IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4395

A Multi Layer Architecture for Integrated Project Memory and Management Systems

Prof. Dr. Ulrich Frank
Institut für Wirtschaftsinformatik, Universität Koblenz-Landau, Rheinau 1, 56075 Koblenz, Germany, E-Mail: Ulrich.Frank@uni-koblenz.de, Tel.: +49 - 261 - 287 2522, Fax: +49 - 261 - 287 100 2522

Bardo Fraunholz
Institut für Wirtschaftsinformatik, Universität Koblenz-Landau, Rheinau 1, 56075 Koblenz, Germany
E-Mail: Bardo.Fraunholz@uni-koblenz.de, Tel.: +49 - 261 - 287 2536, Fax: +49 - 261 - 287 100 2536

Hanno Schauer
Institut für Wirtschaftsinformatik, Universität Koblenz-Landau, Rheinau 1, 56075 Koblenz, Germany
E-Mail: Hanno.Schauer@uni-koblenz.de, Tel.: +49 - 261 - 287 2533, Fax: +49 - 261 - 287 100 2533

ABSTRACT

For an increasing number of companies, methods and technologies to promote efficient project management are becoming critical success factors. An essential prerequisite of successful project management is the fast availability of proper knowledge. Current project management tools focus on planning and monitoring particular projects. They are not designed to develop a corporate project knowledge base. On the other hand, systems that deal with the representation and dissemination of knowledge lack specific concepts to structure knowledge about projects. To fill this gap, this paper introduces an architecture of a project memory and management system that provides for storing, retrieving and disseminating knowledge about projects. It thereby helps project managers, project workers and others involved with a project to plan and monitor projects as well as to prepare for particular tasks within projects. The architecture features various layers that allow to store, re-use and view knowledge on different levels of abstraction. The layers are based on comprehensive semantic models thereby allowing for powerful queries and instructions.

INTRODUCTION

In a global economy with ever-increasing competition and decreasing product life cycles, project management is becoming more and more important [Gray00]. For companies to stay competitive it is not sufficient to adapt their organisational structure to the needs of project management. In order to increase the productivity of projects, purposeful and thorough management of knowledge is a crucial factor. This is especially due to the fact that on termination all the project members' collaborative knowledge falls apart, its context ceases to exist, hence, project knowledge potentially dangles unproductive or gets lost for the organisation. This is a challenge that recommends a set of measures to be taken: Human resource planning, developing appropriate 'project cultures', and last, but not least, specialized knowledge management that is aimed at collecting, storing and disseminating knowledge gained about projects. There is no doubt that, in addition to social and psychological aspects, information technology - if applied in a sensitive way - can be a very effective driver of successful knowledge management. Current project management tools focus on planning and monitoring particular projects. They are not designed to develop a corporate project knowledge base. On the other hand, systems that deal with the representation and dissemination of knowledge, like decision support systems, management information systems, document management systems, groupware systems or organisational memory systems lack specific concepts to structure knowledge about projects. This situation indicates that there is need for specialized knowledge management systems to support project management – an assumption which is backed by publications that emphasize the demand for software to assist knowledge intensive project work, particular to the consulting industry ([HNT99], [Sarv99]). In this paper we present an architecture of a system we call project memory and management systems (PMMS) that is intended to fill this gap.

The concept of a PMMS was inspired by previous work on enterprise modelling, knowledge management systems (KMS), and a project we currently carry out with a number of small and medium sized enterprises (SME). During the last years we have developed a method for enterprise modelling called MEMO [Fran97]. It is aimed at providing for a set of integrated models each of which covers a particular perspective on an enterprise (like 'strategy', 'organisation', 'information system'). For this purpose, MEMO includes an extensible set of modelling languages, like an object-oriented language to model information (MEMO-OML, [Fran98]) or a language to model organisational structures and business processes (MEMO-OrgML——, for an overview of the modelling languages see [Fran99]). An instantiated enterprise model can be regarded as a repository of corporate knowledge, which can be accessed from different perspectives. It includes all relevant process types, organisational units, resources etc. The MEMO language architecture proved to offer a reasonable conceptual foundation for the design of knowledge management systems [Fran00].

It makes perfect sense to deploy knowledge management technologies in large companies to foster the collection and dissemination of knowledge that would otherwise be inaccessible to most of the employees. But even in small companies appropriate technologies of this kind would be of great help. This at least is an insight we gained from various projects we carry out with SMEs. One of those projects funded by the Foundation for Innovation at the Ministry for Commerce Rheinland-Pfalz is focussing on IT support for project management in SMEs. We found that in SMEs there is a high demand for systems that provide relevant knowledge about projects. This is due to limited human resources, the prominent position of the owner and his intuitive knowledge. For these reasons, one of the project's goals is to develop a prototypical system to support project related knowledge management in SMEs.

To reflect these considerations and findings, a PMMS should provide for storing, retrieving and disseminating knowledge and data about projects. It should thereby support project managers, project workers and others involved with a project to plan and monitor projects as well as to prepare for particular tasks within projects.

REQUIREMENTS

In order to distinguish PMMS from related software types, such as project management software, project management information systems [Mere95] or organisational memory systems, we will develop more specific requirements. Although this work was in part motivated by the insights we gained on SMEs, the following considerations should be valid for large companies, too.

Prerequisite: Extra Value for Users

Numerous people may be involved and/or interested in a project. At first site, the corresponding roles mainly include project managers and project workers. But potential users of a PMMS could also be line managers (who may have to provide resources), suppliers, customers, external consultants etc. All of these roles may range from novice to expert level. For a PMMS to be an attractive alternative to existing software, manuals and training, it is essential that it provides an additional value for its prospective users. To depict this field, we will consider a number of questions and instructions a PMMS should be able to answer faster, better and more reliable than existing alternatives.

- Is there a generic structure that can be applied for any project?
- What are the resource types generally to be taken into account for planning a project or projects of a certain purpose or type?
- Are there any critical success factors?
- What are promising approaches to render projects in an intuitive way?
- What cost categories should be differentiated in general to provide for powerful project accounting/controlling?
- How can a project be integrated with an existing organisational structure?
- What are relevant features of qualification a project manager should have?
- Select all employees who are qualified to manage a certain project type.
- What is the average deviation of cost and time from the original targets in a certain project type?
- Is the company able to complete a particular project within a certain time frame?
- Suggest a team of possibly available employees that would form an efficient team for a certain type of project.
- What are the most frequent reasons for project failures – in our company and related industries in general?
- Is Jim Clark an efficient and reliable team worker?
- What is the current state of project X?
- What are appropriate measures to optimise cost or time in project X?
- Was the customer satisfied with the outcome of project Y?
- What is the relevant knowledge we gained from project Y?

Analysing these questions reveals that we are dealing with different levels of abstraction. Some questions are concerned with projects in general while others focus on special project types or even on peculiarities of a particular project. In addition to answering questions or performing instructions, a PMMS should also foster communication between those involved in a project. Often the members of a project do not share the same professional, educational or cultural background. This is especially the case for projects that involve many companies on an international scale. In order to avoid friction caused by misunderstandings and deviating methods, common standards are required: For organising projects (roles, responsibilities etc.), for project accounting, for structuring and indicating resources. A PMMS should serve as a repository to store and manage the corresponding concepts in a consistent way.

Economic Consideration: Emphasis on Re-use and Adaptability

The questions and instructions demonstrated in the previous section indicate that the content of a PMMS is only in part specific to a particular company. It also includes knowledge, which can be found in textbooks. This relates to general concepts ('What is a project?'), but also to more specialized knowledge about certain project types – for instance about planning software development projects. From an economic point of view it is recommended to equip a PMMS with this generic body of knowledge. This requires an appropriate terminology. One of the difficulties we found while studying project management was the wide interpretability of terms and their definitions. Taking a look at milestones, *Litke* e.g. defines them as the end of a phase with a scheduled result. This milestone is reached only when the given result has been approved by the quality control [Litk95]. The decision of where it is appropriate to have a milestone is left to the manager. Type and number of milestones are to be suitable for the type and risk of the project. In contrast a milestone can also be seen as an easily identifiable key activity or event at the boundary between phases of a project and should coincide with the completion of each package of the WBS [Lock96]. While milestones can also be seen more flexible as a natural, important control point in time, that are easily identifiable by the participants [Gray00]. For this reason, it is essential for a PMMS to introduce an appropriate terminology to store generic knowledge and to guide the consistent collection of further knowledge.

Sometimes the concepts provided by a PMMS will not satisfy individual requirements. Maybe a given project type does not entirely reflect the situation in a particular company – or there is no predefined project type that comes close to a required one. In these cases, a PMMS should allow to define own concepts (like customized project types). To avoid conceptual redundancy, it should be possible to build on existing knowledge – for instance by specializing existing concepts.

Consequence: Need for Multiple Levels of Abstraction

The requirements we have discussed so far suggest that a PMMS has to be based on elaborate concepts that help to structure existing knowledge and guide the user with organizing new insights about project management. In addition to that it should cover different levels of abstraction. Sometimes it will be sufficient to retrieve a description of a project that provides only an outline of the temporal relationships between high-level tasks. In other cases it may be important to get a comprehensive description of every task within the project as well as the required resources. A PMMS should feature different layers to support the user with differentiating these various levels of abstraction and to take advantage of (re-use) relationship between concepts on various levels. In addition to different conceptual levels, the system should satisfy diverse preferences to view concepts – for instance by supporting various diagram types used to render projects.

THE ARCHITECTURE

The following architecture reflects the requirements outlined above. It is inspired by the architecture we used for the implementation of MEMO Center, the tool that accompanies MEMO [Fran94]. The architecture assigns different levels of conceptual abstraction to different layers. In addition to these content layers, it includes a presentation layer. The differentiation of three content layers serves to promote a high level of re-use and flexibility.

The ontology level layer, which one could also call 'generic level layer', represents the highest level of abstraction. It includes an object-oriented reconstruction of basic terms/concepts that are suited to describe projects or knowledge about projects. The domain level layer serves to capture knowledge about specific project types in certain domains. This includes specifications of project structures, the required resources, roles, specific documents etc. Only the project layer level serves to represent particular project instances. While in principle any of the layers may be modified, users should avoid changing the ontology level layer: Modifications to the ontology level layer jeopardize the integrity of the models on the domain level layer.

The presentation level layer includes an extensible set of editors and viewers that allow accessing certain parts of the various content layers. The editors and viewers are assigned to certain types of diagrams and documents. Notice that the architecture puts emphasis on conceptual aspects. It does not take into account aspects like distribution or persistence. Fig. 1 gives an overview of the architecture. Each layer will be described in more detail below.

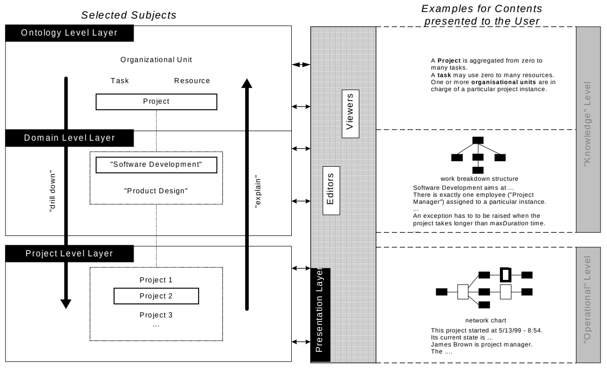

Selected Subjects
Examples for Contents presented to the User
Ontology Level Layer
Organizational Unit
Task
Resource
Project
Domain Level Layer
"Software Development"
"Product Design"
"drill down"
"explain"
Project Level Layer
Project 1
Project 2
Project 3
...
Viewers
Editors
Presentation Layer
A Project is aggregated from zero to many tasks.
A task may use zero to many resources.
One or more organisational units are in charge of a particular project instance.
work breakdown structure
Software Development aims at ...
There is exactly one employee ("Project Manager") assigned to a particular instance.
...
An exception has to to be raised when the project takes longer than maxDuration time.
network chart
This project started at 5/13/99 - 8:54.
Its current state is ...
James Brown is project manager.
The ....
"Knowledge" Level
"Operational" Level

Fig. 1: Overview of the architecture

THE ONTOLOGY LEVEL LAYER

To separate the ontology level layer from the domain level layer, we applied the following rule: Any concepts/propositions that apply to all types of projects should be located on the ontology level layer. Concepts that are specific to a limited set of projects belong to the domain level layer. Unfortunately, there are no theories available about projects that would provide a comprehensive set of propositions valid for all projects. Therefore the content of the ontology level layer is currently restricted to generic definitions of relevant terms, like project, resource, etc. There are two distinct options to model this layer. A meta model would allow to define a *language* to describe project types – including relevant resource types, role types etc. This language would then be used on the domain level layer to define a specific project type as an instance of the meta model. A meta model approach has the advantage to allow for a high level of flexibility. It also supports the construction of (at least syntactically) valid models. While these features would be nice to have, meta models imply a restriction that would cause a severe drawback for our purpose: A meta model specifies how to define a project type (like the UML meta model specifies how to define a class within an object model), but it does not allow to specify features that are common to all project instances. But this is exactly what we want to do according to the rule outlined above: Any feature that applies to all instances of all project types belongs on the ontology level layer. For instance, every single project starts and terminates at a certain time.

The second option to model the ontology level layer is to use abstract classes that constitute a generic body of knowledge about projects. In order to allow for more specific 'memory structures' for certain project types, these abstract classes can be specialized into classes that reside on the domain level layer. Using a specialization instead of an instantiation relationship between the ontology level layer and the domain level layer allows specifying features that are valid for any possible project instance already on the ontology level layer. For this reason we decided on a specialization relationship. Unfortunately this approach has its pitfalls, too. While specializing associated classes (Project and Role in Fig. 2) is a powerful abstraction (it corresponds to the 'Abstract Factory Pattern' in [Gam+95]), it may lead to the notorious covariance problem (for possible solutions see [Meye97]). A further problem relates directly to the purpose of a PMMS, namely storing knowledge about projects. For this purpose it is essential not only to describe particular projects according to a certain conceptual structure, but also to store insights/information about a project type. This, however, is not possible with classes that represent sets of project instances. For example, one could not express a feature like 'averageDuration' of all projects of a certain type. To allow for this important level of abstraction, we enhanced the model with

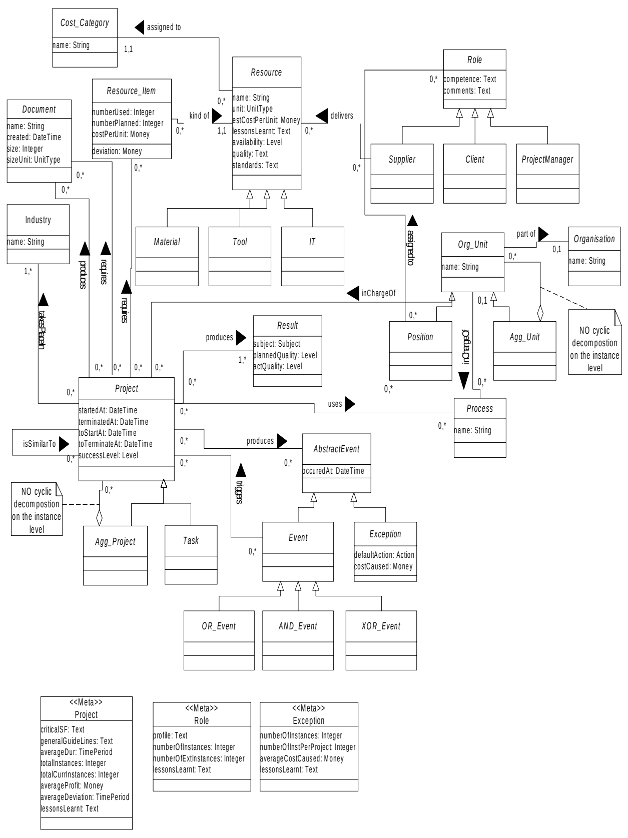


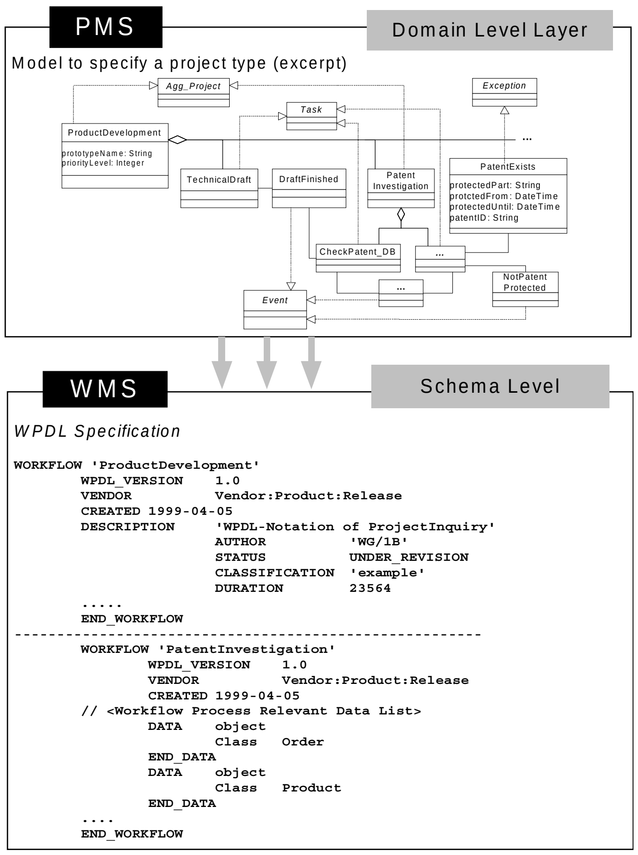
Fig. 2: Excerpt from the object model of the ontology level layer

metaclasses. A metaclass serves to describe features of a class, which would be its only instance. Hence, it is possible to express that 'numberOfInstances' or 'averageDuration' are features of a certain project type (represented by a class). Except for Industry all classes on this layer are abstract classes that need to be refined on the domain level layer. Fig. 2 shows a simplified excerpt of the object model (in UML notation) that is the conceptual foundation of the ontology level layer. It also shows a few metaclasses that correspond to classes with the same name. The instance of a metaclass is initialised on the layer the class belongs to.

The implementation of metaclasses depends on the concepts provided by the implementation level languages. Implementation is convenient with languages like Smalltalk that feature metaclasses. Otherwise one might define special classes with a sole instance that serves to represent features of a class. In addition to that one has to protect the semantic integrity of the relationship between a virtual meta class and the corresponding class.

THE DOMAIN LEVEL LAYER

This layer serves to create, edit and store domain specific knowledge, hence, knowledge about certain types of projects and the related types of resources, roles etc. The domain level layer includes a number of project types. They are specified by specializing the required classes from corresponding abstract classes on the ontology level layer and by instantiating objects from Industry. In addition to predefined project types, the users of a PMMS may add further types by specializing existing classes. For users of a PMMS it can be very helpful to navigate to project types that are similar to a given type. Since we did not find a convincing formal concept of similarity (looking at common features is certainly not enough), we decided to leave it to those who create or maintain the domain level layer. They may associate two project types as being similar.



```
WORKFLOW 'ProductDevelopment'
        WPDL_VERSION    1.0
        VENDOR          Vendor:Product:Release
        CREATED 1999-04-05
        DESCRIPTION     'WPDL-Notation of ProjectInquiry'
                        AUTHOR          'WG/1B'
                        STATUS          UNDER_REVISION
                        CLASSIFICATION  'example'
                        DURATION        23564
        .....
        END_WORKFLOW
-------------------------------------------------------
        WORKFLOW 'PatentInvestigation'
                WPDL_VERSION    1.0
                VENDOR          Vendor:Product:Release
                CREATED 1999-04-05
        // <Workflow Process Relevant Data List>
                DATA    object
                        Class   Order
                END_DATA
                DATA    object
                        Class   Product
                END_DATA
        ....
        END_WORKFLOW
```

Fig. 3: Object model of a project type and corresponding workflow specification (excerpts)

The conceptualisation of projects used on the ontology level layer bears obvious similarity to the conceptualisation of (business) processes. Therefore it seems reasonable to deploy technologies that are used to support business processes, such as workflow management systems, also for projects. That requires identifying common abstractions of projects and processes or – more straightforward – to map concepts used to describe projects to those used for the description of processes. Workflow management systems seem to be an appropriate technology to support project management. The architecture of workflow management systems proposed by the Workflow Management Coalition (WfMC) includes a 'workflow engine' that controls a workflow according to a declarative specification of a corresponding workflow type. Fig. 3 shows an example that illustrates how to define a project type (and the corresponding 'memory' structure) and how to map it to the specification of a workflow type using the Workflow Process Definition Language (WPDL).

THE PROJECT LEVEL LAYER

The project level layer consists of objects that store information about particular projects according to the specification of the classes on the domain level layer. Therefore its main purpose is to support the monitoring of projects. In addition to that it contributes also to the knowledge management function of a PMMS. While we would not regard descriptions of singular projects as general knowledge, they are an important source of knowledge creation. The generalisation of experience gained in singular projects recommends employees with a specialized qualification. For companies that conduct a large number of projects, automated, search for common patterns can provide valuable hints for those employees. The rich structure of the concepts a PMMS is based on providing a promising prerequisite for corresponding inductive procedures. Together with the other content layers, the project level layer fosters learning processes by supporting individual cognitive preferences. A user can access a PMMS either on the project level layer or on a higher level of abstraction. Afterwards he may want to drill down to concrete examples or get a more general view. Provided they are initialised appropriately, the three content layers of a PMMS allow handling all the questions and instructions listed above in a satisfactory way.

THE PRESENTATION LEVEL LAYER

Like any other interactive system a PMMS needs a user interface. To satisfy different perspectives on its content, the presentation level layer includes an extensible set of editors and viewers. They allow to edit or view diagrams of various types (for instance: Work breakdown structures or network diagrams) as well as various types of documents. Usually, common diagram types can be applied to represent project types as well as project instances. Documents can be used for any content level. Additional object models that refer to the corresponding classes on the content layers define the semantics and abstract syntax of a diagram type. The concrete syntax (graphical notation) is in part defined by graphical symbols that can be selected by a user to customize the graphical representation of a diagram type.

While the various users of a PMMS may want to decide for different editors/viewers, they should always see the same state of the system. For this purpose the presentation level layer and the content layers interact according to the model view controller paradigm: The content layers serve as model, while the presentation layer includes controller and view. This ensures that every view (within an editor or viewer) will be transparently informed about any relevant changes of the model's state.

USE AND MAINTENANCE

The benefit a company can gain from a PMMS depends crucially on the quality of the stored knowledge. This recommends the application of special attention to the maintenance of a system's content. The administration effort varies significantly depending on the level of the architecture. Most users will update the system only on the project level layer. The elaborate structure guides the user with entering data in an appropriate way. Hence, there is no apparent need for additional qualification or further education on this level.

Maintenance on the domain level layer requires highly qualified employees. Not only that they should have a deep understanding of project management. Furthermore they need a well-developed ability to find proper abstractions. But qualification may not be enough to give an employee write access to the domain level layer. To ensure a consistent growth of knowledge it may be necessary to restrict administration rights for this level to one administrator only. Obviously, maintenance of the domain level layer may be not an option for many SMEs. However, it is not very likely that there is need for change on this level anyway. Even for those companies with sufficiently qualified employees the question remains whether it is economic to modify the domain level layer or to have an external expert do the job ('make or buy'). In the best case there will be reference specifications of many project types that can be acquired off the shelf or from specialized service providers. As a default, users should not change the ontology level layer. This is only an option, if it makes sense for a (large) company to develop its own project ontology.

The maintenance of a PMMS depends on its use. The more people enter data on the project level layer, the better are the chances to detect useful generalisations for the upper level layers. Also, the more people access the system, the better the quality management of its content – as long as users have a chance to report misconceptions they detected. For this reason, it is of crucial importance to establish effective incentives for users of all layers of a PMMS. Experience with incentives to motivate software re-use or the development of reusable artefacts respectively indicates that quantitative measures (like lines of text entered) tend to be contra-productive. Instead, it should be a sufficient motivation for many people to get credit from those they could help with their input. This can be accomplished by asking a user to rank a particular piece of knowledge that was provided by somebody else.

CONCLUSION AND FUTURE WORK

The architecture we proposed in this paper allows to integrate project planning and monitoring with the management of knowledge about projects. Different from document management systems that are currently used by some companies to store knowledge about projects, a PMMS features concepts on a much higher level of semantics. Thereby it allows to handle queries and instructions that would be out of reach for a text retrieval system. By separating different levels of abstraction, the architecture supports a reasonable organisation of knowledge, fosters re-use of existing generic knowledge and provides guidelines for the maintenance of knowledge. Therefore a PMMS should be valuable source of knowledge for expert and novice project workers.

Our focus is currently on refining the proposed project ontology and expanding its nucleus to an enterprise ontology based on MEMO [Fran97]. Further we will provide for *multi-lingual dictionaries* and support for spatially distributed projects in different languages. At present time the architecture is restricted to conceptual models. However, we plan a prototypical implementation (using Smalltalk or Java) to evaluate the concepts in projects carried out by co-operating SMEs. Within this case study relevant project management content will be exchanged between participating companies by encoding selected classes to XML DTDs and objects to XML documents respectively.

REFERENCES

[Dunc96] Duncan, W. R.: A Guide to the Project Management Body of Knowledge, Newtown Square 1996

[Euze96] Euzenat, J. Corporate Memory Through Cooperative Creation of Knowledge Bases and Hyperdocuments, in Proceedings 10th Banff Workshop on Knowledge Acquisition for Knowledge-Based Systems. SDRG Publications, 1996 pp. 1-18

[Fran94] Frank, U.: MEMO: A Tool Supported Methodology for Analyzing and (Re-) Designing Business Information Systems. In: Ege, R.; Singh, M.; Meyer, B. (Hg.): Technology of Object-Oriented Languages and Systems. Englewood Cliffs 1994, pp. 367-380

[Fran97] Frank, U.: Enriching Object-Oriented Methods with Domain Specific Knowledge: Outline of a Method for Enterprise Modelling. Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 4, Juli 1997

[Fran98] Frank, U.: The Memo Object Modelling Language (MEMO-OML). Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 10, Koblenz 1998

[Fran99] Frank, U.: Memo: Visual Languages for Enterprise Modelling. Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 18, Koblenz 1999

[Fran99] Frank, U.: Applying the MEMO-OML: Guidelines and Examples. Arbeitsberichte des Institut für Wirtschaftsinformatik der Universität Koblenz-Landau. Nr.11, Universität Koblenz-Landau 1999

[Fran00] Frank, U.: Multi-Perspective Enterprise Models as a Conceptual Foundation of Knowledge Management Systems. In: Proceedings of the Hawaii International Conference on System Sciences, Los Alamitos, Ca. 2000

[Gray00] Gray, Clifford F./Larson, Eric W.: Project Management, Newtown Square 2000

[HNT99] Morton T. Hansen, Nitin Nohira, Thomas Thierney: What's Your Strategy for Managing Knowledge? In: Harvard Business Review, March-April 1999, S. 106 – 116

[Litk95] Litke, Hans-D.: Projektmanagement, 3rd Edition, München 1995

[Lock96] Lock, Dennis: Project Management, 6th Edition, Brookfield 1996

[McD99] Richard McDermott: Why Information Technology Inspired But Cannot Deliver Knowledge Management. In: California Management Review. Vol 40 No. 4, p. 103 – 117

[Sarv99] Miklos Sarvay: Knowledge Mangement and Competition in the Consulting Industry. In: California Management Review. Vol 41 No. 2, p. 95 - 107