

Workshop on Knowledge Management and Organizational Memories

Rose Dieng (INRIA ,France), Nada Matta (UTT, France), Stefan Decker (Universität Karlsruhe, Germany), Knut Hinkelmann(GmbH, Germany), Ann Macintosh(International Teledemocracy Centre, United Kingdom), Juergen Mueller(Deutsche Telekom AG, Germany), Agostino Poggi (Dipartimento di Ingegneria dell Informazione, Italy), Ulrich Reimer(Swiss Life Information Systems Research, Switzerland), Carla Simone(University of Torino, Italy), Steffen Staab(AIFB ,Germany)

Knowledge Management and Organizational Memories

Knowledge Management (KM) is one of the key progress factors in organizations. It involves explicit and persistent representation of knowledge of (geographically) dispersed groups of people in the organization, so as to improve the activities of the organization. Although KM is an issue in human resource management and enterprise organization beyond any specific technology questions, there are important aspects that can be supported or even enabled by intelligent information systems. Especially AI and related fields provide solutions for important parts of the overall KM problem.

- The know-how of an organization may consist of problem solving expertise in functional disciplines, experiences of human resources, and project experiences in terms of project management issues, design technical issues and lessons learned. The coherent integration of this dispersed know-how in a corporation, aimed at enhancing its access and reuse, is called "corporate memory" or "organizational memory" (OM). It is regarded as the central prerequisite for IT support of Knowledge Management and is the means for knowledge conservation, distribution, and reuse. An OM enables organizational learning and continuous process improvement.
- Identification and analysis of a company's knowledge-intensive work processes (e.g., product design or strategic planning). Knowledge Engineering and Enterprise Modeling techniques can contribute to this topic. The analysis of information flow and involved knowledge sources allows to identify shortcomings of business processes, and to specify requirements on potential IT support.

Activities underlying knowledge management in an organization can comprise detection of needs, construction, distribution, use and maintenance of the corporate memory. It demands abilities to manage disparate know-how and heterogeneous viewpoints, to make it accessible and suitable for adequate members of the organization. When the organization knowledge is distributed on several experts and documents in different locations all over the world, the Internet or an Intranet inside the organization and World Wide Web (WWW) techniques can be a privileged means for acquisition, modelling, management of this distributed knowledge.

The main objectives of KM/OM workshop is to discuss as well as the coherent integration of the dispersed know-how in a corporation, knowledge capitalization, distribution, reuse, and knowledge engineering and enterprise modeling techniques.

Program Committee

Hans Akkermans, University of Twente (NL)
Jean-Paul Barthès (UTC-Compiègne, France)
John Debenham, University of Technology, (Sydney, Australia)
Stefan Decker, Universität Karlsruhe (Germany)
Rose Dieng, INRIA (France)
John Domingue, Open University, (UK)
Jean-Louis Ermine, CEA, (Paris, France)
Jérôme Euzenat, INRIA Rhône-Alpes (Grenoble, France)
Knut Hinkelmann, Insiders Information Management GmbH (Germany)
Robert Jasper, Boeing, (USA)
Ann Macintosh, International Teledemocracy Centre (United Kingdom)
Dirk Mahling, University of Pittsburgh (USA)
Nada Matta, University of Technology of Troyes/Tech-CICO (France)
Frank Maurer, University of Calgary (Canada)
Enrico Motta, Open University (UK)
Juergen Mueller, Deutsche Telekom AG (Germany)
Philippe Pérez, ATOS (France)
Agostino Poggi, Dipartimento di Ingegneria dell Informazione (Italy)
Joel Quinteton, LIRMM (France)
Ulrich Reimer, Swiss Life Information Systems Research Group Postfach (Switzerland)
Myriam Ribière, SRI (California, USA)
David G. Schwartz, Bar-Ilan University (Israel)
Rudi Studer, University of Karlsruhe, (Germany)
Mike Uschold, Boeing (USA)
Carla Simone, University of Torino (Italy)
Steffen Staab, AIFB (Germany)
Gertjan van Heijst, CIBIT (Utrecht, The Netherlands)

Table of Content

A Case-Based Reasoning Framework for EnterpriseModel Building, Sharing and Reusing <i>Yun-Heh Chen, David ROBERTSON, Jussi STADER</i>	5
Integrating Textual Knowledge and FormalKnowledge for Improving Traceability <i>Farid CERBAH, Jérôme EUZENAT</i>	10
Learning from experience of incidents in public transportation : A new form of Experience reflection for organizational learning <i>Cheila COLARDELLE & Jean-Luc WYBO</i>	17
Multiple Views on Consensual Categories: A Contribution for Corporate Memory Management <i>Sylvie DESPRES</i>	27
Agent Architecture and Interaction Protocols for Corporate Memory Management Systems <i>Federico BERGENTI, Agostino POGGI, Giovanni RIMASSA</i>	39
On the convergence of core technologies for knowledge management and organisational memories: ontologies and experience factories <i>Yannis KALFOGLOU</i>	48
Unifying or reconciling when constructing Organizational Memory? Some open issues. <i>Carla SIMONE</i>	56
Structuring Organizational Memories using Multi- Dimensional Knowledge Networks <i>Tang-Ho LÉ, Luc LAMONTAGNE</i>	61

A Case-Based Reasoning Framework for Enterprise Model Building, Sharing and Reusing

Yun-Heh Chen-Burger¹ and David Robertson² and Jussi Stader¹

Abstract. Enterprise model development is essentially a labour-intensive exercise. Human experts depend heavily on prior experience when they are building new models making it a natural domain to apply *Case Based Reasoning* techniques. Through the provision of model building knowledge, automatic testing and design guidance can be provided by rule-based facilities. Exploring these opportunities requires us not only to determine which forms of knowledge are generic and therefore re-usable, but also how this knowledge can be used to provide useful model building support. This paper presents our experiences in identifying and classifying the knowledge which exists in *IBM's BSDM Business Models* and applying AI techniques, *CBR* and *Rule-Based* reasoning together with a symbolic simulator, to provide more complete support throughout the enterprise model development life cycle.

Key-words Enterprise Modelling, Model Development Life Cycle, Case Based Reasoning, Business Modelling, Process Modelling, Knowledge Management, BSDM, Formal Method.

1 Introduction

The main task of *BSDM's Business Modelling* is to identify two conceptual components: entities and dependencies. Entities are things that a business needs to manage and dependencies are the relationships between these things. Certain kinds of scenarios or relationships between entities are common to many businesses. Hence, one would expect that the corresponding *BSDM Business Model* maps reflect these commonalities.

In practice, IBM provides a catalogue of such generic entity models [8]: some of them are standard and example models from the method and some of them were specifically developed for selected industries. Provided with these generic models, BSDM practitioners help clients build their business model by using this information implicitly or explicitly. For BSDM consultancy, King[9] suggested three possible ways of re-using generic/known models when addressing a new problem domain.

- *Back-Pocket Approach*: the clients are made aware of the existence of these generic models, but they are only used to support consultancy. The client will see little or none of the generic model. A consultant keeps these generic models at the back of his/her mind and tailors them to the clients' special requirements.

- *Reference Model Approach*: supply the client with a relevant and complete generic model with detailed description, together with a contractual consultancy service which provides help for the interpretation and use of the model.
- *Software System Solution*: provide developed software systems as packages which are based on generic industrial models. These software systems can then be used by the clients. The client may or may not see the generic business model which was used to develop the required software system.

The fact that similar practices are exhibited in many different businesses and business models are reusable in practice make them a perfect domain candidate for applying *CBR* techniques. *Case-Based Reasoning (CBR)*[10] was inspired by observing human reasoning when learning how to solve new problems by remembering solutions that were applied to similar problems in the past, thus becoming more competent in dealing with wider range of problems over time. In the same way, a *CBR* system solves a new problem by comparing it with old problems and their solutions, which are stored in the system's memory, a *Case Library*. Several *CBR* systems have been built to support design: *Cadet*[13][15] supports better conceptual design for electro-mechanical devices; *Cadsyn*[12] provides guidance for architectural design and adapts existing designs for new buildings; *Casacad* [11] and *AskJef* [2] use multimedia technology to store and present their cases to the user, the former in the domain of architectural design, the latter in the domain of human-machine interface design. Other example *CBR* systems are *Archie-II*[5], *Cadre*[6], *Kritik-II*[14] and *Julia*[7].

In the context of *BSDM*, the standard and example models from the method and the generic models built for a particular industry can be stored in the *Case Library*. The next step is to understand how one can make use of these models and provide useful automatic support for the modeller. BSDM also provides a semi-formal step-by-step procedure for building a business model which includes modelling rules, check lists and recommendations of different strength about good modelling style. This knowledge also forms a natural source for constructing error-checking and advisory rules. However, not all model building knowledge can be formalised. For example, the rule which requires the user to examine whether all of the important concepts are included in the model can not be formalised and automatically checked by our logical rules. The initial BSDM business model is a static model with system dynamic implications. To demonstrate the dynamic aspects of the model, we have extended its original notation and enabled a model execution phase in our *Business Model Simulator*. Both pieces of work are described in more detail in [4].

This paper presents how knowledge which is possessed by different stake-holders: in the business modelling method, in the built industrial models, and in individual practitioners, can be captured and

¹ AIAI, The University of Edinburgh, 80 South Bridge, Room E32, Edinburgh EH1 1HN, UK, email: jessicac@aiai.ed.ac.uk, jussi.stader@aiai.ed.ac.uk

² Department of Artificial Intelligence, The University of Edinburgh, 80 South Bridge, Room E13, Edinburgh EH1 1HN, UK, email: dr@dai.ed.ac.uk

formalised to provide coherent and comprehensive support throughout the *model development life cycle*. It considers two issues: is such knowledge generic and reusable, and how can this knowledge be used to provide automatic support. The paper first describes how *Case Based Reasoning* techniques can be used to provide a common platform for knowledge sharing. It then presents to which extent this knowledge can be formalised and provide assistance for model building activities.

2 The Modelling Support Framework

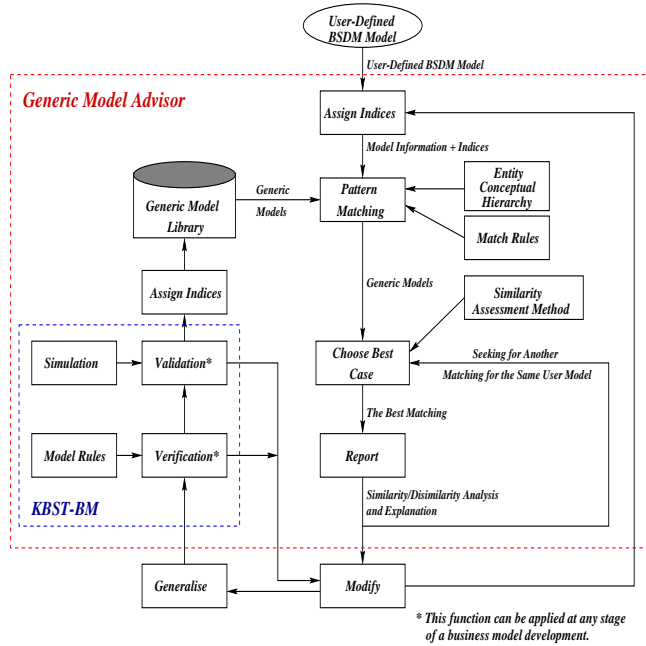


Figure 1. Architecture of Generic Model Advisor

Figure 1 shows the modelling framework which provides automatic facilities to support the *iterative plan-build-test-refine* modelling development life cycle as shown in Figure 2.

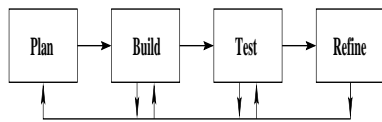


Figure 2. The Plan-Build-Test-Refine development cycle

Two integrated knowledge based support tools, *Generic Model Advisor*(GMA) and *Knowledge Based Support Tool for Business Modelling*(KBST-BM), have been built. Since a BSDM's business model is organised and presented in *views* and *diagrams*, these are the "units" that GMA stores and retrieves. GMA identifies and assigns indices (features which characterise a model) to the problem, i.e. the user-defined BSDM model. These indices, together with the embedded domain knowledge, in our case the *Entity Conceptual Hierarchy* and *Match Rules*, are passed to the pattern matching algorithm which compares the indices of the problem and those of the generic models

in the *Generic Model Library* to retrieve a set of reference models which exhibit similar characteristics to the input model.

At this stage the retrieved similar generic models are not yet examined to determine which is a better match for the current problem. For such a comparison, GMA provides a flexible *Similarity Assessment Function* which enables the deployment of a built-in heuristic method or the users can dynamically make up their own evaluation method to explore specific matches based on the identified indices of the model.

The best matching case, according to the chosen similarity assessment method and an analysis report of similarities and differences between the user model and the retrieved reference model together with suggestions about how to eliminate the causes of the differences, are given to the user. The user can then read the report and/or ask the system to present a different matching result for another generic model. Matches are shown in the descending order of their scores in the chosen similarity assessment method. A summary of all of the matches shown to the user is produced separately which records the similarity measurements of each match to give the user an overview of all possible mappings and to allow revisiting of selected generic models.

A user-defined model may be matched with more than one generic models. The user can choose to modify his/her model and repeat the above modelling cycle as a part of an iterative process. If the user has decided to use the reference model as a basis to generate a new model, the user can export the chosen reference model from the library. At any stage of the model development, the user can choose to use the *verification* and *validation* facilities provided by KBST-BM to check for the completeness, soundness and appropriateness of the built model.

When the user is sufficiently satisfied with his/her model, he/she can retain this new model, i.e. write it back to GMA, by firstly generalising the new model, verifying and validating the generalised model using the integrated tool KBST-BM, and then storing the new generic model back to the *Generic Model Library*. The *Case Based Reasoning Cycle* is now completed, and GMA's knowledge can be enriched and evolved through time via the inclusion of newly acquired knowledge during operations. GMA does not provide an automatic adaptation facility for two reasons. First, there is no absolute standard which fits all businesses in determining whether or not a particular design is the most *appropriate* one for a business. Secondly, although common practices are shared by many businesses, business models are in general organisation-dependent and building a good model requires understanding of the organisation's operation and a consensus within the organisation which may not be available or formalisable due to the size and nature of the required knowledge[4]. Both issues have to be resolved before high quality automatic adaptation can be provided.

The inner KBST-BM system box in the Figure 1 illustrates how KBST-BM can assist in completing the *CBR cycle*. It provides an independent *verification* and *validation* (V&V) facilities (from the user) and is included in the "Test" activity in the standard model development process shown in Figure 2. This V&V approach and implementation details of KBST-BM are given in [3].

3 Indexing, Matching and Similarity Assessment

Indices are features which can be used to distinguish models in the case memory and to find appropriate matches between a given problem and previous models. In the context of a BSDM business model, these distinguishing characteristics are embedded in the semantics of

entities, the architecture of a business model, and the business area that a model describes.

Simply comparing the graphical representation of business models is not sufficient. For example, drawing an existing model upside-down does not make it a different model, the semantics of the inter-relationships (dependencies) between entities must be taken into account. Furthermore, business contextual similarities may be disguised. For instance, if a business model is a more elaborated or specialised version of another one (or vice versa), then these two models normally will not have the same architecture (e.g. one may expand parts of the model in some areas), and often they do not share the same entities (e.g. using domain specific vocabularies instead). However, because they are essentially describing the similar business operations, it will be useful to refer one to the other.

To be able to make meaningful comparisons between BSDM models, one must have an integral understanding of the business context which is described in both the architecture of a model as well as the business context that each entity represents. We capture part of this context through typing of entities via a concept hierarchy.

3.1 Entity Conceptual Hierarchy (ECH)

BSDM provides *Entity Families* which provide entities in groups according to where and how they can be used in a business model. BSDM modellers use *Entity Families* as a starting point when trying to identify entities for a new model. They also use it as a guideline to check the architecture of the model. We organise information given in the *Entity Families* in a taxonomic hierarchy, called the *Entity Conceptual Hierarchy*.

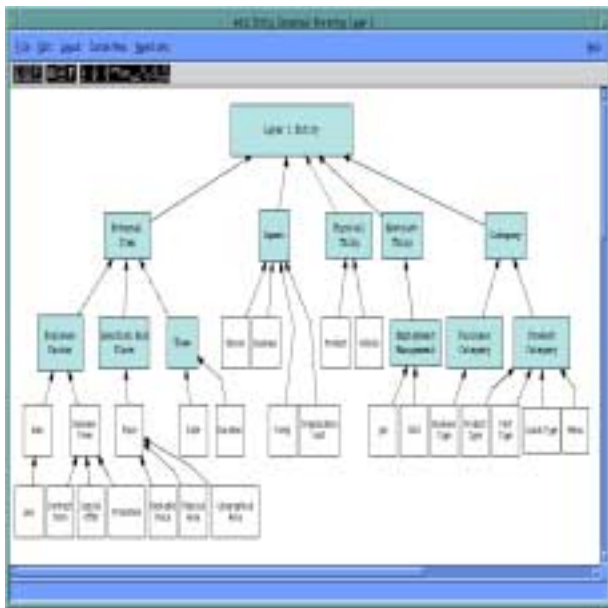


Figure 3. A Part of Entity Conceptual Hierarchy (ECH)

Figure 3 shows a screen shot from *GMA* which captures a part of the *Entity Conceptual Hierarchy* which contains the suggested entities for the top layer (layer 1) of a BSDM business model. Two types of classes have been used to describe entities: the shaded rectangular boxes represent the *Abstract Entity Types*, and the clear rectangular boxes represent the *Concrete Entity Types*. *Abstract Entity*

Types provide a structure to allocate conceptual categories and normally describe more “general” concepts. *Concrete Entities* present more specialised concepts and include entities which are used in real business models (as opposed to a generalised model). An arrow from entity B to entity A indicates an *is-a* relationship from B to A, i.e. B *is-a* A.

The *Entity Conceptual Hierarchy* captures the semantics of all of the entities (in the user and reference models) as well as the relationships between them and it can be used to identify and match similar entities used in the user and reference models.

3.2 Case Retrieving and Similarity Assessment

The *Pattern Matching Algorithm* compares the contextual and architecture information of the given user model with that of all of the reference models stored in the *Generic Model Library*. Several types of information is taken into account. Do these models describe a similar business area? Are they capturing similar concepts? Do they follow similar business rules? The contextual and architecture information is stored in the business area, *view*, links, *dependencies*, and in the *entities*.

Provided with knowledge embedded in *ECH*, one can now match views, dependencies and entities to determine if two different models are sufficiently similar. To match entities, for instance, entities which have the same name in both user and reference models produce a positive match. However, similar but variant entities (sibling relationships in the *ECH*), or “stream-line” specialisations (e.g. parent and child, or grandparent and grandchild relationships) may also produce a positive match. When deciding which is a better match between entities, the closer the relationship is between the two entities on the *ECH*, the better quality of a match it is.

A user model may include several generic models. On the other hand, a generic model may include or partially overlap with the user model. Figure 4 shows the possibilities how a user model may be mapped to a generic model.

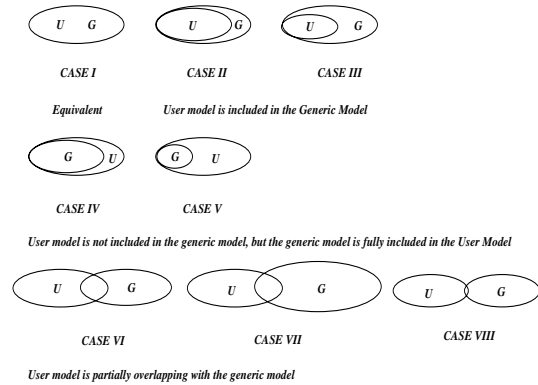


Figure 4. Possible Matching between User Models and Generic Models

As our aim is to seek for the best or better match, naturally a 100% match is always given the highest priority, therefore CASE I. The second preference goes to a match in which an user model is fully included by the selected generic model, hence CASE II and III. However, CASE II is superior than CASE III because its generic model is more similar to the user model: it has a smaller difference compared between the two models.

When a user model is not fully covered, we prefer a match where the user model has a better coverage from the selected generic model, hence CASE IV is superior to CASE V, VI and VII which are all more superior than CASE VIII. In the case of V, VI and VII where the coverage of commonality of the user model are similar, the quality of the matched generic model should be taken into account, i.e. a generic model which is more similar to the user mode should be preferred. Since the generic model in CASE V is entirely included in the user model, it is the most similar (or relevant) one to the user model, CASE VI is in second place, and CASE VII is the least similar one to the user model since it has a comparatively smaller common portion with the user model.

Based on our preferences, five discriminating criteria are identified: the matching result of the captured business areas, the matching ratio of links (dependencies) in the selected reference model, the matching ratio of entities in the selected reference model, the matching ratio of links (dependencies) in the user model and the matching ratio of entities in the user model.

HEURISTIC SIMILARITY ASSESSMENT FUNCTION

Given two matches, X and Y

if $\text{match-view}(X) > \text{match-view}(Y)$ then SELECT X

else if $\text{match-view}(X) = \text{match-view}(Y)$ and
 $\text{match-data-link}(X) > \text{match-data-link}(Y)$ then SELECT X

else if $\text{match-view}(X) = \text{match-view}(Y)$ and
 $\text{match-data-link}(X) = \text{match-data-link}(Y)$ and
 $\text{match-data-entity}(X) > \text{match-data-entity}(Y)$ then SELECT X

else if $\text{match-view}(X) = \text{match-view}(Y)$ and
 $\text{match-data-link}(X) = \text{match-data-link}(Y)$ and
 $\text{match-data-entity}(X) = \text{match-data-entity}(Y)$ and
 $\text{match-case-link}(X) > \text{match-case-link}(Y)$ then SELECT X

else if $\text{match-view}(X) = \text{match-view}(Y)$ and
 $\text{match-data-link}(X) = \text{match-data-link}(Y)$ and
 $\text{match-data-entity}(X) = \text{match-data-entity}(Y)$ and
 $\text{match-case-link}(X) = \text{match-case-link}(Y)$ and
 $\text{match-case-entity}(X) > \text{match-case-entity}(Y)$ then SELECT X

else SELECT Y

Figure 5. The Heuristic Similarity Evaluation Function

Figure 5 shows the heuristic evaluation method provided by *GMA*. It provides a means to use the evaluation criteria in selecting a better model which complies with the preference order demonstrated earlier. This method produces good results using our test data (see Section 4). Alternatively, the user can dynamically design their own evaluation methods using *Weighted City-Block* evaluation function based on the above criteria, if they wish to explore specific aspects of models in the library.

4 Evaluation

For evaluation purposes, we obtained a variety of *BSDM* models from different domains. Part of a real industrial model which was developed by an international automobile company.³ One generic business model for small and medium-sized restaurant was developed based on interviews of three independent family restaurant (ex-)owners to enlarge our testing base. We also captured example and standard models from *BSDM* and stored them in our *Generic Model*

³ The company wishes to keep its identity confidential.

Library. In total, the library contains about a dozen of models described in 15 different *views*, represented in 25 diagrams.

The evaluation was concerned with the following issues: (1) to which extent can the tool provide a starting point to help build a new model; (2) how capable is the tool in helping detect model errors by retrieving the appropriate reference models; (3) how well can the system help to retain new knowledge and store it for future reuse. In other words, we are interested in determining how well the tool can help to speed-start model building, encourage good modelling practice and accumulate model building knowledge.

Althoff *et al* [1] proposed a useful evaluation framework to test both the theoretical and practical aspects of a *Case-Based Reasoning* system. Adapting their method, four types of tests were designed and carried out. Firstly, by giving only very little information, a test was carried out on *GMA* to determine if it can provide any useful assistants by retrieving similar models. Secondly, to test the capability of *GMA* to cope with “noisy” models, pre-determined portions of data were deleted from the original models which were then used as input for *GMA*. The result was used to compare with the expected (i.e. perfect) result when the original model was used.

Thirdly, the above automobile industrial model was used as the user-defined model. Since the automobile model was developed independently by and for a real business, it would be a good testing vehicle to demonstrate if *CBR* techniques can be used to contribute to general business model building exercises. The intention was also to determine whether or not *GMA* could retrieve similar cases from the library, given sufficiently different model architecture and entity names.

One vital step for a *Case Based Reasoner* is in its ability to retain and reuse new knowledge. Therefore, the final test was to use *GMA* as a modelling tool to develop, generalise, verify and validate (with the help of *KBST-BM*) and retain a business model, and then export it from the *Generic Model Library* as a new model. The results obtained demonstrated that even when provided with only partial and noisy models, the system was still able to retrieve all relevant reference cases where they existed in the library. We also observed that the matching result was largely influenced by the matching of the view name of the data model. However, in the absence of a matching view name, *GMA* still retrieved good matching cases from the library. In fact, out of the 10 different tests and 29 different sets of data, all of the tests successfully retrieved the best and good matching cases.

Although the above tests are encouraging, it is possible to produce scenarios where the system may not produce similarly successful results, i.e. instead of using a correct partial model, it gives an erroneous model containing vital mistakes. For example, when a business model uses an entirely wrong view name or a business model which is grossly misrepresented. When the input model is given in such a way it will misguide the system to believe that it is more similar to another reference model, hence the retrieval case will probably be incorrect. We, however, believe that the modellers normally have sufficient judgement not to make such vital mistakes.

During the third test, i.e. given an input model with significant different architecture and entity names, *GMA* was also able to retrieve all of the similar reference models for it, and present them in a reasonable order of preference. The testing result showed that although some of our cases in the library are much less complicated and smaller in scale and most of them indeed describe a different domain of business, useful similarities (in the same business areas across sectors) are still being identified using *GMA*. This also demonstrated the fact that at this level of abstraction common practices are exhibited in different business environments and can be reused.

KBST-BM integrates with *GMA* together provide an adequate framework for *CBR*, i.e. automatic indexing input data, retrieving relevant cases from library, comparing and analysing input with selected cases, revising cases for current problem, verifying and validating input, and retaining the new inputs for future reference. This allows us to use the larger *KBST-BM BSDM* modelling environment in the adaptation phase of the *CBR* cycle. We tested this route using the automobile and restaurant models.

5 Conclusion

Successful business model development requires both methodological and application domain knowledge and experience. Unfortunately, few people possess all of these capabilities. Our studies of applying *CBR* and Rule-Based techniques which are based on a coherent underlying formal method shows how model building knowledge can be obtained, reused and used to provide automatic verification and validation facilities. We believe that with this support we are able to enhance the level of knowledge sharing, and ability of problem solving. More importantly, it adds to our understanding of how this sort of seemingly informal method can fit into parts of the design lifecycle which require formal models.

REFERENCES

- [1] Klaus-Dieter Althoff, Eric Auriol, Ralph Barletta, and Michel Manago, *An AI Perspectives Report: A Review of Industrial Case-Based Reasoning Tools*, An AI Perspective Report, AI Intelligence, P.O.Box 95, Oxford OX2 7XL, 1995.
- [2] J. Barber, S. Bhatta, A. Goel, M. Jacobsen, M. Pearce, L. Penberthy, M. Shankar, and E. Stroulia, *Integrating Case-Based Reasoning And Multimedia Technologies For Interface Design Support*, In *Artificial Intelligence In Design*, Editor: J. G. Boston, Kluwer Academic Publisher, 1992.
- [3] Yun-Heh Chen-Burger, Dave Robertson, and Jussi Stader, 'Knowledge-Based Automatic Verification and Validation for Business Models', *DARPA-JFACC Symposium on Advances in Enterprise Control*, (November 1999).
- [4] Yun-Heh Chen-Burger, David Robertson, and Jussi Stader, 'Formal Support for an Informal Business Modelling Method', *The Special Issue for The International Journal of Software Engineering and Knowledge Engineering*, (February 2000).
- [5] E. Domeshek, J. Kolodner, and C. Zimring, 'The Design of a Tool Kit for Case-Based Design Aids', *Proceedings of the Third International Conference on Artificial Intelligence in Design*, (1994).
- [6] B. Faltings, *Case Reuse By Model-Based Interpretation*, in *Issues and Applications of Case-Based Reasoning in Design*, Editor: M. L. Maher, P. Pu, Lawrence Erlbaum Associates, Hillsdale, N.J., 1997. pp. 30-60.
- [7] T.R. Hinrichs, 'Towards an Architecture for Open World Problem Solving', *Proceedings of CBR workshop*, pp. 182-189, (1988). Morgan Kaufmann, San Francisco.
- [8] IBM United Kingdom Limited, 389 Chriswick High Road, London W4 4AL, England, *Business System Development Method: Business Mapping Part1: Entities*, 2nd edn., May 1992.
- [9] Martin King, 'Knowledge Reuse in Business Domains Experience with IBM BSDM', Technical report, Artificial Intelligence Application Institute, (1995).
- [10] Janet Kolodner, *Case-Based Reasoning*, Morgan Kaufmann Publishers, Inc., 2929 Campus Drive, suite260, SanMateo, CA, USA, 1993.
- [11] M. L. Maher, B. Balachandran, and D. M. Zhang, *Case-Based Reasoning In Design*, Lawrence Erlbaum, 1995.
- [12] M. L. Maher and A. Gomez de Silva Garza, 'Developing Case-Based Reasoning for Structural Design', *IEEE Expert, Intelligent Systems and Their Applications*, **11**(3), (June 1996).
- [13] S. Narashiman, K. Sycara, and D. Navin-Chandra, *Representation and Synthesis of Non-Monotonic Mechanical Devices*, In *Issues and Applications of Case-Based Reasoning in Design*, Editor: M.L. Maher, P.Pu, Lawrence Erlbaum Associates, Hillsdale, N.J., 1997.
- [14] E. Stroulia and A. K. Goel, 'Generic Teleological Mechanisms and Their Use in Case Adaptation', *Proceedings of the Fourteenth Annual Conference of the Cognitive Science*, (1992). Northvale, N.J., Erlbaum.
- [15] K. Sycara, R. Guttal, J. Koning, S. Narasimhan, and D. Navinchandra, 'CADET: A Case-based Synthesis Tool for Engineering Design', *International Journal for Expert Systems*, **4**(2), pp. 157-188, (1992). <http://www.cs.cmu.edu/afs/cs.cmu.edu/project/cadet/ftp/docs/CADET.html>.

Integrating Textual Knowledge and Formal Knowledge for Improving Traceability

Farid Cerbah¹ and Jérôme Euzenat²

Abstract. This article deals with traceability in knowledge repositories. More precisely, we concentrate on the role of terminological knowledge in the mapping between (informal) textual requirements and (formal) object models. We show that terminological knowledge facilitates the production of traceability links and model generation, provided that language processing technologies allow to elaborate semi-automatically the required terminological resources. The presented system is one step towards incremental formalization from textual knowledge. As such, it is a valuable tool for building knowledge repositories.

1 INTRODUCTION

Knowledge management has for long been preoccupied by the relationships between formal and informal knowledge. The informal is richer and familiar to any user while the formal is more precise and necessary to the computer. It is recognized that linking formal knowledge to informal knowledge has several benefits in the context of knowledge management including, (1) establishing the context for formalized knowledge and documenting it, and (2) providing a natural way to browse through formalized knowledge. A software tool for supporting link generation, like the one presented in this paper, is an opportunity to kick off incremental, corpus-driven formalization.

In the field of knowledge management, there have been attempts to provide tools supporting the linking of knowledge sources [11, 15, 18]. However, the computational support provided was quite limited. The links had to be established manually and thus were error-prone and time consuming (not only the initial setting of the links but, above all, the updating operations). Besides, the browsing capabilities from formal knowledge to the informal documents were minimal (e.g., the hyperlinks had only one target document). In the meantime, several works focused on the advantages of using a corpus-based terminology for supporting formal knowledge acquisition [4, 1, 2]. These contributions emphasize the central role of terminological resources in the mapping between informal text sources and formal knowledge bases. We put forth an architecture, centered around a terminology extraction and man-

agement tool, which enables to generate models from texts and navigate from one to another through the terminology. We describe a fully implemented system that provides high-level hypertext generation, browsing and model generation facilities. From a more technical viewpoint, we introduce an original XML based model for integrating software components.

The rest of the paper is organized as follows. Section 2 introduces the main concepts of our approach and the basic tasks that should be performed by a user support tool which exploits terminological knowledge for improving traceability. Section 3 gives a detailed and illustrated description of the implemented system. Finally, section 4 briefly compares our contribution to related works and the conclusion provides some directions for further research.

2 PRINCIPLES

2.1 Traceability in software engineering and knowledge repositories

In software engineering, it is often stressed that design and implementation decisions should be “traceable”, in the sense that it should be possible to find out the requirements impacted, directly or indirectly, by the decisions. In a similar way, when building a somewhat formal (or at least structured) repository from document sources, the concepts in the formal repository must be linked to their original sources in the texts. This mapping is useful in many respects:

- It helps to ensure exhaustiveness: By following traceability links, the user or a program can easily identify the concepts which are not represented in the repository.
- It facilitates the propagation of changes: At any time in the elaboration process, traceability information allows to find out the elements impacted by changes (upstream and downstream).
- When traceability is established with hyperlinks, the browsing capabilities of the overall repository are increased.

Moreover, in the context of generalized knowledge management, traceability of elaborated knowledge from raw text provides both grounding and arguments for decisions.

In an object-oriented framework, many traceability links aim at relating textual fragments of the documents in natural language and model fragments. Putting on these links manually

¹ Dassault Aviation - DPR/DESA - 78, quai Marcel Dassault 92552 cedex 300 Saint-Cloud - France - E-mail: farid.cerbah@dassault-aviation.fr

² Inria Rhône-Alpes - 655, avenue de l'Europe 38330 Monbonnot St Martin - France - E-mail: Jerome.Euzenat@inrialpes.fr
<http://www.inrialpes.fr/exmo/>

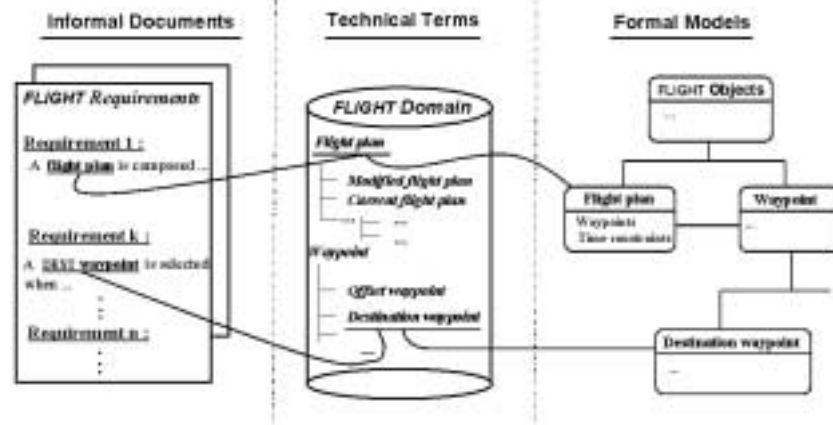


Figure 1. Using terminological items to link textual requirements and object models

is a tedious and time consuming task and current tools for requirement analysis or knowledge acquisition provide no significant help for doing that job (except [15]).

2.2 The role of terminological resources

In many information systems where both textual knowledge and formal knowledge are involved to describe related concepts, terminology can play an intermediate role. As mentioned earlier, previous works in the fields of knowledge acquisition and natural language processing have shown that terminological resources extracted from corpora can help in the incremental formalization processes from texts to formal models. There exists other demonstrative examples in related domains, such as product data management and software engineering.

For example, in the DOCSTEP project [9], which deals with product data management, terminological resources are used to connect multilingual technical documentation and items of product trees. Hyperlinks are established between term occurrences in documents and corresponding objects in product trees.

In software engineering, the role of terminological knowledge in the modeling process has often been pointed out [19, 12, 3]. One of the first step in the modeling process consists in a systematic identification of the technical terms (simple and compound nouns) in the documents, namely the terminology used to describe the problem. Some of these technical terms represent concepts which will be subsequently introduced in the formal models. These terms can be seen as an intermediary level between the textual requirements and the formal models. (see figure 1).

2.3 Functional view of a system that exploits terminology

A system that takes advantage of terminological resources may involve techniques pertaining to several technological areas, and particularly natural language processing, information retrieval and knowledge management:

Terminology Extraction. In technical domains, many precise and highly relevant concepts are linguistically represented by compound nouns. The multi-word nature of the technical terms facilitates their automatic identification in texts. Relevant multi-word terms can be easily identified with high accuracy using partial syntactic analysis [4], [13] or statistical processing [6] (or even both paradigms [8]). Terminology extraction techniques are used to automatically build term hierarchies that will play the intermediate role between documents and models.

Document and Model Indexing. The technical terms are used for indexing text fragments in the documents. Fine grained indexing, i.e paragraph level indexing, is required while most indexing systems used in information retrieval work at the document level. Besides, most descriptors used in this kind of indexing are multi-word phrases. The terms are also used for indexing the model fragments (classes, attributes ...).

Hyperlink Generation. The terminology driven indexing of both texts and models with the same terminology is the basis of the hyperlink generation mechanisms. Furthermore, hyperlink generation should be controlled interactively, in the sense that the user should be able to exclude automatically generated links or add links that have not been proposed by the system.

Model Generation. It is quite common that the concept hierarchies mirror the term hierarchies found in the documents. This property can be used to generate model skeletons which will be completed manually.

These features are implemented in the system presented in the next section.

3 A USER SUPPORT TOOL FOR IMPROVING TRACEABILITY

The implemented system consists of two components, XTerm and Troeps. XTerm deals with the document management and linguistic processing functions, more particularly terminological extraction and the document indexing. Troeps

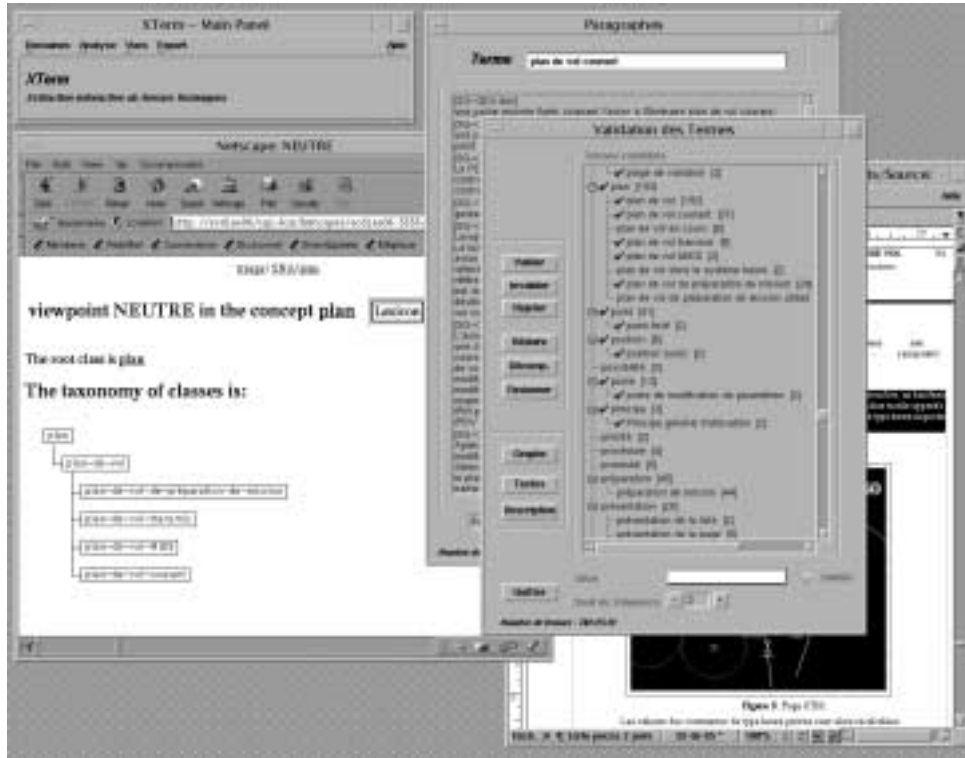


Figure 2. The integrated system based on XTerm and Troeps.

deals with knowledge management and model indexing. The model generation function is spread over both components.

3.1 Terminology extraction with XTerm

XTerm [5] is a natural language processing tool that provides two services to end users:

- Terminology acquisition from documents. It analyzes a French or English technical documentation in order to build a hierarchy of potential technical terms. The user can explore and filter the extracted data via a graphical interface.
- Terminology-centred hypertext navigation. XTerm can be seen as a hypertext browser. The extracted terms are systematically linked to their textual contexts in the documents. The user can easily access the textual fragments containing term occurrences.

Starting with a document collection, XTerm scans all document building blocks (paragraphs, titles, figures, notes) in order to extract the text fragments. These word sequences are then prepared for linguistic processing. Additionally, it provides the mechanisms for indexing and hyperlink generation from technical terms to document fragments. Hyperlink generation is a selective process: To avoid overgeneration, the initial set of links systematically established by the system can be reduced by the user.

The first linguistic processing step is POS tagging. We used a rule based tagger based on the Multex morphological parser

[17]. POS tagging starts with morphological analysis which assigns to each word its possible morphological realizations. Then, contextual desambiguation rules are applied to choose a unique realization for each word. At the end of this process, each word is unambiguously tagged.

As mentioned in section 2.3, the morpho-syntactical structure of technical terms follows quite regular formation rules which represent a kind of local grammar. For instance, many French terms can be captured with the pattern “*Noun Preposition (Article) Noun*”. Such patterns can be formalized with finite state automata, where transition crossing conditions are expressed in terms of morphological properties. To identify the potential terms, the automata are applied on the tagged word sequences provided by the POS tagger. A new potential term is recognized each time a final state is reached. During this step, the extracted terms are organized hierarchically. For example, the term “*flight plan*” (“*plan de vol*” in figure 2) will have the term “*plan*” as parent and “*modified flight plan*” as a child in the hierarchy.

The candidate set obtained after this step is still too large. Additional filtering mechanisms are involved to reduce that set. Grouping rules are used to identify term variants. For instance, in French technical texts, prepositions and articles are often omitted for the sake of concision (the term “*page des buts*” can occur in the elided form: “*page buts*”) ³. Term variants are systematically conflated into a single node in the term

³ Whose English literal translations are respectively: “*page of the waypoints*” and “*page waypoints*”. A plausible equivalent term in English could be “*Waypoint page*”.

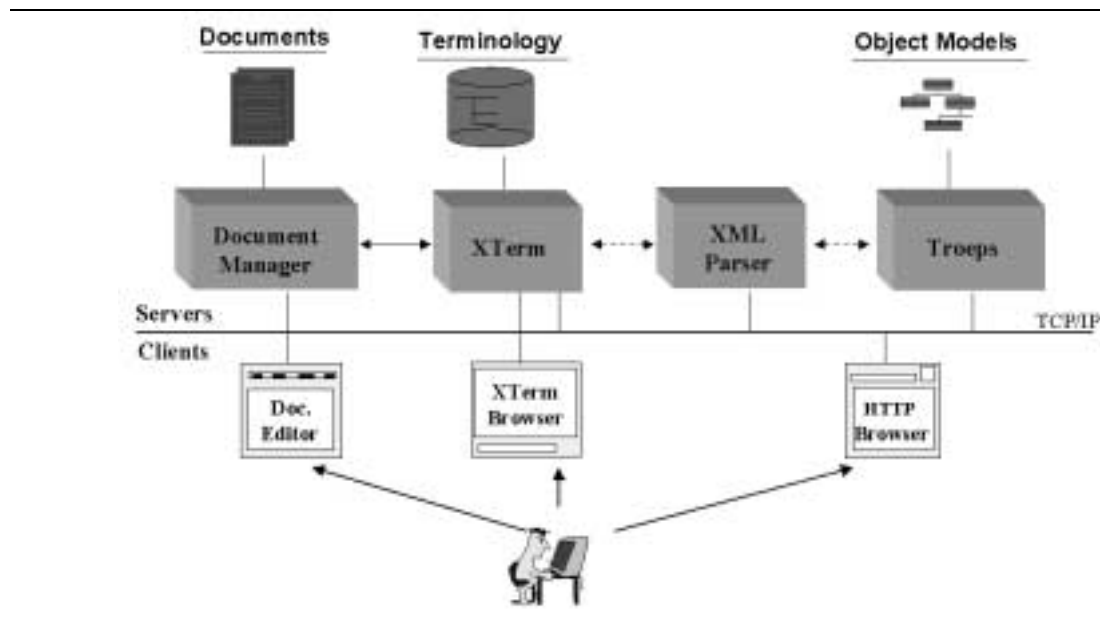


Figure 3. System architecture

hierarchy.

XTerm is highly interactive. Many browsing facilities are provided to facilitate the manipulation of large data sets (extracted terms + text fragments). XTerm can be used as an access tool to documentation repositories.

3.2 Knowledge modeling with the Troeps system

Troeps [14, 21] is an object-based knowledge representation system, i.e. a knowledge representation system inspired from both frame-based languages and object-oriented programming languages. It is used here for expressing the models.

An object is a set of field-value pairs associated to an identifier. The value of a field can be known or unknown, it can be an object or a value from a primitive type (e.g. character string, integer, duration) or a set or list of such. The objects are partitioned into disjoint concepts (an object is an instance of one and only one concept) which determines the key and structure of its instances. For example, the “*plan*” concept identifies a plan by its number which is an integer. The fields of a particular “*plan*” are its time constraint which must be a duration and its waypoints which must contain a set of instances of the “*waypoint*” concept.

Objects can be seen under several viewpoints, each corresponding to a different taxonomy. An object can be attached to a different class in each viewpoint. For instance, a particular plan is classified as a “*flight plan*” under the nature viewpoint and as a “*logistic plan*” under the functional viewpoint. This is unlike other object systems, which usually allow only one class hierarchy.

Object-based knowledge representation provides various facilities for manipulating knowledge among which filtering queries (which find objects of a concept satisfying fields and attachment constraints), similarity queries (function of field

values or attachment classes) involving a distance measure, value inference (through default values, procedural attachment, value passing or filtering), position inference (classification and identification) in which the possible positions of an object or a class in a taxonomy are computed.

Troeps knowledge bases can be used as HTTP servers whose skeleton is the structure of formal knowledge (mainly in the object-based formalism) and whose flesh consists of pieces of texts, images, sounds and videos tied to the objects. Turning a knowledge base into a HTTP server is easily achieved by connecting it to a port and transforming each object reference into an URL and each object into a HTML page. If HTML pages already document the knowledge base, they remain linked to or integrated into the pages corresponding to the objects. The Troeps user (through an Application Programming Interface) can explicitly manipulate each of the Troeps entities. The entities can also be displayed on a HTTP client through their own HTML page. The Troeps program generates all the pages on demand (i.e. when a URL comes through HTTP). The pages make numerous references to each others. They also display various documentation (among which other HTML pages and lexicon) and give access to Troeps features. From a Troeps knowledge server it is possible to build complex queries grounded on formal knowledge such as filtering or classification queries. The answer will be given through a semantically sound method instead of using a simple full-text search. Moreover, it is possible to edit the knowledge base. The system presented here takes advantage of this last feature.

3.3 Communication between the components

The communication between the linguistic processing environment and the model manager is bidirectional: Upon user request, XTerm can call Troeps to generate class hierarchies

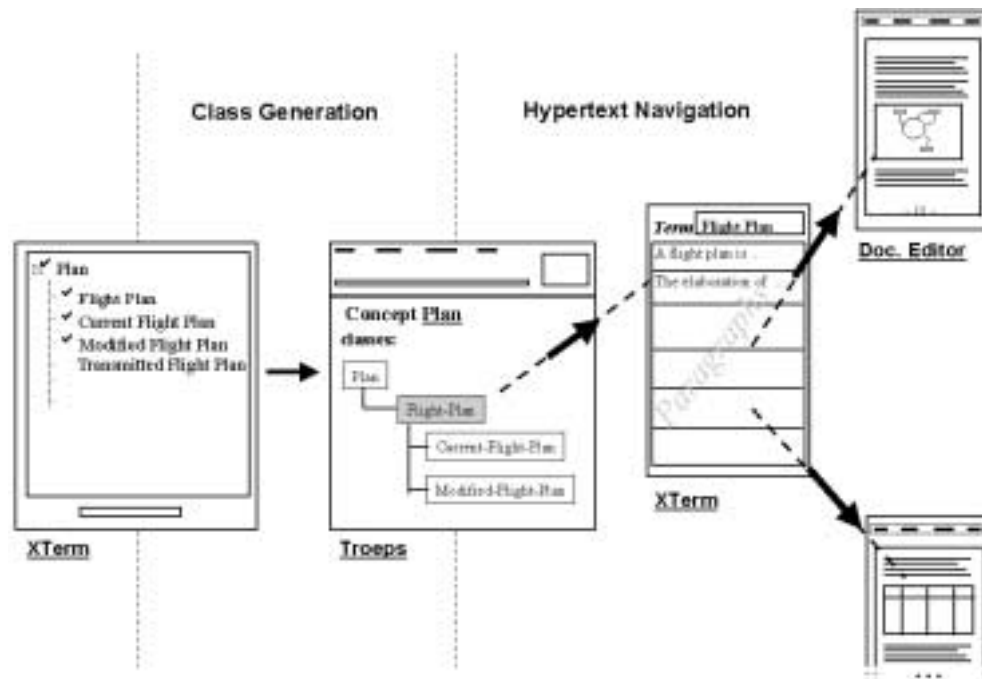


Figure 4. Class generation and traceability through hyperlinks

from term hierarchies. Conversely, Troeps can call XTerm to provide the textual fragments related to a concept (via a technical term).

For example, figure 4 illustrates the class generation process from a hierarchy of terms carefully validated by the user (a hierarchy rooted in the term “Plan”). The class hierarchy constructed by Troeps mirrors the hierarchy of the validated terms (under the root “Plan”).

At the end of the generation process, the created classes are still linked to their corresponding terms, which means that the terminology-centred navigation capabilities offered by XTerm are directly available from the Troeps interface. As illustrated by figure 4, the Troeps user has access to the multi-document view of the paragraphs which concern the “Flight-Plan” concepts⁴. From this view, the user can consult the source documents if required.

Data exchanges between XTerm and Troeps are based on the XML language (see figure 3). Troeps offers an XML interface which allows to describe a whole knowledge base or to take punctual actions on an existing knowledge base. This last feature is used in the interface where XTerm sends to Troeps short XML statements corresponding to the action performed by the user. These actions correspond to the creation of a new class or a subclass of an existing class and the annotation of a newly created class with textual elements such as the outlined definition of the term naming the class. For example, to generate classes from the term hierarchy rooted at the term “plan”, XTerm sends to Troeps an XML stream containing a sequence of class creation and annotation statements.

⁴ More precisely, this view displays the paragraphs where the term “flight plan” and its variants occur.

XML representation of object models. We give below an extract of this sequence, corresponding to the creation of classes “Flight-Plan” and “Current-Flight-Plan”:

```
<trp:ADD>
  <trp:CLASS>
    <trp:CLASSDSC name="Flight-Plan">
      <trp:CLASSREF name="Plan"/>
    </trp:CLASSDSC>
  </trp:CLASS>
</trp:ADD>

<trp:ADD>
  <trp:CLASS>
    <trp:CLASSDSC name="Current-Flight-Plan">
      <trp:CLASSREF name="Flight-Plan"/>
    </trp:CLASSDSC>
  </trp:CLASS>
</trp:ADD>

<trp:ANNOTATE label="comment">
  <trp:CLASSREF name="Flight-Plan"/>
  <trp:CONTENT>
    A flight plan is a sequence of waypoints...
  </trp:CONTENT>
</trp:ANNOTATE>
```

The term definition filled out in the XTerm description of the term is added as a textual annotation in the class description. After these automated steps, the classes can be completed manually.

This XML interface has the advantage of covering the complete Troeps model (thus it is possible to destroy or rename classes as well as adding new attributes to existing classes). Moreover, it is relatively standard in the definition of formalized knowledge so that it will be easy to have XTerm generating other formats (e.g. XMI [16] or Ontolingua) which share the notion of classes and objects.

More details about this approach of XML-based knowledge modeling and exchange are given in [10].

4 RELATED WORK

Terminology acquisition is one of the most robust language processing technology [4, 13, 8] and previous works have demonstrated that term extraction tools can help to link informal and formal knowledge. The theoretical apparatus depicted in [4], [1] and [2] provides useful guidelines for integrating terminology extraction tools in knowledge management systems. However, the models and implemented systems suffer from a poor support for traceability, restricted to the use of hyperlinks from concepts and terms to simple text files. On this aspect, our proposal is richer. The system handles real documents, in their original format, and offers various navigation and search services for manipulating “knowledge structures” (i.e., documents, text fragments, terms, concepts ...). Moreover, the management services allow users to build their own hypertext network.

With regard to model generation, our system and Terminae [2] provide complementary services. Terminae resort to the terminologist to provide a very precise description of the terms from which a precise formal representation, in description logic, can be generated. In our approach, the system does not require users to provide additional descriptions before performing model generation from term hierarchies. Model generation strictly and thoroughly concentrates on hierarchical structures that can be detected at the linguistic level using term extraction techniques. For example, the hierarchical relation between the terms “*Flight Plan*” and “*Modified Flight Plan*” is identified by XTerm because of the explicit relations that hold between the linguistic structures of the two terms. Hence, such term hierarchies can be exploited for class generation. However, XTerm would be unable to identify the hierarchical relation that hold between the terms “*vehicle*” and “*car*” (which is the kind of relations that Terminae would try to identify in the formal descriptions). As a consequence, the formal description provided by our system is mainly a hierarchy of concepts while that of Terminae is more structural and the subsumption relations is computed by the description logic system.

A recent contribution in the field of knowledge management is that of [20] which provides automatic indexing of mail messages in a corporate context. However, the indexing mechanisms do not involve terminological resources.

In the field of software engineering, object-oriented methods concentrate on the definition of formal or semi-formal formalisms, with little consideration for the informal-to-formal processes [19, 12, 3]. However, to identify the relevant requirements and model fragments, designers should perform a deep analysis of the textual specifications. The recommendations discussed in section 2.2 on the use of terminological resources can be seen as a first step.

The transition from informal to formal models is also addressed in [22]. The approach allows users to express the knowledge informally (within texts and hypertexts) and more formally (through semantic networks coupled with an argumentation system). In this modeling framework, knowledge becomes progressively more formal through small increments. The system, called “Hyper-Objet substrate”, provides an active support to users by suggesting formal descriptions of terms. The integrated nature of this system allows to make suggestions while the users are manipulating the text,

and to exploit already formalized knowledge to deduce new formalization steps. Our system, whose linguistic processing component is far more developed, could be coherently embedded in this comprehensive modeling framework.

Our work is also related to the WEB→KB system [7] whose goal is to automatically build large knowledge bases by analyzing the World Wide Web. The system starts with a predefined domain model, composed of classes and relations between them. Potential instances are identified on the Web using machine learning techniques. “Informal instances” of predefined classes and relations may correspond to Web pages, hyperlinks or text fragments. Our approach concentrates on the extraction of model fragments whereas this work focuses on instance identification. No linguistic processing is involved in this system. Textual material is simply viewed as bag of words (without stemming). However, some learning techniques developed in this context could be adapted for model generation.

5 CONCLUSION

Structured knowledge repositories are by nature highly relational and the various relations that hold between knowledge fragments are often expressed through hyperlinks. However, hypertext editing is an expensive and time-consuming activity which, nowadays, is hardly processed automatically, even partially. Our approach emphasizes the need for an active support to hypertext editing. We have presented a fully implemented system that helps users to link formal models to their informal sources.

We assumed in this work that the sources had a low degree of formality, roughly documents with a poorly structured content. Further investigation will address the problem of link generation from semi-formal sources such as SGML and XML documents. With the success of XML, the availability of such semi-formal sources tends to increase. We think that link generation can be significantly improved when the sources are semi-formal. In particular, XML tagging provides useful information about the content structure that allows to accurately identify the potential link anchors.

We also addressed in this work the issue of model generation from informal sources. We proposed robust class generation mechanisms that take advantage of term hierarchies automatically built with NLP techniques. Further work will address automatic generation of more complex knowledge structures such as relations between classes and attributes.

ACKNOWLEDGEMENTS

This work has been partially realized in the GENIE II program supported by the French ministry of education, research and technology (MENRT) and the DGA/SPAé.

REFERENCES

- [1] N. Aussenac-Gilles, D. Bourigault, A. Condamines, and C. Gros, 'How can knowledge acquisition benefit from terminology?', in *Proceedings of the 9th Knowledge Acquisition for Knowledge Based System Workshop (KAW '95)*, Banff, Canada, (1995).
- [2] B. Biébow and S. Szulman, 'Une approche terminologique pour la construction d'ontologie de domaine à partir de textes : TERMINAE', in *Proceedings of 12th RFIA Conference*, pp. 81–90, Paris, (2000).
- [3] G. Booch, *Object-Oriented Analysis and Design with Applications*, Addison-Wesley, 2d edn., 1994.
- [4] D. Bourigault, 'Lexter, a terminology extraction software for knowledge acquisition from texts', in *Proceedings of the 9th Knowledge Acquisition for Knowledge Based System Workshop (KAW '95)*, Banff, Canada, (1995).
- [5] F. Cerbah, 'Acquisition de ressources terminologiques – description technique des composants d'ingénierie linguistique', Technical report, Dassault Aviation, (1999).
- [6] K. W. Church and P. Hanks, 'Word association norms, mutual information and lexicography', *Computational Linguistics*, **16**(1), 22–29, (1990).
- [7] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery, 'Learning to construct knowledge bases from the World wide Web', *Artificial Intelligence, Special Issue on Intelligent Internet Systems*, **118**(1-2), 69–113, (2000).
- [8] B. Daille, 'Study and implementation of combined techniques for automatic extraction of terminology', in *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, eds., J.L. Klavans and P. Resnik, MIT Press, Cambridge, (1996).
- [9] K. Elavaino and J. Kunz, 'Docstep — technical documentation creation and management using step', in *Proceedings of SGML '97*, (1997).
- [10] Jérôme Euzenat, 'XML est-il le langage de représentation de connaissance de l'an 2000?', in *Actes des 6eme journées langages et modèles à objets*, pp. 59–74, Mont Saint-Hilaire, CA, (2000).
- [11] B. Gaines and M. Shaw, 'Documents as expert systems', in *Proceedings of 9th British society expert systems conference*, ed., Cambridge University Press, pp. 331–349, (1992).
- [12] I. Jacobson, *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, 1992.
- [13] J. S. Justeson and S. M. Katz, 'Technical terminology: Some linguistic properties and an algorithm for identification in text', *Natural Language Engineering*, **1**(1), 9–27, (1995).
- [14] O. Mariño, F. Rechenmann, and P. Uvietta, 'Multiple perspectives and classification mechanism in object-oriented representation', in *Proceeding of 9th ECAI*, pp. 425–430, Stockholm, (1990).
- [15] P. Martin, *Exploitation de graphes conceptuels et de documents structurés et hypertextes pour l'acquisition de connaissances et la recherche d'information*, Ph.D. dissertation, Université de Nice-Sophia Antipolis, 1996.
- [16] OMG, 'XML Metadata Interchange (XMI)', Technical report, OMG, (1998).
- [17] D. Petitpierre and G. Russell, 'MMORPH – the Multext morphology program', Technical report, Multext Deliverable 2.3.1, (1995).
- [18] F. Rechenmann, 'Building and sharing large knowledge bases in molecular genetics', in *Proceedings of 1st International Conference on Building and Sharing of Very Large-Scale Knowledge Bases*, pp. 291–301, Tokyo, (1993).
- [19] J. Rumbaugh, *Object-Oriented Modeling and Design*, Prentice-Hall, 1991.
- [20] D. Schwartz, 'When email meets organizational memories', *International journal of human-computer studies*, **51**(3), 599–614, (1999).
- [21] Projet Sherpa, 'Troeps 1.2 reference manual', Technical report, Inria, (1998).
- [22] F. Shipman and R. McCall, 'Supporting incremental formalization with the hyper-object substrate', *ACM Transactions on information systems*, **17**(2), 199–227, (1999).

Learning from experience of incidents in public transportation

A new form of Experience reflection for organizational learning

Cheila COLARDELLE & Jean-Luc WYBO

Ecole des Mines de Paris, Pôle Cindyniques
P.O. box 207, F-06904 Sophia Antipolis (France)
Colardelle@cindy.cma.fr

Abstract

Experience reflection is a management method in which people having participated in the management of an action (an incident or an accident) analyze the development of the situation, learn lessons and apply decisions to avoid problems in the future. The following article describes a reflection review process, which is distinct from the typical processes, in that it focuses on a new form: the *positive experience reflection* (PER). The PER method uses the development of a real event as an opportunity to collect individual experience of several actors and assemble them into a collective experience. It was successfully validated in 1999 when applied to several rail incidents of the RATP, the French collective transportation company operating in the region of Paris.

The PER method accentuates the need for capitalization of all types of experience and know-how, by proposing a simple and structured way in which these can be shared and perpetuated among all actors of a particular system at any given time.

By means of three main graphic supports, the PER allows for a reliable analysis of past incidents and near miss incidents. It traces the unfolding in time and propagation in space of each incident and potential incident situations. It equally allows for a reliable and complete evaluation of the degree of danger and status of the failing system. The dynamic evolution of dangerous situations is thus better understood and the actors' reaction to the system breakdown and crisis management skills is also greatly improved.

Keywords: Experience reflection, tacit knowledge, organizational learning, rail incidents

1. Introduction

The development of emergency management experience depends largely on debriefing sessions and individual learning from operations. Debriefing consists of the analysis of the crisis and lessons learnt from the evaluation of decisions and actions. Individual learning is the result of the analysis that each manager carries out of events, decisions and actions, from his point of view (hierarchical level, position during the incident, etc.).

Collective learning is mainly based on an *experience reflection*¹ (ER) process. It is a post operational evaluation activity that is used to learn from incidents, accidents and crisis to reduce their occurrence. ER is composed of four phases: collect events, analyze events, learn lessons and apply new decisions.

¹ From the French expression "Retour d'expérience"

The ER process is practiced widely today and is commonly viewed as one of the unavoidable building blocks in all effective security plans.

The ROE process makes it possible:

- To react to the probability of risks by avoiding the repetition of past errors,
- To react to the gravity of risks by studying how to limit danger,
- To intervene more effectively during the evolution of crisis situations.

The ER process, as we know it today, was practiced informally by the RATP as early as the beginning of the 20th century. By the early 90's, the ROE method was officially established as a compulsory procedure when analyzing rail incidents. Since the last 5 years, procedures are established by the RATP to store information about incidents and accidents in a database.

Although the ER process significantly reduced the probability of occurrence and the gravity of the effects of rail incidents in general, it was however, far from being fully effective.

One of the reasons was that the ER consisted of applying a “bottom-up approach” by identifying technical problems and supplying solutions at the design and operational levels.

The benefits that the field agents received from the ER process were insignificant if not non-existent, limiting the effectiveness of the ER in practice.

The idea and birth of the *positive experience reflection* (PER) method was thus imminent. There was a need for a more interactive ER method, placing emphasis on the sharing of experience between actors belonging to a dysfunctional system.

2. The Positive Experience Reflection method

The PER method takes into account the complexity of the systems to which it is applied, one of the reasons being that danger cannot be fully assessed in isolation. This is the law of *Cindynics reticularity*, one of the concepts proposed by G. Y. Kervern [Kervern 94]: all the layers of a system must be considered in order to fully grasp the correlation between complexity and vulnerability.

The complexity of the system can be represented by the analysis of three sub-systems:

- Human: employees of all activities;
- Organizational: documents and procedures;
- Technological: technical equipment and machinery.

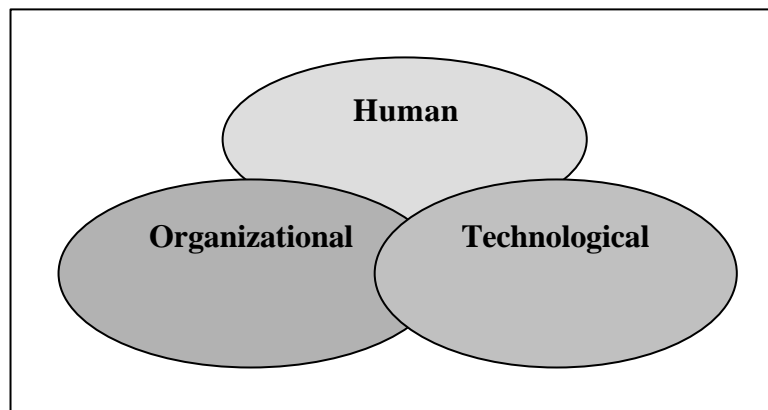


Fig. 1: The three sub-systems

We present the application of the PER method to:

- The derailling of a rail gravel car, in the process of renewal of rail lines².
- The malfunction of electrical devices³ (remote-controlled switchboards).

These two incidents merited special attention in that:

- They were considered by the actors of the system as being “normal”, an acceptable part of the system, due to the frequency with which they occur (twice a year, on the average). Our aim was to highlight the fact that these types of incidents, no matter how apparently negligible, generate substantial losses in the long term and should therefore be diminished if not altogether eradicated from the system.
- These particular incidents had taken place a little over a month before the beginning of this study, therefore the actors still had the incidents “fresh in their minds”, making interchange of experience all the more interactive,

The PER method has been developed to be applied to any past, present and future incident or malfunction⁴. Five separate but complementary steps make up the PER method:

- Step 1: Perception. The maximum amount of data dealing with the incident is collected from available sources to constitute a comprehensive database on the incident. The data sources are: observation of the work environment, written reports, interviews, departmental meetings, informal conversations, etc. This step is the most time and energy-consuming one due to the length of investigation and the variety of dispersed accounts.
- Step 2: Analysis. The data is vetted and a questionnaire is compiled, which provides questions covering a *string of key events*⁵ in the systems breakdown. The questionnaire (series of key events) is presented to two categories of actors: those that were directly involved and those that were not present when the incident unfolded. The incidents are dissected in order to identify key events that led to their occurrence. Once each key event is identified, it is detailed into the 4 phases of what we call a “particle of experience”: a situation, a decision, an action and its effect.
- Step 3: Validation. After having been interviewed individually, the actors receive the results of their interview (a set of particles of experience) and approve or modify them. Lastly, a collective meeting of all the actors of the system is convened in which the final results are discussed and, if necessary, modified. This is the “mirror effect” stage, practiced at individual and collective levels.
- Step 4: Modeling and support to the sharing of experience. Three graphic support methods are used: the String of Key Events Graph used mainly in step 3, the Fault Tree and the Cindynics HyperSpace, used in step 5.
- Step 5: Proposal of measures. Practical measures are proposed for the three sub-systems: Human/Organizational/Technological, based on the experience of actors.

3. Formalization of experience

The first method that can be used is to store incidents or accidents as elementary items. This is

² in French: “Déraillement d’une ballastière lors du renouvellement des voies ballastées”.

³ in French: “Dysfonctionnement des sectionneurs d’isolement télécommandes”.

⁴ Past: referring to the use of past written or oral records; Present: incidents which have just taken place; Future: study of procedures.

⁵ From the French expression “Fil Conducteur”.

the approach chosen in most databases of accidents, because it is appropriate for statistical use and for epidemiology of accidents. An analysis of accidents and incidents in different databases shows that each type of accident corresponds to a series of events that differ by the context, the development or the consequences. A study of mental representation of actors in the development of risk management activities [Therrien 98] has permitted the identification of a generic structure for the development of the incident or accident, based on key events. A more detailed analysis shows that each of these meaningful events is associated with a *decision cycle*: identification of context and event, situation analysis and actions. These key events are more frequent than incidents, as each incident or accident is very often the succession of such key events.

They constitute, with the associated decision cycles, the basis of experience of actors that they use for the management of new incidents. Analogy is the main reasoning process used with experience: if the actor has already experienced the same event in a similar context, he uses the memory of his analysis and actions, weighted by the corresponding effects (decrease or increase of danger) to take a decision in the current situation.

Our objective is to develop a methodology to collect experience and to promote its sharing amongst actors. We therefore use a key event, the associated decision cycle and the evaluation of effects to propose the concept of a *particle of experience*. It represents the smallest element of experience that still holds onto its properties, that still renders information without distortion and hence preserves most of the complexity of the situation.

A particle of experience is composed of four main aspects:

- Situation: what was happening at that particular moment in time (event & context),
- Decision: after analyzing the situation, what decisions concerning actions are taken,
- Action: what is the action taken,
- Effect: what is the effect of the action taken until the next key event.

This structure is an adaptation of the model proposed by H.A. Simon [Simon 96] for the representation of the decision process.

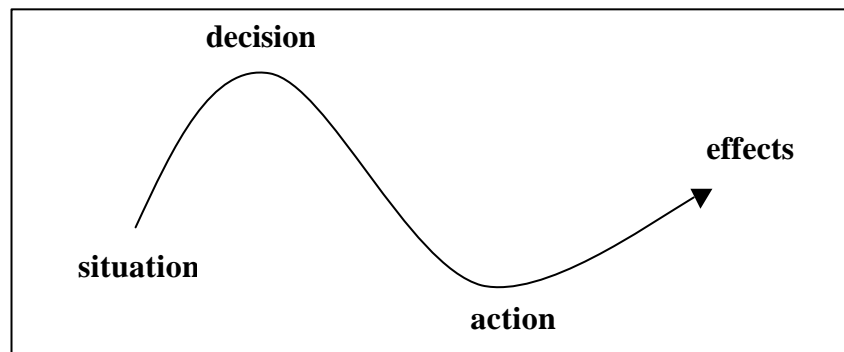


Fig. 2: Particle of Experience: key event and decision cycle

From this generic model, we have defined the set of data corresponding to the two types of incidents that we have studied. Once all the relevant information has been identified, a questionnaire is drawn up. The questionnaire serves to identify the key events in the unraveling of the incident. Each key event (particle of experience) is presented on one page, to serve as a basis of discussion.

The questionnaires are completely anonymous, having no mention of the names or functions of the actors that participate. The aim is to eliminate the fear of being reprimanded and to thus

create a free and open dialogue. As we will see later, this proved to be effective. The objective of this questionnaire is to make it possible to communicate on common ground with the actors of the system and to collect particles of experience.

The collection of particles of experience is achieved in three steps:

- One of the actors initiates the process by telling the story of the incident development, using the questionnaire. This story is formalized as a set of particles of experience.
- Each of the other actors is interviewed on the basis of the current set of particles of experience (one at a time), to collect his individual experience: did he experienced a similar key event? Does he have proposals to manage differently the same key event? This step represents an opportunity to collect positive experience from actors.
- When all actors have been interviewed, the full set of particles of experience that has been assembled constitutes the collective experience.

In order to facilitate the collection of particles of experience and the validation along the capitalization process, a graphical representation has been designed: the *string of key events* (SKE).

This SKE graph (Fig. 3) is used after individual interviews and, later, during collective interviews in order to confirm as closely as possible the accuracy and hence the reliability of the data. The string of key events is designed to guide the actors in their line of reasoning, in order for their answers to be complete and structured.

To collect the positive experience reflection, the actors are going through each key event and are asked what they would have done in a similar situation. Two kinds of possibilities were identified:

- Possible positive action: action that could have been applied in order to avoid/stop to the system breakdown. Had it been applied, this action may have made it possible to avoid the occurrence of the incident altogether. This is a possible action that the actors would want to encourage;
- Possible negative action: this relates to feasible action that could have been applied, but that would have led to another incident or that may have even aggravated the deterioration of the system. This is a possible action that actors would want to avoid.

Emphasis is placed not only on what actually happened, but also, and more importantly, on what could have happened. Each of the actors interviewed is able to provide his or her know-how freely and share this experience with other participants.

4. Validation

After each interview, the actors receive a copy of the questionnaire results in the form of a SKE Graph that traces:

- The exact events that took place during the period in which the system malfunctioned in the form of real particles of experience (i.e. tracing what really happened).
- The possible positive and negative events that could have occurred in a similar situation in the form of hypothetical particles of experience.

We call this step “mirror effect meeting”. It was applied to each actor, on a one-to-one basis (to an actor after his or her individual interview) and in a group meeting that was held after all the actors had partaken in the individual interviews.

This collective validation step allows for the coming together of actors in the same department, actors belonging to different departments, and those with different functions and responsibilities, who would not normally meet or be seen together. It allows for communication between the actors, elicitation of tacit knowledge and sharing of positive experience because each actor is encouraged to submit and review possible actions and solutions.

Individual and collective learning thus takes place and the actors acquire an increased sense of team spirit. This was confirmed by a number of actors after the collective “mirror effect meeting”: the actors felt motivated by this participative process and expressed the feeling that they had at long last been able to communicate their feelings, to interchange ideas, be heard and even to learn from the experience of others.

5. Modeling and support to the sharing of experience

The efficiency of the PER method is supported by three graphic representations: SKE graph (String of Key Events), Fault Tree graph and Cindynics hyperSpace. The benefits of graphic representations should in no way be underestimated. Indeed, they proved to be an invaluable comprehension aid, allowing the actors to better visualize the different aspects of the incidents.

The String of Key Events Graph (Fig. 3) is a visual support, tracing the actual and possible key events that played a part in the evolution of the system malfunction. Real and possible key events were identified thanks to the individual and collective experiences of the actors. In this way, the PER method was able to draw from experience and know-how in order to identify real and could-be key events:

- From the start (system deterioration);
- During the actual incident (system breakdown);
- Through to the end (normalization of the situation: system is back in its “normal” state).

The central line represents the development of the incident as the succession of key events and decision cycles that really occur. On the left side are represented the decision cycles that make the situation easier (decrease the danger or stop the incident), and on the right side, those that aggravate the situation (increase danger or create a new incident) are given.

The Fault Tree Method (Fig. 4) was incorporated in the PER method for various reasons: Firstly, because it is one of the most commonly used methods in the field of safety analysis. Secondly, and contrary to the SKE Graph that traces the unraveling in time of an incident, the Fault Tree Method represents, at a given moment in time, all the possible combinations of events that could lead to the system breaking down. It includes also *safety barriers* (devices, procedures or human actions aiming at reducing/stopping the propagation of the incident). The fault tree is created during the design of the system. At each occurrence of an incident or accident that was not identified, it must be updated to take into account the causes and determine the appropriate safety barriers. The PER method has proved to be efficient in supporting this update.

The Cindynics HyperSpace Method (Fig. 5) helps to grasp the level of danger inherent to the malfunctioning system at any given time: before, during and after the incident. This qualitative method can be seen as a framework to assess the global level of danger, which is represented by means of five *Cindynics dimensions*:

- Statistical dimension: information on the system⁶, past incidents, databases,
- Epistemic dimension: models, representations of the system,
- Objectives dimension: the direct finalities and objectives of the system,
- Rules dimension: organizational rules and procedures governing the system,
- Values dimension: the fundamental values of the system.

Before and after each particle of experience, each dimension of the HyperSpace is analyzed and elements affecting the danger of the situation are highlighted along the corresponding axis. Through this approach, it is possible to visualize the variations in the level of danger during this cycle.

6. Conclusions

In this application to rail incidents and accidents, the PER method has demonstrated a significant potential to improve the efficiency of incident and accident management by eliciting tacit knowledge. It allows for:

- Crisis Management. The actor draws on past experiences and is thus better equipped to deal with incidents, should these arise.
- Prevention. The sharing of experiences allows for the compilation of security barriers, better adapted measures to be applied in order to reduce the likelihood of occurrence and the gravity of the incidents.
- Establishing of a collective memory. The PER method promotes the capitalization and the perpetuating of the experience and know-how of the actors in a system.
- Strengthening communication channels between actors. Cooperation of actors in the same department as well as different departments are reinforced if not completely set up.

It is a straightforward and structured method, applicable to any type of incident of the past, present and future. Despite the benefits of the PER method, there were obstacles that arose, namely:

- Difficulty in assuring the participation of all the actors in the PER method. The usual ER process is viewed mainly as an inquisition, in that senior management often associates the experience reflection process to inspection, reprimands and sanctions.
- Different levels of involvement on the part of certain departments. Some departments especially those not directly involved in high-risk activities, felt less concerned with safety and security measures, and gave little importance to their participation in the PER method. Indeed, there is still much to be accomplished in the field of risk awareness.

There are plans in place for the PER method to be applied to several problems of risk management in industrial companies, within the framework of a research and development group involving both academic and industrial partners.

7. References

- [Kervern 94] Kervern G.Y., *Latest advances in Cindynics*, Economica, 1994
- [Simon 96] Simon H.A. "The sciences of the artificial", MIT Press, 1996
- [Therrien 98] Therrien M.C., "Pragmatisme et modèles systémiques pour la compréhension des processus de gestion des feux de forêt: apprentissage et expérience lors d'événements complexes", Ph.D. thesis, Ecole des Mines de Paris, November 1998

⁶ By System, we mean human, technical and organizational aspects.

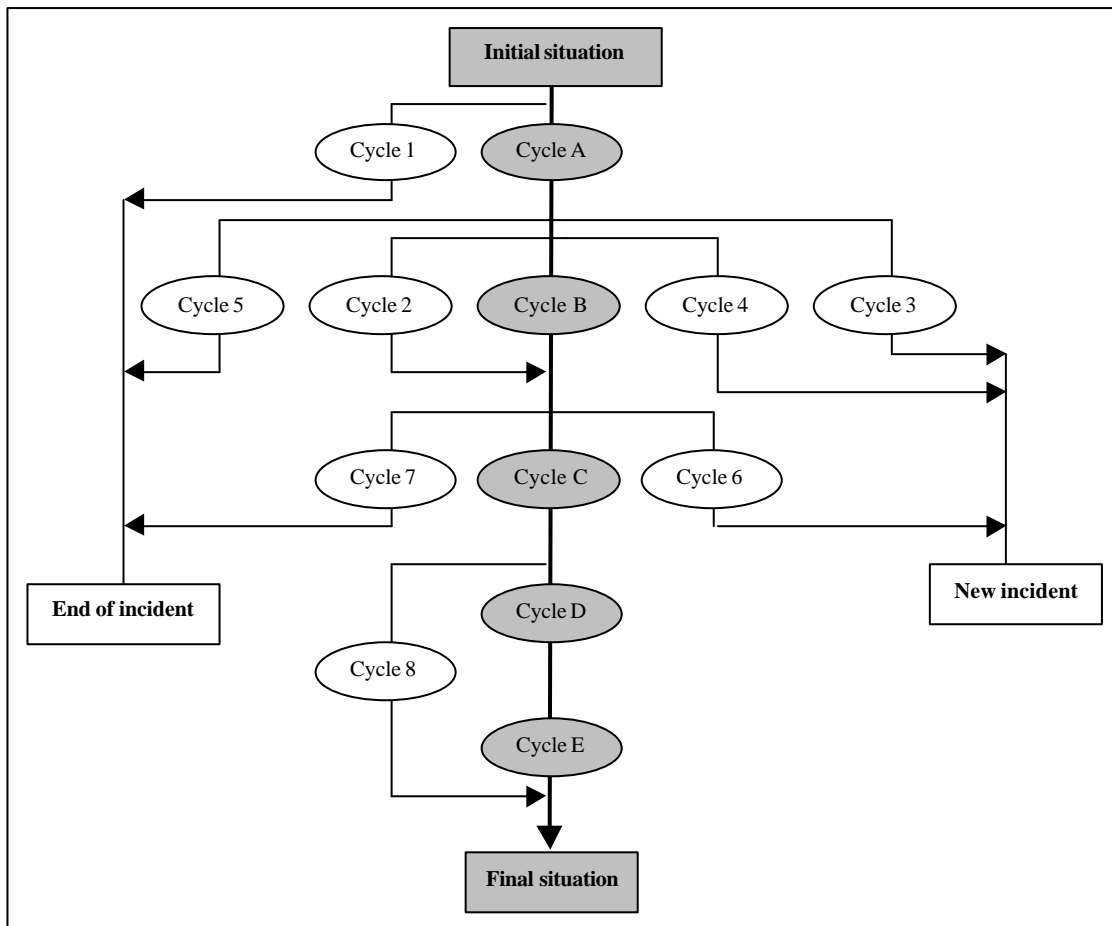


Fig. 3: the String of Key Events Graph

Real Cycles

Cycle A: Definition and cutting electrical current of sectors where works will take place,

Cycle B: Power restored to sectors that underwent works,

Cycle C: Incident on line due to the non-uniformity of electrical current,

Cycle D: First measures applied following electrical current incident,

Cycle E: Technical solution found for the electrical current incident.

Hypothetical Cycles

Positive

Cycle 1: Maintenance of infrastructures outside working hours,

Cycle 2: Power supply switched off individually for all the electrical devices in the sector where the works will take place,

Cycle 5: Electrical device testing included in the works planning schedule,

Cycle 7: Increase of responsibility actor awareness,

Cycle 8: Information Supplied to passengers.

Negative

Cycle 3: Power on, and works done on the electrical devices in the work site,

Cycle 4: Lack of respect for safety regulations before works on site,

Cycle 6: Lack of respect for rules and regulations that should be observed in a work site.

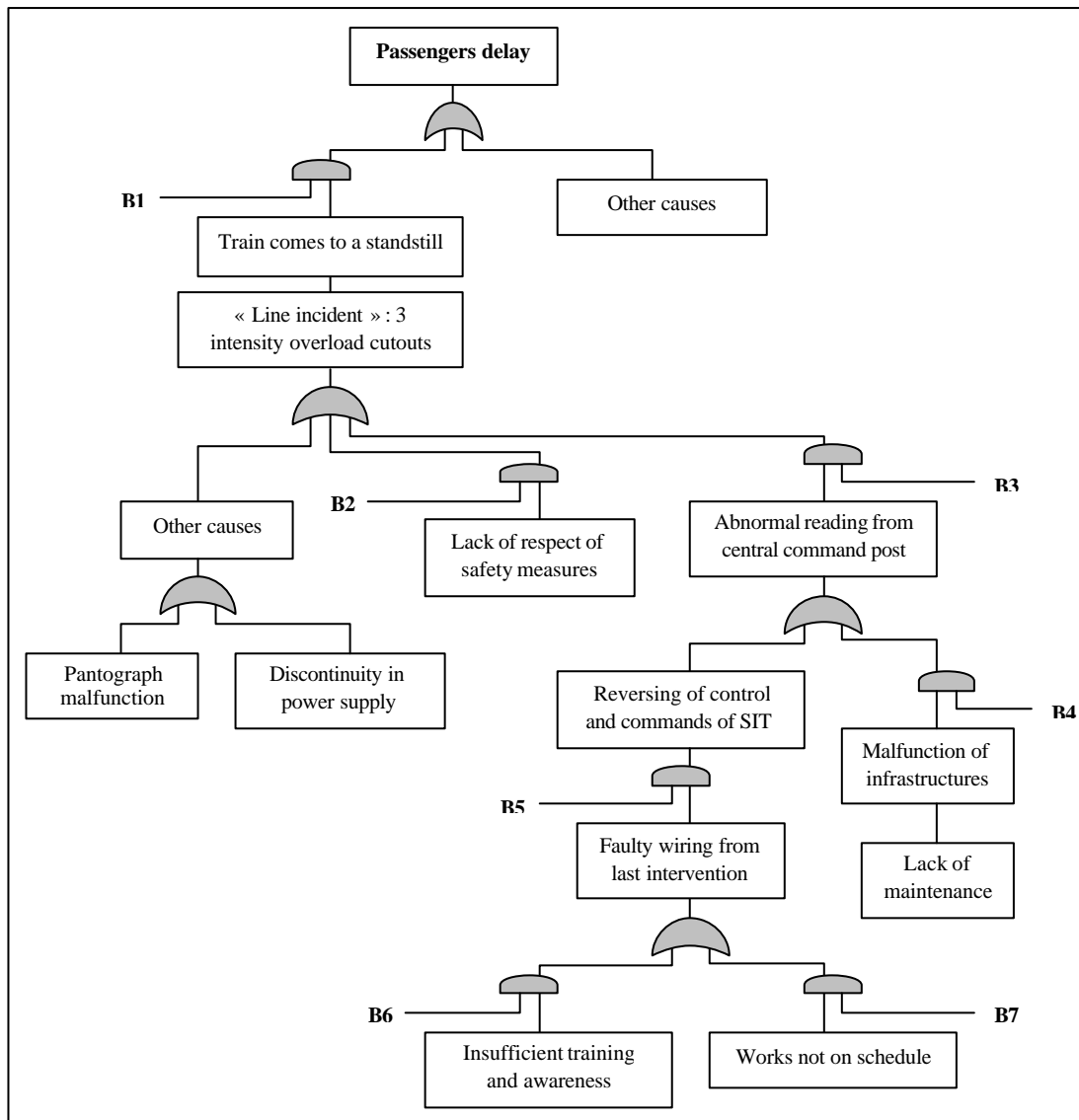


Fig. 4: the Fault tree Graph

Safety Barriers

B1: Passengers notified via loudspeakers and visual announcements

B2: Awareness of latent risks

B3: ESE department notified of malfunction; emergency plan activated

B4: Drawing up of control measures following works

B5: Annual electrical tests outside working hours

B6: Awareness of actors promoted about the consequences of unsatisfactory work

B7: Begin works on time and if not possible, do less during the night

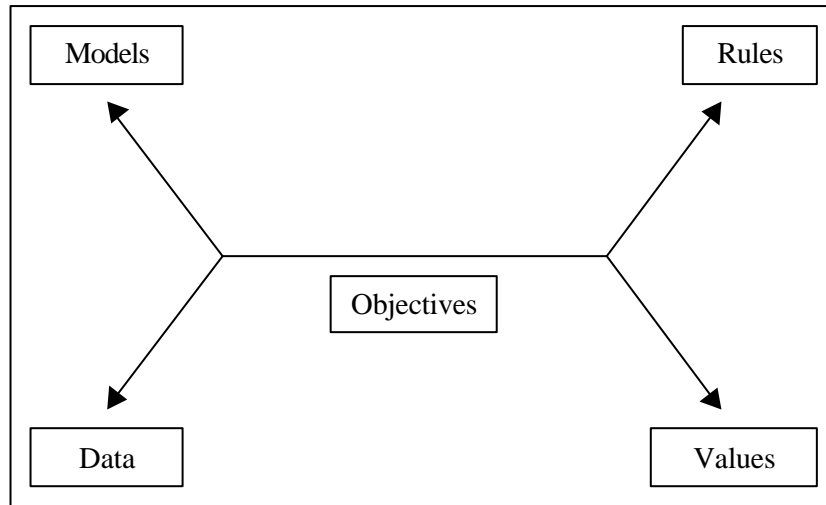


Figure 5: the Cindynics HyperSpace graph

Dimension of danger	Start of the incident (after Cycle A)	End of the incident (after Cycle C)
Data	Experience of actors in this type of incident	Experience of other electrical current incidents
Models	Methods applied during works and maintenance	Methods applied towards the end of working hours
Objectives	Carry out working orders 525 and 527	Handing over of fixed infrastructures in good working order and on time
Rules	Night works planning schedule and safety regulations	Measures to be implemented after electrical current incidents
Values	Stick to the works planning schedule	Observe safety regulations

Multiple Views on Consensual Categories: A Contribution for Corporate Memory Management

Sylvie DESPRES
UFR Mathématiques et Informatique
Université René Descartes
45, rue des Saints-Pères
75006 PARIS
tel : (33) 01 44 55 35 43
email : sd@math-info.univ-paris5.fr

1- Introduction

In this paper, the properties of a system providing access to a core concept in the knowledge management of an organization are defined. One of these properties is the ability of the system to deal with multiple points of view on a concept. This work refers to accidentology and the studied concept is the accident scenario.

This concept of accident scenario has been used since the late eighties in French research on road safety. Various research tasks on road accidents and on safety-study methods led to the progressive development of this concept. A scenario can be defined as a prototypical unfolding corresponding to a set of accidents which are similar from the point of view of the chain of facts and causal relations, throughout the different accident stages. This concept is a means for synthesizing knowledge extracted from accident case studies, based on in-depth investigation methods, or on detailed analyses of police reports. Applications of this concept are developed both in the area of traffic accident research and in the area of safety studies(diagnoses), thus preparing engineering measures or local safety policies.

In the field of safety studies, the design of scenarios proved its utility to reach a total view of the local phenomena and associated preventive possibilities by starting from case analyses. In the long run, researchers feel that case analysis in the framework of diagnoses tends more and more to concentrate on the recognition of the accident scenario, already precisely described and documented. Researchers underline the need for continuing the research on the concept of scenario when considered as accident knowledge. The robustness of these scenarios with respect to expertise, to road site differences, to mathematical methods and to tests of similarity should be clearly established before using them as an efficient help in accident analysis.

The research works conducted at CRIP5 since 1995 on the representation of accident scenario and system for automatic recognition of accident scenario, in co-operation with MA (Salon de Provence Mechanisms Accident Department) of INRETS (Institut National de recherche sur les Transports et leur Sécurité) on the one hand, and on the other hand with INRETS and INRIA (knowledge intranet server), are also related to this concept. In this framework, the concept of scenario is used for accident type recognition, for investigators' training, and for accidentology knowledge improvement.

Therefore, the scenarios constitute a core concept of capitalized knowledge at INRETS. They are created and used by its researchers. In the end, they will be used by the investigators of this organization and by people outside it. There exists many reports from different authors that describe the concept of scenario from their own points of view. They have been written during the last ten years. Moreover, each of these authors have elaborated scenario corpora which are available in paper form. Some recent papers have been written to take stock of the definition and the use of the scenario in accidentology.

The corporate memory of MA is already well developed. The aim of this work is to define the properties of a tool which will make possible it to use and to maintain this corporate memory in an interactive way, to support and even promote its evolution. A scenario editor integrated into a knowledge server or a recognition system seems a suitable tool.

To define the properties of this tool, we have determined the various representations useful for the reasoning of a recognition system and for the assistance to the reasoning of users, other than the designers of these categories. The scenarios established by the researchers and the contexts in which they were produced are also studied. A co-operative activity has been initiated to build a first corpus of consensus scenarios.

The studied co-operative activity consists in the design of consensus scenarios from a corpus of heterogeneous ones. A scenario from this corpus constitutes a prototypic unfolding of events, representative of a cluster of similar unfoldings. The corpus heterogeneous nature is due to the multiplicity of authors' representations, to the tasks they were intended for and to the authors' aims. Therefore, building consensus scenarios requires that authors agree on the relevance and the formulation of their initial scenarios, then on the form of the resulting scenarios.

The study of this collective design activity reveals the founding structures of these scenarios: description logic, description language, representation mode. It also provides representation modes which rely on these structures. In this study, the cooperation process summons up knowledge to clarify the essential elements that give structure to these categories. Cooperation is considered as a knowledge acquisition technique for a collective design task in a context of multi expertise.

Work about the psychology of expertise, about the role of multiple representations in training, about terminology and ontological engineering, about the studies of the links between experts' expertise and language, enabled us to determine the forms of representation and the suitable language of description to be retained. They also enabled us to justify these choices. We retained the description logic, the description language and the modes of presentation of the initial representations as fundamental elements to construct these consensual representations. We propose the use of a terminology of the field and the use of an ontology to give access to these various components of the scenarios.

In the following sections, after presenting the domain concerned, the properties of the scenario editor are described first, then the protocol used for the collective design of consensus scenarios. After, the elements on which the experts rely upon for starting their cooperation are introduced and the consensus scenarios are characterized. Finally, the conclusion shows that in the case of a previously built memory, techniques of knowledge acquisition are useful to elicit the implicit knowledge structures and therefore to give access to these structures.

2-Domain of accidentology

Understanding accidents, reconstitution and knowledge unfolding branches of activity in accidentology.

The road accident

A road accident is presented as a complex event. The accident is a process defined by a sequence of facts necessarily located in time (logic sequential course of events) and space (logic of the followed trajectories, generally rather linear) and by a causal logic which relates these facts between them (former events will determine posterior events). In a systemic approach, an accident is seen as a symptom of a malfunction of the HVI system (Human, Vehicle, Infrastructure), i.e. a failure of the adaptations of the system components [Fleury, 1998]. The HVI system helps accidentologists in their analysis and understanding of accidents.

The models

Models in the domain theory permit to represent and explain the unfolding of an accident. Two models are considered here because they structure the unfolding representation: the functional model and the sequential model.

Actually, the unfolding of an accident is seen, by the experts, as a chronological succession of situations by means of the sequential model : the driving situation before the accident; the accident situation; the emergency situation; the crash situation. The exit of each phase determines conditions for entry into the following phase. The functional model is a driver's model. It is based on psychological functions and the psychomotor activities implied in vehicle driving.

The accident scenario

An accident scenario is defined as a prototypical unfolding corresponding to a set of accidents which presents similarities from the point of view of chain of facts and causal relations, through different accident steps. It results from an expert analysis of the accident. As such, the various situations of the sequential model are clearly identified. The scenario is presented according to the sequential model. In each situation, the concerned mechanisms are described. The scenario is also an important synthesis tool in accidentology. It helps to draw more general conclusions in the case of accurate accident analysis and leads to global action on infrastructure. But it strongly depends on the experts' preferences, and on the tasks they have to achieve.

3- The scenario editor

Corporate memory in this paper is considered as *“an explicit, disembodied, persistent representation of knowledge and information in an organization, in order to facilitate its access and re-use by members of the organization, for their tasks”*[O'Leary and al.,1998 ; Van Heijst and al., 1996 ; Rabarijaona and al., 1999].

Activities underlying knowledge management can be decomposed in six steps : (1) Detection of needs in corporate memory, (2) Construction of the corporate memory, (3) Diffusion of the corporate memory, (4) Use of the corporate memory, (5) Evaluation of the corporate memory, (6) Maintenance and evolution of the corporate memory [Dieng et al., 1998].

Cooperative practices in multi disciplinary work at MA have led the researchers to build a corporate memory. This memory is made up of accidentology knowledge, of disparate know-how and heterogeneous points of view. The scenarios constitute a significant part of the

accumulated knowledge. The tool described here will make it possible to use and maintain this memory in an interactive way, to support and promote its evolution. Consequently, this contribution mainly addresses points 2, 3, 4 and 6.

A brief recall of the important points to consider in each of these three steps is presented below. In step (2), the adopted techniques depend on the sources from which the corporate memory can be built (in this study: experts, paper reports, technical documents and case studies) and the nature of the needed corporate memory according to the intended users. In step (3), the distribution to specially selected members of the organization [Van Heijst and al., 1996] can be passive or active: either the user can search for the needed information or knowledge distribution can be decided systematically and taken in charge by an adequate department of the organization. In step (4), the information search by the authorized members of the organization should be adapted to the users' needs, to their activities and to their work environment. In step (6), problems related to the addition of new knowledge, removal of obsolete knowledge and coherence underlying a cooperative extension of the corporate memory must be tackled.

The scenario editor design relies on these considerations. It enables selective accesses to a persistent scenario basis, through visualization, creation and modifying capabilities. These accesses are supported by a related ontology and conceptual models of the domain.

The scenario editor is designed with two tasks in mind: diagnosis and design.

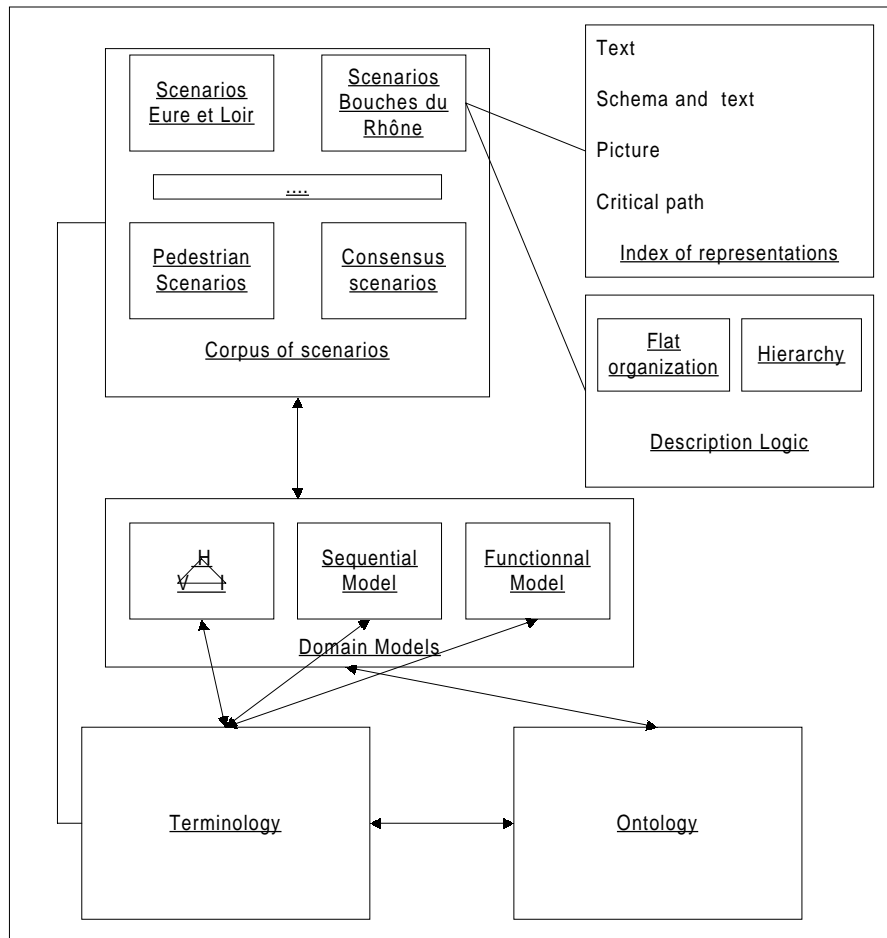
Diagnosis is a task carried out during accident analysis. The end-user is an investigator or a researcher and he only uses the consultation mode of the scenario editor. Consulting a scenario gives access to its multiple representations (a text, a schema, a picture, a critical path). It also makes the description logic of this scenario understandable. The description language used to describe scenarios is accessible by the links established between the domain terminology and the verbs used for scenario description. An ontology gives a view of term organization according to the type of action considered and the situation in which they occur with their arguments.

Design is a task which is generally accomplished by the experts at the time of activities such as the diagnosis (prospective study, safety project, urban project), the thematic study (prevention campaign, etc.). In this case, the end-user writes scenarios which are representative of a set of similar accidents (according to their unfolding). According to the current task, criteria of writing are adopted. The terminology helps as a memory assistance for the activity of writing. However, the terminology must be able to evolve in order to describe new incoming scenarios. Acting as a feedback for the expert activity, the tool will help to both enhance and precise the domain terminology.

The description logic is left to the experts' choice. The scenarios are organized according to a hierarchy or a flat organization and can be retrieved in both ways. In any case, once a new scenario is written, it is appropriate to check if it already appears in the basis or if it must be added. A measurement of similarity is currently under evaluation in order to determine the proximity between the considered scenarios.

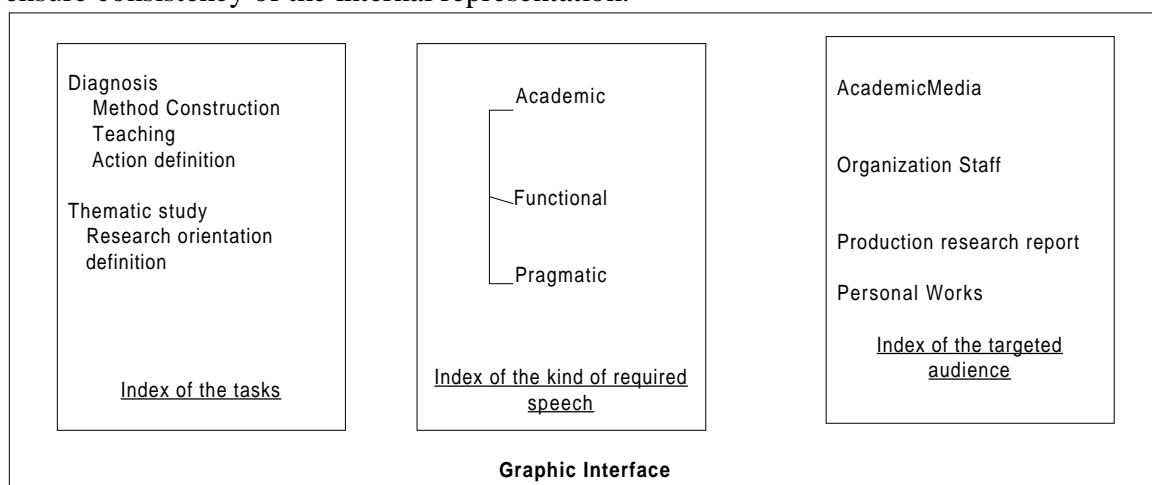
Another useful feature during the design task is the ability to clone an existing scenario and, by slightly modifying its representation, to create a new one, refining an existing category.

The basic architecture of the scenario editor is sketched in the following figure.



Scenarios are stored in a database combining, for each scenario, a set of fixed attributes and an object model. Classification attributes are used for selection purpose while the object model supports the different kinds of representations for a scenario. This model heavily relies on the strong concepts of the domain, namely the HVI model, the sequential model, the functional model.

Terminology proposition stored in the ontology database takes its arguments (H, V, I) from the object model. Therefore the link with the ontology is made via these arguments in order to ensure consistency of the internal representation.



For the user, at the graphic interface level, different views (four) may be alternatively displayed corresponding to different representations of the same scenario. Editing is done with the help of both the ontology database and a library of predefined objects (schematic or drawn).

From now on, it will be possible to choose the representations in adequacy with the tasks undertaken. The database should be able to evolve in the course of time and must reflect further improvements concerning the conceptual aspects of the scenarios. Thus, the researchers should be able to add or modify elements related both to scenarios and to their categorization. More beginners should be able to understand the underlying reasoning and to learn how to design such scenarios. The idea is also to couple this database with a scenario recognition system and to use it as a layer of additional knowledge associated with a knowledge server.

4-Method for defining the properties of the scenario editor

The method for defining the properties of such a tool consists in a consensus design protocol of scenario and in an analysis of the result.

The consensus scenarios are conceived to be used by the researchers, the investigators and outside users. It is thus mandatory to make the various elements of the components understandable: the description logic, the description language and the various representations. The idea is to have a scenario basis made up of the corpora of existing scenarios in a sufficiently universal form. This will allow the re-use the scenarios in different use and study context.

Universal structures involve a risk of standardization of the initial structures i.e. the drastic gumming of differences. This led to the dissatisfaction of the researchers who did not recover all the elements of their own structures. Moreover, such consensual structures are useless because of their non-representativeness. To avoid this risk and in agreement with the researchers, the decision to release the particularities of the various representations and of the different modes of reasoning was taken. This orientation made it possible to focus more particularly on these differences and to understand the description logic better.

The analysis of the collective construction of consensual categories is based on the use of a cooperation process in a multi expertise context. This presentation is supported by works relating to:

the psychological aspects of the expertise [Caverni, 1988,1991], [Chi, Feltovitch, Glaser, 1981], [Dubois, 1992], [Dubois, Bourguine, Resche-Rigon, 1992], [Falzon et Visser, 1988, 1989], [Gobbo, Chi, 1986],[Visser et Falzon, 1992];

the relations between expertise and language [Prince, 1991, 1992,1996];

the status of multiple representations [Alpay, Giboin, Dieng, 1999], [Bromme, Nückles, 1999], [Tabachneck-Schijf, Leonardo, Simon, 1994, 1997], [Hermina, Tabachneck-Schijf, Simon, 1999].

These various works enabled us to understand better the role played by the cognitive and socio-cognitive aspects. This analysis was carried out in order to point out the useful elements for the construction of a consensus scenario database.

4-1 Protocol for collective design of consensual categories

Before going any further it is necessary to define the terminology that is used in this paper. *Categorization* refers to the activity of category design and to its result i.e. the whole of the designed categories. *Recognition* refers to the activity which consists in judging if an object is one instance or belongs to a subcategory of an already existing category. Categorization is obviously not fixed once for all. The categories obtained at the conclusion of a categorization activity provide data on the organization and nature of these categories (categorical standards, information on typicality, and the subcategories' basic-level). The results relating to a categorization process can be reported according to the logic of description, the level of description and the extension of categorization. The logic of description refers to the categories obtained and the reasoning used to constitute them. The level of description underlines the links with the studied cases. The extension of the category relates to the coverage of the number of possible cases. It is also established that according to the types of expertise and the tasks carried out, categorizations differ. The experts involved in this study had the task to collectively rebuild a categorization, starting from already established categorizations, within the framework of diversified tasks (diagnosis, thematic study, information campaign, etc.). These new categories are those on which the end-users engaged in tasks of different nature (re-writing, designing, training) will be able to rely during their activities.

The activities of designing and writing were carried out by a group of researchers who were experts in scenario design. The protocol used was broken up into two sorts of working sessions: with an expert alone, and with a group of experts, building consensus scenario collectively. The individual sessions included an initial meeting and working sessions on corpora provided by other authors.

The purpose of the initial meeting was to make sure that researchers agreed on the concept of consensus scenario. Once the interest of the notion of consensus scenario was accepted, the cooperation for designing these scenarios was established efficiently between the experts. The first result was the definition of what, within this particular framework, a consensus scenario is. A consensus scenario is an abstract entity which corresponds to cases of accidents presenting an overall similarity in their unfolding and whose relevance, from the point of view of the phenomena, is the object of a consensus between experts.

The experts worked collectively on a corpus of generic scenarios which they had already designed or which had been built by beginners in accidentology supervised by the members of the group. The design of generic or consensus scenarios is founded on common models which form part of the corporate culture.

The individual working sessions enabled to clarify the authors' description logic. Each expert uses his/her own description logic, his/her own representation modes and his/her own description language of scenarios. The description language is generally shared by all the researchers, but their description logic differs in some ways and the representation modes vary according to researchers. These various elements play a part in the collective design but generally they are not explicitly quoted.

4.2 The designed consensual categories

The design of consensual categories (consensus scenario) was carried out during collective sessions. This design relies on studies and comparisons of scenarios from studied corpora.

The experts' comparison strategy is based on two essential points, the agreement on the *existence* and on the *writing* of the considered scenario.

The scenario existence relates to the validity of the regrouping made by an expert during a particular study. The discussion on the regrouping emphasizes the specificity of each expert's description. The differences between the experts are explained quite easily when the reasoning which led to the regrouping is available. The adjustment can raise more difficulties and even fail if the ways of reasoning are too far apart. This occurs primarily when regrouping is worked out by "novices". The scenarios which describe different points of view are preserved.

Then, when a consensus is acquired on the relevance of clustering, the experts may not share the way adopted to account for the phenomenon associated with clustering. In case of disagreement, the discussion deals with the elements structuring the prototype. The conflicts between experts are easily solved insofar as the elements characterizing a scenario are relatively standard (infrastructure, maneuver, human component, etc.).

Finally, once a consensus is obtained both on the existence and on the way of clustering, a *writing* remains to be adopted. Two processes underlie the writing: the reference to cases of accidents and the recourse to memory. The description level is dependent on the adopted process. The closer the expert is to cases, the more detailed the scenario is. According to Tulving [1976,1983], the descriptive elements of scenarios written by using a recollection process can be considered as indices of recovery.

The researchers implied in this study have different initial backgrounds, professional experiences and varied expertise. They all are engineers and experts in accidentology in varying degrees. Their position in the organization is different but they all have contributed to the development of generic scenarios. Their professional activity can be divided in three parts: fundamental research, finalized research and participation in the development policy of the organization.

According to the kind of speech they use, the researchers do not report the results in an identical way. For each researcher, speech includes the academic, functional and practical levels in varying degrees. Their activity is essentially interdisciplinary. They build models for the comprehension of complexity by using inductive and hypotetico-deductive reasoning. They know about the academic speech but are more familiar with the systemic one. Practical work is essential for them to understand and act.

Variability in the use of the various types of knowledge mentioned above is related to "cultural pre-built", to know-how and to individual preferences. The use of these various types varies according to the activities carried out (diagnosis, thematic studies), the tasks done (method construction, teaching, action definition, research orientation definitions, result dissemination) and the audience concerned (academic media, organization staff, production research report and books published for their organization, or works published in their own name). Whoever the researchers are, their productions include re-formulations and demonstrations, generally allotted to the academic experts, descriptions tuned to the objectives of the organization and pervaded with the corporate culture generally associated to the industrial experts and their skills.

Consequently the representations associated with the consensual scenarios must reflect these various productions. Differences are clarified in the writing of the generic scenarios. The form of speeches of the researchers reveal a language situated at the various levels previously quoted. These languages are relatively close.

The researchers use multiple representations to write the generic scenarios. The choices for these representations depend mainly on the tasks carried out, on the aspects they wish to highlight and on their preferences regarding result presentation. Moreover, certain aspects can be explained better in a representation than in another, which makes the various kinds of representations complementary. The different representations are used for knowledge assembling (coordination of the various representations during the reasoning) and represent different points of view (cognitive component of the expertise connected to the task).

Three great classes of scenarios emerge:

- The robust scenarios which are based on experts' agreement.
- The hypothetical scenarios which are clusters. The appearance of new cases reinforces or questions these clusters. They have a very fragile basis, they are likely to be questioned or to be transformed.
- The aberrant scenarios, which are so few that they can't be seen as clusters, but which are sufficiently representative to be preserved. They are generally associated with a single study. In this case, the accident instance delimits the scenario.

The extension of categorization depends entirely on the mode of category clustering.

4.3 Clarification of the elements required for accessing scenarios

The protocol analysis enabled us to clarify fundamental elements for designing consensus scenarios: the description logic, the description language and the multiple representations to write the scenarios.

These elements give access to the consensual categories. It is consequently advisable to define the means needed to translate these elements into a form readable and comprehensible for the end-users. An ontology was conceived in order to associate the tasks of the experts with their types of expertise in order to determine the useful representations. The description logic is also dependent on the chosen representation. We built a terminology from two corpora of accident scenarios (pedestrians, open-country) to make it possible to understand the terms employed by the researchers.

The description logic

The research about expert activities [Bourguin 1989, Gobbo and Chi, 1986.] has shown that expertise comes into the picture not only to modify the abstraction level of categories but also leads to a deeper reorganization of category structure in building hierarchies from different principles. The generic scenario organization relies clearly upon the basic representation level. For consensus scenarios, clustering is made according to the mode chosen by the end-user. A presentation of the different clustering organization (flat or hierarchical) ways is explicitly provided by means of examples for the various kinds of logic available.

The description language

The description language of the scenarios is common to all researchers. We thus built a terminology of the field which can allow the writing of new consensual scenarios. The

construction of this terminology fits in the framework of the theoretical and methodological proposals formulated during TIA conference [Bourigault and Al,1999]. We went from the text down to the terms. A set of terms denotes a concept which is here centered on accident analysis. The terms selected and described are verbs because they are predominant in the expert's description as stable action descriptor. The result of the terminology description can take several forms such as a tree structure for terms or a flat table of terms with their related definitions.

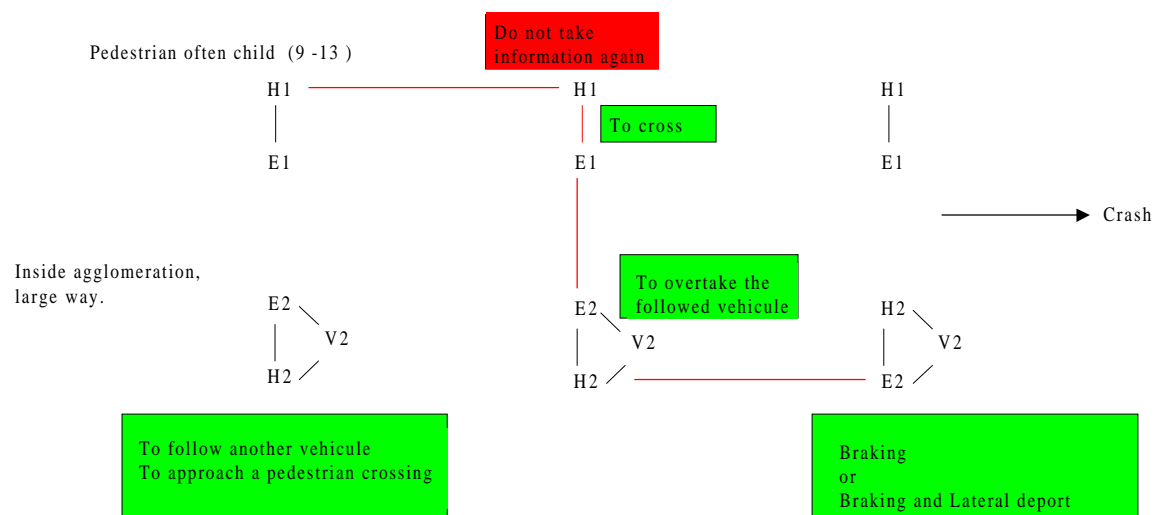
The corpus used to build the terminology consists of a set of scenarios concerned with pedestrians. The analysis of this corpus was carried out manually. We extracted the relevant verbs. We structured this terminology, respecting the two great phases required by the construction of a terminology [Daugan and *al.*, 1994]. The verbs were first of all extracted from a relevant corpus for the application. Then we added semantic links in order to obtain a more complex terminological network. Thus, the verbs are typified according to the situations in which they appear and are directly related to the sequential and functional models.

Multiple representations

In the prototype theory [Rosch, 1978; Dubois D., 1991; Kleiber G., 1990], the concepts are organized according to a basic level, an on-ordered level and an under-ordered level. The basic level is the operational level of the concept. It evolves according to the expertise because of the purposes required by the tasks to which the individuals are confronted.

In the case of the scenarios, the textual and diagrammatic representations are both at the basic level but serve different purposes. Thus, rather than choose a single representation for the consensus scenarios, we worked out a textual representation and a diagrammatic representation. The drawing representation is associated to these two representations, for it gives important information on the maneuver. The textual representations of the consensus scenarios are mitigated and consequently more synthetic since description moves away from the cases. A list of accident factors is preserved, the explanatory elements are gummed. The diagrammatic representations are little transformed. They can sometimes be enriched by the introduction of accident factors.

During this study another representation named "critical path" has been designed. It is based on the domain models and on the established terminology. It constitutes a means to clarify the implicit knowledge used for analyzing accidents.



The critical path contains the terms used both in scenarios and accidents. For that reason, it is suitable for different uses (diagnosis, analysis, creation). Although this representation is based on parts of existing models, it is new in the accidentology field.

5- Conclusion

For INRETS, the need to effectively develop a corporate memory which already exists (mostly in paper form) and which is already structured, excluded the choice of a simple document archiving even if the documents were easily accessible and shared. Indeed, in addition to a specific terminology, this memory encapsulates knowledge and modes of analysis established during many years of know-how. This is why a thorough analysis of these concepts and these modes was necessary and could not be made without resorting to the analysis of a consensual co-operative activity. The notion of critical path and the scenario editor constitute a core concept and a central tool to manage the corporate memory of the MA department dynamically.

The resulting tool, intended to make this memory evolve is thus more than a simple repository for these data. It must have properties which enable it to enrich this memory starting from the corpora of existing scenarios. The construction of a terminology of the field was essential to allow the end-users to grasp the meaning of the terms commonly used. Ontology explicits the relations binding knowledge inside the scenarios and is connected to the various elements useful for the tasks carried out by the experts.

6- References

- Alpay L., Giboin A., Dieng R. - Accidentology : An Example of Problem Solving by Multiple Agents with Multiple Representations, in *Learning with Multiple Representations*, Pergamon, pp.1998.
- Bourgine R – Contribution à une théorie de l'auto-modélisation. Thèse de l'Université d'Aix en Provence, 1989.
- Boshuizen H.P., van de Wiel W.J. – Using Multiple Representations in Medicine : How Students Struggle with them, , in *Learning with Multiple Representations*, Pergamon, pp.1998.
- Bromme R., Nückles M. - Perspective-Taking Between Medical Doctors and Nurses : A Study on Multiple Representations of Different Experts with Common Tasks, in *Learning with Multiple Representations*, Pergamon, pp.1998.
- Brenac T., Delcamp J., Pelat S., Teisseire G. - Scénarios types d'accidents de la circulation dans le département des Bouches du Rhone. Contribution à l'élaboration d'un diagnostic pour le Plan départemental d'actions de sécurité routière. Rapport MA 9611-2, 1996.
- Brenac T., Megerbi B. - Diagnostic de sécurité routière sur une ville : intérêt de l'analyse fine de procédures d'accidents tirés aléatoirement. *Recherche Transports Sécurité*, n°52, 1996.
- Caverni J.P. – Psychologie de l'expertise : élément d'introduction. *Psychologie française*, n°Spécial « Psychologie de l'expertise », n°33, pp.114-125, 1988.
- Caverni J.P. – La verbalisation comme source d'observables pour l'étude du fonctionnement cognitif. In J.P. Caverni, C. Bastien, P.Mendesohn, G. Tiberghien (eds), *Psychologie cognitive : modèles et méthodes*. Grenoble, PUG, pp.253-273, 1991.
- Chi M.T., Feltovitch, Glaser R. – Categorization and Representation of Physics Problems by Experts and Novices, *Cognitive Science*, 5, 121-152, 1981.
- Diday E., Lemaire J., Pouget J., Testu F. – *Eléments d'analyse de données*. Dunod, 1982.

- Dieng R., Giboin A., Amergé C., Corby O., Després S., Alpay L., Labidi S., Lapalut S. – Building a corporate memory for Traffic Accident Analysis, *AI Magazine*, 19(4), 80-100, Winter 1998.
- Dubois D. - Sémantique et cognition. Catégories, prototypes et typicalité. Editions du CNRS, Paris, 1991, 342p.
- Falzon P., Visser W. –Eliciting Expert Knowledge in a Design Activity : Some Methodological Issues, Rapport de recherche INRIA, n°906, 1988.
- Falzon P., Visser W. – Variations in expertise : Implications for the design of Assistance Systems. *Designing and Using Human-Computer Interfaces and Knowledge Based Systems* edited by G. Salvendy and M.J. Smith. Elsevier Publishers B.V., Amsterdam, 1989.
- Fleury D. - Sécurité et urbanisme. La prise en compte de la sécurité routière dans l'aménagement urbain. Presses des Ponts et Chaussées, 1998, 299p.
- Gobbo et Chi – “ How knowledge is structured and used by expert and novice children ”. *Cognitive Development*, 1, pp.221-237, 1986.
- Grize J.B. – Logique naturelle et langage. Conférence donnée à l'ISCC, Orsay , 1993.
- O'Leary – Enterprise Knowledge Management. *Computer*, 31(3),54-61pp., 1998.
- Pinson S. – Credit-Risk Assessment and Meta-Judgment. *Theory and Decision*, vol.27, n°1/2, pp.117-134, 1989.
- Prince V. – Expertise naturelle, expertise artificielle, vers quels paradigmes cognitifs ? *Intellectica*, n°12, pp.7-31, 1991.
- Prince V. – L'automatisation de l'expertise peut-elle rendre compte des automatismes des experts ? *Revue Internationale de Systémique*, n° spécial « Connaissances explicites vs connaissances implicites » n vol.6, n°1-2, pp.159-166, 1992.
- Prince V. – Vers une informatique cognitive dans les organisations. Le rôle central du langage. *Masson Collection Sciences Cognitives*, 190p. , 1996.
- Rabarijaona A., Dieng R., Corby O. – Building a XML-based Corporate Memory, 1999.
- Rastier F. – L'analyse linguistique des textes d'experts. *Génie logiciel*, n°23 ; pp. 16-23, 1991.
- Rosch – Principles of Categorization, in E. Rosch , B.B. Lloyd (eds), *Cognition and categorization*, Hillsdale, (N.J.) : L. Erlbaum, pp.27-47, 1978.
- (Tabachneck-)Schijf H.J.M., Simon H.A. – One Person, Multiple Representations : An Analysis of Simple, Realistic, Multiple Representation Learning Task, , in *Learning with Multiple Representations*, Pergamon, pp.1998
- Tabachneck-Schijf H.J.M., Simon H.A. – CaMeRa : A Computational model of Multiple Representations
- Tabachneck H.J.M., Leonardo A. M., Simon H.A. – How does an expert Use a Graph ? A Model of Visual and Verbal Inferencing in Economics, *Proceedings of the 16th Annual Conference of the Cognitive Society*, August 1994.
- Van Heijst G., Van der Spek R., Kruizinga E. – Organizing Corporate Memories. In Gaines B., Musen M. eds, *Proc. Of KAW'96*, Banff, Canada, 42.1-42.17, 1996.
- Visser W., Falzon P. – Catégorisation et type d'expertise : une étude empirique dans le domaine de la conception industrielle, in D. Dubois (ed.), *Intellectica : Connaissances et rationalités*, 15(3), 27-54, 1992.

Agent Architectures and Interaction Protocols for Corporate Memory Management Systems

Federico Bergenti, Agostino Poggi and Giovanni Rimassa¹

Abstract. This paper discusses what are the most suitable agent architectures and interaction protocols to implement a multi-agent system for the management of a corporate memory helping the users into two main tasks: the insertion of new documents and the retrieval of information. Moreover, the paper presents a software framework called JADE (Java Agent Development Environment), and shows how it can be used to develop the agent architectures and interaction protocols we need to realise such a multi-agent system. These guidelines should be taken into account during the realisation of such a system we are developing together with some other research centres inside the European Commission IST project “CoMMA - Corporate Memory Management through Agents”.

1 INTRODUCTION

A corporate memory is defined in [21] as an “explicit, disembodied, persistent representation of knowledge and information in an organisation, in order to facilitate its access and reuse by members of the organisation, for their tasks”.

In order to realise such kind of system, some recent works shown that organisations can take advantages of internet and intranet technologies and of agents. Internet and intranet technologies simplify the diffusion of knowledge. In fact, the Web can serve as a basis to distribute information in a uniform way independently of the way the information is stored [6,9,21]. Agents can help the user supporting retrieval of the relevant information from the corporate memory and adapting the interaction with the system to the user’s preferences [2].

In this paper, we discuss what are the most suitable agent architectures and interaction protocols to implement a multi-agent system for the management of a corporate memory we are developing together with some other research centres inside the European Community IST project “CoMMA - Corporate Memory Management through Agents” [5]. Moreover, the paper presents a software framework called JADE (Java Agent Development Environment), and shows how it can be used to develop the agent architectures and interaction protocols we need to realise such a multi-agent system.

2 AGENT TYPES FOR CORPORATE MEMORY MANAGEMENT

The corporate memory management multi-agent system we are developing should help the user in three main tasks: the XML annotation and insertion of new documents and the search of documents, and should autonomously push her/him information about new interesting documents.

To perform these tasks, the system needs five categories of agents:

- a set of Interface Agent (IAs), one for each user, operating as personal assistant guiding its user both in the search of documents in the corporate memory and in the insertion of new documents in the corporate memory;
- a set of User Profile Agents (UPAs) managing user profiles and driving information pushing;
- a set of Document Ontology Agent (DOAs) managing document ontologies used to annotate documents;
- a set of Search Agent (SAs) searching documents in the corporate memory;
- a set of Archivist Agents (AAs) storing documents in the corporate memory.

Other kinds of agents as, for example, Directory Facilitators, Mediators and Resources Managers might be present.

The XML annotation and insertion of new documents can be described as follows. At the beginning of the XML annotation the IA negotiates with the DOAs which of them will help it to guide the user in the annotation. During the annotation of a document, the IA changes information with the user’s UPA receiving information to prevent user’s actions and sending information about what the user really do. Finally, when the annotation is finished and the document is ready to be stored the IA negotiates with the AAs which of them will store the document. Note that each user has a fixed UPA because the IA negotiates its help after user’s registration.

The search of documents can be described as follows. At the IA negotiates with the DOAs which of them will help it to guide the user to build the query. During the generation of the query, the IA changes information with the user’s UPA receiving information to prevent user’s action and sending information about what the user really do. Finally, when the query is ready to be executed the IA negotiates with the SAs which of them will search the document. The SA agent tries to directly execute the user’s query, if the query has not results, the SA negotiates with the DOAs which of them will help it to refine the query. Moreover, the SA can use the partial results of the query execution to build new queries (e.g., if the query is “find all the

¹ Dipartimento di Ingegneria dell’Informazione, University of Parma, Parco Area delle Scienze, 181A, 43100, Parma, Italy, *email*: {bergenti, poggi, rimassa}@CE.UniPR.IT

documents whose author is Mayer and whose topic is agent systems” and the SA finds only a document about it where Mayer is one of the authors, then this SA refines the query, adding the other authors of the found document as possible alternative to Mayer, executes the new query adding its results to first found document). After that the SA gives the results to the IA that, helped by user’s UPA, orders the results and presents them to the user.

If the corporate memory is distributed, the IA negotiates with the SAs which of them will search the document, but the chosen SA builds a team of SAs (one for each memory node), searches in a node of the corporate memory delegating the search in the other nodes to the other team members. At the end, it collects their results and then sends the results to the IA. In this case, a SA might work independently from the other SAs involved in the search; however, they may co-operate to improve the search. They may co-operate in the generation of a common global query. For example, if all the team members start the search with the user query, but none of them succeeds in finding enough information, they can refine the query using both the DOAs help and the information of the partial results, and then they can propagate the new query to the other team members (e.g., if the query is “find all the documents whose author is Mayer and whose topic is agent systems” and only an SA finds a document about it where Mayer is one of the authors, then this SA refines the query adding the other authors of the found document as possible alternative to Mayer, executes the new query and informs the other team members about this new opportunity).

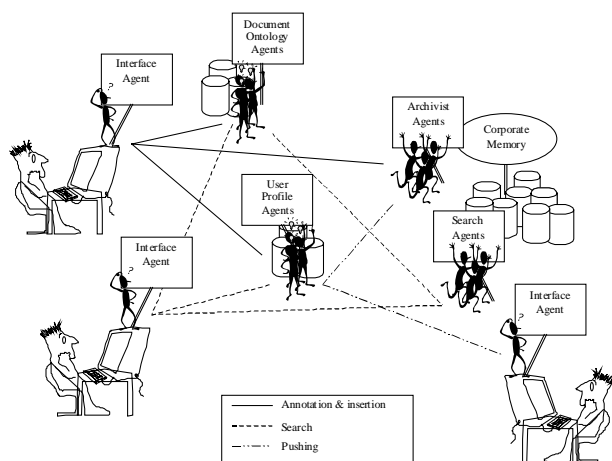


Figure 1. Graphical description of the interactions among the agents during the annotation and insertion, search and pushing tasks.

The pushing of documents is driven by:

- unsatisfied user’s requests (i.e., a user is not satisfied by the result of a query),
- a persistent user’s requests (i.e., a user subscribes herself to a specific interest), or
- a foreseeable interest (i.e., the UPA reasoning on its users profiles determinates that some of the new information might be of great interest to an its user).

This task can be described as follows. When an AA stores a new document in the corporate memory, it informs the UPAs. These agents check the unsatisfied and persistent users requests and, if the document do not satisfy such requests for some users,

the UPAs reason on their profile to check if this document might be of interest for their users. When the IA informs its UPA that the user enters the system, the UPA sends to the IA the information about the new documents that might be of interest to its user.

3 AGENT ARCHITECTURES FOR CORPORATE MEMORY MANAGEMENT

The first step to realise such a multi-agent system is to choose some appropriate agent architectures for the different types of agent the management of our corporate memory needs. This choice is done through an analysis of the agent types involved in the management of the corporate memory on the basis of the weak notion of agent [24], and through an analysis of the most known agent architecture models.

3.1 Agent types characterisation

The weak notion of agent characterises an agent through the following properties:

- autonomy, an agent can operate without the direct intervention of humans or other agents, and has a complete control over its action and internal state;
- reactivity, an agent can perceive its environment and respond in a timely fashion according to changes that occur within that environment;
- pro-activity, an agent does not simply act in response to its environment, but it is able to exhibit opportunistic, goal directed behaviour by taking the initiative where appropriate;
- social ability, an agent can interact with other agents and, possibly, with humans in order to performs its activities.

IAs, DOAs and AAs are autonomous, reactive and social, but they are not pro-active. In fact, all of them operate without the direct intervention of humans or other agents, react to the stimuli of the environment (i.e., the other agents and humans), interact with other agents and the IAs with humans too, but none of them show a goal oriented behaviour.

UPAs and SAs show all the properties of the weak notion of agent. In fact, a UPA starts pushing actions and a SA builds a team and modifies user queries with the goal to satisfy user interests. Moreover, a UPA shows additional properties that is the capability to learn user profiles: it builds user profiles from the information received by the IAs and uses such profiles to drive the IAs actions.

3.2 Agent architecture models

The weak notion of agent can be used to classify agent architecture models too. In fact, while some of these models are designed to offer complete reactive, pro-active and social agents, other models are oriented to offer task-oriented agents showing the balance among reactivity, pro-activity and social ability it needs to perform a particular kind of tasks.

We consider three main classes of agent architectures: reactive architectures, Belief-Desire-Intention architectures and layered architectures.

Reactive architectures do not have any internal, symbolic models of their environment, and they act using a

stimulus/response type of behaviour by responding to the present state of the environment in which they are embedded [4,10].

The behaviour of such architectures can be described through two functions. The first function, called *see*, maps input data from the environment to perceptions. The second function, called *action*, maps perceptions to actions. Figure 2 shows a schematic diagram describing the behaviour of such architecture.

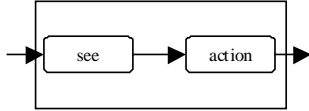


Figure 2. Schematic diagram of a reactive architecture.

The big advantage of reactive architectures is simplicity, however they have two main disadvantages: they cannot be pro-active because of their stimulus/response behaviour and they cannot learn from experience because of they have not an internal state.

Belief-Desire-Intention (BDI) architectures represent a system as a rational agent having certain mental attitudes of beliefs, desires, and intentions representing respectively the information, motivational and deliberative states of the system. These mental attitudes determine the system's behaviour and are critical for achieving adequate or optimal performance when deliberation is subject to resource bounds [3,18,22].

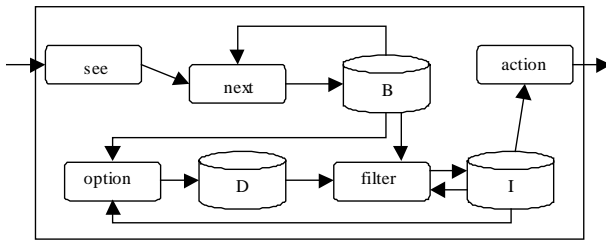


Figure 3. Schematic diagram of BDI architecture.

The behaviour of such architectures can be described through five functions. The first function, called *see*, maps input data from the environment to perceptions. The second function, called *next*, builds a set of new beliefs on the basis of current sets of perceptions and beliefs. The third function, called *options*, maps the new set of beliefs and the current set of intentions in a set of desires. The forth function, called *filter*, updates the set of intentions on the basis of the previous set of intentions and the current sets of beliefs and desires. The fifth function, called *action*, maps intentions to actions. Figure 3 shows a schematic diagram describing the behaviour of such an architecture.

BDI architectures are attractive for two main reasons. The first reason is that its behaviour is intuitive for humans. The second reason is that they present a clear functional decomposition indicating what sorts of subsystems might be required to build an agent. The main problem is to realise an efficient implementation.

Given the requirement that an agent should be capable of reactive and pro-active behaviour, an obvious decomposition involved the use of two separate subsystems to deal with these

two different behaviours. This idea leads naturally to a class of architectures, called layered architectures in which the various subsystems are arranged into a hierarchy of interacting layers.

Layered architectures are divided in two subclasses: horizontal and vertical layered architectures. In horizontally layered architectures, the software layers are each directly connected to the sensory input and action output [4]. In vertically layered architectures sensory input and action output are each dealt with by at most one layer each. In particular, vertically layered architectures are divided into one pass control and two pass control architectures. In one pass control architectures, control flows sequentially through each layer until the final layer generates action output [11]. In two pass control architectures, information flows up the architecture and then control flows back down. Figure 4 shows the schematic diagrams of three main categories of layered architectures [20].

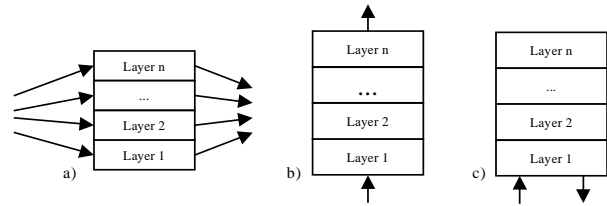


Figure 4. Schematic diagrams of three main categories of layered architectures: a) a horizontally layered architecture, b) an one pass control vertically layered architecture and c) a two pass control vertically layered architecture.

The great advantage of horizontally layered architectures is their conceptual simplicity to build an agent that exhibits some different types of behaviour. The problem is when these behaviours are in competition for some resources or to perform actions. In this case, it is necessary to introduce a mediator that implements a central control, but also introduces a bottleneck into the agent behaviour. Vertically layered architectures have not the problem of the central control and the way to pass control and information presents some similarities with the way that organisations work, but has a limited flexibility because control must pass between each different layer.

3.3 Agent architectures

At this point, we have the information to choose the most appropriate architecture for the different agent types we need in the multi-agent system for corporate memory management.

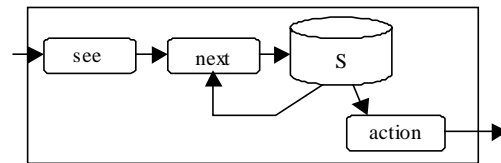


Figure 5. Schematic diagrams of a "reactive with state" layer of a horizontally layered architecture.

IAs, DOAs and AAs are not pro-active and the tasks they must perform can be easily defined eliminating possible conflicts on actions and resources. Therefore, a reactive and horizontally layered architecture, where each layer is dedicated to the execution of a particular task of the agent, seems to be the most

appropriate. However, it is not possible to use a purely reactive architecture because the execution of some tasks is influenced by history. Then, we can realise each layer extending the reactive architecture model to manage agent state: the behaviour of each layer, we call “reactive with state” layer, can be described through three functions. The first function, called *see*, maps input data from the environment to perceptions. The second function, called *next*, builds the new agent state on the basis of the current agent state and perceptions. Finally, the third called *action*, maps action state to actions. Figure 5 shows a schematic diagram describing the behaviour of a “reactive with state” layer.

UPAs and SAs are pro-active; therefore a BDI or a vertically layered architecture seem to be the most appropriate architectures. From a first analysis, it seems that the tasks these agents must perform can be easily defined limiting the possible conflicts on actions and resources. Therefore, the BDI architecture seems to have some advantages because with few conflicts among agent tasks the work mainly consists in the mapping of agent tasks in agent intentions. However, the use of a vertically layered architecture is also possible with the limited effort to decompose the tasks on the basis of the competence of the different layers.

Some of the tasks that UPAs and SAs must perform are reactive. One of BDI architecture limits is that tasks performing agent intentions are executed by monolithic processes. Therefore, a BDI architecture does not optimally support reactivity, in a sense that it does not provide mechanisms allowing, for example, to recognise emergency situation in time and to stop current task to start the task managing this emergency, but delegates the implementation of such mechanisms within the individual tasks realising agent intention. This is not a big problem, because the reactive tasks performed by UPAs and SAs do not manage emergency situations and so they can wait for the end of the running task without any problem for the agent.

4 AGENT INTERACTION PROTOCOLS FOR CORPORATE MEMORY MANAGEMENT

The second step to realise the multi-agent system for corporate memory management is to identify the different types of protocols that allow the interaction between the agents of the system. The simplest protocols come from concurrency and distributed systems research; the others mainly come from distributed artificial intelligence research. From the task description of section 2, we identified three protocols of the first class: request, query and subscription protocols, and two of the second class: contract net and partial global planning protocols.

Request and query protocols are usually the simplest and the most used protocols in an agent system and respectively allow an agent to request the execution of an action or to request some information to another agent. In this case, the protocol is based on two agents exchanging two messages: an agent sends a request/query message and the other agent replies a response message.

Subscription protocol allows an agent to receive automatically some kinds of information from another agent when they become available to this agent. In this case, the protocol is based on two agents and a sequence of messages: an

agent sends a subscription message, the other agent sends an *inform* message when it acquires new information related to the subscription and, finally, the first agent sends a message to remove its subscription.

In our multi-agent system the subscription protocol is used by UPAs that subscribe themselves to IAs and AAs respectively to receive information about user actions and to receive information about the new documents.

Contract net protocol allows an agent, called manager, to assign a task to one among a set of agents, called contractors, through a negotiation [23]. In this case, the following steps can describe the protocol:

- the manager announces a task to be performed to the potential contractors;
- the contractors respond either declining the offer or proposing a bid;
- the manager awards a contract to a contractor;
- the contract accepts the contract;
- the contract reports the results of the tasks.

In our multi-agent system the all the agents use the contract net protocol. In particular, IAs and SAs act as managers and UPAs, DOAs and AAs act as contractors.

Partial global planning protocol allows to build a global plan starting from the partial plans that a set of agents can independently generate [8]. In this case the protocol can be described by:

- an initial step, in which each agent sends a partial plan to the other agents, and
- an iterative step, in which each agent build a partial global plan and sends it to the other agents; the iteration ends when all the agents converge to the same partial global plan.

In our multi-agent system, if the corporate memory is distributed, the partial global planning protocol is used by SAs that build a global query from the partial queries generated by the different SAs.

5 AGENT ARCHITECTURES AND PROTOCOLS WITH JADE

The realisation of such a multi-agent system will be simplified through the use of an agent-oriented development environment.

5.2 JADE

JADE (Java Agent Development Environment) is a software framework to make easy the development of agent applications in compliance with the FIPA specifications [12] for interoperable intelligent multi-agent systems [1]. JADE is an Open Source project, and the complete system can be downloaded from JADE Home Page [15]. The goal of JADE is to simplify development while ensuring standard compliance through a comprehensive set of system services and agents. To achieve such a goal, JADE offers the following list of features to the agent programmer:

- FIPA-compliant Agent Platform, which includes the AMS (Agent Management System), the default DF (Directory Facilitator), and the ACC (Agent Communication Channel). All these three agents are automatically activated at the agent platform start-up.

- Distributed agent platform. The agent platform can be split on several hosts. Only one Java application, and therefore only one Java Virtual Machine, is executed on each host. Agents are implemented as one Java thread and Java events are used for effective and lightweight communication between agents on the same host. Parallel tasks can be still executed by one agent, and JADE schedules these tasks in a co-operative way.
- A number of FIPA-compliant additional DFs (Directory Facilitator) can be started at run time in order to build multi-domain environments, where a domain is a logical set of agents, whose services are advertised through a common facilitator.
- Java API to send/receive messages to/from other agents; ACL messages are represented as ordinary Java objects.
- FIPA97-compliant IIOP protocol to connect different agent platforms.
- Lightweight transport of ACL messages inside the same agent platform, as messages are transferred encoded as Java objects, rather than strings, in order to avoid marshalling and unmarshalling procedures.
- Library of FIPA interaction protocols ready to be used.
- Graphical user interface to manage several agents and agent platforms from the same agent. The activity of each platform can be monitored and logged. All life cycle operations on agents (creating a new agent, suspending or terminating an existing agent, etc.) can be performed through this administrative GUI.

The JADE system can be described from two different points of view. On the one hand, JADE is a runtime system for FIPA-compliant Multi Agent Systems, supporting application agents whenever they need to exploit some feature covered by the FIPA standard specification (message passing, agent life-cycle management, etc.). On the other hand, JADE is a Java framework for developing FIPA-compliant agent applications, making FIPA standard assets available to the programmer through object oriented abstractions. In this paper, we centre the discussion on the second aspect. A detailed discussion about the first aspect can be found in [1].

5.2 JADE agent model

JADE can be seen as a software framework for agent-based applications, with application programmers concentrating themselves on writing agents and on specifying each agent role within the agent society. Since complex tasks in multi agent systems are usually tackled using collaboration among many agents, a single agent is typically a strongly cohesive piece of software. On the other hand, asynchronous message passing with pull consumer messaging model grants a very loose coupling between different agents. Furthermore, no implementation inheritance (and no code reuse) is considered when dealing with software agents. In this way, software agents bear strong resemblance with actors, and indeed JADE execution model has its roots in actor languages.

The abstraction used to model agent tasks is the *Behaviour*: each JADE agent holds a collection of behaviours which are scheduled and executed to carry on agent duties. Behaviours represent logical threads of a software agent implementation. According to *Active Object* design pattern [19], every JADE agent runs in its own Java thread, thereby satisfying autonomy

property; instead, in order to keep small the number of threads required to run an agent platform, all agent behaviours are executed co-operatively within a single Java thread. So, JADE uses a *thread-per-agent* execution model with co-operative intra-agent scheduling.

Using a single Java thread to handle multiple agent behaviours needs some sort of scheduling policy. JADE relies on a “*co-operative scheduling on top of the stack*”, in which all agent behaviours are run from a single stack frame without context saving (*on top of the stack*) and a behaviour continues to run until it returns from its main function and cannot be pre-empted by other behaviours (*co-operative scheduling*); of course ordinary pre-emption is still active between different agent threads and among JADE system threads: co-operative scheduling is strictly an intra-agent policy.

Using co-operative behaviours to model multiple agent conversation is a lightweight approach to concurrency, trying to achieve low latency by working entirely in user space. Similar techniques are customary in modern high performance network protocols and messaging libraries [7]. Besides, JADE model is an effort to provide fine-grained parallelism on coarser grained hardware. A likewise, stack based execution model is followed by Illinois Concert runtime system [17]; willing to provide a runtime environment for parallel object oriented languages, Concert can execute concurrent method calls optimistically on the stack, reverting to real thread spawning only when the method is about to block, saving the context for the current call only when forced to.

However, Concert system relies on compiler support to select concurrency strategy (really parallel or lazily stack based) and to save method context when needed. On the other hand, JADE is not a new language but a Java development framework: saving the context when an agent behaviour is blocking would put an heavy burden on JADE users, because they should write state saving code themselves, since no compiler would be going to write it for them.

Choosing not to save behaviour execution context means that agent behaviours start from the beginning every time they are scheduled for execution and local variables are reset every time. So, behaviour specific state that must be retained across multiple executions is to be stored into behaviour instance variables. A general rule for transforming an ordinary Java method into a JADE behaviour is the following:

1. Turn the method body into an object whose class inherits from *Behaviour*.
2. Turn method local variables into behaviour instance variables.
3. Add the behaviour object to agent behaviour list during agent startup.

The above guidelines apply the *reification technique* [16] to agent methods, according to *Command* design pattern [13]; an agent behaviour object reifies both a method and a separate thread executing it. While reification yields enhanced flexibility in task scheduling and execution, it often results in verbose program text (a problem commonly encountered in reflective code, too). A whole new class must be written and instantiated for each agent behaviour, and this can easily lead to programs hard to understand and maintain. JADE application programmers can compensate for this shortcoming using Java *Anonymous Inner Classes*; this language feature is similar to Smalltalk code blocks and makes the amount of code necessary for defining an

agent behaviour only slightly higher than for writing a single Java method.

Sometimes real intra-agent multithreading may seem unavoidable: for example, an agent acting as a wrapper onto a DBMS could issue multiple queries in parallel, or an agent might want to block on a stream or socket while still being able to engage in ordinary conversations. Really, this kind of problems occur only when an agent must interact with some non-agent software; FIPA acknowledges that these are boundary conditions for the execution model and deals with them in a separate part of the standard (namely, FIPA part 1 is about agent management and FIPA part 3 deals with external, non-agent software).

JADE *thread-per-agent* execution model can deal alone with all the most common situations involving only agents: this is because every JADE agent owns a single message queue from which all ACL messages are to be retrieved. Having multiple threads but a single mailbox would bring no benefit in message dispatching, since all the threads would still have to synchronise on the shared mailbox. On the other hand, when writing agent wrappers for non-agent software, there can be many interesting events from the environment beyond ACL message arrivals. Therefore, application developers are free to choose whatever concurrency model they feel is needed for their particular wrapper agent; ordinary Java threading is still possible from within an agent behaviour, as long as appropriate synchronisation is used.

5.3 Using behaviours to build complex agents

The developer who wants to implement an agent must extend the *Agent* class and implement the agent-specific tasks by writing one or more *Behaviour* subclasses, instantiate them and add the behaviour objects to the agent. User defined agents inherit from the *Agent* class the basic capability of registering and deregistering with their platform and a basic set of methods that can be called to implement the custom behaviour of the agent (e.g. send and receive ACL messages, use standard interaction protocols, register with several domains). Moreover, user agents inherit from their *Agent* superclass two methods: *addBehaviour(Behaviour)* and *removeBehaviour(Behaviour)*, which allow managing the behaviour list of the agent.

JADE contains ready made behaviours for the most common tasks in agent programming, such as sending and receiving messages and structuring complex tasks as aggregations of simpler ones. For example, JADE offers a so-called *JessBehaviour* that allows full integration with JESS [14]. JESS is a scripting environment for rule programming written in Java offering an engine using the Rete algorithm to process rules. Therefore, while JADE provides the shell of the agent and guarantees the FIPA compliance, JESS allows using rule-oriented programming to define agent behaviours and exploiting its engine to execute them.

Behaviour is an abstract class that provides the skeleton of the elementary task to be performed. It exposes three methods: the *action()* method, representing the "true" task to be accomplished by the specific behaviour classes; the *done()* method, used by the agent scheduler, that must return *true* when the behaviour has finished and can be removed from the queue, *false* when the behaviour has not yet finished and the *action()* method must be executed again; the *reset()* method, used to restart a behaviour from the beginning.

Because of the non pre-emptive multitasking model chosen for agent behaviours, agent programmers must avoid to use endless loops and even to perform long operations within *action()* methods. This is because when some behaviour's *action()* is running no other behaviour can proceed until the end of the method (of course this is true only with respect to behaviours of the same agent: behaviours of other agents run in different Java threads and can still proceed independently).

Moreover, since no stack context is saved, every time *action()* method is run from the beginning: there is no way to interrupt a behaviour in the middle of its *action()*, yield the CPU to other behaviours and then start the original behaviour back from where it left.

For example, suppose a particular operation *op()* is too long to be run in a single step and is therefore broken in three sub-operations, named *op1()*, *op2()* and *op3()*. To achieve desired functionality one must call *op1()* the first time the behaviour is run, *op2()* the second time and *op3()* the third time, after which the behaviour must be marked as terminated. The code will look like the following:

```
public class my3StepBehaviour {
    private int state = 1;
    private boolean finished = false;
    public void action() {
        switch (state) {
            case 1: { op1(); state++; break; }
            case 2: { op2(); state++; break; }
            case 3: { op3(); state = 1;
                    finished = true; break; }
        }
    }
    public boolean done() {
        return finished;
    }
}
```

Following this idiom, agent behaviours can be described as finite state machines, keeping their whole state in their instance variables.

When dealing with complex agent behaviours (as agent interaction protocols) using explicit state variables can be cumbersome; so JADE follows a compositional approach to allow application developers to build their own behaviours out of the simpler ones directly provided by the framework. Applying the *Composite* design pattern [13], *ComplexBehaviour* class is itself a *Behaviour*, but can have an arbitrary number of sub-behaviours or *children*; this class also defines the two methods *addSubBehaviour(Behaviour)* and *removeSubBehaviour(Behaviour)*, allowing to define recursive aggregations of several sub-behaviours. Since *ComplexBehaviour* extends *Behaviour*, the agent writer has the possibility to implement a structured tree composed of behaviours of different kinds (including *ComplexBehaviours* themselves). The agent scheduler only consider the top-most tasks for its scheduling policy: during each "time slice" (which, in practice, corresponds to one execution of the *action()* method) assigned to an agent task only a single subtask is executed. Each time a top-most task returns, the agent scheduler assigns the control to the next task in the ready queue.

Besides *ComplexBehaviour*, JADE framework defines some other direct subclasses of *Behaviour*: *SimpleBehaviour* class can

be used by agent developer to implement atomic steps of the agent work. A behaviour implemented by a subclass of *SimpleBehaviour* is executed by JADE scheduler in a single time frame. Two more subclasses carry out specific actions: *SenderBehaviour* and *ReceiverBehaviour*. Notice that neither of these classes is abstract, so they can be directly instantiated passing appropriate parameters to their constructors. *SenderBehaviour* allows sending a message, while *ReceiverBehaviour* allows receiving a message which can be matched against a pattern; the behaviour blocks itself (without stopping all other agent activities) if no suitable messages are present in the queue.

The two classes *ComplexBehaviour* and *SimpleBehaviour* leave to their subclasses the choice of their termination condition and, for complex behaviours, the actual children scheduling policy. For atomic behaviours, two subclasses are provided: *OneShotBehaviour* is an abstract class that models behaviours that must be executed only once. *CyclicBehaviour* is an abstract class that models behaviours that never end and must be executed forever.

ComplexBehaviour class fulfils the responsibility of organising its children, but it defers children scheduling policy to subclasses; JADE provides ready made subclasses with common policies, but application programmers are free to implement their own.

SequentialBehaviour is a *ComplexBehaviour* that executes its sub-behaviours sequentially, it blocks when its current child is blocked and it terminates when all its sub-behaviours are done.

NonDeterministicBehaviour is a *ComplexBehaviour* that executes its children behaviours non-deterministically, it blocks when all its children are blocked and it terminates when a certain condition on its sub-behaviours is met. The following conditions have been implemented: ending when all its sub-behaviours are done, when anyone among its sub-behaviours terminates or when at least N sub-behaviours have finished.

JADE recursive aggregation of behaviour objects resembles the technique used for graphical user interfaces, where every interface widget can be a leaf of a tree whose intermediate nodes are special container widgets, with both rendering and children management features. An important distinction, however, exists: JADE behaviours are reifications of execution tasks, so task scheduling and suspension are to be considered, too.

Thinking in terms of software patterns [13], if *Composite* is the main structural pattern used for JADE behaviours, on the behavioural side we have *Chain of Responsibility*: agent scheduling directly affects only top-level nodes of the behaviour aggregation tree, but every composite behaviour is responsible for its children's scheduling within its time frame. Likewise, when a behaviour object is blocked or restarted a notification mechanism built around a bi-directional *Chain of Responsibility* scheme provides all necessary event propagation.

5.4 Interaction Protocols

Interaction protocols are used to design agents interaction providing a sequence of acceptable messages and a semantic for those messages. FIPA defines a set of general-purpose interaction protocols that provide a well-known interaction means to promote the social and cooperative nature of agents.

While each FIPA ACL message kind is given a formal semantics based on the speech-act theory, this is still not enough

to satisfy the need for sociality of agent systems. This is because a typical agent interaction encompasses more than a single message, so more comprehensive abstractions are needed. FIPA specifications provide this abstractions as a collection of interaction protocols. When an agent engages in a conversation according to some interaction protocol, it can play two roles: the *initiator* role is the one who creates the conversation, by sending the first ACL message belonging to the new conversation; the *responder* role is the one who receives the first ACL message.

5.5 FIPA standard interaction protocols

The *fipa-request* protocol allows an agent to request another agent to perform some action; this is similar to ordinary request/response protocols used in client/server systems, but with a significant difference. Since software agents are autonomous entities, an agent can refuse to perform the requested action even if able to do so. So, while a client/server call either succeeds or fails raising an exception, a *fipa-request* interaction can succeed, can fail for a lack of the receiver-agent capabilities, but can also fail for the unwillingness of the receiver to perform the task at hand.

The *fipa-query* protocol is to be used to request an agent to provide some information to the protocol initiator. Since queries to an agent knowledge are not supposed to interact with external software systems, the *fipa-query* protocol shares the outcome set with the *fipa-request* protocol, but has a flat structure.

The *fipa-request-when* protocol is a variation of the *fipa-request* protocol: the initiator agent requests the responder agent to perform an action in the future, and after an initiator-supplied precondition is true. At that time the responder agent will try executing the action and will notify the initiator using an inform message if the attempt will have succeeded or using a failure message otherwise.

The *fipa-contract-net* protocol is the first of the higher level FIPA interaction protocols. An agent, called *manager*, can initiate a *fipa-contract-net* protocol sending a call for proposal, or *cfp*, message to a set of other agents; some agents will answer *not-understood* or *refuse* and will be discarded, but some others will hopefully send back a *propose* message, declaring their will to perform the requested action under a constraint set, such as price, time, or some other kind of constraint. The manager agent can then evaluate all the proposals and select one among them according to some criterion.

The *fipa-iterated-contract-net* protocol allows the manager to incrementally refine its call for proposal until a suitable contract is made. The only difference with respect to what previously described is that a manager can refuse all proposals, with *reject-proposal* messages, and issue a revised *cfp* using the same conversation identifier.

fipa-auction-english protocol resembles an auction for selling goods, led according to English style, using a low initial price and rising it gradually until no buyer declares its intention to pay. At that point, the last buyer can, and must, acquire the goods for sale paying the amount he or she accepted before.

The *fipa-auction-dutch* protocol realises the Dutch style for auctions, used for flowers market and working basically the opposite as the English auction: the auctioneer starts with a price much higher than the real market value of the goods he or she is trying to sell, then lowers the price gradually until one of the buyers accepts the suggested price and buys the goods.

5.6 Implementing interaction protocols with JADE

JADE implements the communication policies between agents by means of protocols defining the set of possible conversations between the agents, in particular, for every protocol that JADE supports a couple of classes is provided: one for agents playing the initiator role and another for responder agents. For each designed protocol, a semantic is also provided in order to let third-parties agents interact with the implemented ones. Besides providing FIPA interaction protocols, JADE allows the definition of new interaction protocols. However, the user is suggested to realise interaction protocols as a composition of FIPA protocols in order to ease their implementation with JADE and to support inter-operability with other FIPA-compliant systems.

To show how JADE supports FIPA interaction protocols, the *FipaRequestInitiatorBehaviour* class will be used as an example; this will also give a practical application of JADE concurrency model. This class must be used by agents willing to start new *fipa-request* conversations, asking some other agent to do something for them.

Looking at FIPA97 specification, one finds a graphical representation of the protocol, reported in Figure 6. The white boxes in the figure represent communicative acts that must be issued by the initiator agent, whereas the grey ones represent communicative acts issued by the receiver agent. So, from the initiator point of view, ACL messages corresponding to white boxes must be sent and the ones corresponding to grey boxes must be received. From the figure can also be seen that, after sending a *request* message, the initiator can expect either a *not-understood*, a *refuse* or an *agree*; in the last case, another message will arrive, i.e. either a *failure* (some problem occurred), or an *inform* (in two different flavours for actions without result and with one).

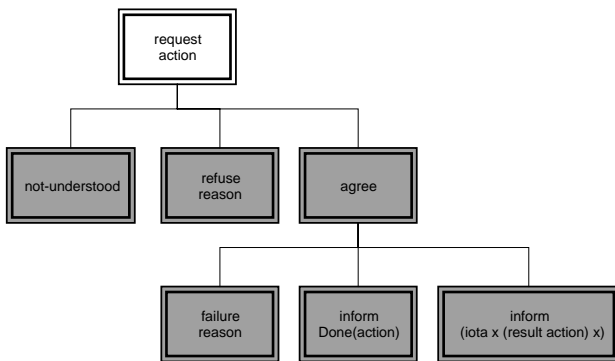


Figure 6. FIPA Request Interaction Protocol

Looking at the documentation of *FipaRequestInitiatorBehaviour* class, one finds a constructor accepting as parameter the ACL *request* message that will start the conversation; various abstract methods are provided to handle the various kinds of messages that can arrive during the protocol. Application programmers simply write their subclasses, implementing the abstract methods, named *handleAgree()*, *handleRefuse()* and so on. This approach relies on classic OO techniques such as abstract classes and tries to present a familiar programming style to application developers,

but the inner workings of *FipaRequestInitiatorBehaviour* are a bit different.

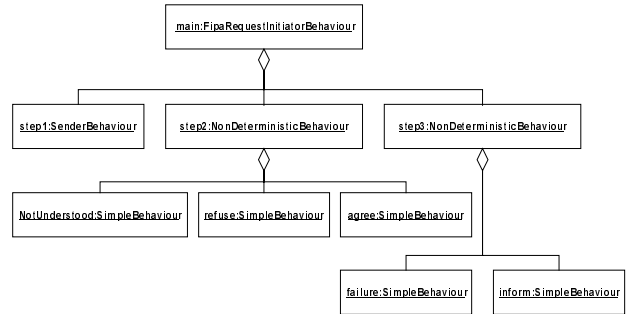


Figure 7. Internal structure of a JADE *FipaRequestInitiatorBehaviour* behaviour object.

This class relies on several private inner classes, which build an interaction protocol engine based on delegation, fully exploiting JADE intra-agent concurrency model. Looking again at Figure 5, the following algorithm can be written:

1. Send the request message.
2. Receive the first reply and handle it.
3. If the first reply was an *agree*, then receive the second reply and handle it.

When a complex behaviour is made by a list of steps, JADE library offers *SequentialBehaviour* class, which *FipaRequestInitiatorBehaviour* will be a subclass of. A *SenderBehaviour* can easily be used to carry out the first step, while the second one is more interesting.

The second step is a disjunction of three possibilities, seen as branches in the graph of Figure 6; it completes when any one among the three events (receiving a *not-understood*, receiving a *refuse* or receiving an *agree*) occurs. This observation allows to exploit the *NonDeterministicBehaviour* class to express the three possible branches as concurrent activities; the non deterministic behaviour object will be constructed so that it terminates as soon as one of its children receives something.

A similar approach is followed for the third step, but the second *NonDeterministicBehaviour* object will be added to the main *FipaRequestInitiatorBehaviour* by the handler of the *agree* message. All the atomic behaviours are implemented as *SimpleBehaviour* subclasses, which first receive their specific kind of message (using JADE pattern matching capabilities) and then invoke the handler method (e.g. *handleNotUnderstood()*). The overall object structure is drawn in Figure 7, showing also how similar the behaviour structure is to the original protocol structure in Figure 6.

5.7 Using JADE for building the multi-agent system for corporate memory management

From the previous description, JADE seems to be an appropriate tool to realise our multi-agent system.

In fact, JADE agent model is based on a horizontally layered architecture, where each behaviour corresponds to a layer. Therefore, this agent model can directly realise the “reactive with state” and horizontally layered architecture, we need to

implement IAs, DOAs and AAs, if each behaviour implements the three functions *see*, *next*, and *action* described in section 3.3.

JADE does not offer a BDI architecture we need to implement UPAs and SAs; however, a BDI architecture can be implemented with a limited effort on the top of JADE agent model through an appropriate composition of a set of behaviours implementing the different functions defining the BDI architecture.

Finally, JADE offers an implementation of all the FIPA interaction protocols. Therefore, all the interaction protocols, except the partial global planning protocol, come with a limited effort from JADE. The realisation of the partial global planning requires a big effort, but can be guided by the implementation of FIPA interaction protocols.

ACKNOWLEDGEMENTS

We like to thank all the partners of the CoMMA project, because some of the ideas presented in this paper derive from discussions we have with them and from project documents they wrote for the project.

This research is partially supported by the European Commission through the contract *IST-1999-12217 - CoMMA - Corporate Memory Management through Agents*.

REFERENCES

- [1] F. Bellifemine, A. Poggi and G. Rimassa. JADE - A FIPA-compliant Agent Framework. In Proc. Fourth International Conference on the Practical Application of Intelligent Agent and Multi Agent Technology (PAAM99), pp. 97-108, London, UK, 1999.
- [2] B. Berney and E. Ferneley. CASMIR: Information retrieval Based on Collaborative User Profiling. 1999.
- [3] M.E. Bratman. Intentions, Plans, and practical Reason. Haward University Press, Cambridge, MA, 1987.
- [4] R.A. Brooks. A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation, 2(1):14-23, 1996.
- [5] CoMMA. The CoMMA Project Home Page. Available at <http://www.ii.atos-group.com/sophia/comma/HomePage.htm>.
- [6] O. Corby and R. Dieng. A CommonKADS Expertise Model Web Server. In Proc. Fifth Int. Symp. on the Management of Industrial and Corporate Knowledge, 1997.
- [7] D. Dunning, G. Regnier, G. McAlpine, D. Cameron, B. Shubert, F. Berry, A.M. Merritt, E. Gronke and C. Dodd. The Virtual Interface Architecture. IEEE Micro, 18(2):58-64, 1998.
- [8] E.H. Durfee. Coordination of Distributed problem Solvers. Kluwer Academic Press, Boston, MA, 1988.
- [9] J. Euzenat. Corporate Memory through Cooperative Creation of Knowledge Base Systems and Hyper-Documents. In Proc. of Knowledge Acquisition Workshop (KAW'96), Banff, Canada, 1996.
- [10] J. Ferber. Simulating with reactive agents. In E. Hillebrand and J. Stender (Eds.), Many Agent Simulation and Artificial Life, pp. 8-28, IOS Press, Amsterdam, Netherlands.
- [11] I.A. Ferguson. Integrated control and coordinated behaviour: a case for agent models. In M. Wooldridge and N.R. Jennings, editors. Intelligent Agents: Theories, Architectures and Languages. LNAI-890, pp. 203-218, Springer Verlag, Berlin 1995.
- [12] FIPA. FIPA Specifications 1997. 1997. Available at <http://www.fipa.org>.
- [13] E. Gamma, R. Helm, R. Johnson and J. Vlissides. Design Patterns: Elements of Reusable Object Oriented Software. Addison Wesley, 1995.
- [14] E.J. Fridman-Hill. Java Expert System Shell. 1998. Available at <http://herzberg.ca.sandia.gov/jess>.
- [15] JADE. The JADE Project Home Page. 2000. Available at <http://sharon.csel.it/projects/jade>.
- [16] R.E. Johnson and J.M. Zweig. Delegation in C++. The Journal of Object Oriented Programming, 4(7):31-34, 1991.
- [17] V. Karamcheti, J. Plevyak and A. Chien. Runtime Mechanisms for Efficient Dynamic Multithreading. Journal of Parallel and Distributed Computing, 37:21-40, 1996.
- [18] D. Kinny and M.P. Georgeff. Commitment and effectiveness of situated agents. In Proc. 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91), pp. 82-88, Sidney, Australia, 1991.
- [19] G. Lavender and D. Schmidt. Active Object: An object behavioural pattern for concurrent programming. In J.M. Vlissides, J.O. Coplien, and N.L. Kerth, Eds. Pattern Languages of Program Design. Addison-Wesley, Reading, MA, 1996.
- [20] J.P. Muller, M. Pischel and M. Thiel. Modelling reactive behaviour in vertically layered agent architectures. In M. Wooldridge and N.R. Jennings, editors. Intelligent Agents: Theories, Architectures and Languages. LNAI-890, pp. 261-276, Springer Verlag, Berlin 1995.
- [21] Rabarijaona, R. Dieng and O. Corby. Building a XML-based Corporate Memory. In Proc. European Knowledge Acquisition Workshop (EKAW'99). LNAI 1621, Springer-Verlag, 1999.
- [22] A.S. Rao and M.P. Georgeff. Modeling rational agents within a BDI-architecture. In j. Allen, R. Fikes and E. Sandewall, editors. Proc. 2nd Int. Conf. on Principles of knowledge Representation and Reasoning, pp. 473-484, Morgan Kaufmann, San Mateo, CA, 1991.
- [23] R.G. Smith. The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver. IEEE Trans. on Computers, 29(12): 1104-1113, 1980.
- [24] M. Wooldridge and N.R. Jennings. Intelligent agents: Theory and practice. The Knowledge Engineering Review, 10(2):115-152, 1995.

On the convergence of core technologies for knowledge management and organisational memories: ontologies and experience factories

Yannis Kalfoglou¹

Abstract. In this paper we argue for the convergence of core technologies for knowledge management and organisational memories. Most of the work reported in the literature regards knowledge management and organisational memories as intertwined areas. However, the technologies used to implement and support them are not treated in the same fashion. Usually, they are conceived, developed, and deployed separately. This prevents us from fully exploiting their strength. We identify two such technologies in this paper: ontologies and experience factories, originating in these different communities. We elaborate on their strengths as potential core technologies and show, through an example case, how their convergence could be of mutual benefit. We generalise the approach and speculate on the impact of such convergence in the broader context of knowledge management and organisational memories.

1 Motivation

Nowadays, we are witnessing increasing interest in knowledge management(hereafter, KM), and organisational memories(hereafter, OMs). The former refers to: “the formal management of knowledge for facilitating creation, access, and reuse of knowledge”([35]). The latter aims to provide the means for storing, retrieving and distributing knowledge from an organisation’s repositories. The goal of OMs is to preserve knowledge whereas KM aims to develop and deploy knowledge. Clearly these two areas are intertwined and centred upon the enhancement of an organisation’s competitiveness by improving the way it manages its knowledge.

There are various ways of implementing KM systems in an organisation as well as technologies for supporting OMs development. Ontologies and experience factories have been studied as the means to support KM and OMs, respectively. Ontologies were originally investigated in the Artificial Intelligence(hereafter, AI) community as a way to represent consensual knowledge about a domain of interest in reusable and sharable formats. On the other hand, experience factories, studied in the Software Engineering(hereafter, SE) community, have been explored as the means to promote and manage experiences collected throughout the life-cycle of a software project. Recently, both they have begun to be applied in KM and OMs as core technologies to support their implementation. O’Leary investigates the role of ontologies in KM systems and argues that: “a KM system depends on ontologies to facilitate communication between its multiple users and links between multiple knowledge bases”([35]). Similarly,

experience factories and their constituents, experience bases, have been applied as an OM to support exchange of experiences(see, for example, [6]).

Despite the fact that KM and OMs are intertwined the technologies that are employed to implement them are developed and applied separately. For example, most of applications of ontologies do not mention nor use any kind of organisational framework(i.e., in the form of an OM), but merely are deployed in order to achieve knowledge sharing and reuse benefits. In the same line, experience factories amid their successful implementations as OMs, do not employ particular types of ontologies, or even when they do these are often distant from the ontologies reported in the AI literature. This prevents us from exploiting the full potential of these two core technologies.

In this paper we argue for their convergence in order to exploit their potential more fully: knowledge sharing and reuse achieved by deploying ontologies could improve the development and deployment of KM, whereas the presence of an OM, in the form of an experience factory, could allow us better to organise, characterise, and store ontologies.

O’Leary argued in [36] for the need to “convert individual knowledge to group-available knowledge”. This converting process requires knowledge sharing and collection in a form that can be generated and reused. The author refers to this as the *knowledge harvesting*, which “must identify knowledge that it desirable to share, worth converting, and usable by others”([36]). In this paper we paraphrase the above definition in the context of ontologies as follows:

Ontology harvesting must identify ontologies that are desirable to share, worth converting, and usable by others.

We argue that this *harvesting* process can be accomplished by employing OMs technologies, such as experience factories.

In section 2 we review the role of ontologies in KM and support our claim that they are a core technology to achieve KM goals which we identify in that section. Experience factories could effectively support OMs as we describe in section 3. These two technologies, however, should complement each other and we elaborate on an example case in section 4 where we employed experience factories to support ontologies deployment. We generalise the approach and speculate on the convergence of ontologies and experience factories in section 5. We conclude the paper in section 6.

2 Ontologies in support of KM

Many researchers are investigating the role of ontologies in KM. Recently, the reports of O’Leary in [36] and of Benjamins and colleagues in [11], summarise the contributions made and give us an

¹ University of Edinburgh, School of Artificial Intelligence, Division of Informatics, 80 South Bridge, Edinburgh, EH1 1HN, Scotland, email: yannisk@dai.ed.ac.uk

insight for their potential. O’Leary describes uses of ontologies in KM systems with example cases drawn from the private sector, in particular consulting firms([36]). In [11], Benjamins and colleagues argue for an ontology-based KM which results in an intelligent access to knowledge assets as they shown in an example case of KM in a virtual organisation.

We briefly recapitulate here the conclusions drawn in these reports and we also mention additional efforts from the ontology community to achieve effective KM. First, we identify areas of KM where ontologies are suitable for exploitation. In a review of KM systems([35]), O’Leary identified three factors that lead organisations to use KM. These were classified as *environmental pressures* imposed by the increasingly competitive global market place, *technological advancements* arising from recent developments in Internet technology, and the ability to *create valuable information* by converting individually available knowledge into group or organisationally available knowledge. Whereas the *environmental pressures* and *technological advancements* give an organisation the reason and the means to pursue KM activities, *creating valuable information* is the goal of KM.

The latter is achieved by the *converting and connecting* processes as identified in [36]. These are summarised as follows: convert (i)individual to group knowledge, (ii)data to knowledge, (iii)text to knowledge, and connect (iv)people to knowledge, (v)knowledge to knowledge, (vi)people to people, and (vii)knowledge to people. We argue that ontologies are present in most of these processes, either by playing a major role or by supplying the supporting infrastructure that helps an organisation to implement them. In the following paragraph we mention indicative examples from the ontology research literature to justify this claim. A complete listing is out of the scope of this paper which is to investigate the potential of convergence of KM and OMs technologies. However, we point the interested reader to field reviews in [44],[13],[22], and [25].

In particular, ontologies provide part of the infrastructure for conversion processes(i to iii as listed above) and play a major role in the connection activities(iv to vii as listed above). Conversion processes (i) seem to benefit more from the presence of ontologies as this is the underlying principle in their construction. Methodological([44]) and collaborative approaches([40],[10]) in ontology building, convert individual to group knowledge in the form of an ontology. Processes (ii) and (iii) use other AI technology like data and text mining techniques with ontologies being the guide to the ‘right’ data or text repository([16]). Ontologies are more active in the connecting processes. Process (iv) is concerned with the so called, ‘pull’ technology, which aims at pulling knowledge residing in vast repositories to people. The means which used to pull that knowledge are, mainly, search engines and intelligent agents. Examples of ontology use in this area are given in [32] and [24]. Process (v) actually highlights the main contribution of ontologies: enabling communication and interoperability between systems. The best way to cite indicative work here is to point to reviews and collections such as [44] and [23]. Process (vi) is not directly related to ontologies as it is more concerned with technological means such as Intranets. However, we should mention the work on collaboration and discussion aided by ontologies([39]). In contrast with process (iv), process (vii) is concerned with ‘push’ technology. Means to achieve this are designated systems that focus on content and push knowledge to the user instead of waiting for the user to pull out that knowledge. As in (iv), ontologies play a major role here since they are concerned with content and semantically enriched information. Example uses are described in [19] and [16].

We now return to the conclusions drawn in the two reports men-

tioned above([36] and [11]). In [36], O’Leary argued for a number of factors that drive the need for ontologies in KM. In brief, these are: formation of discussion groups on particular topics of interests, improved search capabilities by semantically-enriched query/answering facilities, filtering facilities to capture the desired knowledge, reusability of artefacts, and enabling communication between different systems and/or people by using an interlingua. In the other report([11]), Benjamins and colleagues argued that four basic activities of KM, namely, knowledge gathering, organisation and structuring, refinement, and distribution could be effectively supported by ontologies as they shown in their report with the use of ontologies in each of these activities.

3 Experience factories in support of OMs

While in the area of KM, as we saw in the previous section, ontologies have being acknowledged by many as a core technology to support KM, there is no such distinguishable technology in OMs. A reason for this may be the variety of activities that OMs are concerned with. For example, Reimer defines OMs as: “the means by which knowledge from the past is brought to bear on present activities, thus resulting in higher or lower levels of organisational effectiveness”([37]). Its purpose, Reimer continues, is to “ensure coherence between organisational units, coherence between cooperating companies, secure knowledge, and provide knowledge”. There is no single technology that can accommodate all those activities and recent implementations combine several. For example, Abecker and colleagues identified potential research areas that contribute to OMs implementation([1]).

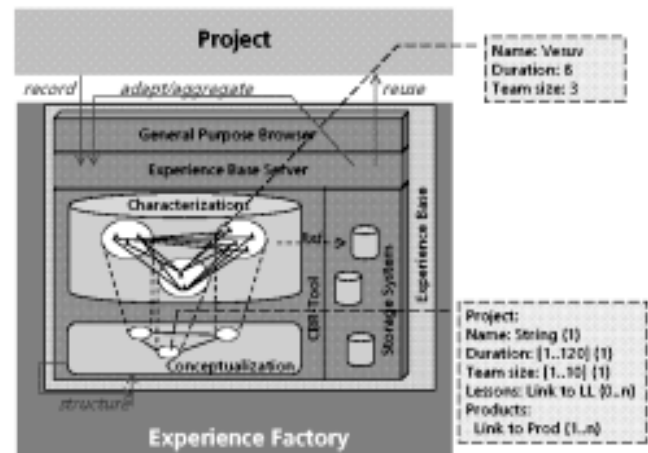


Figure 1. Main experience factory tasks and experience base architecture(adopted from [4]).

However, there is an area of research, stemming from the SE community, which is being promoted as a potential candidate for implementing OMs. This is the area of experience factories². These were first investigated in the context of the TAME project([9]), and further generalised in the early nineties([8]) as the means to promote reuse of “all-kinds” of artefacts in an organisation. In figure 1, we illustrate

² In the SE literature, it is also referred to as “experience management” or “experienceware”.

the main experience factory tasks and an experience base architecture, taken directly from [4]. The core of an experience factory is the experience base which acts as the OM. The key idea is to install an OM to support exchange of all kinds of experiences in the life cycle of a software project. The main focus of an experience factory is to support ‘learning from experience’ on a technology-independent organisational level. An experience factory stores the collected experiences in an experience base. In [5], it was argued that Case-Based Reasoning(hereafter, CBR) plays an important role in the experience factory paradigm. As CBR provides both the technology and a methodology for ‘learning from experience’ in the context of case-based knowledge systems([3]), it was natural to use it for implementing continuous learning in an experience factory style. In [4], Althoff and colleagues deployed CBR to manage the retrieval and adaptation of experiences in an experience factory.

The value of a repository of cases is also acknowledged by KM researchers as O’Leary writes with respect to the conversion of individual to group available knowledge: “Although individuals might have generic knowledge to contribute, case histories are particularly robust”([35]). Such repositories of cases are suitable for applying CBR techniques which provide the means for retrieval and adaptation.

As in KM systems, ontologies also found in OMs implementations, as for example in [2]. Reimer elaborates on a generic architecture for OM which “[...]provides content description that uses a domain ontology to refer to such knowledge areas as employee skills, company regulations, office tasks, organisational structure, products, marketing channels, etc.”([37]). Liao and colleagues describe the use of ontologies for knowledge retrieval in OMs([31]). Recently, work has begun to integrate ontologies and experience factories, mainly in the SE community ([41]). In the next section, we argue for the benefits of such convergence via an example case in the area of ontology verification which we originally presented in [27].

4 Ontologies and Experience Factories

Most of research in ontologies is focussed on development issues and we have witnessed products and tools such as methodologies(i.e., [44], [20]) and guidelines([21]) for building ontologies, online environments([18],[40],[17]) for collaborative construction, agent-based systems([7]) for supporting the selection task, and generic ontologies([30]) to be used as foundations for domain-specific ones. However, there is a dearth of tools to support ontology deployment and maintenance.

To deploy ontologies correctly we need not only reuse-oriented tools and technologies, as for example in the HPKB([15]) and IBROW([19]) projects. It is also necessary to record and organise our experiences in having applied them in order to improve future ontology deployment. It is hard to gain this sort of experience from the literature because few cases of comprehensive ontology reuse and deployment on a large scale are reported([43], [12], [46]). Even those which are reported do not normally discuss the hidden assumptions and tradeoffs identified during testing. This section explains how OMs technologies, like experience factories, may address this issue.

We argue that experience factories can be useful in ontology development and deployment as a way of managing the experiences collected from various agents participating in ontology building and usage. This brings, crudely speaking, OMs technology into KM core technology - ontologies - which in turn has important advantages for the organisation that employs KM activities. We elaborate on these in section 5 while here we focus on a simple architecture we built

to make the idea more concrete. The architecture, which is centred upon the notion of ontology verification, is given diagrammatically in figure 2.

The left-hand side of figure 2, depicts the task of verification. In particular, we are interested in verification of ontologies at the application level([28]). That is, we verify that ontological constructs are not misused by applications that adhere to an ontology. After applying our verification mechanism we accumulate, temporarily, the results in an experience base. These are code-testing results and we regard them as experiences. The experience base is then imported by an experiences editing tool which allows for further additions and modification of the description of existing experiences. It allows us to customise the experiences to the particular project as it provides a way of expressing information usually not obtainable through code-testing. We then select the experiences we want to validate and send them to a designated tool for verifying their correctness with respect to test results. This tool embodies the verification mechanism we deploy in the first step but here we apply it to verify the correctness of the results themselves. After the selected experiences have been validated we store them in the final experience base to be part of the experience factory.

This cycle can be repeated as many times as we wish in the same or other ontologies to collect and manage the knowledge accumulated during verification and testing. Ultimately, this will result in an experience factory of ontology verification and testing that can be deployed in similar projects in order to facilitate ontology use. It is important to mention that the similarity of projects to benefit from this sort of experience factory dictates the ability to transfer experiences. How easily can we identify similar projects in order to transfer previous experiences? An answer to this question is given by the ontology characterisation framework mentioned later in this section.

In [27] we applied this approach in an example case where we deployed and verified an existing ontology at the application level. The case is explained thoroughly in the aforementioned paper, while here we briefly describe by following the architecture given above. The ontology which we used, is the PHYSSYS ontology documented in [12]. It is a formal ontology based upon system dynamics theory and comprises of 7 different ontologies related via an inclusion lattice. We deployed two of these, the *mereology* and *topology* ontology in an exemplar application provided by the PHYSSYS developers themselves. We used the verification mechanism we invented([26]) to check the correctness of *mereological* and *topological* definitions used in the exemplar application. We devised and introduced artificial errors in the application in order to demonstrate the usage of the verification mechanism. Surprisingly, we also detected errors occurred in the original ontologies. We accumulated and verified all the discrepancies found, by storing them as experiences in the final experience base mentioned above. The size of such experience base varies in accordance with the project being applied. In our prototypical case we created a small experience base of a dozen of test cases.

Apart from this specific information on testing we also wanted to store additional information with regard to the ontologies used which will allow us to retrieve and adapt the experiences in different settings. We didn’t dealt closely with retrieval and adaptation techniques as we intend to adopt those made available by the underlying technology, the CBR method(an example of which is described in [47]). This extra information has already been identified as ‘ontology characterisations’ and an early attempt to operationalise them is given by Uschold in [42]. Moreover, in [45], Uschold and Jasper describe possible scenarios where these characterisations could be of help in understanding ontology applications. As Uschold points

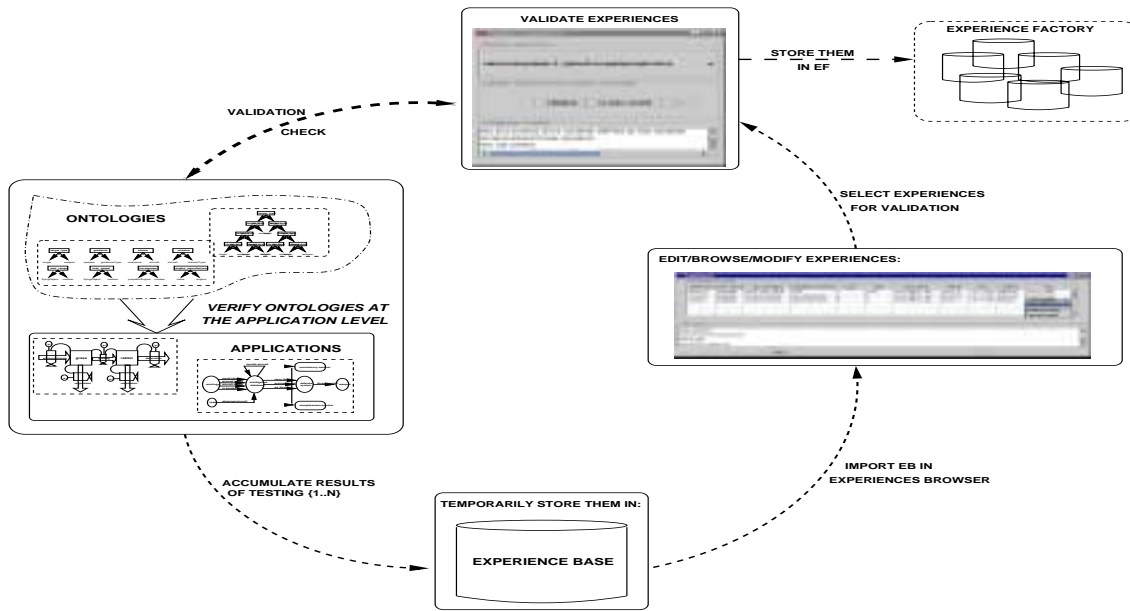


Figure 2. Experience-based architecture to support ontology verification(adopted from [27]).

out in [42]: “if this kind of description was provided for all applications, we would quickly get an overview of the state-of-the-art of ontology application, and a way to compare applications. It would be relatively easy to see which techniques have been used to apply particular kinds of ontologies in specific contexts”. We found the goal of this research similar to our aim, which is to provide the means to organise and collect experiences gained during ontology testing and verification. Therefore, we found it practical to instantiate the proposed framework for characterising ontologies in our verification and testing scenario.

To make our case more precise we list below instantiations of the proposed framework tailored to our scenario. Each item represents a category, followed by its instantiation in the particular context. We also include a short explanation of each category in a parenthesis following its name.

- **Purpose:**(*the purpose of the ontology*) In our tests, both mereology and topology provide the building blocks for PHYSSYS ontology. They formalise generic mereotopological relations as described in the literature(i.e., [38] and [14]);
- **Representation languages and paradigms:**(*Ontolingua? Description Logics? Prolog? Clips? XML?*) The mereology and topology ontology were implemented in Ontolingua, we translated them in the implementation language we use: Prolog;
- **Meaning and formality:**(*to what extent and how formal is the specification of the meaning of each term?*) The Ontolingua versions of mereology and topology provide primitive terms with axioms restricting their use by placing constraints on relationships between types of entities. The implemented versions we used(in Prolog), include this information along with application specific constructs;
- **Subject Matter:**(*is it domain-specific or general?*) Both mereology and topology provide generic relations and axioms to be used in the PHYSSYS ontologies set;

- **Scale:**(*how big is the ontology?*) Both ontologies are quite small, each of them formalises less than 10 relations;
- **Development:**(*the degree to which the application is specified, developed and/or fielded*) The mereology and topology ontologies are research prototypes but the PHYSSYS which includes them has been used in the context of the OLMECO project([12]);
- **Conceptual architecture:**(*what are the main components in the ontology application, and how do they relate to each other*) A single research prototype system was used, the *hospital heating system*, implemented in Prolog, which includes ontological constructs;
- **Mechanisms and techniques:**(*what specific mechanisms and techniques were invoked to make use of the ontology*) We deployed a multi-layered architecture([28]) to embed ontological constructs in the application. We translated ontological constructs to the target implementation language: Prolog, and translated ontological axioms to a designated constraints format used to detect discrepancies with respect to misuse of ontological constructs([26]);
- **Implementation platform:**(*the particular implementation platform and context*) The ontologies are described textually in the literature and were also written in Ontolingua syntax, which we translated in Prolog. The testbed application was also implemented in Prolog. The tests were executed through a Java front-end.

The role of these characterisations is to provide ontology-specific information which can be used for organising and retrieving experiences on testing in similar ontology deployment efforts.

Recall figure 1 from section 3 where we presented the main experience factory tasks as realised in [4]. Here we reproduce that figure along with its instantiations in the context of our case: ontology verification.

As illustrated in figure 3, we have added information obtainable from our experiences with the ontology verification task. These are

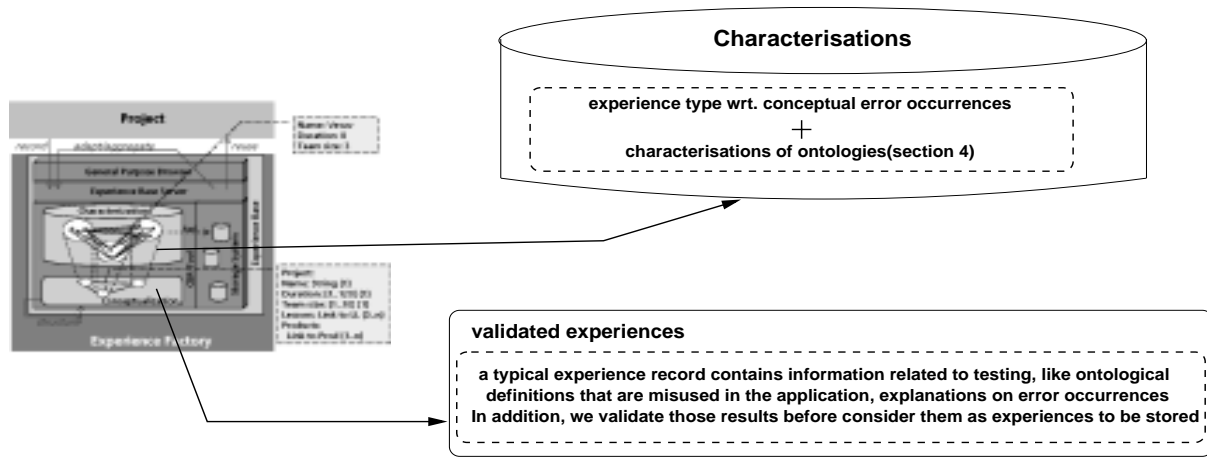


Figure 3. The experience factory instantiated with the ontology verification scenario(adopted from [27]).

results from code-testing, along with additional information such as explanations for error occurrences and are contained in an experience record which we validate before we store it in the experience base. In the characterisation phase we can give generic information, like the experience types with respect to conceptual error occurrences. In addition, at this phase we also provide links to the ontology characterisations described above, which are edited and linked to the relevant experiences through a designated experience table(described in [27]). This helps us to browse through the collected experiences and ultimately supports the idea of “learning from experience” which is an acknowledged need in ontology deployment.

5 OMs in KM

The development and use of KM ontologies hides important caveats. As O’Leary reports in [35]: “Each consulting firm we have been examining has built or is building its own ontologies. Because these enterprise ontologies are so costly to develop and maintain and are constantly changing, ontology or taxonomy issues are emerging as some of the most important problems in knowledge management”. O’Leary analyses further these problems from the KBSs development point of view in [34]. In this section we elaborate on our vision of “bringing OMs technology into KM core technology: ontologies”, which could, potentially, alleviate the situation. To explain the idea we illustrate the approach in figure 4.

On the left hand side of figure 4, we place within a box surrounded by a dash-lined border the OMs technologies. At the bottom of that box we place a potential technology for supporting OMs implementation: experience factories. The diagram included there is actually a reduction of figure 2 presented in section 4. It denotes an example use of experience factories in the area of ontology verification as we described in that section. There can be other technologies for implementing OMs. We point the interested reader to [1] where Abecker and colleagues elaborate on potential technologies for OMs implementation.

On the right hand side of figure 4, we illustrate the main KM tasks and activities. We identify four main KM tasks: *acquiring*, *analysing*, *using*, and *preserving* knowledge. We argue that these tasks are accomplished by activities which are supported by ontologies. In par-

ticular, the knowledge acquisition task, is accomplished by *identifying* activities which are supported by ontologies. This results in the application area of information extraction and/or content-matching. In the same manner, ontologies in the area of knowledge representation are used to *model* and *assess* the environment, which are activities employed in the *analysing* knowledge task. The *using* knowledge task, includes the *apply*, *share*, and *reuse* activities, which are supported by ontologies with such application areas as knowledge sharing and reuse, and KBSs. The last task is *preserving* knowledge. It is accomplished by activities such as *organising*, *maintaining*, and *capitalising* knowledge which are partially aided by ontologies. The resulting application area is that of libraries of reusable knowledge components and experience repositories. The knowledge preservation task and its accompanying activities along with the relevant ontologies are the area of overlap with experience factories as denoted by the dashed box surrounding the task in figure 4.

The way in which the two boxes of figure 4 are related summarises the linkage we are suggesting in this paper. In the introductory section, we argued that OMs and KM are intertwined areas. In this figure we illustrate how the technologies used to implement them can also be intertwined. As can be seen from the curly arrow connecting the OMs technologies with the KM tasks/activities box, technologies such as experience factories, can be employed to *organise* ontologies used to support main KM tasks/activities. The latter, in turn, can *support* OMs implementation by *acquiring*, *analysing*, *using*, and *preserving* knowledge to be processed by an OM.

There are mutual benefits for integrating OMs technology in KM ontologies. On one hand, an OM framework could help to improve ontology development and deployment, facilitate understanding, and ease reuse. A better organised ontology could, in turn, overcome some of the problems identified in [34]: “perfect ontology hype, library ontologies, scale-up, interface, formality”, and analysed from a cost-benefit point of view in [25]. We envisage deployment of experience factories, especially experience bases, in the whole spectrum of ontology life-cycle: experience bases on testing(an example of which was presented in section 4, taken from [27]), on development, on maintenance, etc. Ideally, all these experience bases could be placed alongside with their ontology counterparts in large repositories of reusable knowledge components. A metaphor of this idea

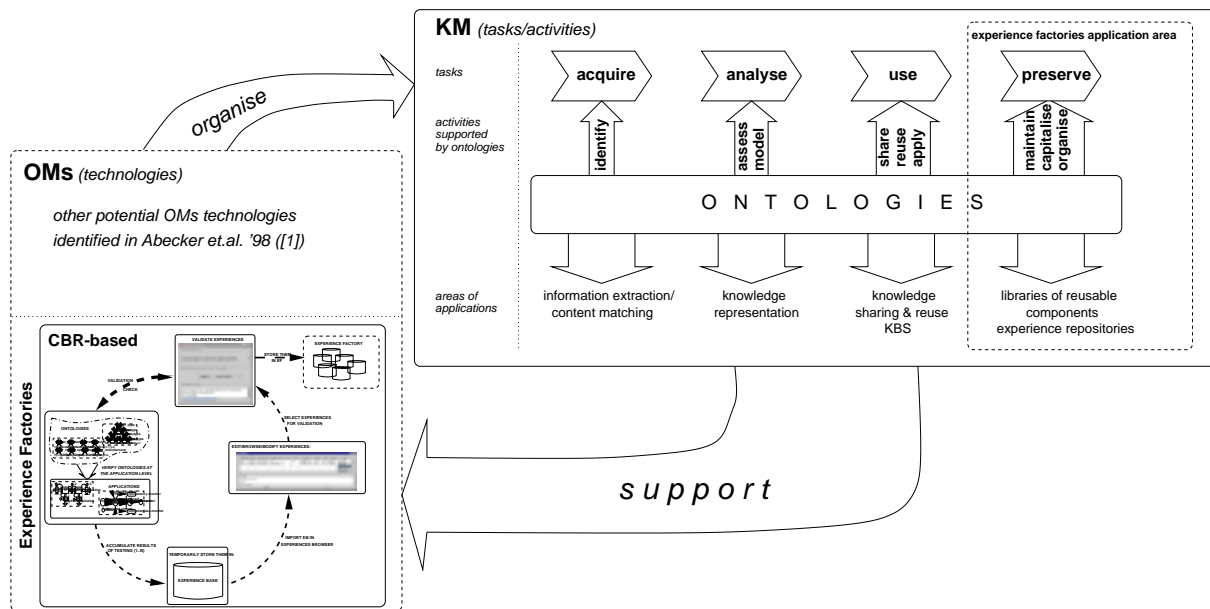


Figure 4. OMs in KM: OMs technology used to organise KM tasks/activities which in turn support the implementation of OMs.

are the implemented on-line libraries of ontologies, like for example, *Ontolingua*, *Ontosaurus*, and *WebOnto*.

On the other hand, better ontologies could help to meet practical requirements for the implementation of OMs. These were identified by Abecker and colleagues in [1]: “(i)collection and systematic organisation of information from various sources, (ii)ability to minimise up-front knowledge engineering, by taking advantage of readily available information, (iii)exploiting user feedback for maintenance and evolution, (iv)integration into existing work environment, (v)active presentation of relevant information”. Requirements (i),(iv) and (v) could benefit more from ontologies as we briefly described in section 2 and diagrammatically present in figure 4. Requirements (ii) and (iii) could benefit more from the presence of an experience factory. In [2], Abecker and colleagues show how they satisfied these requirements in the *KnowMore* system which realises proactive knowledge delivery in order to enable employees solve cooperatively knowledge-intensive tasks.

6 Conclusions

In this paper we explored the possibility of employing OMs technology to improve understanding and enhance the usage of ontologies. The latter are a core technology for KM as we discussed in section 2. Despite the fact that OMs and KM are intertwined areas and usually are treated as a whole, we argue that the technologies used to implement and support them are not. We discussed the potential convergence of technologies for OMs, in particular experience factories, with ontologies in section 4 where we presented an example case in the area of ontology verification and testing. We generalised the approach in section 5 and speculated on the benefits of such convergence for both OMs and KM.

The intersection of these two technologies also highlights the role of knowledge engineering. As Milton and colleagues argue in [33] there is a change of focus in KM, from IT-based solutions to Know-

ledge Technology(KT)-based ones. To quote the authors: “[...] the main purpose of KT is to provide solutions to certain key problems associated with KM, such as scoping what knowledge is to be captured and disseminated, dealing with tacit knowledge, and facilitating better understanding”. The field of knowledge engineering has been studying and practising solutions to these problems for years.

Applying these solutions to KM could help us to realise the ‘ultimate goal’ of OMs as envisaged by Kuhn and Abecker: “[...] to provide the necessary knowledge whenever it is needed. To assure this, [OMs] realize an active knowledge dissemination approach which does not rely on users’ queries but automatically provides knowledge useful for solving the task at hand. To prevent information overload, this approach has to coupled with a highly selective assessment of relevance”([29]). The CBR-based experience factories coupled with ontologically-based reasoning are a way of achieving this ‘selective assessment of relevance’.

In accordance with the ‘ontology harvesting’ notion we mentioned in the introduction we conclude the paper with a motto: *if ontologies are to be a cornerstone for successful KM, we need to manage them by bringing OMs technology into their development, deployment, and maintenance phases, which in turn, will result in a more effective OM.*

ACKNOWLEDGEMENTS

The research described in this paper is supported by a European Commission’s Marie Curie Fellowship(programme: Training and Mobility of Researchers). Thanks to David Robertson for his comments on an early draft of this paper and to anonymous reviewers for their comments.

REFERENCES

- [1] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kuhn, and M. Sintek, ‘Toward a Technology for Organizational Memories’, *IEEE Intelligent Sys-*

- tems, **13**(3), 40–48, (June 1998).
- [2] A. Abecker, A. Bernardi, and M. Sintek, 'Proactive knowledge delivery for enterprise knowledge management', in *Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering(SEKE'99)*, Kaiserslauten, Germany, pp. 120–127, (June 1999).
 - [3] K-D. Althoff, P. Bergmann, and L.K. Branting, eds. *Case-Based Reasoning Research and Development(ICCBR'99)*, number 1650 in Lecture Notes in Artificial Intelligence. Springer Verlag, July 1999.
 - [4] K-D. Althoff, A. Birk, S. Hartkopf, W. Muller, M. Nick, D. Surmann, and C. Tautz, 'Managing Software Engineering Experience for Comprehensive Reuse', in *Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering, SEKE'99*, Kaiserslauten, Germany, pp. 10–19, (June 1999).
 - [5] K-D. Althoff, F. Bomarius, and C. Tautz, 'Using Case-Based Technology to Build Learning Software Organizations', in *Proceedings of the ECAI'98 Workshop on Building, Maintaining, and Using Organisational Memories(OM-98)*, Brighton, England, (August 1998).
 - [6] K.-D. Althoff, M. Nick, and C. Tautz, 'Improving organizational memories through user feedback', in *Proceedings of the Learning Software Organizations(LSO'99) workshop*, Kaiserslauten, Germany, ed., F. Bomarius, pp. 27–44, (June 1999).
 - [7] J. Aspirez, A. Gomez-Perez, A. Lozano, and S. Pinto, '(onto)2agent: An ontology-based www broker to select ontologies', in *Proceedings of the Workshop on Applications of Ontologies and Problem-Solving Methods, ECAI'98*, Brighton, England, pp. 16–24, (August 1998).
 - [8] V. Basili, G. Caldiera, and D. Rombach, 'Experience Factory', in *Encyclopedia of Software Engineering*, ed., J. Marciniak, volume 1, 469–476, John Wiley & Sons, (1994).
 - [9] V.R. Basili and H.D. Rombach, 'The TAME Project: Towards Improvement Oriented Software Environments', *Transactions on Software Engineering*, **SE-14**(6), 758–773, (June 1988).
 - [10] R. Benjamins and D. Fensel, 'The Ontological Engineering Initiative-KA2', in *Proceedings of the 1st International Conference on Formal Ontologies in Information Systems, FOIS'98*, Trento, Italy, ed., N. Guarino, pp. 287–301. IOS Press, (June 1998).
 - [11] R. Benjamins, D. Fensel, and A. Gomez-Perez, 'Knowledge Management through Ontologies', in *Proceedings of the 2th International Conference on Practical Aspects of Knowledge Management(PAKM'98)*, Basel, Switzerland, (October 1998).
 - [12] P. Borst, H. Akkermans, and J. Top, 'Engineering Ontologies', *International Journal of Human-Computer Studies*, **46**, 365–406, (1997).
 - [13] B. Chandrasekaran, R. Josephson, and R. Benjamins, 'What Are Ontologies, and Why Do We Need Them?', *IEEE Intelligent Systems*, **14**(1), 20–26, (January 1999).
 - [14] B.L. Clarke, 'A calculus of individuals based on "connection"', *Notre Dame Journal of Formal Logic*, **22**, 204–218, (1981).
 - [15] P. Cohen, R. Schrag, E. Jones, A. Pease, A. Lin, B. Starr, D. Gunning, and M. Burke, 'The DARPA High Performance Knowledge Bases project', *AI Magazine*, **19**(4), 25–49, (1998).
 - [16] S. Decker, M. Erdmann, D. Fensel, and R. Studer, 'Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information', in *Proceedings of DS-8, Semantic Issues in Multimedia Systems*, Boston, MA, USA, ed., R & et.al. Meersman, pp. 351–369, (1999).
 - [17] J. Domingue, 'Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web', in *Proceedings of the 11th Knowledge Acquisition, Modelling and Management Workshop, KAW'98*, Banff, Canada, (April 1998).
 - [18] A. Farquhar, R. Fikes, and J. Rice, 'The ontolingua server: a tool for collaborative ontology construction', *International Journal of Human-Computer Studies*, **46**(6), 707–728, (June 1997).
 - [19] D. Fensel, V.R. Benjamins, E. Motta, and B. Wielinga, 'UPML: A Framework for knowledge system reuse', in *Proceedings of the 16th International Joint Conference on Artificial Intelligence, IJCAI'99*, Stockholm, Sweden, pp. 16–21, (August 1999).
 - [20] M. Fernandez, A. Gomez-Perez, and N. Juristo, 'METHONTOLOGY: From Ontological Arts Towards Ontological Engineering', in *Proceedings of the AAAI-97 Spring Symposium Series on Ontological Engineering*, Stanford, CA, USA, pp. 33–40, (March 1997).
 - [21] T.R. Gruber, 'Towards principles for the design of ontologies used for knowledge sharing', *International Journal of Human-Computer Studies*, **43**, 907–928, (1995).
 - [22] N. Guarino, 'Formal Ontology and Information Systems', in *Proceedings of the 1st International Conference on Formal Ontologies in Information Systems, FOIS'98*, Trento, Italy, ed., N. Guarino, pp. 3–15. IOS Press, (June 1998).
 - [23] N. Guarino, ed. *Formal Ontology In Information Systems*, Frontiers in Artificial Intelligence and Applications. IOS Press, June 1998. ISBN: 90-5199-399-4.
 - [24] N. Guarino, C. Masolo, and G. Vetere, 'OntoSeek: Content-Based Access to the Web', *IEEE Intelligent Systems*, **14**(3), 70–80, (May 1999).
 - [25] Y. Kalfoglou, T. Menzies, K-D. Althoff, and E. Motta, 'Meta-knowledge in systems design: panacea...or undelivered promise?', *The Knowledge Engineering Review(submitted)*, (2000).
 - [26] Y. Kalfoglou and D. Robertson, 'Managing Ontological Constraints', in *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods(KRR5)*, Stockholm, Sweden, <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18/>, (August 1999). Also as: Research Paper No.948, Dept. of AI, University of Edinburgh.
 - [27] Y. Kalfoglou and D. Robertson, 'Applying Experienceware to support ontology deployment', in *Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering, SEKE2000*, Chicago, IL, USA, (July 2000).
 - [28] Y. Kalfoglou, D. Robertson, and A. Tate, 'Using Meta-Knowledge at the Application Level', *Journal of Artificial Intelligence Research*, submitted, (2000). Also as: Research Paper No.956, Dept. of AI, University of Edinburgh.
 - [29] O. Kuhn and A. Abecker, 'Corporate Memories for Knowledge Management in Industrial Practice: Prospects and Challenges', *Journal of Universal Computer Science*, **3**(8), 929–954, (1997).
 - [30] D.B. Lenat and R.V. Guha, *Building large knowledge-based systems. Representation and inference in the Cyc project*, Addison-Wesley, Reading, Massachusetts, 1990.
 - [31] M. Liao, A. Abecker, A. Bernardi, K. Hinkelmann, and M. Sintek, 'Ontologies for Knowledge Retrieval in Organizational Memories', in *Proceedings of the Learning Software Organizations(LSO'99) workshop*, Kaiserslauten, Germany, ed., F. Bomarius, pp. 19–26, (June 1999).
 - [32] L.D. McGuinness, 'Ontological Issues for Knowledge-Enhanced Search', in *Proceedings of the 1st International Conference on Formal Ontology in Information Systems(FOIS'98)*, Trento, Italy, ed., N. Guarino, pp. 302–316. IOS Press, (June 1998).
 - [33] N. Milton, N. Shadbolt, H. Cottam, and M. Hammersley, 'Towards a knowledge technology for knowledge management', *International Journal of Human Computer Studies*, **51**(3), 615–641, (September 1999).
 - [34] D. O'Leary, 'Impediments in the use of explicit ontologies for KBS development', *International Journal of Human-Computer Studies*, **46**(2), 327–337, (1997).
 - [35] D. O'Leary, 'Knowledge Management Systems: Converting and Connecting', *IEEE Intelligent Systems*, **13**(3), 30–33, (June 1998).
 - [36] D. O'Leary, 'Using AI in Knowledge Management: Knowledge Bases and Ontologies', *IEEE Intelligent Systems*, **13**(3), 34–39, (June 1998).
 - [37] U. Reimer. Building, Maintaining, and using Organisational Memories. Invited talk in the ECAI'98 Workshop on Building, Maintaining, and Using Organisational Memories(OM-98), Brighton, England, August 1998.
 - [38] P. Simons, *Parts: A Study in Ontology*, 5–128, Oxford: Clarendon Press, 1987.
 - [39] T. Summer and S. Buckingham-Shum, 'From Documents to Discourse: Shifting Conceptions of Scholarly Publishing', in *proceedings of the CHI'98: Human Factors in Computing Systems*, Los Angeles, CA, USA, pp. 95–102. ACM Press, (1998).
 - [40] B. Swartout, R. Patil, K. Knight, and T. Russ, 'Toward Distributed Use of Large-Scale Ontologies', in *Proceedings of the 10th Knowledge Acquisition, Modeling and Management Workshop(KAW'96)*, Banff, Canada, (November 1996).
 - [41] C. Tautz and C. Gresse von Wangenheim, 'A Representation Formalism for Supporting Reuse of Software Engineering Knowledge', in *Proceedings of Workshop in Reuse in Developing Knowledge-Based Systems(XPS'99)*, Wurzburg, Germany, (March 1999).
 - [42] M. Uschold, 'Where are the Killer Apps?', in *Proceedings of Workshop on Applications of Ontologies and Problem Solving Methods, ECAI'98*, Brighton, England, ed., Gomez-Perez,A. and Benjamins,R., (August 1998).
 - [43] M. Uschold, P. Clark, M. Healy, K. Williamson, and S. Woods, 'An Experiment in Ontology Reuse', in *Proceedings of the 11th Knowledge Acquisition Workshop, KAW98*, Banff, Canada, (April 1998).

- [44] M. Uschold and M. Gruninger, 'Ontologies: principles, methods and applications', *The Knowledge Engineering Review*, **11**(2), 93–136, (November 1996).
- [45] M. Uschold and R. Jasper, 'A Framework for Understanding and Classifying Ontology Applications', in *Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods(KRR5)*, Stockholm, Sweden, (August 1999). <http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18/>.
- [46] A. Valente, T. Russ, R. MacGrecor, and W. Swartout, 'Building and (Re)Using an Ontology for Air Campaign Planning', *IEEE Intelligent Systems*, **14**(1), 27–36, (January 1999).
- [47] G. von Wangenheim, K-D. Althoff, and M.R. Barcia, 'Intelligent Retrieval of Software Engineering Experienceware', in *Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering(SEKE'99)*, Kaiserslauten, Germany, (June 1999).

Unifying or reconciling when constructing Organizational Memory? Some open issues.

Carla Simone¹

Abstract. In the paper surveying tools and methods for Corporate Knowledge Management [1], Distributed Corporate Memory (DCM) is listed among the possible materializations of knowledge and CSCW among the techniques possibly used to realized it. In this paper we want to accept this challenge and try to reinforce the bridge between DCM and CSCW. Despite the apparently natural connection between them, this attempt is still a challenge since the two frameworks rarely meet in integrated proposals where each side provides its results to improve the other one. As in any challenging attempts, results are often more about open issues than coherent solutions.

1. ABOUT THE ORGANIZATION MODEL

Knowledge Management (KM) can be seen as the definition of a producer/consumer process that can take different forms in relation to the organizational structure it is based upon. The latter shows in general a distributed structure for what concerns the producers. In fact, it is well accepted that the sources of information are spread around in the organization. The same holds for the consumers. They can access the Organizational Memory (OM) at any granularity of organizational units. The main point distinguishing the above structures is the possible presence of a central node with a distinguished role. Typically, this node is where locally produced knowledge is collected, filtered, "normalized" before becoming available, with the status of Corporate Knowledge (CK), to the other nodes. Often, within sufficiently big organizations, a dedicated office is envisaged where people's job is to elaborate the collected knowledge, and organize it for future use. This is for example a solution reported by a world wide spread consulting company (in a workshop held in Torino in November 1999). At the opposite extreme, there are fully distributed structures where each producer is responsible of the collection and publication of locally generated knowledge. This case applies in general to organizations showing a highly flat structure where is difficult, if not impossible, to assign distinguished roles in KM. Consider, for example, the typical cases of WEB-based communities of knowledge workers or virtual enterprises. In between the two extremes, there could be all possible mixes of the two above mentioned structures. However, in our experience with companies planning or realizing forms of KM, the intermediate structures are not so frequent. What can be recognized is the presence of loosely coupled subnetworks, either centralized or fully distributed.

In any case, the structure is determined by both the organizational culture and the type of knowledge under concern.

The awareness that knowledge is an important component of the everyday work is not a new phenomenon: knowledge management did already exist before OM/KM. What is new is the fact that companies are increasingly realizing it, pushed by very pressing and well known motivations. To respond to this pressure, their culture can lead them to approach the solution according to the following idea: recognize a new problem and construct a new structure to solve it. Or according to the alternative view: see the problem as diffused in the organization and enhance existing solutions. The first approach naturally leads to the identification of a centralized ad hoc structure, the second one naturally to a more distributed one on top of what already exists. Accordingly, the use of the technology is interpreted in a different way: to propose new functionality and information flows vs. better supporting the existing ones.

Also the type of knowledge plays a role in the definition of the structure. What is perceived as 'objective' or 'commonly interpretable' knowledge easily reaches the status of CK: for example, part of what in [1] refers to Profession or Technical Knowledge. The related contents are likely to be considered as universals that have to be collected and maintained in a coherent heap, more easily achieved by a centralized organization. (Parts of) Other types of knowledge can be less naturally interpreted as universals since they are more dependent on the context in which they are generated (local universal [2]). A typical example is the part of knowledge related to the experience (successful cases, best practices, and so on). This part of knowledge is either lost or (informally) maintained close to where it makes sense.

2. THE SOURCES OF KNOWLEDGE

By considering the structure from the point of view of the nodes providing information, one can make similar considerations as in the previous section. Again, providing knowledge can be seen as an activity which is fully immersed in everyday work or as an additional one occurring asynchronously in relation to it.

This distinction has implications on different policies an organization can apply to promote this activity: for example, incentives, personnel management, and so on. The point we want to take in this paper is the one of work content and supporting technology. The asynchronous view leads to the well known problem of doing something in an episodic way so that it is likely that more pressing, interesting or rewarding activities are done in place. The opposite point of view does not deny the necessity to spend additional effort to produce knowledge but it incorporates this effort as part of the normal work. The advantage is twofold: the produced knowledge is likely to cover both a greater part of the activities potentially generating it and

¹ University of Torino, e-mail: simone@di.unito.it

a greater amount of detailed knowledge from each single activity on the other one, since it is recorded in a easier way. In this situation the technology plays a relevant role. In fact the degree by which the effort to produce knowledge is smoothly meshed with the normal work depends on the degree by which the technological support of both of them is integrated. Hence, words, a supportive technology for normal work should to be designed so that knowledge production is part of its basic functionality.

Despite the increasing attention to KM this is not yet the general case. More often KM technology is a specialized one which is put aside the other ones supporting other activities. This view has as obvious consequences not only the possibility that some knowledge is lost because of the technological hiatus but also the possibility that some activities are considered as not producing knowledge. This however contrasts with the common experience that knowledge is produced by the combined and continuous effect of a variety of apparently not knowledge generating activities. Knowledge is produced when people write documents, communicate through e-mail, surf the web and so on. Now, how many applications supporting these 'simple' actions do contain knowledge oriented functionality? Knowledge workers are claimed to be the only ones surviving the evolution of the current organizations but knowledge is rarely taken into account when technology supporting work is designed. There are of course some exceptions (e.g., [3]).

Generally the most widely applied idea is to consider that a common repository, possibly populated and accessed in a distributed way, through the WEB, can be a good solution to the problem without paying attention on the problems raised by its usage. The metaphor of OM as a repository has been thoroughly criticized in [4] by deconstructing the common interpretation of OM based on it. To simplify their argument, we can say that memory metaphor is misused since instead of bringing in the technology features drawn from human memory, it interprets human memory in term of the technological counterpart, thus reducing later usage of the collected information to a simple retrieval. This brings to the argument of how to select the knowledge to be recorded, in which form to record it and how to organize it in the memory.

3. DE-CONTEXTUALIZING AND RE-CONTEXTUALIZING

The answers of the questions at the end of the previous section has to deal with the issue that can be called de-contextualization and (re-)contextualization. This is, in our opinion, the core problem of KM.

Recording calls for a degree of de-contextualization of the collected knowledge for reasons that go from feasibility and efficiency up to the necessity to extract the true "universal" from it. Re-use calls for a degree of (re-)contextualization to make possible the act of interpreting the retrieved information. Interpretation is the fundamental component of organizational "remembering" as a creative act of organization members [4].

The appropriate balance of two above 'degrees' is difficult to reach since they do not necessarily match. In full agreement with [5], we can rephrase this point by saying that the pyramid proposed there (going from Explanation, through Experience up to Classification) has to be gone along up and down to support the whole KM life-cycle and that in any case not all the knowledge about the various ladder can be adequately covered by a technological support.

The tension between de-contextualization and (re-)contextualization can be dealt with by means of different, hopefully complementary, strategies. First, the recording process can build

and maintain different descriptions of the same piece of reality, at different levels of simplification and abstraction. This idea is again similar to the one proposed in [5]. Our emphasis is however more on the possibility of having different level of visibility of the same type of knowledge while there the emphasis is on different types of knowledge related to the same group of experiences). We will come back to this point in Section 5.

Second, informal knowledge or organizational behaviors that cannot be easily consolidated can be captured and made available by building kinds of 'knowledge maps' like: who knows what, who is expert of, who had similar experiences as, who did or is likely to be helpful in this case, and so on. If human beings play an unavoidable role in KM, then information on them becomes a fundamental part of CK. Of course, this view contrasts with the idea to record things that will be used far away in the future when people could no more be there. However, this is in our view a goal which applies to very particular types of knowledge and situations. More often, the past experience is no more applicable because the context (organization, culture, technology, and the like) did change in a way that makes it useless. In any case, still being able to use in an appropriate way the existing knowledge for current situations would be an invaluable benefit for the organization and people working in it. Maintaining 'knowledge maps' in presence of (high) turn-over should become an action, among others, in charge of the organization itself. The relevance of these 'knowledge maps' has been recognized for example in [6, 7]. The two approaches show a basic difference: the first one derives the map directly from computational features incrementing the functionality of a communication support based on Speech Act theory, while the second one treats (as implicitly do many other proposals) the management of this type of knowledge in an asynchronous way with respect to the processes producing it.

The third strategy makes KM a more distributed process in terms of both the collected knowledge and its maintenance. In this view OM becomes a network of local memories managed by distributed organizational units. Local memories can serve people acting in the different units who are then more motivated to spend effort in maintaining them, as well as people outside these units, especially when the 'knowledge map' mentioned in the previous point is realized. Hence, KM becomes not only more distributed but also more 'cooperative': in terms of [8] KM constructs communities of people actively and consciously sharing interests and experiences. In this scenario, re-contextualization becomes a cooperative learning process that helps both the providers and the consumers of the knowledge itself. The former can validate the quality of the knowledge they are providing and improve it, the latter are adequately supported in interpreting it and will be more motivated in being supportive when the roles will be exchanged in the future. To be sustainable, cooperative KM has to be supported by adequate tools, organizational and technological. This is where KM and CSCW could meet and positively influence each other. Following the above idea that knowledge oriented functionality should be embedded in applications, KM could suggest how to enhance CSCW applications and the principles leading the latter could support cooperative KM.

Recently, the relevance of contextual information was recognized as a fundamental means for improving communication and coordination. The keyword "awareness" was introduced to denote this specific type of knowledge: awareness is far from being univocally defined within CSCW [9]. However, it is common opinion that it is about knowledge on what is going on 'around' a specific activity performed by single or groups of individuals. In the framework of KM this idea was put at work in [8]: "Research on digital libraries are dominated by indexing and retrieval

mechanisms. The usefulness of such tools is not in dispute, however this is not the whole picture of what libraries can and should be about. ... The observed behavior of others in a situation contributes to a sense of the situation's ambience, or social atmosphere". What is needed is to enlarge this view to additional functionality which fully takes into account that cooperative KM is a distributed process (as any other form of cooperation does [10]). Hereafter, we will focus on two issues we are working on. They can be considered as examples of an exercise that could lead to a possibly quite long list of additional functionality.

4. MANAGING ONTOLOGIES

Ontologies play a central role in almost all approaches to KM. Motivations for their introduction and use can be found in [11]. One of the most relevant aspect that matters in the context of KM is that ontologies can be shared (possibly together with the knowledge they help to organize). In this view, identifying a single 'organization' ontology or merging different ontologies into a common one, or translating an ontology into another one are considered as feasible goals to achieve uniformity, sharing and reuse. In addition, formal definitions of concepts and relations linking them is seen as a factor improving portability.

This optimistic view contrasts with other experiences in the use of classification schemes [12, 13]. The latter share with ontologies some basic aspects, like: the decision of what matters, the relations among categories/concepts used to classify or capture the essence of a reality.

In [13] is reported the experience of adopting a 'universal' classification scheme in a big civil engineering company in order to organize the documentation of large projects, like bridge design. The standard was systematically circumvented not because it was recognized poor; rather, because it was impossible to define one (or classes of) scheme(s) serving the different needs. The supportive technology had to be designed taking local adaptability as primary goal to become accepted and used to enhance knowledge circulation about projects.

In [12], again on the basis of empirical evidence, the emphasis is put on how subtle cultural, social and political factors play a role in the choices of basic concepts and relations to capture the essence of work activities, and the consequence they bring along. It is impossible to give a satisfactory account of all the arguments discussed in the book. However, the point we borrow from them is that any type of ontology/classification should be seen as a "boundary infrastructure" serving the purpose of a mediation space among different local communities. "What we gain with the concept of boundary infrastructure over the more traditional unitary vision of infrastructure is the explicit recognition of the different constitution of information objects within the different communities of practice that share a given infrastructure (p. 314)". In addition, ontologies/classifications "always represent multiple constituencies. They can do so most effectively through the incorporation of ambiguity... Designers should recognize these zones of ambiguity, protecting them where necessary to leave the schemes to play their organizational work. (p. 324)".

Now, why to spend an effort in trying to enforce standardization when "the history to attempt to standardize information systems" shows that "standards do not remain standard for very long, and one person's standard" can easily become "another's confusion and mess (p. 293)", as shown for example by the experience reported in [13]? Instead, the same effort could be spent in supporting people to live with the degrees of ambiguity and partiality the boundary infrastructure encompasses, and to actively participate in the

maintenance of such ambiguity and partiality in a continuous learning process triggered by "interruptions of expectations". As stressed in [12], ontologies/classifications do matter for lots of reasons. We agree, and would emphasize that they make sense within communities of practice where their existence is part of cooperation: the latter is not only based in articulation work [14]

but also in categorical work ("distribution of work, and its different meanings in different communities, must be managed for cooperation to occur. The juggling of meanings is what we call categorical work. ([12], pag. 310). Across communities, more flexible links can better take into account the inherent differences. In this view, unification of concepts and relations into a common shared framework should smoothly coexist with the reconciliation of concepts and relations that reflect local culture and work organization.

In our research effort to enhance articulation work through the design of CSCW applications supporting communication and coordination among cooperating people we met the problem of different ways by which these groups look at their reality and organize the related information. Specifically, we designed two frameworks aimed to support local activities and actions that cross groups in order to achieve coordination in organizations that can be conceived of as autonomous interacting communities constructed around shared objectives and work modalities. To this aim we introduced the notion of coordination mechanism: in our terminology, they are dyads of coordinative protocols and artifacts. Moreover, a language (Ariadne) has been developed for defining, specifying, and executing them [15]. Its elemental categories and composition laws, which have been identified empirically and can be modified at any time [14], capture how actors identify the basic units of work and information for articulating their activities and the relations between these units. For this to be feasible requires a degree of expressiveness such that no specific modelling approach is imposed: that is, any type of flow of work up to the light-most shared working space should be representable. Moreover, since classification schemes and coordinative artifacts play an important role in articulation work, the basic categories and relations must allow for their definition as well.

The second framework is based on a software component (called Reconciler [16]) that supports the interoperability of different 'coordination mechanisms' in terms of the mutual alignment of their boundary objects and events. The problem of managing mutual alignment of such applications becomes a crucial part of coordination and can be formulated in terms of 'interoperability raised at the semantic level of articulation work', in contrast to current research efforts that aim at the interoperability of applications at the system level. The Reconciler provides a means for managing the tension between the requisite local perspective and the shared meaning required to interpret the boundary objects and intersecting events that characterize inter-group cooperation. This is achieved by recording the conventions users establish to maintain the correspondence between the different definitions and views of the boundary objects and crossing events. The recorded conventions are then used to improve inter-group cooperation in terms of 'naturalization' of concepts and relations ("naturalization means stripping away the contingency of their creation and their situated nature" by taking them for granted ([12] p. 299); 'translation' when possible: that is when contingency has to be maintained but correspondences can be established; or, when impossible, to support the related learning process by highlighting discrepancies and increase mutual awareness of ambiguities and partiality generated by local conventions.

Although not specifically oriented to KM, this experience led us to reflect on how knowledge is created and used in communication

and cooperation. In the light of the claimed integration of supports of cooperative work and KM, we believe that the principles underpinning the above mentioned prototypes could be helpful in avoiding approaches that create a hiatus between KM and everyday activities, between pieces of knowledge that can be 'shared' and pieces of knowledge that make sense only in context, and last but not least in making publicly visible information structures that materialize the relevant knowledge on the work organization and the information resources on which it is based. Specifically, we refer to the coordinative artifacts that in Ariadne are taken as primary category in cooperation and constitute a privileged source of knowledge to enhance sharing and reuse of experience in supporting coordination.

5. DIFFERENT DESCRIPTIONS AS ABSTRACTIONS

In discussing the tension between de-contextualization and (re)-contextualization (Section 3) we mentioned the need of having different descriptions of the same piece of information. This redundancy can be achieved by identifying different aspects. For example, different perspectives have been taken to organize knowledge, and are there illustrated mainly in relation to textual information. There are however sources of knowledge that provide descriptions that are based on objects and relations between them. This is the case when the target organization makes use of applications supporting the coordination of activities, typically workflow systems, which, as constitutive part of their usage, presuppose the definition of causal relations among activities and the record of their actual executions in specific instances. In [15] we discussed the role of these descriptions as a constituent part of CK. The description of processes managed by workflow systems is increasingly based on formal theories that allow for the definition of tools to control process modifications, both at definition and execution time. The point we want to make here is that these tools can be viewed as means to provide the above different descriptions in the specific case of processes, that is, when the knowledge is about how activities are conceived of and combined to achieve a result, and actualized in concrete experiences (instances). In most of the cases, tools controlling process modifications are based on a hierarchical description of the target process, where the links between nodes of the hierarchy are expressed in terms of refinements of behavior (as in classical top-down approaches) preserving behavioral properties. Although this approach produced nice results and applications, it is problematic in respect to its potential application in KM. First, top-down approaches naturally lead to a global view of the system and are not suited to represent the partiality and openness characterizing distributed (workflow) systems. Secondly, intermediate nodes of the hierarchy carry symbolic names that make sense for the community supported by that system but raise the obvious problem when they have to be re-interpreted by members outside the community.

An approach that is more suited to avoid these problems is based on a different interpretation of the hierarchy that we have drawn from [17]). Leaves contain the most detailed description of system components together with the most abstract view of the rest of the system, as perceived from the point of view of the component itself. This abstract view accounts for partial visibility of the 'world' by, and openness of, the component itself. Intermediate nodes contain abstractions of the children ones and, again, of the of the rest of the system in relation to this node. Hence, each node contains different abstractions of the whole system, at different level of detail, up to the root which mainly contains the description of the interface of

the system with the rest of the organization. The second key point is that abstraction is based on the notions of aggregation of states and observability of actions [18].

Leaving aside the technical details (that can be found in [19]), the basic idea is that aggregations are defined by the point of view the observer wants to take on the system. For example, the root does not account for the number and structure of the components but only on the external interface. So, it can be interpreted as the protocol the specific system wants to offer to its environment and therefore expresses the constraints it has to follow or is able to impose when interacting with it. Going down the hierarchy, more detailed information can be obtained about the children of the considered node, while the rest of the system is still at the most abstract level, and plays recursively the role of constraint for those components.

Obviously, the way in which components are organized in a partial order within the hierarchy is 'context dependent', in principle as the 'names' assigned to intermediate nodes in standard refinement. However, unlike refinements, the underlying theory allows for an easy specification, on the fly, of the shape of the hierarchy. In fact, the latter can be constructed starting from the information contained in the leaves (or possibly in any intermediate node if one wants to reduce the level of detail of the most concrete descriptions). It is sufficient to specify which components one wants to see at which level of aggregation. Observability of actions, irrespective of the level of aggregation of the considered components, can be defined by the observer too. Usually the choice falls on actions that represent communications with the environment of the (aggregated) component. However, nothing prevents the observer from choosing different actions and to specify the level of control of the abstraction process, in traversing the hierarchy nodes. For example, if actions are classified according to some criteria (e.g., control actions, communications, check-points, and the like) the observer can just specify the type of actions s/he is interested in, and the resulting abstraction is automatically done (by algorithms that take away the other ones). Abstraction keeps the consistency level specified by the observer too: for example, a strict maintenance of the causality induced by the language of the observed actions or other (theoretically defined) weaker notions of consistency.

Reading the above (very condensed) description of the approach from the point of view of KM in the case of processes, we can say that from knowledge constructed in the normal course of action within the organization one can record some basic knowledge (e.g., the components or some abstraction of them) and then make this knowledge available, at the different levels of detail that make sense to its reader. If, as we advocate, a KM informed definition of the actions used to define the distributed workflow as well as of the hierarchy nodes specifies additional information (referring to explanation, administrative aspects and so on) they can be organized accordingly and shown following the visibility policy defined by the reader. A similar reasoning can be done for executions of processes. Meaningful case histories can be recorded and read in different ways, using the observability approach. This goes in the same direction of what was proposed in [20]. The above presentation sketches what we would like to see as an integration of KM and CSCW.

6. CONCLUSIONS

As mentioned in the foreword, the attempt to reinforce the links between KM and CSCW did not lead us to the identification of solutions. The previous sections discussed issues that can be seen as the starting point of a rich research agenda. The contribution

CSCW can provide to this topic is not only technological but also about the experience that within this area has been accumulated on how people work, both individually and cooperatively, in a smooth way. One of our biggest concerns is about the risk to make KM a fragmented process, because it uses a loosely integrated technology and is conceived of as an asynchronous activity with respect to the other ones. Fragmentation has been recognized as one of the main sources of problems in communication and cooperation. Having infrastructures able to support integrated applications is one of the main research efforts in CSCW as a means to make available to the cooperating actors supports to different modes and means of cooperation, to be used flexibly in different situations. In the view of KM as a fundamentally communication and cooperation process, fragmentation risks to make any effort to achieve it less effective than expected, in relation to the organizational and technological effort put in it. On the other hand, all applications, and CSCW applications among them, should be designed to contain functionality that makes them as a portion of the overall technological support to KM and OM. This implies that the two areas share many research issues where their specialized methodologies and techniques can be used in a combined way. In particular, we see the maximum benefit to solve the conflict between de-contextualization and re-contextualization. In fact, as we have sketched, the latter require enhanced representation capabilities, for both static and dynamic knowledge, and in particular rich and flexible abstractions tools that go beyond the current capabilities. Cooperative KM necessitates the same proactive support as other cooperative applications, to trigger and enhance organizational learning as a fundamental part of KM. In other words, how much the 'pull' approach has to be combined with a 'push' one in dealing with KM? In this framework, in order to achieve the above claimed joint effort, the recent experience gained in dealing with 'awareness' management (which is mainly 'push'-based) needs to be extended to knowledge concerning past experiences. In any case, both awareness information and CK require support for their interpretation in contexts different from where they have been generated. Last but not least, up to now KM is based on the metaphor of a "memory" and focuses the core of the attention on the action of 'remembering'. What about the second action typical of a memory, that is 'forgetting'? In [12] there is a chapter on this subject which is stimulating reflections on the necessity to have tools supporting this too. Our everyday experience is that people remember and forget in a mixed way, at different depths which are functional to improve the effectiveness of their memory. Now, one could ask: how can the technology support this combined action? Which methods and tools are needed to make it feasible? An intriguing interdisciplinary research topic which requires both field research and innovative knowledge representation and management capabilities.

REFERENCES

1. Dieng, R., et al., *Methods and tools for corporate knowledge management*. Human Computer Studies, 1999. **51**(3): p. 567-598.
2. Timmermans, S. and M. Berg, *Standardization in action: achieving universalism and localization in medical protocol*. Social Studies of Science, 1997. **27**: p. 273-305.
3. Schwartz, D.G., *When email meets organizational memories: addressing threats to communication in learning organization*. in [1], p. 599-614.
4. Bannon, L. and K. Kutti, *Shifting Perspectives on Organizational Memory: From storage to Active Remembering*, in *Proceedings of the 29th IEEE HICSS, vol. III, Information Systems - Collaboration Systems and Technology*. 1996, IEEE Computer Society Press: Washington. p. 156-167.
5. Landes, D., K. Schneider, and F. Houdek, *Organizational learning and experience documentation in industrial software projects*. in [1] p. 643-661.
6. Benjamins, V.R., D. Fensel, and S. Decker, *(KA)2: building ontologies for the internet: a mid-term report*. in [1], p. 687-712.
7. Simone, C. and M. Divitini, *The CHAOS project: a coordination support integrating communication contexts*. CSC: the Journal of Collaborative Computing, 1998. **8**(3).
8. Robertson, S. and K. Reese, *A virtual library for building community and sharing knowledge*. in [1], p. 663-685.
9. Heath, C., K. Schmidt, and T. Rodden, ed. *Special Issue on Awareness in CSCW. CSCW: an International Journal*, 2000, Kluwer: .
10. Simone, C. and K. Schmidt. *Taking the distributed nature of cooperative work seriously*. in *6th Euromicro Workshop on PDP'98*. Madrid (Spain), 1998: IEEE Computer Society.
11. Chandrasekaran, B., R. Josephson, and V.R. Benjamins, *What are Ontologies, and Why do we need them?* Intelligent Systems, 1999. Jan/feb 1999: p. 20-26.
12. Bowker, G.C. and S.L. Star, *Sorting things out: classification and its consequences*. 1999, The MIT Press.
13. Dourish, P., J. Lamping, and T. Rodden. *Building bridges: customization and intelligibility in shared category management*. in *ACM-Group99*. 1999. ACM Press.
14. Schmidt, K. and C. Simone, *Coordination Mechanisms: Towards a conceptual foundation for CSCW systems design*. CSCW.: An International Journal, 1996. **5**(2-3).
15. Simone, C. and M. Divitini, *Ariadne: Supporting Coordination Through a Flexible Use of Knowledge on Processes*, in *Information Technology for Knowledge Management*, U.M. Borghoff and R. Pareschi, Editor. 1998, Springer: Berlin-Heidelberg. p. 121-148.
16. Simone, C., G. Mark, and D. Giubbilei. *Interoperability as a means of articulation work*. in *WACC'99*. 1999.: ACM Press.
17. Buchholz, P., *A framework for the hierarchical analysis of discrete dynamic systems*. 1996, University of Dortmund:
18. Milner, R., *A Calculus of Communicating Systems*, LNCS 92. 1980, Springer-Verlag: Berlin.
19. Donatelli, S., M. Sarini, and C. Simone. *Towards a Contextual Information Service supporting adaptability and awareness in CSCW systems*. in *COOP 2000*. 2000. Sophia-Antipolis (Fr):
20. Dourish, P., et al. *Getting some perspectives: using process description to index document history*. in *ACM-Group99*. 1999. ACM Press.

Structuring Organizational Memories using Multi-Dimensional Knowledge Networks

Tang-Ho Lê¹ and Luc Lamontagne²

ABSTRACT. In this paper, we present an approach for the structuring and exploitation of organizational memories. We propose a system to build organizational memories (OMBS) with multiple dimensions, each dimension being defined for a different exploitation mode. An advantage of this OMBS approach resides in the incremental construction of domain knowledge networks including numerous knowledge units and links. We begin by discussing some ideas related to the structuring of an organizational memory (OM) using flight safety as an application domain. Then we describe the purpose, the formalism and the structuring of the knowledge networks. We also propose some directions to exploit the OMBS system along its various dimensions.

1 Introduction

Over the past few years, the construction of organizational memories has generated much interest within academic and industrial communities. Recent progress in interactive information technology (mostly web-related) has provided a technological infrastructure for the implementation of these knowledge repositories. Moreover OM favor the implementation of knowledge management (KM) practices within organizations in order to enable people to "know what they know". Some authors [1] even claim that the construction of an OM should be considered as the first step in the KM cycle.

With different viewpoints being presented in the KM literature, OM are becoming "overworked and confused" (see [2]). Some authors define an OM as "the collective data and resources of a company including project experiences, problem solving expertise, design rationale, etc." (see [3]); others view it as "a repository of knowledge and know-how of a set of individuals working in a particular firm" (see [4]). Even with the latter definition, knowledge is such a vague subject that it is difficult for developers to start the construction of an OM.

There is a lack of conceptual ground on the approaches for the structuring and integration of OM (see [1]). More specifically, proposed methodologies do not offer a compromise between vague structuring guidelines (as extensions of digital libraries) and excessive knowledge formalization (AI flavored approaches). The goal of our work is to experiment with multi-dimensional networks in the structuring and exploitation of knowledge assets and try to determine a well-balanced approach through experimentation with examples from our application domain, flight safety. By multi-dimensional network, we

refer to a directed graph where the nodes represent static and "how-to" domain knowledge and where the links provide guidance on the usage of the network.

In this paper, we report on the approach we followed in the structuring and integration of OM using the knowledge network (KN) formalism. We discuss some of the choices made for the implementation of an OM for our application domain (sections 2-3), the compromise leading to the structuring approach (sections 4-5) and the schemes implemented (section 6-7). Finally we propose some directions on how to pursue this research effort.

2 A KM Perspective of Flight Safety

The Flight Safety program of the Canadian Forces aims at eliminating accidental loss of aviation resources. These measures are essential to preserve vital resources and to maintain operational potential for transportation, emergency management and/or combat purposes.

The program is based on the principle that by effectively disseminating analysis of air incidents, pilots can learn from the experiences of others and hence avoid repeating the same mistakes themselves. Understanding why safety occurrences happen (determine the cause) and develop pilot awareness (correcting their causes and implement preventive measures) are the keys to an effective accident prevention program.

From a KM perspective, the main knowledge assets of the Flight Safety program are the lessons learned from the incident reports and the expertise of the Flight Safety officers. The efforts of the program are mainly concentrated on effective and timely development of the incident reports. The efficiency of the system depends mainly on the quality of the reports (assured by open and honest reporting of the incidents) and their effective dissemination.

The expertise and experience of the Flight Safety officers conducting the analysis of incidents is also a crucial knowledge asset. As the majority of incidents have human root causes, officer's understanding of aviation principles and human factors is of great importance. However, due to their prior training and high qualifications, the program puts less emphasis on managing knowledge practices of the officer's community.

The KM cycle of the program can be described according to the following four (4) steps (see [5]): knowledge development, knowledge preservation, knowledge usage and knowledge dissemination.

Knowledge Development: this involves the gathering of local information by the Flight Safety team, the investigation and analysis of incidents to determine possible causes, the monitoring of new findings of incidents occurring at other units and the estimation of their relevance to local operational characteristics.

¹ Computer Science Department, University of Moncton, Moncton (NB), E1A 3E9, CANADA, email: letangho@UMoncton.CA

² Defence Research Establishment Valcartier, 2459 Blvd Pie-XI north, Val Bélair (Qc) G3J 1X5, CANADA, email: luc.lamontagne@drev.dnd.ca

Knowledge Preservation: the production and storage of reports, news letters, videos and other documents resulting from the knowledge development activities.

Knowledge Usage: to determine corrective actions and to increase pilot awareness by the assimilation of preventive measures to reduce the chances of an occurrence, to provide novice pilot with an access to Flight Safety background knowledge, to spot trends and to determine the magnitude of typical/unusual problems.

Knowledge Dissemination: to communicate findings of new incidents to other organizations and to the appropriate level in the chain of command (e.g. Wing Commander), and to provide advices and training to the personnel.

3 The Construction of an OM

We view an OM as a knowledge system combining domain collections accumulated by an organization and some structured knowledge depicting how the collections can be exploited by its various users (Figure 1).

Typically, organizations have accumulated collections ready to be exploited. For instance, the Flight Safety program has a large collection of reports describing findings of incidents over the last decades, manuals provided to pilots during their aviation training and other material (e.g. videos, news letter, web sites) promoting Flight Safety practices among the pilot community. These collections can be distributed throughout different sites.

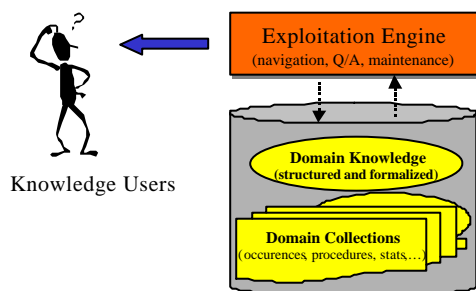


Figure 1. A diagrammatic view of an Organizational Memory

Domain collections being provided, the process of building an OM relies on the choice of scheme to integrate the collections and to exploit them. This experimental process, following either a bottom-up or top-down approach, progressively migrates knowledge from a implicit state (non proven expertise, sometimes tacit to the beholder) to a better structured and formalized formulation (section 4). The construction process implies the selection of organizational knowledge assets to be preserved, the level of structuring/formalization to be reached, and the choice of schemes to exploit the memory. In our Flight Safety case study, the sharing of experiences and the reinforcement of basic safety principles are targeted as the key assets (sections 6-7).

In our approach, we propose a paradigm for structuring domain knowledge and a framework to exploit the domain knowledge in conjunction with domain collections. Our approach relies on 3 aspects:

- To limit our structuring efforts on explicit task-oriented knowledge;
- To incrementally structure knowledge, through informal descriptions of knowledge units (KU). Our goal is to reach a compromise between rigid and formalized knowledge and ill-structured knowledge as often encountered in documents (like frequently asked questions);
- To take into account the exploitation of knowledge during the structuring phase.

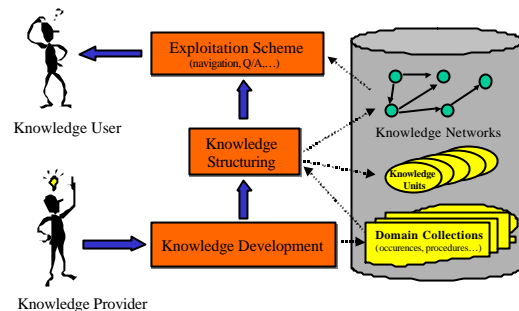


Figure 2. Organizational Memory Building Process

The process we follow to build the OM is a combination of steps to structure the KU, to expand this knowledge along various dimensions and to provide knowledge required to the exploitation schemes. Further details are provided in the next sections.

4 Knowledge Levels

Acquiring and structuring corporate knowledge has proven to be the bottleneck in the design of knowledge systems. To overcome this difficulty and to ease the work of the analyst building the OM, the choice of knowledge structuring/formalization approaches and the type of knowledge to capture are critical.

For each domain, a body of knowledge exists and is maintained in different forms (books, documents, procedures, database, expert systems) and by expert humans or employees. The issue of what knowledge should be considered as candidate for OM can be clarified if one distinguishes the different layers of knowledge existing in an organization. In our work, we classify knowledge according to three (3) different layers: basic knowledge, innovative knowledge and creative knowledge.

To have the abilities to work within a domain, one must learn or be trained, i.e. be familiarized with the first knowledge layer, the basic knowledge of the underlying domain. For example, to fly an aircraft, a pilot must learn the basic knowledge of meteorology, aeronautical navigation and must be trained to control a specific type of aircraft (as by Intelligent Tutoring systems).

Over time, changes occurred in the domain and in the environment a solutions must be devised for new situations. The basic knowledge evolves and more knowledge is available that forces the organization and employees to adapt to their work in order to improve them, keep competitiveness or just to be better. That is the second knowledge layer, the innovative knowledge. For example, all pilots must learn how to use new aircraft

instruments and develop abilities on how to react during critical flying situations.

This innovative knowledge comes from creativity and expertise validated after much experimentation. This knowledge is precious because it contributes to the community's global understanding of their domain. Normally, this kind of knowledge takes time to be formed since tacit knowledge and unproven skills must be leveraged, clarified, formalized or validated. That is the third knowledge layer, the creative knowledge.

From the above distinction of knowledge layers, tools are needed to determine where to apply the innovative knowledge level or on how to stimulate the creation of new knowledge. Also it must overcome basic training knowledge as knowledge can not be reduced to information primitives. To effectively achieve the KM functions, "organizations must create a set of roles and skills to do the work of capturing, distributing, and using knowledge" [6]. We aim to build a tool to facilitate the "active collection and diffusion of knowledge (knowledge pump)" [7], with a special emphasis on the development of the innovative knowledge layer.

5 The Structuring of OM

Some authors (see [8], [9]) propose methods and techniques to build an OM as the starting point of a KM process. Without the distinction of the three knowledge layers as mentioned above, the OM structuring task is difficult. One obstacle is the mass of knowledge for a domain such as Flight Safety being proportional to the many years of college studies, training and pilot experience. Moreover, this kind of basic knowledge is normally well structured and formalized. Some powerful intelligent tutoring systems are already developed to carry out this task. As this basic knowledge has already been specified in the criteria to hire employees, it is not an essential function for an OM system to pursue active development of these knowledge assets. Frequent reinforcement of these principles should be sufficient to ensure adequate operations. For instance, it is fair to assume that most pilots fully understand the effects of weather conditions on navigation. So reminders on a few critical principles (ex: how to avoid convective clouds, appropriate clothing in case of emergency) will bring corrections to observed deficiencies.

The important knowledge to leverage intellectual operations is rather the innovative knowledge. It is the knowledge in evolution that must be captured, formulated and disseminated through an organization. For the Flight Safety domain, this knowledge mainly resides in the findings of incident analysis and in the proposed corrective actions. To do that, we need an OM that describes the actual tasks of the operators (in our case the pilots) as well as the actions and skills required in each specific situation. Employees must be encouraged to review this knowledge and to leverage the tacit knowledge or informal expertise of other employees, based on their work experiences. The task review process must be carried out periodically and occasionally, especially when environmental changes are observed.

Finally the creative knowledge normally appears when there is a problem solving challenge, for instance inductive speculation by observing similar incidents or deductive

speculation by applying some rules/heuristics in a situation. An initial form of creative knowledge may be tacit knowledge. An OM can help to leverage this knowledge by showing to users the relation between tasks or situations, by associating each situation with related actions. Users can then examine and survey many cases and current experiences before finding a new solution. The generation of creative knowledge can be stimulated in several ways in an organization but this activity is different from *knowledge elicitation*, i.e. the process of making clear tacit knowledge. From our point of view, individual tacit knowledge is the kernel of creative knowledge, if not, where would it (tacit knowledge) come from? This point of view is a little different from the one of [10] who emphasis on the conversion of tacit knowledge to explicit knowledge (i.e. the organizational knowledge).

In [11], the authors propose an approach to build up an organizational memory from existing documents to avoid employee's resistance and work disruption. This is an adequate approach to start up the building of an OM; however tools are required to support cognitive analysts to achieve this task.

6 Organizational Memory Building System

An Organizational Memory Building System (OMBS) is a tool to help in the structuring of an organizational memory and in depicting the exploitation schemes. In the OMBS infrastructure we are developing, we offer a framework to describe the KU which can be exploited along various dimensions. Current efforts support the exploitation of three (3) types of knowledge assets: "know-what" (concepts of the domain), "know-how" (internal processes) and "know-who" (knowledge providers, sources of information, relevant agencies, etc...).

6.1 Knowledge Units

In the current implementation of the OMBS, we distinguish two kinds of KU: "Static" and "How-to" KU.

Static KU contain domain concepts, facts and information describing the specifics of situations. For instance cloud conditions or instrument descriptions and settings are represented as static KU. It is presumed in the system design that the only way to learn about static knowledge is to memorize it, no provision being made on how to reason about it.

In "How-To" knowledge are embedded the skills and expertise to be used by an employee in a given situation. This kind of KU is task-oriented and contains the procedure to follow through (actions and/or tasks). It refers to others Static KU when necessary. This knowledge can include both "how to do" and "how to think" descriptions. The frame of a "How-to" KU has the following attributes:

Task name: a descriptor of the nature of the activity. It can also be perceived in some situations as a goal to achieve. From a system point of view, it is the index by which is described the underlying knowledge.

Domain: and ontological description of the sub-domain. For instance engine shutdown procedures would be associated to the AVIATION /NAVIGATION / ARRIVAL sub-domain.

Done by: refers either to the analyst or the knowledge provider (employee) who creates the KN.

Situation: a textual description of the conditions where the how-to unit is applicable.

Actions: a textual description of the expertise and procedures. It may include some primitive actions, some subtasks or both.

Subtasks: a main task can be achieved by carrying out many subtasks; consequently, a hierarchy of tasks can be formed and presented in the system using visual features.

Remark: frequently used to highlight reminders suggested by domain analysts.

Consequences: anticipated states and damages if the "how-to" is applied.

Reference cases: a link to textual documents of the collection, providing explanation of pertaining incidents. We also refer to sources (persons or documents) to accumulate supporting information and evidences.

Demo link: a reference to videos, photos, graphs, diagrams and other pedagogical material. A link allows activation of the multimedia resource.

6.2 Knowledge Networks

The OMBS system makes use of a set of separate KNs. A KN is a directed graph where the nodes are a group of KU, each of them being related to some others by links of different types. For the sake of a better exploitation of the OM, we introduce the notion of KN dimensions. A dimension is a subset of the network that includes nodes related by links with specific meaning and used for a specific purpose. Users can thus exploit the networks along different dimensions according to their goals and intentions.

In the formalism we propose, an organizational memory can be formally defined as (not including the lexical dimension):

$$OM = \{KN_1, KN_2, \dots, KN_n\}$$

$$KN = \{ \text{pedagogical DIMENSION}, \text{organizational DIMENSION}, \text{logical DIMENSION} \}$$

$$\text{DIMENSION} = \{KU_1, KU_2, \dots, KU_i\}, \{LINK_1, LINK_2, \dots, LINK_p\}$$

In the current implementation, we propose four (4) conceptualized dimensions:

1. the **organizational dimension** reflects the work flow between the KU in the underlying organization. The workflow links allow users to examine the works of other employee which are related to the actual KU. By considering this organizational dimension of all KN in an OM, users understand how an organization attains its goals.
2. the **lexical dimension** provides users with explanations on domain terms. The multitude of work-centered terms for an application domain can hinder the understanding of the users of the system. Even for the same organization, terms can have different meanings. Also abbreviations and acronyms, as frequently encountered in the military world, can cause confusion among users. So for each application domain must be prepared a lexicon containing frequently domain terms and abbreviations (as well as acronyms) being frequently used. To some extent, an ontological research is required to come up with a widely accepted glossary. User can refer to the domain terminology through the static KU.

3. the **pedagogical dimension** contains prerequisites KU for understanding the actual KU. The required knowledge is called prerequisite knowledge.

4. the **logical dimension** provides information on the logical relationships between KU (e.g. a KU being deduced from another one). Logical links are to be provided to support description of reasoning schemes. Experimentation is required to determine how the generation of creative knowledge can emerge from problem solving activities.

For each dimension we provide users with links that he can establish between KU. These links allow users to navigate through the network in order to learn and to support further reasoning for problem solving. A local unit index is maintained by the system. KU can be selected and depending on the availability of the links, other units may also be selected and displayed by an appropriate activation.

6.3 Overview of the OMBS Authoring tool

The OMBS Authoring tool provides the analyst with visual functions for the iterative construction of KNs. Given a specific domain, the analyst can use the system to create KU and link them together, with the possibility to modify and update previous descriptions.

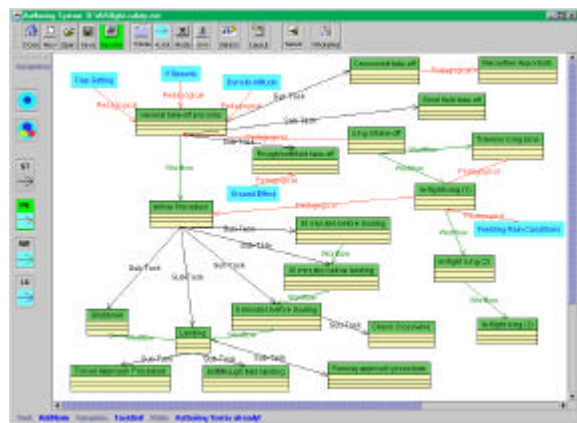


Figure 3. Graphical Interface of the OMBS Authoring Tool.

The figure 3 is an example of a portion of a KN representing some of the *how-to* knowledge to be carried out by a pilot, from departure to arrival at an airport, and their links to other KU. Each time a KN is opened, the virtual workspace of the whole KN is presented in "GlobalView" mode. A link from a Static KU to a How-to KU is normally specified as pedagogical link (prerequisite link) because this knowledge is necessary to understand the How-to KU. If a KU has one or more pedagogical links, clicking on this component will bring the KUs up to the screen. It will be the same system behaviour when user clicks on the organizational or the logical component.

Figure 4 (next page) illustrate an example of KU frame. A KU frame is opened in "Detail View" mode. Users can fill in the various fields to create the KU. If the *Reference-Cases* field points to a document, clicking on it will open the pointed document. If the *Demonstration* field points to a multimedia resource, clicking on it will open the pointed resource. A subtask name is automatically added by the system in the *Subtasks* field whenever the user establishes a subtask link.

7 Exploitation of the Knowledge Networks

The essential functions of a KM system are to improve work-centered tasks, applying the innovation when possible and sharing knowledge between employees. These functions combined with an appropriate compensation policy can motivate employee to leverage creative knowledge. Links between "how-to" KU must be established by the analyst to accommodate different usage (or dimensions). For instance, in our OMBS formalism, an employee can lay his expertise before leaving his job; this expertise will help a novice or a new employee to learn (by using pedagogical links). Other employees can also share knowledge by using organizational links.



Figure 4. Example of a knowledge unit frame (detail view)

The logical links can help employees in problem solving activities by stimulating their reasoning on the underlying situations. If available, demonstration videos, photos, graphs or diagrams can also be linked to each task to make the description more concrete and therefore help the learning process in an effective way.

By referencing to the classification of [1], our system is intended to be knowledge-based and case-based corporate memory. It allows to reason about KU describing experiences and cases already encountered. We are currently in the process of developing exploitation schemes for our system that will allow users to manipulate KNs according to its two capabilities: retrieval of KU and navigation along various dimensions.

The retrieval scheme consists of matching KU of the system with a partial description of what the user intends to search for in a network. This partial description is called "pivot unit". A pivot unit contains the description of what the user intends to search for in a network. By introducing this pivot in the system and do partial matching of the various attributes structuring the units, the user can obtain the units most relevant to the partial description. As most of the attributes contain textual descriptions, statistical (Tf-Idf, n-grams) and semantic (e.g. edge-counting) similarity techniques can be used to exploit the units through the retrieval scheme as described by [12]. Finally the

navigation scheme currently relies in the capabilities of the system to visualize elements of the networks and browse through the networks following various dimensions.

8 Conclusion

In this paper, we presented an approach to build an organizational memory. We distinguish three knowledge layers: the basic knowledge, the innovative knowledge and the creative knowledge. We argue that the last two layers are essential for the KM and that an OMBS is needed to help organization starting up the first task of KM. Next, we set up the system objectives that insist on the ease to use and the supporting of KM functions. The structure of our OMBS is then described with its multi-dimensions and visual interface. Possible exploitation schemes are also discussed. In the near future, we will complete the implementation of the exploitation schemes and expand the system to its fourth dimension, the logical dimension. We believe that this dimension can help user in problem solving activities and for leveraging the creative knowledge. We also foresee the merge of KU and KN in a knowledge space as a mechanism for exploiting KN.

Acknowledgement: Thanks to Ruibiao Guo for implementing the first version of the KN Authoring tool.

REFERENCES

- [1] Dieng, R.; Corby, O.; Giboin, A. and Ribi re, M. [1998]. Methods and Tools for Corporate Knowledge Management. In *Proceedings of KAW'98. Eleventh Workshop on Knowledge Acquisition, Modeling and Management*. Banff, Alberta, Canada.
- [2] Ackerman, M and Halverson, C. [2000]. Reexamining Organizational Memory, *Communication of the ACM*, vol. 41, no 1.
- [3] Nagendra Prasad, M. V. N. and Plaza, E. [1996]. Corporate Memory as Distributed Case Libraries. In *Proceedings of KAW'96*. Gaines, B. and Musen, M. (eds.) Banff, Alberta, Canada.
- [4] Euzenat, J. [1996]. Corporate Memory through Cooperative Creation of Knowledge Bases and Hyper-documents. In *Proceedings of KAW'96*. Gaines, B. and Musen, M. (eds.) Banff, Alberta, Canada.
- [5] MacIntosh, A.; Filby, I. and Tate, A. [1998]. Knowledge Asset Road Maps, in *Proceedings of PAKM98*, Basel, Switzerland, 29-30 Oct. 1998.
- [6] Davenport, T. and Prusak, L. [1998]. *Working Knowledge*. Harvard Business School Press. Boston, Massachusetts.
- [7] Van Heijst, G.; Van der Spek, R. and Kruizinga, E. [1996]. Organizing Corporate Memories. In *Proceedings of KAW'96*. Gaines, B. and Musen, M. (eds.) Banff, Alberta, Canada.
- [8] Spek, R. and Hoog, R. [1998]. *Methods and Techniques for Knowledge Management*. Tutorial document published at *The fourth World Congress on Expert Systems*, Mexico city.
- [9] Wiig, K. M. [1993]. *Knowledge Management Foundations*. Schema Press, Ltd. Arlington, Texas.
- [10] Nonaka, I. and Takeuchi, H. [1995]. *The Knowledge-Creating Company*. Oxford University Press. New York.
- [11] Abecker, A.; Bernardi, A.; Hinkelmann, K.; Kuhn, O. and Sintek, M., [1997]. Towards a Well-Founded Technology for Organizational Memories. In *Artificial Intelligence in Knowledge Management*. Papers from the 1997 AAAI Spring Symposium, Technical Report SS-97-01. AAAI Press.
- [12] Watson, I. P. [1997]. *Applying Case-Based Reasoning - Techniques for Enterprise Systems*, Morgan Kaufmann Publishers, San Francisco, California.