. En plus, au lieu de se limiter à un simple transfert de valeurs (fonction identité), on peut appliquer une fonction de transfert sur les valeurs des attributs source pour obtenir les valeurs des attributs destination. Cette fonction est désignée au niveau d'un translateur. Ainsi, un transfert est obligatoirement composé d'au moins un translateur pour établir la propagation entre les entités liées et décrire la fonction appliquée.

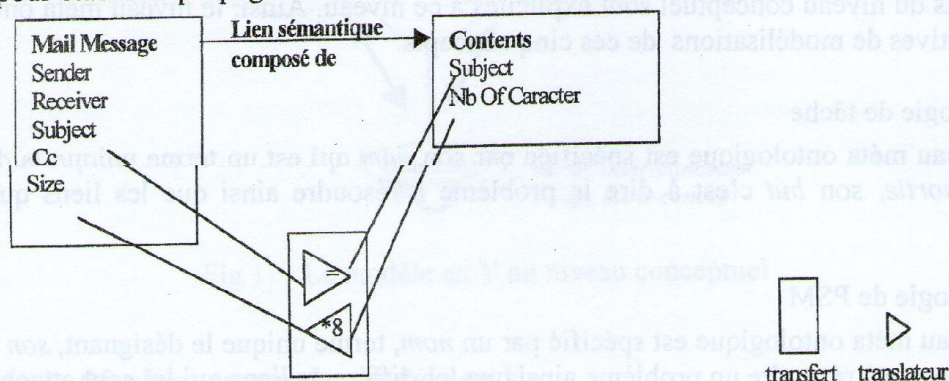


Fig 4. Exemple de lien de transfert

Définition d'un translateur : Un transfert est composé de translateurs qui permettent de décrire toutes les correspondances pouvant être réalisées entre les concepts reliés par un lien de transfert. Chaque translateur établit une correspondance entre les attributs des concepts reliés et définit la fonction de transfert à appliquer sur les valeurs des attributs source pour obtenir les valeurs des attributs destination. Plusieurs attributs d'un même concept peuvent être impliqués dans la propagation.

3.4 Niveau ontologique (ontologie de bibliothèque)

A ce niveau, différentes sortes d'ontologies de tâches, de PSM, de domaines et de liens sont décrites (exemples fig. 8). Ces composants seront stockés au niveau de la bibliothèque.

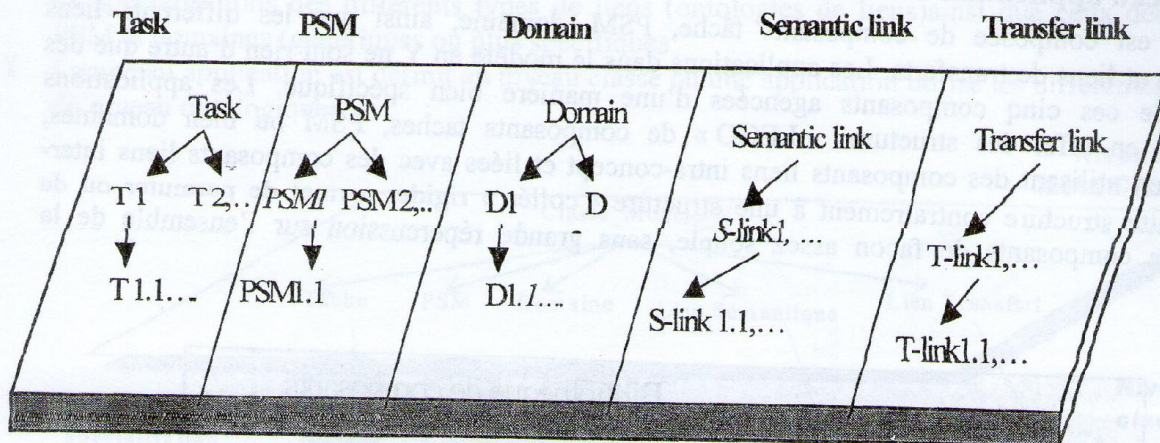


Fig 5. Ontologies de la bibliothèque

3.5 Niveau application

le niveau application permet quant à lui de décrire des applications. Dans le modèle en Y, une application est vue comme un assemblage de composants tâches, PSM, et domaines spécialisés en utilisant des liens intra-concept et liés en utilisant des liens inter-concepts.

Les concepts traversent ainsi les niveaux en acquérant de plus en plus de spécifications jusqu'au niveau implantation.

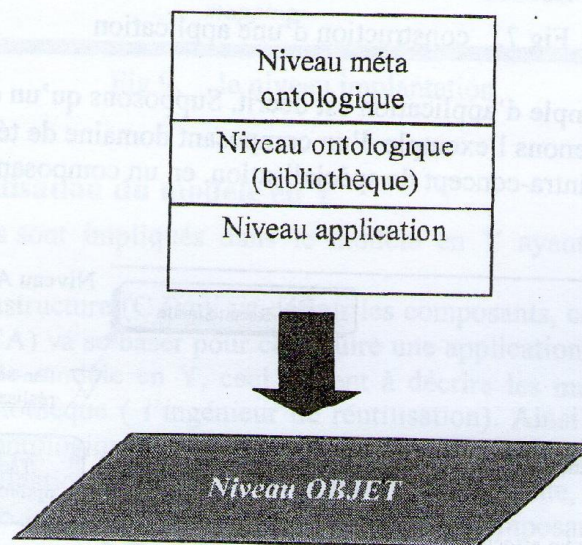


Fig 6. Le niveau implantation

3.6 Niveau objet (symbol level)

Nous utilisons un *niveau objet* comme niveau d'implantation pour tous les mécanismes qu'il offre facilitant la réutilisation (héritage, instanciation, composition et le polymorphisme). Il peut être remplacé par tout autre type de langage qui permettrait de représenter les niveaux plus hauts, tel que *logique* ou *frame*. Les niveaux méta ontologique, ontologique et application seront représentés au *niveau objet*¹ en un ensemble de méta classes, classes, attributs et méthodes. Dans ce qui suit, nous allons voir plus en détails les niveaux application et objet.

¹ Le niveau conceptuel n'est pas représenté au niveau objet car il nous sert à identifier les concepts clés que nous retrouvons d'ailleurs au niveau méta ontologique.

4 Le niveau application

La librairie est composée de composants tâche, PSM, domaine, ainsi que les différents liens sémantiques et liens de transferts. Les applications dans le modèle en Y ne sont rien d'autre que des structures de ces cinq composants agencées d'une manière bien spécifique. Les applications représentent en effet des structures « LEGO » de composants tâches, PSM ou bien domaines, spécialisés en utilisant des composants liens intra-concept et liées avec des composants liens inter-concepts. Cette structure contrairement à une structure « collée » rigide, permet de permuter ou de changer des composants de façon assez souple, sans grande répercussion sur l'ensemble de la structure.

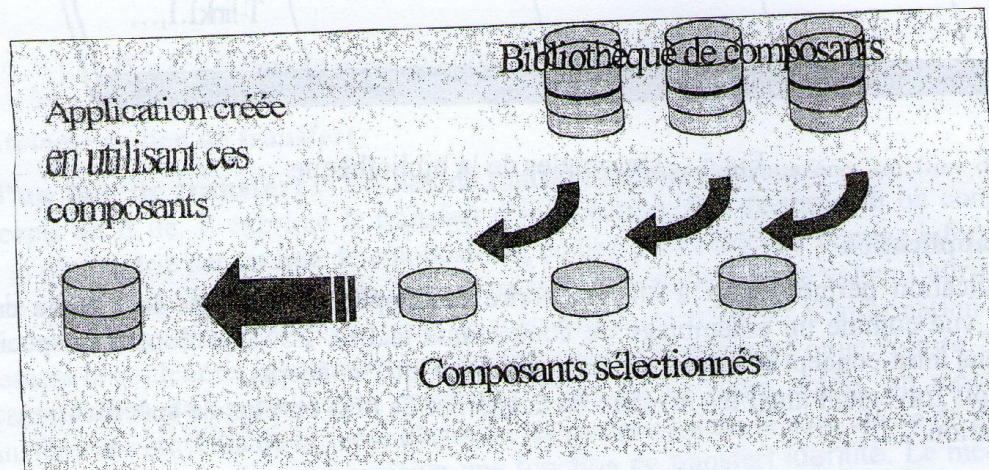


Fig 7. construction d'une application

Dans la figure qui suit, un exemple d'application est décrit. Supposons qu'un ensemble de composants existe dans la bibliothèque. Prenons l'exemple d'un composant domaine de télécommunications. Il est spécialisé, en utilisant un lien intra-concept de spécialisation, en un composant de réseau d'accès.

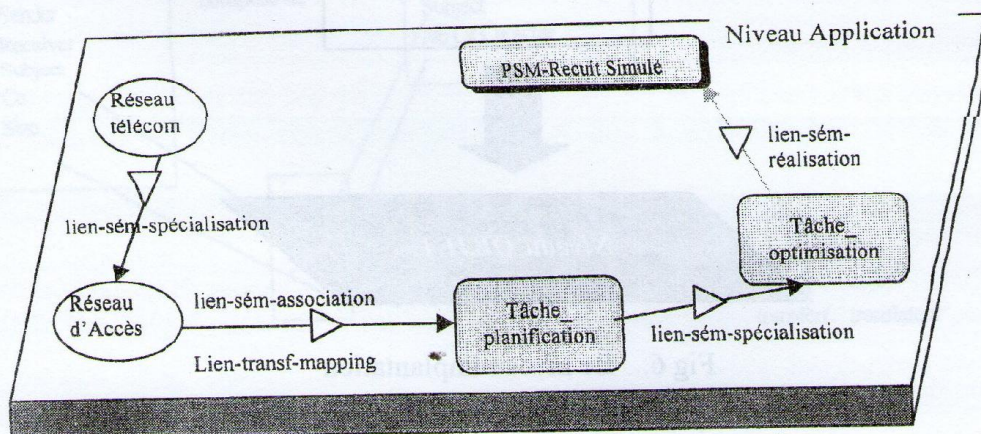


Fig 8. Le niveau application

5 Le niveau objet

Les différentes hiérarchies du modèle en Y sont représentées en utilisant la notion de méta classes, classes et instances.

1. Le niveau *meta-ontologique* est implanté par un niveau Méta classes. Les méta ontologies de tâche, PSM, domaine, lien sémantique et lien de transfert sont ainsi des méta classes.
2. Le niveau ontologique (ontologie de bibliothèque) est décrit par instantiation des méta classes du niveau supérieur. Les différentes sortes d'ontologies de tâches, PSM, domaines, liens sémantiques et liens de transferts sont des classes. Le graphe d'héritage nous permet de maintenir les niveaux

- de spécialisations des différents types de liens (ontologies de liens) ainsi que ceux des tâches ; PSM et domaines (génériques ou plus spécifiques).
3. Le niveau application est défini au niveau classe où une application utilise les différentes classes du niveau ontologiques.

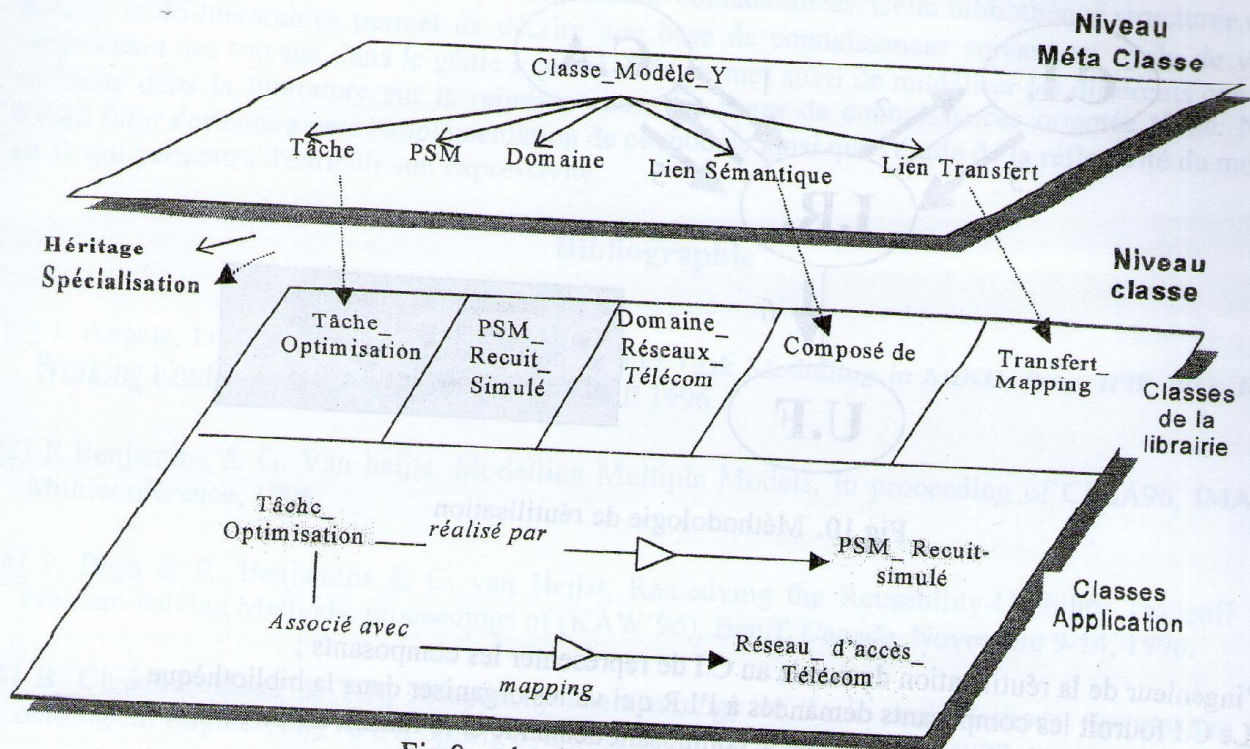


Fig 9. le niveau implantation

6 Méthodologie d'utilisation du modèle en Y

Quatre types d'utilisateurs sont impliqués dans le modèle en Y ayant chacun une fonction bien spécifique. Nous avons :

1. Le constructeur d'infrastructure (C.I.) qui va définir les composants, concepts de base sur lesquels l'expert du domaine (CA) va se baser pour construire une application. Les composants sont de ce fait génériques. Dans le modèle en Y, ceci revient à décrire les méta-classes qu'il fournira au gestionnaire de la bibliothèque (l'ingénieur de réutilisation). Ainsi le C.I. est concerné par les niveaux conceptuel et ontologiques.
2. Le constructeur d'application (C.A.), qui est l'expert du domaine, va utiliser les composants génériques pour décrire ses connaissances. Il construit des composants applicatifs spécifiques à une application et des applications réutilisables construites en assemblant des composants applicatifs donc, réutilisables pour les mêmes classes d'applications. Il est concerné quant à lui par le niveau application.
3. Le gestionnaire de bibliothèque que nous appelons ingénieur de la réutilisation (I.R.), va construire et gérer la bibliothèque. La bibliothèque est composée de parties réutilisables soit des composants génériques du C.I., soit des composants applicatifs et des applications réutilisables du C.A. Avant d'intégrer un composant, il doit vérifier
 1. Que les composants sont bien documentés et bien testés ;
 2. Que les normes sont respectées ;
 3. Que les applications utilisent bien les composants dans les règles de l'art ;
 4. Qu'on ne développe pas des composants d'application qui existent déjà.
4. L'utilisateur final va utiliser les applications décrites, fournir les données de son problème (données initiales) et instancier les applications.

La méthode de réutilisation est divisée en deux étapes. La première appelée pour la for-reuse décrit la construction d'un composant et la seconde, by reuse, décrit la construction d'une application par adaptation à partir de composants déjà construits.

1. Le C.A identifie les composants dont il a besoin puis, établit sa demande pour l'ingénieur de la réutilisation ;

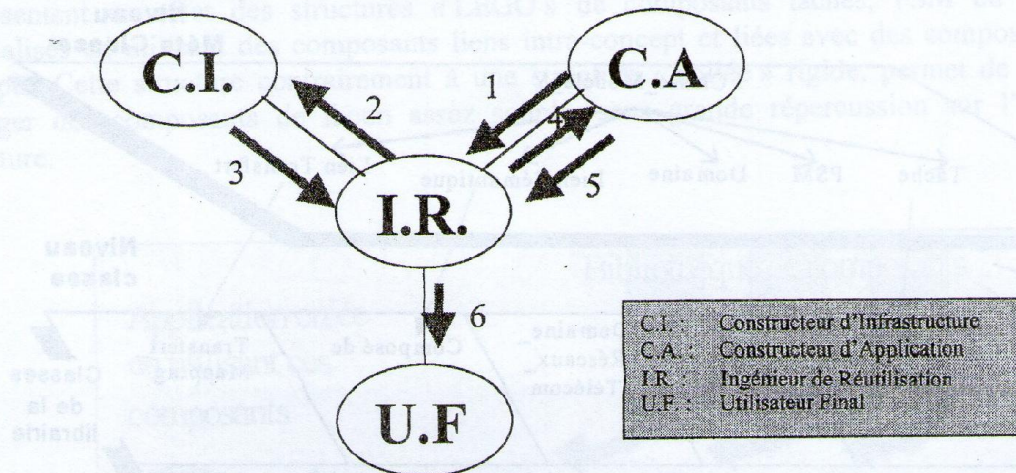


Fig 10. Méthodologie de réutilisation

2. L'ingénieur de la réutilisation demande au C.I de représenter les composants ;
1. Le C.I fournit les composants demandés à l'I.R qui va les organiser dans la bibliothèque ;
2. Le R.I. identifie dans sa bibliothèque le composant demandé et le fournit au C.A ;
3. Le C.A. va adapter le composant et l'intégrer dans son application.
4. Finalement, l'U.F instancie et utilise les composants applicatifs.

« L'I.R. joue le rôle d'un pivot entre les différents utilisateurs. »

Nous pouvons schématiser la librairie comme dans la fig 11. La partie lib_CI va représenter les composants de haut niveau : méta classes du modèle ainsi que les différentes sorte de classes de relations sémantiques et de transfert.

Le CA quand à lui, va manipuler les classes d'ontologies ainsi que des spécialisations de relation de composition.

L'U.F va manipuler les applications construite par le CA.

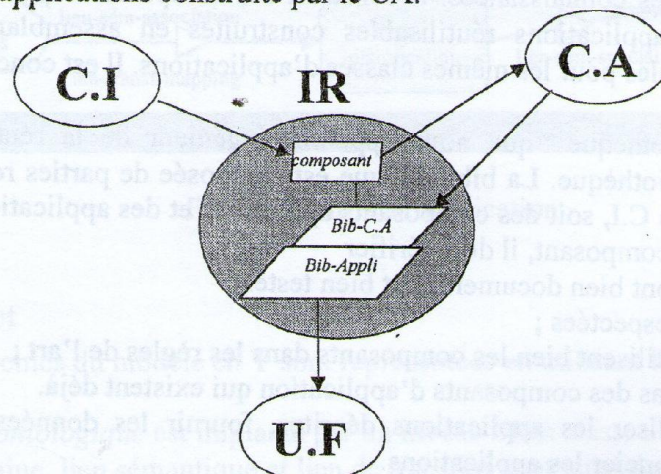


Fig 11. Librairie au niveau implantation

7 Conclusion

Nous avons défini dans cet article une bibliothèque pouvant décrire toute base de connaissances orientée tâche permettant d'intégrer et réutiliser des tâches, des PSM et des domaines à différents niveaux de descriptions. Pour cela nous nous sommes basés sur les travaux effectués dans le domaine du génie logiciel, des objets, et la représentation de connaissances. Cette bibliothèque structurée en un modèle multi-hiérarchies permet de décrire une base de connaissances suivant un cycle de vie se rapprochant des travaux dans le génie logiciel. Elle permet aussi de modéliser les différents concepts existants dans la littérature sur la représentation des bases de connaissances orientée tâche. Notre travail futur s'orientera vers l'implémentation de ce modèle ainsi que l'étude de la réflexivité du modèle en Y, qui permettra d'enrichir son expressivité.

Bibliographie

- [1] J. Angele, D. Fensel & R. Studer, Domain and Task Modelling in MIKE, Proc. IFIP WG. Joint Working Conference, Geneva. Chapman & Hall 1996.
- [2] R. Benjamins & G. Van heijst, Modelling Multiple Models, In proceeding of CESA96, IMACS Multiconference, 1996.
- [4] P. Beys & R. Benjamins & G. van Heijst, Remedying the Reusability-Usability Tradeoff for Problem-solving Methods, proceedings of (KAW'96), Banff, Canada, Novembre 9-14, 1996.
- [5] B. Chandrasekaran & T.R. Josephson, The Ontology of Tasks and Methods. Symposium on ontological Engineering, AAAI Spring Symposium Series, Stanford, CA. 1997.
- [6] B. Chandrasekaran, T.R. Johnson & J.W Smith, Task-Structure Analysis for Knowledge Modelling, Communication of the ACM, 35(9) :124-137, 1992.
- [7] E. Coelho & G. Lapalme, Describing Reusable Problem- Solving Methods with a Method Ontology, proceedings of (KAW'96), Banff Alberta, Canada, Novembre 9-14, 1996.
- [8] D. Fensel : The tower of adapter Method for developing and reusing problem solving methods. In E. Plaza & al (eds.), *Knowledge acquisition, modeling and management*, (LNAI), 1319, Springer-Verlag, Berlin, 1997.
- [9] D. Fensel, H. Erikson, M. Musen & R. Studer, Conceptual and Formal Specifications of Problem-Solving Methods. In International Journal of Expert Systems, 9(4), 1996.
- [10] T. R. Gruber: A Translation Approach to Portable Ontology Specifications, Knowledge Acquisition, 5(2), 1993.
- [11] G. Klinker & Al, Usable and Reusable Programming Constructs, Knowledge Acquisition, 3:117-136, 1991.
- [12] M. Magnan, Objets complexes, In Oussalah & Al (Eds): *Ingénierie objet concept et techniques*, InterEditions, Masson, (in French) 1997.
- [13] J. McDermott, Preliminary steps towards a taxonomy of problem solving methods, in S. Marcus ed, Kluwer academic publisher, Boston 88.

- [14] M. Molina, Y. Shahar, J. Cuena, & M. Musen, A Structure of problem-Solving Methods for Real-time Decision Support : Modelling Approaches Using PROTEGE-II and KSM. proceedings of (KAW'96), Banff Alberta, Canada, Novembre 9-14, 1996.
- [15] E. Motta, Trends in knowledge modelling: Report on 7th KEML Workshop. The Knowledge Engineering Review, Volume 12/Number 2/June 1997.
- [16] K. Messaadia, M. Oussalah using semantic links for reuse in KBS, in proceeding of DEXA'98.
- [17] A. Newel, The knowledge level, artificial Intelligence 18, 1982, 87-127.
- [18] C. Oussalah & al, A framework for modeling the structure and behavior of a system including multi level simulation, IASTED INT. Symp. On Applied Simulation an Modeling, ASM, Grindelwald, Switzerland, February 1988.
- [19] L. Steels, component of expertise, AI magazine, 11(2), 28-49, 1990.
- [20] Uta Priss, Relational Concept Analysis: Semantic Structures in Dictionaries and Lexical Databases. Dissertation, TH-Darmstadt, october 96.
- [21] R. Studer, H. Eriksson, J.H. Gennari, S.W. Tu, D. Fensel & M. Musen, Ontologies and the Configuration of Problem-Solving Methods, proceedings of (KAW'96), 9-14, 1996.
- [22] A. Valente, J. Breuker & B. Bredeweg, Integrating Modelling Approaches in the CommonKADS Library, In proceedings of the AISB93, 121-130, IOS Press, Amsterdam, 1993.
- [23] G. Van Heijst, A.Th Shreiber & B.J Wielinga, Using explicit ontologies in KBS development, International Journal of Human-Computer Studies, 46(2/3):128-292, 1997.
- [24] B. Wielinga, A. Schreiber & J. Breuker, KADS: A Modelling Approach to Knowledge Engineering. Knowledge Acquisition. 4, 5-53. 1992.
- [25] M.E Winston, R. Chaffin, D.J, Herrmann, A Taxonomy of Part – Whole Relations, Cognitive Science 11, 417-444, (1987).