A Collaborative E-authoring Tool for Knowledge Assets

Tang-Ho Lê[1] and Luc Lamontagne[2]

[1] Computer Science Department
Université de Moncton, 165 Massey Ave, Moncton, (N.B.)
CANADA, E1A 3E9
letangho@umoncton.ca
[2] Département d'informatique et de recherche opérationelle
Université de Montréal, C.P. 6128, Succ. Centre-Ville, Montréal (Québec)
CANADA, H3C 3J7
lamontal@iro.umontreal.ca

Abstract. The development of e-government enhances not only the public-agency relation but also inter-organizational cooperation between governmental agencies. In this context the promotion of knowledge distribution favors the application of existing techniques and approaches in Knowledge Management Systems. Especially, to exchange "knowledge in evolution" from different disciplines, one needs some groupware knowledge management tools to support knowledge worker communities via the Internet. To be effective, these tools should have visual features for several presentation issues like distributed tasks, evolution trace keeping, ontological discussion and action demonstration. In this paper, we provide an overview of our groupware tool, called Collaborative e-Authoring Tool for Knowledge Assets (CATKA) allowing to create, visualize, exploit and interchange two kinds of knowledge: declarative and procedural knowledge. We also detail the knowledge base updating technical issues for knowledge exchange process between knowledge workers to carry out an e-authoring project step by step, from the beginning to the final phase.

1 Introduction

In the current explosion of the Web, the development of e-government must include on one hand, the e-services that carry out the public-agencies relation, and on the other hand, the groupware to support inter-organizational cooperation between governmental agencies. For example, search and rescue operations, which imply several agencies, as the police, the Air force, the Coast guard , etc., actually need some kind of knowledge networks built with distributed tasks between these agencies. Several knowledge workers from distant sites must create this new multi-disciplines knowledge during a period of time. The procedures for such operations to follow may be revised several times and then disseminated to responsible workers. The development of such a tool pertains mostly to procedural knowledge. In the actual state of the Knowledge Management field, this kind of tools is also necessary.

From our point of view, the difficulties to realize such tools can be identified in terms of the lack of visual features for several presentation issues like distributed

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-540-44836-5_33

M.A. Wimmer (Ed.): KMGov 2003, LNAI 2645, pp. 175–185, 2003.

tasks, evolution trace keeping, ontological discussion and action demonstration. According to [1], the need to better represent procedural knowledge is still poorly understood. Indeed, intensive research has been conducted over several years to devise formal methods to understand and to determine declarative knowledge based on ontology; for example, several proposals were made for ontology languages and ontology builder such as DAML, Protege 2000, Ontolingua, etc. But fewer research efforts were devoted to model and specify knowledge being procedural in nature, and to combine this with declarative knowledge. In this paper, we describe our groupware tool, called Collaborative e-Authoring Tool for Knowledge Assets (CATKA) allowing to create, visualize, exploit and interchange two kinds of knowledge: declarative and procedural knowledge. This tool was adapted from an e-learning project, realized during the last year, granted by the Human Resources Development Canada. First, we introduce some basic concepts related to practical Knowledge Management (KM) tools; then we present the modeling scheme for Procedural Knowledge used by our software. Finally, we detail the knowledge base updating technical issues for knowledge exchange process between knowledge workers to carry out an e-authoring project step by step, from the beginning to the final phase.

2 Basic Concepts Related to Practical KM Tools

KM tool developers and users have now reached maturity with some basic concepts included in these tools; for instance, the *granularity issue* for defining Knowledge Units (KU) that take into account the limitation of human cognitive capacity, the *visualization feature* allowing to visualize and directly manipulate KU on the computer's screen and the *modeling* of declarative and procedural knowledge. These essential concepts and features are further described in the following.

In the design of a Knowledge Management system, either as a process or as a product [2], one has to cope with the granularity problem. Indeed, if knowledge workers want to efficiently transfer the desired knowledge to one another, what would be its suitable volume? To answer this question, we believe that one must take into account the user's cognitive capability. Without determining this cognitive limitation, the transferred knowledge will be useless or/and not be reused. This issue is not clearly addressed in the current literature. We see two reasons for the difficulties to determine such a limitation:

- Because the knowledge is not well defined yet. Fundamentally, what is the knowledge we want to transfer? A declarative knowledge or a procedural one? The former concerns some concept definitions or some factual (short) information; the latter is essentially action(s) to perform some tasks (in the sense of an algorithm). In the both cases, to call them "knowledge" they must be self-understandable by the receiver. How one can ensure that?
- Because of the lack of a clear separation between information and knowledge. Consequently, instead of transferring the right knowledge, one may collect a bulk of information of all types. The receiver of this "knowledge", cannot consume it, has the impression that he/she lacks yet the desired knowledge and wants to know more.

One approach for the transfer of knowledge is to apply some research results already available from the Intelligent Tutoring Systems. In such systems, a knowledge object

(i.e. a KU) corresponds roughly to a teaching subject. Several related subjects form a lesson, what we call a Knowledge Network (KN) that one can teach in a session during about one hour. Moreover, it is proven that the teaching of procedural knowledge will be more efficient if it is accompanied by demonstrations [3]. These demonstrations can be realized by animation files (e.g. by Macromedia Flash) or, if the memory space isn't matter, by some video films. Even in the context of ontological negotiation as proposed in [4], one must firstly frame out the underlying knowledge. To do this, a cognitive analysis will be necessary. From the light of Intelligent Tutoring Systems, we can design a KN as a task hierarchy consisting of just a small number of levels (from one to tree levels is ideal). That is, the KN is named as a global task, and its content can be detailed with some KUs as primitive tasks, which are arranged as a sequence (one level) or as a hierarchy of several levels.

While the ontology construction is based on the entities of the external world, the procedural knowledge results from our epistemological states of mind. This distinction leads to these two important consequences:

1. The ontology construction can apply the principles of *object oriented design* with class definition (a template to classify instances) in which the emphasis is put on the view of entities (expressed by nouns) and the relation between super and sub-classes as generalization/specialization. For example, we can have the superclass "Vehicle" and the subclass "Truck". However, the *actions* are designed as relations between two different class entities; for example, a "Car" being *driven* by a "Person". See more details in [5] and [6].
2. For procedure knowledge, the importance relies on the description of actions or the achievement of tasks. In order to make understandable this knowledge, *pedagogical principles* can be applied, e.g. one can specify the prerequisite (condition) for each knowledge progression step. Thus, as a result, one produces also a hierarchy of *domain knowledge concepts* which are a mix of entity concepts and procedural concepts. The latter can be expressed by verbs and by nouns derived from verbs (not like the entity nouns), for example, "*navigation* in the web site".

We observe another consequence of this distinction regarding the granularity problem of the Semantic Web: while it is difficult to define a manageable unit for an ontology (i.e. as a class, or as a hierarchy of classes, or as a knowledge base with some classes and their instances) it is simpler to define a unit for procedural knowledge, because each unit correspond to a task which already has its limitations. And then, as several related KUs form a KN, its volume is logically enough to describe any general task.

Although the above concepts and features can be realized in some manners for most software tools, a simple and effective tool stills to be desired. Especially, in the context of the knowledge exchange between knowledge workers (of inter agencies) where a small group works on a collaborative project via the Internet. This kind of groupware must be simple enough to not discourage the users in learning how to use the tool, while it is effective enough to allow users to continually visualize and work on the same knowledge bases (i.e. developing and updating their KU) over a period of time.

3 Declarative and Procedure Knowledge Modeling

The design of a Knowledge Management System is essentially the modeling of knowledge so that it is understandable to users, or at least, to provide them with the conditions for an adequate awareness. As many authors in the Knowledge Management field, we make use of frames to model knowledge in the Collaborative Authoring Tool for Knowledge Assets. To clarify the terminology, we use the term "Knowledge Unit" (KU) to designate some knowledge about an *entity* or about a *process*. Thus, we distinguish two kinds of units: "Static" KU and "How-to" KU. A static KU is a declarative KU that contains concept definitions, labels, facts and information related to the underlying domain. The names of these KU form (or are derived from) the domain ontology. A How-to KU is task-oriented (i.e. procedural knowledge). It contains the procedure to follow through (actions or tasks) and refers to static KU when necessary. The description of a How-to KU focuses on a limited scope, and its attributes specify the underlying context, a situation or some conditions. This context allows users to become aware of the conditions to understand it (pedagogically called prerequisites). A How-to KU can also give some references (links) to other available documents.

Although a KU has limited scope, it is not confined to itself. In other words, a KU never exists in isolation and always relates to other KUs. Thus, a KN including several KUs linked together could provide a complete view of the knowledge related to a specific topic. The tool described below allows for the construction of KNs, independently of the application domain. Figure 1 on the next page illustrates an example of a How-to KU frame describing the creation task of Flash files. Figure 2 gives an example of a KN for an authoring project presented in the section 4; the blue node representing the related concept.

To each KU featured in a KN is associated a frame structure. The frame of a How-to KU has two parts with some attributes. In the first part, the following attributes are used for the KU identity:

Name: a term that abstracts the actions of the underlying KU.
Domain: an ontological hierarchy of the domain written as a sequence of terms (i.e. domain/sub-domain/sub sub-domain /...). The last sub-domain is where the actual KU is situated. For instance, in the computer programming domain, if the actual KU is "*for* loop", then the domain field can be specified as "Computer Science / Programming Languages / C++ Programming". This sequence simply is the path on the storage device to locate the actual KU. All together (i.e. the sequence plus the KU name), they form an index entry for the working domain.
Done by: refers to the knowledge worker who creates the underlying KU.

The second part models the procedural knowledge. This modeling takes into account the epistemological states of mind that is realized by describing the first three main attributes, which naturally correspond to what we already know, what we are learning, and what we will know afterwards:

Situation: a textual description of the conditions where the KU is applicable.
Actions: a textual description of some primitive actions or subtasks.

Results: anticipated states and consequences if the KU is applied. This final state will become the initial state (e.g. situation) in the next mind development phase of this individual. Two others attributes have a secondary role for annotation purposes:

Subtasks: a main task can be achieved by carrying out many subtasks; consequently, a hierarchy of tasks can be established and form a KN.

Remark: to highlight reminders suggested by the KU's author.

In the middle there are two optional fields, which may be necessary for the procedural knowledge:

Reference: a link to an existing document or to a web hyperlink (URL). Used to provide more explanation, or just to enlarge the underlying knowledge.

Demo: a reference to a multimedia resource (e.g. flash file, video, photo, graph, diagram, etc.). This link allows the activation of the multimedia resource from another window on screen. This on screen demonstration is an effective learning method applied for procedural knowledge according to [3] and [7].

The content of a KU is described in natural language and can be supported with multimedia resources (the file names are given in the Reference and Demo field). Thus, the underlying knowledge is represented by most available means corresponding to human perceptual senses. Moreover, with this knowledge modeling, several aspects of a specific context are mentioned: origin, environment, initial state and goal state. Some terms used in the attributes of the second part may be ontology terms that are defined in the related Static KU; consequently they are appropriate for further locating.

A KN is a directed graph where the nodes are KUs, each of them being related to some others by links of different types. In the current version, in addition to the subtask links, we implemented two other kinds of links: The *workflow links* reflect the order between KUs in an activity, a project or an organization. The *pedagogical links* connect prerequisite KU for understanding the actual KU. The required knowledge is called prerequisite knowledge, which is usually some Static KUs that provide users with explanations on the domain ontology (e.g. concept definitins). Existing KUs can be reused by integrating them into newly created KNs.

4 A Collaborative Authoring Tool for Knowledge Management

Our tool (CATKA) can be seen as an environment in which we can review how new terms emerging from the worker's creative knowledge are integrated within a knowledge hierarchy. This is of interest because an ontology reflects this hierarchy of knowledge [8]. In other words, we would like to intervene upon the emergence of a new term. This task can be seen as the "bottom-up" approach to construct ontology. This approach involves the capturing of the knowledge in evolution to keep trace of the domain progresses.

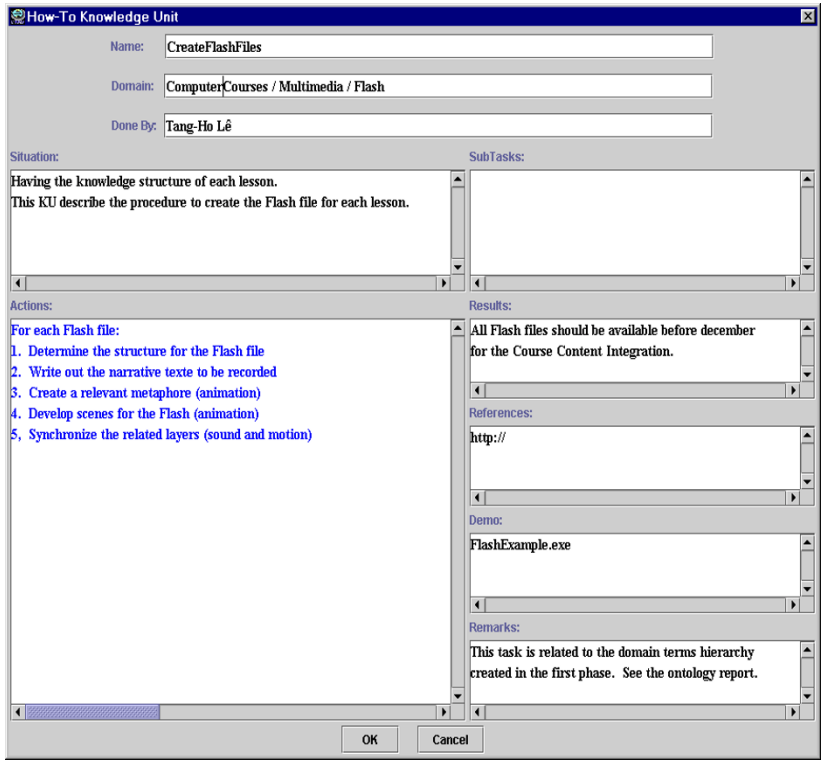


Fig. 1. Example of a Knowledge Unit Frame

In order to understand how it works, we should consider the Knowledge Creation Process (KCP) in a work context of a team. Through the internalization and externalization phases, new terms may be used to describe new concepts. These terms are necessary for knowledge interchange. However, since they were used for the first time, their meaning is well understood only by the team (a very small community). When the KCP comes to maturity with clear definitions of terms, more people may appreciate the new terms and they become popular and, by default, formal (i.e. implicit convention). For instance, we cite here several new terms that appeared in the web domain; most of them being already standard terms: webbot, cobweb, hover button, hotspot, thumbnail image, image map, rollover image, etc. In a KCP, knowledge workers propose new terms normally without taking into account the understanding or the impressions of a large community. So these new terms must be reviewed by a knowledge engineer to eventually justify them. This process consists of applying the consensus of the users community, if reachable. That is comparable to the proposition of [4]. In our CATKA environment, the content of the How-to units is considered as a corpus and each new significant term with its explicit meaning is recorded in a Static KU. When the number of KN reach a manageable size, a grouping task is necessary. We call it the sub-domain promotion (it seems like the promotion of an index level in the B-tree indexing). The knowledge engineer should name a sub-domain by a generic category term which conceptually covers all the underneath KNs. These category terms progressively form the domain knowledge

hierarchy. The knowledge hierarchy is then updated in the domain attribute for these KNs. Note that, as opposed to conceptual graphs, there is no semantic interference between terms in different branches of the ontological tree.

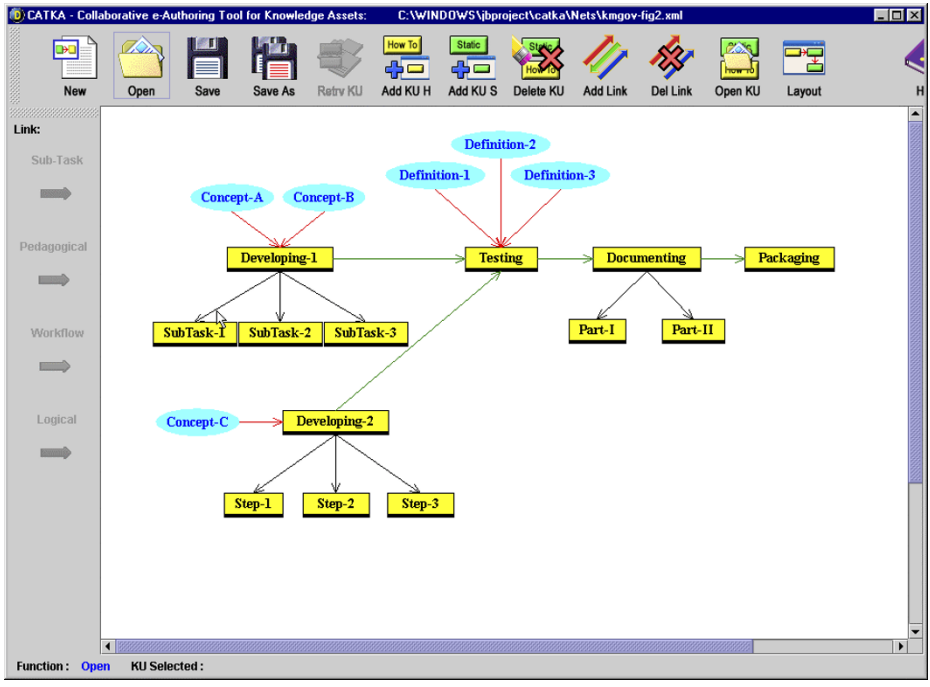


Fig. 2. Example of a Knowledge Network

5 Updating the Distributed Tasks of an E-authoring Project

During the summer of 2001, we conducted an e-learning project to improve the self-learning abilities of forty workers. The CATKA was used as a collaborative tool during the development of course's content and pedagogical material (animation Flash files). In the next steps, we will carry out some real applications in the context of inter-agencies. For this purpose, we generalize below the experience in a more general e-authoring project to well describe the distributed tasks.

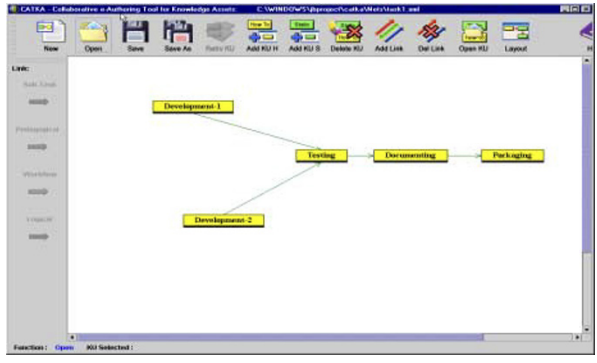
To create and exchange "knowledge in evolution" between project members, the e-Authoring tool will allow for collaborative work on the Internet. In technical terms, to realize this tool, we have designed the visual interface for several presentation tasks like project issues, evolution trace keeping, ontological discussion and action demonstration. Put in others words, the e-Authoring tool allows the exchange of two kinds of knowledge: declarative and procedural knowledge. Finally, the updating of the project work in a client/server knowledge base system also allows for concurrent development by all project members, and this work can continue during several

days/weeks. A member can visualize the recent development of others while elaborating his/her work. Moreover, the server updating does not interfere with the ongoing work of each member. To provide these features, our tool updates the collaborative work represented as a KN of several nodes (KU) basing on each member's distributed tasks. This approach is comparable to record's field updating in databases but with visual feature. Note that although some other groupware (e.g. MS-FrontPage) allows the collaborative work, the common document is totally blocked when editing by the user; all other users having to wait until the current user relinquish it. To show how the CATKA works, we summarize possible member's tasks as followed:

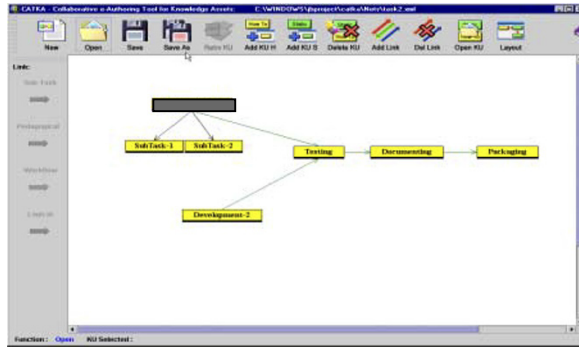
- The project leader sends the original KN of the initial plan to all members. In this KN, each member can recognize his/her distributed tasks, which are represented by some KUs according to the group discussion.
- Each member develops/updates his/her work by adding more KU around her/his distributed KU. Then, she/he sends the updated KN to the server.
- The server receives the member's updated KN, locks the original KN while updating. It updates the KN by importing (integrating) only the new KU to the identified node (KU) into its actual KN. And finally, it sends back the updated KN to the sender.
- A member, who has not updated his/her node yet, can request the recent updated KN before the development of his/her part. He/she always receives a new updated KN after sending her/his current KN. While developing, a member can locally save (at any time) the current KN and continue to work with this KN before sending it to the server.
- Only the project leader can add more distributed nodes (KU) to the KN located in the server and then send a notice message (specifying the added KU-names and their assigned developers) to all members so that on the next updating, members will recognize these changes.

The following scheme concretely illustrates these mentioned tasks:

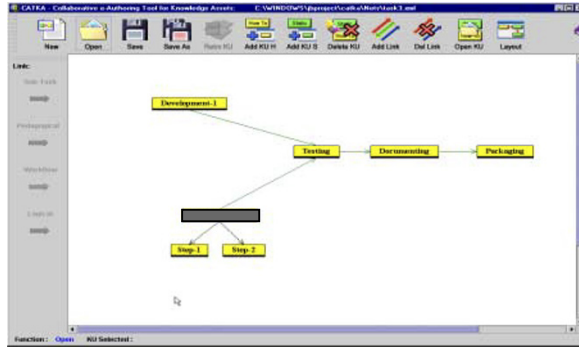
Task #1. *Initial planning of KN*: The project leader builds the initial KN and distributes KU tasks to project members.



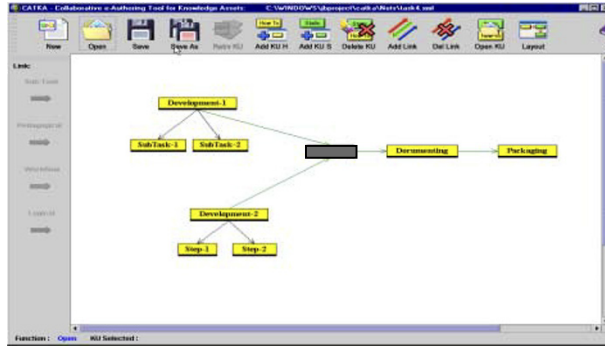
Task #2. *Development-1*: The server receives and updates the KN developed by the first developer.



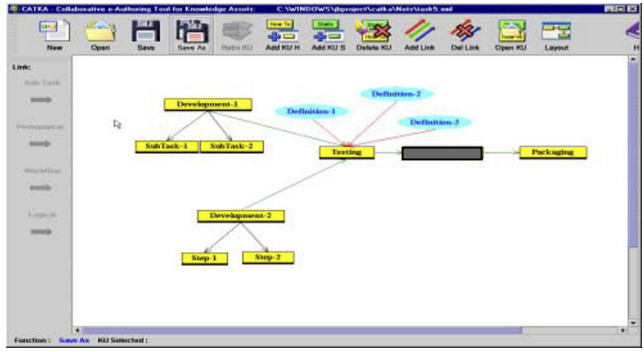
Task #3: *Development-2*: The server receives the KN developed by the second developer. The new added KUs of this developer are updated .



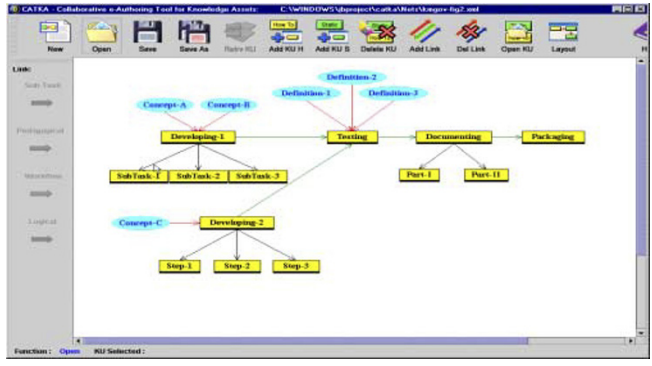
Task #4: *Requesting*: The tester requests the recently updated KN.



Task #5: *Testing:* The tester adds some Static-KUs (ontological terms definitions – rounded boxes). The server receives the updated KN from all members.



Task #6: Writing the Product User Guide. The writer does his work with more KU added. Other members may add mores KU to the KN (as illustrated in the figure 2) and the project leader finalizes the last KN before packaging.



6 Conclusion

The development of e-Government reveals the need of groupware to support inter-agencies collaborative work. However, research efforts are still required to devise adequate formalisms for the management of procedural knowledge assets. In this paper, we provided an overview of an e-Authoring tool to exploit and interchange procedural knowledge represented as networks of semi-structured units. The "bottom-up" approach that can be carried out by this tool is appropriate to gather new terms for the construction of new domain ontology, which will be candidates for the discussion/selection of the underlying community. We detail the updating technique based on the distributed tasks of an e-Authoring project. At this point, we can conclude that group refinement of KN structures offers a simple and intuitive approach to the problem of knowledge interchange. For future work, the exploitation of the logical links created by the CATKA environment will help us to experiment with reasoning schemes, which can be applied to problem solving tasks.

References

1. Multiple authors in Research Challenges and Perspectives of the Semantic Web, Report from the joint European commission and National Science Foundation, Euzenat, J. (Editor), 3-5 October 2001, Sophia Antipolis, France, http://www.ercim.org/EU-NSF/semweb.html.
2. Apostolou, Mentzas, Abecker and Young, 2000. "Consolidating the Product Versus Process Controversy in Knowledge Management: The Know-Net Approach", in the Proceedings of Practical Application of Knowledge Management (PAKeM 2000), Crowne Plaza Midland Hotel, Manchester, UK.
3. Gagné, R.M., Briggs, L.J. and Wager, W.W., 1992. *Principles of Instructional Design*. Harcourt Brace Jovanovich College Publishers. Fort Worth, Texas, U.S.
4. van Elst, Ludger and Abecker, Andreas, 2000. Domain Ontology Agents in Distributed Organizational Memories, *Knowledge Management and Organizational Memories*, Rose Dieng-Kuntz and Nada Matta (Eds.), Kluwer Academic, U.S.
5. Noy, N. and McGuinness, D. L.: Ontology Development 101: A Guide to Creating Your First Ontology. 2001, Stanford Medical Informatics technical reports, SMI Report Number: SMI-2001-0880.
6. Lassila, O. and. McGuinness, D. L.: The Role of Frame-Based Representation on the Semantic Web, in Electronic Transactions on Artificial Intelligence, Volume 5, Number: 2001-03-07, 1403–2031 (Printed version) 1403-204X (Electronic version).
7. Lê, T.H., Gauthier, G. and Frasson, C., 1998. The Process of Planning for an Intelligent Tutoring System, in *Proceeding of The Fourth World Congress On Expert Systems*, Vol. 2, pp. 709–714, Mexico City, Mexico.
8. Chandrasekaran, B., Josephson, J. R. and Benjamins V. R., 1998. The Ontology of Tasks and Methods, in *Proceedings of KAW'98,* Gaines, B. and Musen, M. (eds.), Banff, Alberta, Canada. Foreword of Proceedings of SWWS'01, 2001. The First Semantic Web Working Symposium, Stanford University, CA.