Reuse of KBS components

M. Oussalah, K. Messaadia

IRIN, Université de Nantes, 2 rue de la Houssinière
BP 92208 – 44322 NANTES CEDEX 03, France
E-mail: Mourad.oussalah@irin.univ-nantes.fr

Abstract

This paper proposes a meta modeling technique which permits to describe a KBS according to three axis: the object of reuse axis , the levels of granularity axis and the reuse process axis. The object of reuse axis allows to see a KBS as a set of inter-related components for reuse purposes. The levels of granularity axis allows to describe the KBS components according to different levels of granularity for clarity and reuse purposes. The reuse process axis allows to see the KBS components as (re)usable components.

Keywords:

Meta modeling, Task, PSM, component reuse, semantic link, transfer link

Introduction

Modeling, meta modeling and reusable component libraries are proven techniques, used in software engineering, in order to reduce application development time and helping maintenance and evolution.

As a model is a modeling of a given system, a meta model is a model of (a) given model(s). The Meta modeling technique is a widely used technique in software engineering for integrating and defining models from different domains. It is also used for standardization purposes. Thus, it is a promising way for the definition of different models (or views) describing a system (for example the UML meta model). Component libraries have been described in different K.E approaches [2], [5], [6] , [8], [15]. Studying these libraries, we can see that they are facing the same problems amongst : how to describe a component (descriptive language), what should be the component level of granularity , how to structure and index components so that users can easily: understand, find, configure and integrate their needed components in order to construct specific applications.

In this paper we combine the both techniques: meta modeling and reusable component libraries for reuse purposes in order to describe KBS. In the proposed Y meta modeling a KBS is described according to three axis :

The object of reuse axis : determines the minimal and sufficient components used to describe a KBS. We have identified five components : Task; PSM; Domain; Inter-component link; and Intra- component link.

The levels of granularity axis: determines the KBS components, at a *knowledge* level, following different layers : meta ontology, ontology and application. Thus the five components already identified in the first axis can be described according to different abstraction degrees. Thus a component can be fine grained trough these levels (layers) and users can (re)use a component at the chosen level of description.

The reuse process axis : determines the components with a reuse process vision. The reuse process is spliced into two stages: for-reuse engineering and by-reuse engineering. The for- reuse engineering concern is about the construction of reusable components (library construction). The by-reuse engineering concern is about how to use reusable components (library use).

Existing K.E. methodologies describe a KBS using three distinct and complementary components: task, PSM, and domain. Separating the task to achieve, the reasoning used to achieve a given task and the knowledge about application domains permits to see a KBS application as a combination of these three components. We use these already identified three components along our first axis for describing a KBS, plus two other components called inter-component link and intra-component link. These links describe the interdependencies of the three components thus, keeping them really separate. For more information readers can refer to [13].

In AI [12] the *knowledge* level has been identified for describing a KBS independently from its implementation. This knowledge level description can be done according to different levels [9]. Following this, our second axis identifies three levels : meta ontology, ontology, and application. We call these levels in this paper the *levels of granularity.* Thus, along the levels of granularity axis, the five components are described throughout: meta ontology, ontology, and application. Reuse process in Information System [4] defines the components according to two stages *: for-reuse engineering* and *by-reuse engineering.* Thus, our third axis identifies the components along these two stages. Following the for-reuse engineering, the

components are first identified, secondly, described using a chosen descriptive (or formal or semi formal) language and then, organized in a library using a chosen library organization structure. Following the by-reuse engineering, the components can be found and selected using a search technique which should correspond to both the library organization structure and the language, chosen in the first stage. Then, the components are adapted and integrated, if needed, in a user application.

In order to *separate the concern* thus helping maintenance and facilitating users tasks, we have identified different kind of users. We will see this more in details, in the fourth paragraph. The rest of the paper is organized as follow: we will see the meta modeling KBS components along the three axis. Then, we will see an example of a matrix operations application where the components are constructed following the for-reuse engineering and the by-reuse engineering. We will conclude by our future work

Y meta modeling along the object of reuse axis

KBS representation has shifted from an extraction art to a modeling process of a complex system. The key point of a complex system modeling is its associated modeling process. The associated modeling process we've adopted is Y meta modeling. According to Y first axis, we have to identify the object of reuse. For our case we have to identify KBS as a set of components. For this, we use the Multi-abstractions/Multi-Views modeling technique [14].

The multi-abstractions/multi- views modeling process

The multi-abstractions/ multi-views (MHMV) modeling process describe a system, (a KBS for our case) according to two decompositions: views and abstractions :

Horizontal decomposition : An horizontal decomposition allows to define different views on a given system. Modem S.E combines different models (views) to describe a given system. This multi-views principle is called textitSeparation of Concerns. Having different views on a system clarifies different important aspects of it. Generally, human experts use this multi-views decomposition for describing complex problems. A view can be assimilated to the vision, a specialist in an area, can have on the system.

The number of views is not fixed. It depends on the problem we face and on the domain application. The views are dependent on each others. Thus, two distinct views of a given system can express either complementary aspects, or a common aspect, but using different components or different levels of description. It is important to highlight these inter-dependencies between the views using what we can call inter-view links.

vertical decomposition : defines different abstractions on a given view. Sometimes, a view is complex enough so that we need to describe it using different levels of abstractions (fine degree descriptions). A given abstraction level, introduces some information relegating the more details to sub-abstraction levels. The number of abstractions (abstraction levels) depends on the complexity of the view. The decomposition of a given view isn't necessary isomorphic to the decomposition of the other views. Even if this isomorphism is sometimes imposed in some modeling in order to facilitate their verification and validation.

library : generally, when modeling complex systems, one need to stock context independent information using multi-hierarchical libraries, thus, facilitating the modeling process. End users can reuse these information in their applications so that they have only to give context dependant information. We will see that a (meta) modeling process using such an horizontal and vertical decompositions allows to construct this kind of libraries.

The Y meta model uses an horizontal and vertical decomposition on a given system in order to capture its complexity and to identify its components. Y introduces also an associated hierarchical library in order to organize the components. Applied on a Knowledge Base System, Y describes the known three components: Task, PSM and domain as three distinct views of a KBS a la Components of Expertise approach [16]. In order to have independent descriptions of these three views, thus enhancing reuse for example, reusing a PSM to solve different tasks in different domains, we use intra-view (we call intra-component) links and inter-view (inter-component) links to describe their interdependencies. An inter-component link, links two different components: task/PSM, PSM/domain, Task/domain. An intra-component link links two components of the same type: task/task, PSM/PSM and domain/domain. The Y description differs from the components of Expertise approach in the use of two decompositions with predefined inter-view links and intra-view links and an associated hierarchical library.

Y multi-views decomposition

The horizontal Y decomposition defines : a task view, a PSM view and a domain view. Every view is represented in the Y meta model as a branch. In the task view, we describe different sorts of tasks in the same branch, with different semantic distances. A semantic distance between two tasks is materialized by an intra-component link. For example, the semantic distance between a task T1 and another task T2 defines an intra-component link task/task. Thus, more the distance between a task and the center of the Y is big, more this task is reusable and less it is usable. This holds for the two other branches. More a component is far from the center more it is reusable, less it is usable. This is known as the reusability/usability trade-off [10]. The centric component of the Y· (the center of the three branches) where the three distances between task/ PSM / domain are zero correspond to a *primitive component* where the inter-component links and intra-component links are not needed. This centric component is no more than a *primitive task* associated with a PSM-code *primitive PSM,* described in the terminology of the associated *application domain.* Inter-component links are the inter-branches (inter- views) links used to relate components belonging to two different views. Thus, we can navigate from one view to another one when constructing the components. For example, when we

decompose a task (for instance T1), we can link a PSM (for instance PSM1) to it using a realized-by link. The PSM by itself can introduce sub-tasks following the composed of link.

We have identified three sorts of inter-component links :

1. Task/domain These links describe task/ domain associations. Corresponding for example to the *knowledge roles* in the CommonKADS [2] methodology. In UPML [6], These links correspond to the task/domain adapter. Works on domain-task ontologies [1] can also be classified as being in the task/domain dimension [5] and [1].

2. Task/PSM These links describe the task/PSM associations. Task and PSM are two notions which are not often dissociated [2]. and when they are, the sub-tasks composing the task are always described either in the composite task body or in the PSM-body associated to the task. Thus sub-tasks are not considered as tasks. We can classify here, work on the fixed task/PSM associations like (role limiting methods) or flexible task/PSM associations like in the Task methodology [15] and Components of expertise [16], adapters used to link a task with a PSM [6] and task/PSM mappings in the protegeII methodology [8].

3. PSM/domain These links describe PSM/Domain associations. for example, we can find the work [1] on PSM/domain assumptions.

The associated library

The Y meta model introduces a hierarchical library. We will see that the hierarchies in the Y associated library are the three levels of granularity: meta ontology, ontology and application.

Domain ontologies as a multi-views multi- abstraction decomposition

Work on domain ontologies has shown that domain knowledge is described using *generic ontologies,* more *specific* ontologies, ontologies of *application.* It is significant, so, to model domain knowledge according to various abstractions. On the basis of the principle that a domain representation is a kind of knowledge representation [3], we apply to it the same modeling process using horizontal and vertical decomposition plus a domain library. Thus we have a hierarchical description of a domain ontology using views and abstractions.

Usually, in Software engineering (information system), they use three *principal* views to represent a domain knowledge: structural view, behavioral view and physical view [14]. Each view is described according to levels of abstractions. There are links (inter-view links) which allow to change from one view to another and links (inter--abstraction links) allowing to change from one abstraction to another. The number of views and the number of abstractions per view is not prefixed, thus enabling the enrichment of domain ontology.

Y meta modeling along the levels of granularity axis

Following this axis, we highlight the three levels: meta ontological, ontological and application. Each level clarify the preceding one by adding more specifications :

1. Meta Ontological level The five components of the Y model clarified on this level. Thus, the meta ontology level provides modeling primitives [13] in terms of: task, PSM, domain, inter- component link and intra- component link. This level represents in fact The Y Meta model.

2. Ontological level (ontology of library) at this level, various kinds of ontologies of tasks, PSM, domains and links are described. These components are described using the meta-ontology modeling primitives listed above. This level represents the Y model which is instance of the Y Meta model.

3. Application level the application level is used to describe applications [7]. In the reuse spirit of Y, an application is seen as an assembly of tasks, PSM, and domains components specialized using intra-component links and linked using inter-components links.

In the rest of the paper we will see more in details the components we have identified along the first axis, by describing them along the levels of granularity axis. This Y projection on the levels of granularity axis will help us to identify the components on the meta ontological, ontological and application levels. These components will fill the Y associated hierarchical library.

Meta ontology level

The meta ontological level is used in the Y model to describe the five components presented above. We will first present the task- PSM- and domain -meta ontologies. Then we will see inter- and intra- components meta ontologies more in details.

Task meta ontology

A task at the meta-ontology level is specified by its name, which is a unique term designating it, its input/output, the goal it is to reach and the inter- and intra- component links attached to it. for example :

An inter-component link can be a task/PSM link which can be a composition of :

- semantic realized-by link
- a transfer link with one or more translators
- mapping translators, a value propagation translator, ...

A intra-component link is a Task/task link which can be a composition of: specialization semantic link or a composition semantic link plus transfers with translators.

These links are described more in details in next paragraph.

PSM meta ontology

A PSM meta-ontology level is specified by its name, its input/output, its competence to achieve tasks, and the inter- and intra- component links attached to it. For example :

A inter component link can be a PSM/task link which can be a composition of:

- a semantic composed of link
- a transfer link
- some translators

An intra-component link is a PSM/PSM link which can be a composition of: specialization semantic link or a composition semantic link plus transfers with translators.

Domain meta ontology

The domain meta ontology level can be specified by giving one, two or the three views amongst: structural, behavioral and physical, plus other view types if necessary.

Inter- and intra- component link meta ontologies

An inter- / intra- component link is *composed of* two sorts of links: semantic link and transfer link. By definition, an intra- component link is a sort of inter- component link linking two components of the same type. Thus, an inter- / intra- component link is described by its name, its destination and source concepts, and the semantic and transfer links attached to it.

definition of semantic link and transfer link :

A *semantic link* describes a relation between two entities(components). It has its own semantics and a clean behavior helping the components to communicate and collaborate. Its semantics can express an association (logical, physical, etc.) a composition, an inheritance, etc. A semantic link is defined by:

- a unique name specifying it
- from and to, the connection attributes defining the source and destination components.
- the roles which the source and destination components play within the link.
- semantic attributes describing the semantic properties of the modeled link. These properties enrich the representation of the semantic links and to simplify their handling [11]. the semantic attributes can be for example the cardinality of the linked components, the functional dependence nature of a destination component with the source component.
- A more detailed description of the different sort of semantic attributes can be found in [13] .

A *transfer link* allows to transfer data flow between the related components. It expresses the transfer of information. For example, if one considers a car related to his body by a composition link, this link can convey as information that the color of the car is identical to that of its body. Thus, the mechanism

of information transfer allows to define only the color of the body. The color value can be propagated to the car by using a transfer link. Moreover, instead of limiting itself to a simple transfer of values (identity function), one can apply a transfer function to the values of the source to obtain the values of the destination attributes. This function is indicated using a translator. Thus, a transfer is obligatorily composed of at least one translator to establish the propagation between the dependent entities and to describe the function applied.

- *Translators* describe all the correspondences between the components connected by a transfer link. a transfer link is composed of translators. Each translator establishes a correspondence between the attributes of the connected components and defines the transfer function to be applied to the values of the source attributes to obtain the values of the destination attributes. Several attributes of the same component can be implied in the propagation.

Inter- and Intra- component links in the Y meta model

Now that we have introduced the Inter /Intra link and its associated semantic and transfer links, the different sorts of Y *Inter-component* links are :

- Inter-component links: task/PSM, PSM/domain, task/domain
- Intra-component link: Task/task, PSM/PSM, and domain/domain

We can take the example of a Task/PSM link description :

Name : Task/PSM
source : Task
destination : PSM
semantic link : realised by
Transfer link : Mapping

The realized-by link description associated to the task/PSM link :

name : realized by
from : PSM
to : Task
Rotes : composite component
semantic attributes : dependant/ sharable

The transfer link associated to the task/PSM link:

Name Mapping-Transfer
source PSM-concepts
destination Task-concepts
translators Mapping-

The ontological level

At this level we can describe different sort of ontologies: task, PSM, domain, and links. The Y model at this level, can be seen as a Y-Shape model with several branches.

- Task branch describing types of task such as: diagnosis, configuration, modeling,
- PSM branch describing types of PSM such as: classification, abduction,
- Domain branch describing type of domains such as: networks and telecommunication networks.

links are described in terms of task/PSM, PSM/domain, task /domain, task/task; PSM/PSM; domain/domain link.

The application level

An application is a composition of components described at the ontological level. For more details, readers can refer to [13] . For example, in an application for planning an access network domain, we take a telecommunication domain ontology from the library. We assume that this component has been already created. We specialize this component - using the specialization intra-concept link - into an access network domain. Linking the optimization task with the simulated annealing PSM is done using the realized-by semantic inter-concept link and the mapping transfer inter-concept link.

Y meta modeling along the reuse process axis

Before describing the different components, we will introduce the Y different users, we have identified.

The different Y users

1. The infrastructure builder (I.B) has to define the modeling components used to build the infrastructure. In the Y model, the I.B will define the meta-ontology components which constitutes the descriptive language (the Y infrastructure). His concern is about the *Meta Ontology Level.*

2. The application builder (A.B) is the domain expert and will use the meta-ontology components for describing his/her specific components or specialize the ontological components (or applications). The A.B is concerned by *the Ontology Level and application level.*

3. The Reuse Engineer (R.E) or library manager will construct and manage the library. The library is composed of generic parts: generic meta-ontology components defined by the infrastructure builder, reusable components and reusable applications defined by the application builder. The library is split into different compartments:

- a meta ontology library,
- an ontology library and
- an application library.

The R.E concern is about populating his hierarchical library with the reusable components of different levels of granularity. He needs to make some verifications before inserting an I.B or A.B constructed component. Before accepting a component or an application into the library, the R.E has to verify if:

- the components are well documented and well tested;
- the standards are respected;
- the components are well used in the applications;
- we are not developing already existing components and applications.

4. The End User (E.U) instantiates specific applications in order to solve real problems (feeding in the initial values of the problem).

the reuse process is split in two stages: for-reuse engineering and by- reuse engineering.

The for-reuse engineering

The for-reuse engineering concern is about how to construct reusable components. The meta ontological , ontological reusable components and reusable applications are inserted in the library respectively in the: meta ontology, ontology and application library layers. Before describing the for-reuse engineering in every layer, we will first see the identified for-reuse engineering tasks.

1. Identify

This task is about identification of the users needs. Often, new components emerge from Application Builders needs. The identification can concern components at the meta ontological, the ontological or the application levels.

2. Represent

When a needed component is identified, we need to represent it using a given language. The representation language used can be formal, semi formal or descriptive. For example: CML, OCML, ML, . Y model uses the modeling primitives described at the meta ontology level as a representation language. As the meta ontological level can be modified, by modifying, adding or deleting meta ontology primitives, Y has a flexible modeling infrastructure.

3. Organize

The components are then organized in the library. Thus, the Reuse Engineer has to choose a component organization for the different library layers : meta ontology; ontology and application.

For Reuse Engineering Example We will see an example of matrix operations in Figure 1.

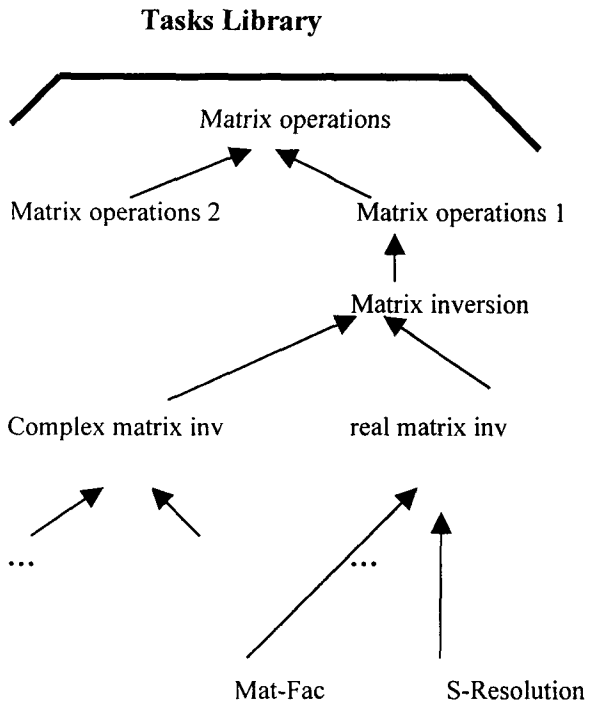

Figure 1 – Example of a Matrix

Meta Ontology level

1. Identify The components we have identified in Y at the meta ontology level are integrated by the R.E in the meta ontology library. Thus we have the : meta-task, meta-PSM, meta-domain, the inter- and intra- meta component links plus the predefined semantic, transfer and translator links. Thus the I.B identifies the task/PSM, PSM/Task, Task/domain and PSM/domain inter-component links and the Task/task, PSM/PSM and domain/domain intra-component links.

2. Represent The meta ontology components are represented in Y using an abject language. Thus all these components are meta-classes.

3. Organize The organization adopted *by the R.E* in Y is done according to a Multi hierarchies /Multi Views description. Thus we have at the meta ontology level two abstractions. In the first abstraction we have the :

- meta-task description
- meta psm description
- meta domain description
- inter component link description
- intra component link description

in the second abstraction, we have the semantic and transfer links organized are in a generalization/specialization tree. For more details about predefined semantic and transfer link trees, reader can refer to [13].

Ontology level

1. Identify The ontological components are constructed by the A.B. For example the A.B want to describe matrix operations task.

2. Represent The A.B. represents his matrix operations as a number of tasks and PSMs using the meta ontology descriptive primitives. We can give the real inversion matrix task in Figure 1.

name : Real MatrixInversionTask
goal : calculate the inverse of a real matrix
input : a real matrix M
output : $^{'}M$
input/output constraints : the matrix should be real
inter component link : T/P Mat-Inv
intra – component link :-

The choleski PSM

name : CholeskiPSM
input : Real Matrix
output : Real Matrix
competence : Mat-Fact; SRI ; SR2
inter component link : P/T Inv Choleski
intra component link : -

Task/PSM Inter-Component-link:T/P Mat-Inv

Name : T/P Mat-Inv
source : Real MatrixInversionTask
destination : CholeskiPSM
Semantic link : realised by
Transfers link : mapping-

PSM/Task Inter-Component-link : P/T Inv Choleski is composed of

- a composed of semantic link to the Fact-Mat task plus a transfer link with translators
- a composed of semantic link to SR task with cardinality = 2 (two link instances towards both SRI and SR2) plus the two transfers and translators.

Name : P/T Inv Choleski
source : CholeskiPSM
destination : Mat-Fact; SR
Semantics links : composed of-Choleski-MatFact; composed of-SR
transfers links : transf-P/Mat-Faact; transf-SRI; transf-SR2

name : composed of-Choleski-MatFact

from : CholeskiPSM
to : Mat-Fact
Rotes : composite component
semantic attributes : not dependant/sharable/ Card= 1

name : composed of-Choleski-SR
from : CholeskiPSM
to : SR
Rotes : composite components
semantic attributes : not dependant/sharable/ Card=2

3. Organize

In existing approaches, the most used organization (for the ontological components)is the task/PSM tree structure. PSM in such organization are usually indexed according to their: competence, assumptions, functionalities, or as a suite of problem types. Other approaches use two separate tree structure: a task tree structure and a PSM tree structure. Then Task/PSM linking is defined dynamically according to an execution context. In the Y library, ontological components are organized in hierarchies according to the user's choice. The user can manipulate separate task trees and PSM trees or task/PSM tree structures describing the different links explicitly. The organization adopted for the matrix operations example is to have three separated hierarchies: Task hierarchy, PSM hierarchy and inter/intra link hierarchy. This organization is described in the Figure 1.

Application level

1. Identify A final user need to make for example an inversion of a real Matrix. The application builder will ask the R.E for the needed components. As this phase, the A.B will link the RealInversionMatrixTASK with the appropriate. for our case, the CholeskiPSM using a TASK/PSM link. We have two scenarii:

- Linking the task with the PSM with an existing (already identified and described)inter-component link.
- if the link doesn't exist in the library, a search (Search-PSM) is done to find the best PSM among the existing PSM for making a Real-MATRIX-inversion.

These scenarii are described as PSM.

2. Represent The A.B. represents his application by representing the missing pieces. For example, the A.B matrix inversion operations application will be a linked TASK and corresponding PSMs with the appropriate inter- and intra- component links.

3. Organize The application insertion in the library is done by the R.E. who will integrate the application, if he decide to do so. The R.E decision about integrating an application as a big piece reusable component depends for example on the frequency use of it. If matrix inversion is often used by F.U, it can be stored as a composite component (components and their links).

The by-reuse engineering

The by-reuse engineering concern is about how to reuse an exiting component in one of the library's compartment, at the meta ontological, ontological or application level. In order to reuse a component, we have identified these tasks :

1. Find and select

These two tasks depends on the adopted library organization along the first stage. For example in case of a TASK/PSM organization, and a predefined hierarchy (fixed task-PSM associations), we can find automatically and easily the PSM associated to a given task. in case of no predefined task-PSM associations, a search is done in the PSM set in order to find the best PSM candidate for achieving a task according to the context. Thus, the search can be considered as a general PSM. Thus, for more flexibility and evolution, we've adopted to use a PSM to describe a search algorithm which is associated to the library organization.

2. Adapt

Adapting a component to a new context is a hard task. In Y the adaptation can be done by describing a new component from an exiting one by : generalization/specialization, by composition,...according to predefined specific adaptation rules

3. Use

Finally the component is ready for use by integrating it to a new application.

Conclusion

The Y is a reflexive model since the three axis we've seen can be considered as task, PSM and domain axis. The task is Reuse, the corresponding PSM is the Reuse Process and the domain is KBS. The table bellow can summarize this idea of applying the Y recursively :

We use the Y for comparing existing methodologies (at the knowledge level) [I], [2], [5], [6] , [8], [15], [16]. By projecting the methodologies following the Y three axis, we can identify the components involved in existing methodologies and the corresponding reuse and use methodology if it exists. Our future work is about comparing the existent known methodologies. For example, if we apply this projection on the components of expertise methodology [16], we can have:

object of reuse : The components used are: task, domain model, case model and method. If we look for the links we can find that the inter-component links used in this methodology are in the method description and they are called: conceptual and pragmatic features. They are not

components by themselves. The intra-component links are not highlighted in this methodology.

levels of granularity : The levels used are the: knowledge level, execution level and code level. The methodology use of these levels makes the navigation from one level to another not *seamless*, as in the spirit of the Y model where the execution level is no more than an object language.

reuse process : The only information we had along this axis is about the library organization. The library organization is a task/PSM tree.

As we have said, our future work is about the comparison of the methodologies. We are also working on the semantic distance between tasks, PSM and domains. This semantic distance which is associated to an inter- or an intra-component link can be calculated by the number of semantic-, transfer- links and translators composing the associated link.

References

[1] Benjamins,V.R. 1993. Problem Solving Methods for Diagnosis. Ph.D diss., University of Amsterdam, Amsterdam.

[2] Breuker, J. A., and Van De, W. eds. 1994.*CommonKADS Library for Expertise Modeling: Reusable problem solving components*, volume 21 of Frontiers in Artificial Intelligence and Applications. Amsterdam, IOS-Press.

[3] Reynaud, C. 1999. Technical report n° 1201, L.R.I Paris sud university.

[4] Barbier, F., and Rolland, C. eds. 1999. Best of ooisʼ98. Revue L'Objet, Masson-editeur.

[5] Chandrasekaran, B ; Johnson, T.R. ; and Smith J. W. 1992. Task-structure analysis for knowledge modeling. *Comm. of the ACM,* 35(9):124-135.

[6] Fensel, D. 1997. The tower-of-adapter method for developing and reusing problem-solving methods. Lecture Notes in Computer Science, 1319:97-111 .

[7] Gaspari, M., and Motta, E. 1994. Symbol-level requirements for agent-level programming. In A. G. Cohn, A.G. eds. In *Proceedings of the Eleventh European Conference on Artificial Intelligence,* 264-268, Chichester.

[8] Gennari , J.H.; Tu, S. W. ; Rothenfluh, T. E. ; and Musen, M. A. 1994. Mapping domains to methods in support of reuse. International Journal of Human-Computer Studies 41(3):399-424.

[9] Guarino, N. 1994. The ontological level. In Casati, R.; Smith, B.; and White, G. eds., Philosophy and the Cognitive Sciences. Holder-Pichler- Tempsky , Vienna.

[10]Linker, K.; Bhola, G. ; Dallemagne, C.; Marques, G.; and McDermott, J. 1991. Usable and reusable programming constructs. *Knowledge Acquisition,* 3:117-136.

[11]Magnan, M. 1994. Réutilisation de composants: les exceptions dans les objets composites. Ph.D. diss., Université des Sciences et Techniques du Languedoc, Montpellier.

[12]Newell, A. 1982. The knowledge level. *Artificial Intelligence,* 18(1):87-127.

[13]Oussalah, M., and Messaadia, K. 1999. The ontologies of semantic and transfer links. In Proceedings of the 11th European Workshop on Knowledge Acquisition, Modeling and Management , v. 1621 of *LNAI* 225-242.

[14]Oussalah, M.C. 1988. Modèle hierarchisés/multi-vues pour le support de raisonnement dans les domaines techniques. Ph.D. diss, Université de Aix-Marseille.

[15]Pierret-Golbreich, Ch., and Talon, X. 1997. Specification of flexible knowledge-based systems. In Proceedings of the 10th European Workshop on Knowledge Acquisition, Modeling and Management v. 1319 of LNAI 190-204.

[16]Steels, L. 1990. Components of expertise. *AI Magazine,* 28-49.