An On-line SE Repository for Germany's SME –An Experience Report–

Raimund L. Feldmann[1], Markus Pizka [2]

[1]University of Kaiserslautern, AG Software Engineering, Postfach 3049,
D-67653 Kaiserslautern, Germany
r.feldmann@computer.org
[2]Technische Universität München, Institut für Informatik – H5,
D-80290 München, Germany
pizka@in.tum.de

Abstract. In order to keep pace with competitors in the ever accelerating business world, software organizations have to continuously improve their products and processes. However, SMEs typically do not have the time to invest in costly training programs and test the newest technology on their own. Most of the time they need support for their daily work processes, which often employ agile methods. In this paper, we exemplify how SE repositories can be employed to support such settings. A taxonomy for the content of such SE repository systems is given. Based on this taxonomy, we describe the internal repository structure of Germany's ViSEK portal: A portal implemented for offering up-to-date SE knowledge to support SMEs in their daily work.

1 Introduction

According to a recent study on Germany's Software industry [3] many companies use and develop software to improve their products and processes. This includes large companies, such as car manufactures, as well as about 20k small and middle-sized enterprises (SMEs). Common to most of them is the fact that traditionally these companies have not been focusing on software development. Hence, they are often missing experience and need support for their daily work processes. The same holds for some of the software houses that mainly focus on software development.

While larger companies and software houses may have enough time and money to hire experts, invest in costly training programs, and/or test the newest technology on their own, this usually does not hold for SMEs. For the mentioned reasons the need for specific Software Engineering technology support of Germany's industry—especially the large number of SMEs—is seen as a key issue for our economy. Therefore, the Department of Education and Research (bmb+f) of the German federal Government funded the ViSEK project [12]. Based on the idea that gained experience from research and practice should be packaged and easily accessible to all companies, an on-line SE repository is installed in the cause of the project. Similar to other repository systems (e.g., [8], [10]), the ViSEK portal offers access to up-to-date Software Engineering technologies of selected application domains. This should help

to improve best practices and support daily work, independent weather a company uses traditional software development methods, or agile methods.

The idea of a German, web-based Software Engineering (SE) repository is similar to the NFS supported CeBase project [4] in the US. As with in CeBase, the basic idea behind the ViSEK portal is the Experience Factory concept, as suggested by Basili et. al. [2], with its SE repository, the Experience Base. However, while CeBase mainly focuses on supporting people from the areas of research and education, ViSEK's main target group are SMEs that develop software an a daily basis.

In this paper we present the current structure of the internal ViSEK repository, which is the result of several discussions and iterations. The starting points for its definition were the process pattern approach developed by Gnatz et. al. [5] and the repository structure architectural framework developed at the University of Kaiserslautern [6]. The remainder of this paper is organized as follows: Section 2 discusses the question how SE repositories like the ViSEK portal are able to support SMEs and their agile processes. Then, in Section 3, we give a taxonomy for the content of such SE repositories in general, before we describe the content and storage structure of the ViSEK portal in detail (Section 4). Finally, we summarize our experiences and conclude with future directions in Section 5.

2 Repository Support for SMEs with Agile Processes

ViSEK's goal to deliver SE expertise to SMEs creates unique challenges. SMEs are usually not able to invest in costly training or consulting activities. Even if trainings itself are offered at low cost our experience shows that SMEs are still often not willing to release their employers from work for training purposes. Hence, to be successful with the ViSEK SE portal in improving SE practices in SMEs we have to understand and respect the specific constraints and needs of SMEs:

1. SMEs are not able to make significant investments into any a priori learning activities without concrete needs. E.g., the temporary loss of a single developer of a company with 10 employees means immediate loss of >10% manpower for production.
2. SMEs need effective ways to quickly disseminate knowledge and experiences among project members on demand, because they are not able to afford groups of experts that can be deployed occasionally.
3. Nevertheless, there is a strong need and demand for long-term learning, because software SMEs operate in a very competitive market where skill is crucial for long-term success and survival.
4. SMEs must often cope with particularly tight schedules and limited budgets while constantly dealing with new challenges. As a consequence they are often in need for ad hoc (~ instantly created for the purpose at hand) methods and solutions. Consequently, SMEs are particularly interested in acquiring more knowledge related with agile software development.

Thus, long-term learning in SMEs is mandatory but must be performed in a delicate balance with the agility of SMEs and their processes. The information needs of SMEs must itself be considered as being "agile". Information must be accessible with low

overhead and concrete answers to questions in certain problem contexts must be readily available. The ViSEK repository scheme aims at supporting this agile style of information delivery by a light-weight repository structure (Section 4), the careful selection and dedicated preparation of information made available (problems, technologies, experiences), and an adaptive, web-based, and low-cost user-interface.

Usage Scenarios

Based on our observations concerning the particular situation in SMEs our strategy to support learning in SMEs is based on the following major usage scenarios for the ViSEK Internet portal:

1. A project team member quickly needs information concerning a specific technology to perform a task he or she has not performed before, such as estimating costs with a certain model. For this purpose, the ViSEK repository provides brief technical descriptions of important SE techniques.
2. A project team or member is facing a concrete SE problem, such as a high defect rate and is looking for possible solutions. To quickly aid in this common situation, the ViSEK repository provides descriptions of known problems with references to experience reports and possible technical solutions. By matching the concrete problem situation with the problem descriptions provided by ViSEK, the user has a chance to quickly identify his problem and possible solutions.
3. To strengthen the position of his organization in the market a SME manager wants to improve aspects of the SE practices of the organization but does not know the economic impact of different technologies, such as inspections. ViSEK supports SME managers by providing experience reports related with technologies, because SMEs can not afford trying new technologies by themselves.

In addition to these scenarios, SMEs are often forced into some kind of ad hoc software development, as stated above. Some of them have already heard about agile methods but do not know “how to do” XP or other agile techniques and “when [not] to do” it. The ViSEK portal will provide answers to these questions by providing SME experience reports, while relying on links to other web-sites for descriptions of technical details where possible.

3 A Taxonomy for the Content of SE Repositories

As indicated by the scenarios, SE repositories can be employed in different ways to support the daily work of SMEs. To describe the content of such SE repositories more precisely, we make use of the terms *data*, *information*, *knowledge*, and *experience* (see Fig. 1). For our taxonomy, we adopt the definitions of these terms as provided by Aamodt & Nygard [1] and Tautz [11]:

Definition 1: Data are patterns with no meaning; they are input to an interpretation process, i.e., to the initial step of decision making[1].

- ***Definition 2: Information*** is data with meaning that a human or an intelligent information system assigns to this data by means of conventions used in their representation[2].
- ***Definition 3: Knowledge*** is the range of learned information or understanding of a human or an intelligent information system.
- ***Definition 4: Experience*** is gained by humans through utilization of information or knowledge.

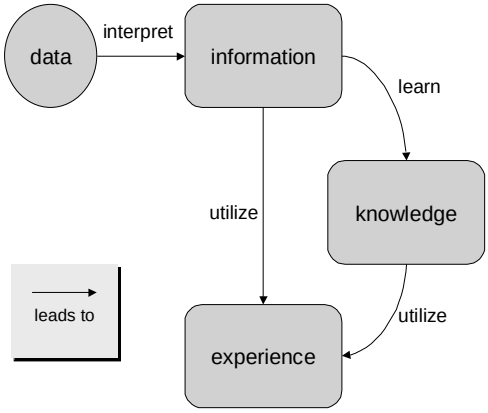

Fig. 1. Relation between data, information, knowledge, and experience

According to Definition 1, all artifacts that occur during the development process of software can be denoted as data, regardless of their quality, quantity, or representation. Examples for data are: (undocumented) code, test cases (i.e., test *data*), requirements documents, or collected measures of effort, errors, or complexity (i.e., measurement *data*). All project related artifacts (i.e., the data) are typically stored in project databases. Since the stored project data are the starting point for creating information, knowledge, or experience by interpretation, learning, or utilization, project databases can be regarded as an integral part of SE repositories.

Creating a project database by following a certain storage structure (i.e., a convention for the data representation) is the first step towards transforming the stored data into information. With a well-defined storage structure, it is possible to automatically create information. Effort distribution models according to the project phases requirements engineering, design, coding, and testing, for instance, can be automatically accumulated from the collected measurement data stored in the project databases. However, information can not always be created automatically. Other information is created by human interactions and has to be integrated into the SE repository manually. Examples for such types of information are: carefully documented C++ or JAVA classes, process and product models gained from former projects that are completely stored in a SE repository, or a collection of (standard) test cases. The common denominator of all this information is the fact that it usually can not and should not be stored as part of the project databases. Hence, storage structures for information form another part of SE repositories.

---

[1] Note that this definition is also similar to the one given by Lehner [9], who describes data as "*raw, unsummarized, and unanalyzed facts*".

[2] Lehner defines information in [9] as "*data processed into a meaningful form*" and adds that "*one person's information can be another's data*". This stresses that the interpretation process from data to information is an individual process depending on the actual human or intelligent information system.

As indicated by Definition 3, knowledge is gained by a learning process based on information. Learning, however, is always an individual (i.e., subjective) process for a human or an intelligent information system, which depends on the actual environment. Based on this environmental setting, the input information is processed and interpreted. Hence, it is important to record data and/or information that describes this environmental setting (i.e., the context) in which the knowledge has been gained. Otherwise, knowledge is not explicit (i.e., applicable) for others in a similarly context, and therefore, would have to be regarded as information again. A process model, for instance, can only be effectively (re-)used when the environment (e.g., the application domain, number of persons in project team, etc.) in which it was already successfully used is *known*. Since we want to avoid mistakes and use the learned knowledge that others have gained before, storage structures for knowledge should be an integral part of SE repositories.

Finally, Definition 4 indicates that the differences between knowledge and experience are:

1. that only humans can gain experience (not an intelligent system), and
2. the fact that experience is utilized.

The second point stresses that one explicitly makes use of the given information or knowledge and *experiences* the results (i.e., one does not only have to believe what others learned or interpreted). According to [11] the validity of experience is bound to the context of the event or activity in which it was gained. This implies that the context in which information or knowledge was utilized has to be stored with the experience, the reason being the fact that when experience is documented, it becomes knowledge for all others who have not experienced it on their own. However, in an SE repository it is possible to differentiate between knowledge and experience. While it is sufficient for knowledge to offer storage structures for the context, the storage structures for experience also have to include the complete environment (i.e., project) where it was gained in.

4 The Internal Repository Schema

With the help of the taxonomy introduced in the previous section, we now describe the main elements of the current storage structure of the ViSEK portal [12]. Our basic considerations for its design can be summarized as follows:

C1: **No data:** Storing pure data would not help ViSEK users in solving their daily problems. This is because they would not be able to compare if the specific item is applicable in their setting (i.e., problem domain) or not. Since the ViSEK consortium does not conduct development projects on its own, it is also not necessary to store complete project databases in the ViSEK portal.

C2: **Knowledge as primary content:** To stress the initial ViSEK goal to provide up-to-date and high quality SE knowledge and to allow goal oriented reuse, we insist of a precise context description for all stored technologies.

C3: **Only selected information:** To provide practical usage examples of described technologies in the ViSEK portal, we allow the storage of *project descriptions* in a compact form. These project descriptions capture the basic information

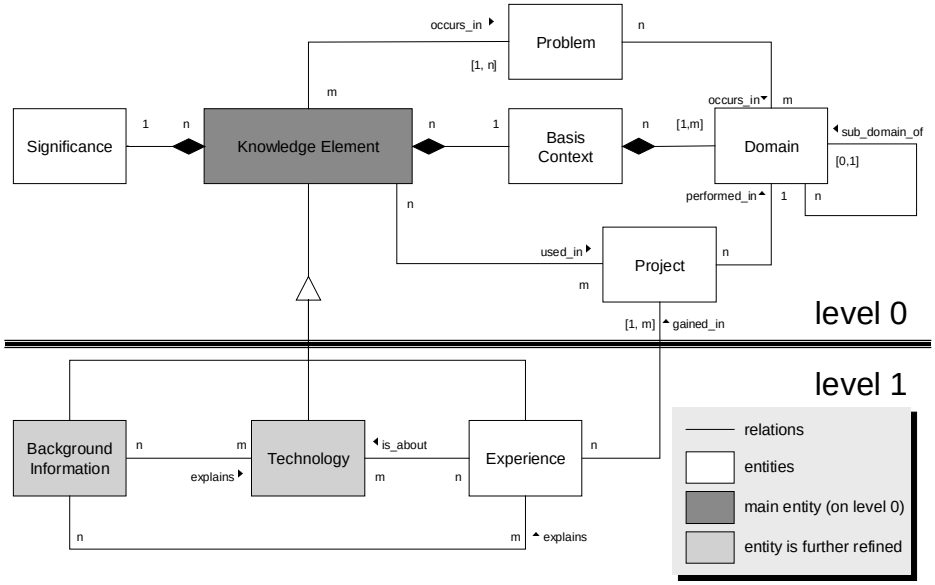

Fig. 2. ViSEK's internal repository schema: levels 0 and 1

about the projects (i.e., consist of accumulated data). Information is further accepted in form of:

- *Glossaries* (i.e., basic definitions independent from a certain context),
- *Literature references* (i.e., standard descriptions of technologies like, for instance, UML), or
- *Contact & expert information* (i.e., similar to literature references, humans or institutions are listed that can be reefer to, if specific questions regarding a certain technology arise).

However, Information that can not be directly related to any of the stored knowledge in the ViSEK portal is not to be stored at all.

C4: **Experiences are represented as lessons learned:** Experiences gained by ViSEK technology providers (i.e., the initial ViSEK consortium or registered users of the ViSEK portal) is documented in form of lessons learned (i.e., knowledge) that include at least one project description (see C3 for details).

In addition to these basic considerations regarding the repository content, we had further design rationales in mind. The most important once can be subsumed as follows:

C5: **Start simple, allow later refinement:** The basic idea was to keep the initial structure simple, but extensible. We wanted a running repository as soon as possible to gain user feedback early in the cause of the project. Based on the experiences gained while filling the initial repository, the underlying schema then should be refinement and/or corrected if necessary.

As depicted in Fig. 2 to Fig. 4, the resulting internal ViSEK schema is hierarchically structured and subdivided into several levels. Starting from the top level (level 0), that contains entities for precise context descriptions of the stored knowledge (compare to

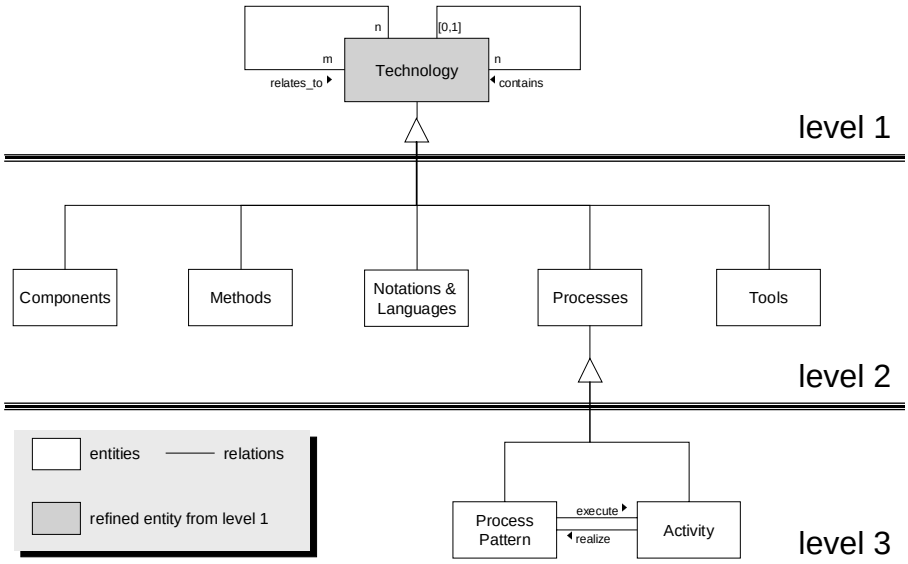

Fig. 3. ViSEK's internal repository schema: level 2 refinement of entity "Technology"

C2), we refine and detail the schema by lower levels holding more specific entities[3]. Consequently, knowledge and experience is represented by a set of entities in the internal ViSEK repository structure. These entities are connected according to predefined relations.

Relations are, if not otherwise mentioned, bi-directional to allow easy navigation in the repository. However, in Fig. 2 to Fig. 4, we only provide the name of relations in one direction to reduce the complexity. For the representation of the schema we use a UML like notation. Cardinalities are expressed by using the numbers *0* and *1*, and the letters *n* and *m*, were *n* and *m* stands for any integer number including zero. Hence, a relation described as 0:n, 1:n, or n:m may or may not exist for a stored knowledge or experience description.

Each instance of an entity is represented by a list of attribute value pairs. Attributes of an entity can be subdivided into the following distinct groups:

1. attributes describing *content* (i.e., technologies, concrete experience, or background information)
2. attributes describing *context* (e.g., application domain or intended user group)
3. attributes describing *significance* (e.g., how useful is a technology description for a concrete user)
4. attributes for *managing the repository* (e.g., user access rights, etc.)

Note that if, in any case, some characteristics can not be represented with the initial set of entities and their attributes, the suggested schema is easily adaptable (compare to C5). This can be achieved by either locally adding attributes to a specific entity, or

[3] Note that not necessarily each and every entity of level *n* must be refined in the next lower level *n+1*.

by introducing further entities and specific relations. Stored knowledge and experience descriptions which do not use the certain entity will not be affected all.

Top level (level 0) entities: These entities for storing descriptions of the "*Basic Context*" (e.g., the intended user group, or legal & technical restrictions), the application "*Domain*", or the "*Significance*" are inherited to all knowledge and experience entries in the ViSEK repository. The "*Problem*" entity allows the storage of problem descriptions for which a "*Knowledge Element*" (i.e., stored knowledge or experience) may be used. In addition, the level 0 entity "*Project*" allows the storage of project information for providing usage examples (compare to C3), or the concrete project in which a lesson learned (i.e., experience) was gained (compare to C4).

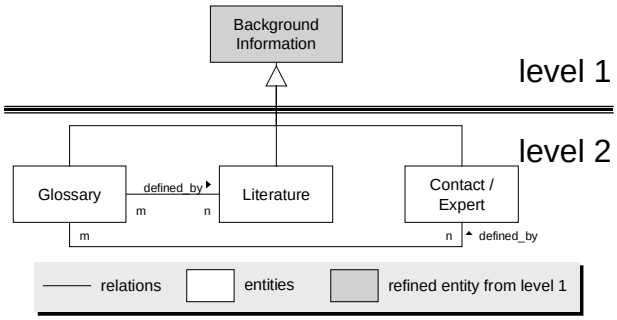

Fig. 4. ViSEK's internal repository schema: level 2 refinement of entity "Background Information"

The Level 1 entities: While entities on level 0 are used to commonly characterize all repository entries, the entities of level 1 are used to classify the stored descriptions. As illustrated in Fig. 2, all of them are directly derived from the level 0 entity "*Knowledge Element*", and therefore, inherit the precise context description. Currently, we differentiate between three different level 1 entities:

1. "*Technology*": This entity is used to store all knowledge descriptions, the main entries of the ViSEK portal (compare to C2). As illustrated in Fig. 3, it is further refined into entities for storing "*Components*" (i.e., code, or patterns), "*Methods*" (e.g., for testing), "*Notations & Languages*" (e.g., UML), "*Processes*" (e.g., How to apply UML), or "*Tools*" (e.g., Rational Rose, or the Gnu C++ compiler). For process descriptions we use the approach suggested by [5]. Therefore, the "*Processes*" entity is further refined on level 3 into an entity for storing "*Process Pattern*" and "*Activity*" descriptions.
2. "*Experience*": This entity is used to store lessons learned, the only form of experience that is stored in the ViSEK portal (compare to C4). Since experience usually is gained with or about a certain technology, a relation between the "*Experience*" and "*Technology*" entity is defined to express about which technology a certain lesson learned was gained. Information about the project in which the lesson learned was "*gained_in*" is stored with the help of the predefined relation.
3. "*Background Information*": This entity is used to store all additional information in the ViSEK portal (compare to C3). As illustrated in Fig. 4, it is further refined into entities for each type of information mentioned in C3. One could argue that

since the entity only stores information, it should be part of level 0. However, we decided that the entity should be placed on level 1, and thereby, can inherit all descriptions from the level 0 entities. This allows in some cases a greater flexibility and an improved quality of the stored information. For instance, it is possible to precisely list the domain(s) for which an expert is listed, or to describe how useful a certain information is for repository users by providing an instance of the entity "*Significance*".

This closes our description of the current structure used for the ViSEK portal. Because of the lake of space, we must referee to [7], for a more detailed description, together with an extended example on how a technology description (i.e., content) is stored using the schema. However, we can state from the first practical experiences while running the repository, that the structure has been accepted by our users. Some adoptions, based on the users experiences, were necessary but could be easily integrated into the storage structure.

5 Conclusion and Future Directions

Although SMEs typically do not have the resources to invest in conventional training programs and for testing new technology, continuous long-term learning is crucial for their survival. The repository structure of Germany's ViSEK portal is targeted towards the specific needs of SMEs by delivering technical information in a goal-oriented way and relating it with practical experiences. The repository structure itself is kept light-weight and leaves flexibility for future extensions and refinements.

The ViSEK Portal was first presented at the CeBit, Hannover in 03/2002. Within 7 days more than 100 visitors demonstrated serious interest in ViSEK and its information services by signing up for a trial membership. We received positive feedback on the concept of an online SE center of competence and the way SE knowledge is going to be presented.

Undoubtedly, our major future challenge is to fill the repository with suitable content. First experiences indicate that the material at hand, i.e. technical reports and research work, is not suitable for the ViSEK repository without change. Significant efforts will be necessary to process this material for publication to SMEs as either a technology description, experience or problem. Furthermore, it seems that this work has to be done by experts of the respective fields, because non-experts seem not to be able to extract and condense the information as needed.

Besides this, we will continuously evaluate the acceptance and the usefulness of the information provided by ViSEK to SMEs as we continue to fill the repository with content that is either requested by SMEs or can be expected to be of use. By this we hope to be able to gain further experience on how to effectively support long-term learning of "agile" SMEs.

Acknowledgements

Part of this work has been conducted in the context of the Sonderforschungsbereich 501 'Development of Large Systems with Generic Methods' (SFB 501) funded by the Deutsche Forschungsgemeinschaft (DFG) and as part of the ViSEK project supported by Department of Education and Research (bmb+f) of the German federal Government.

References

1 A. Aamodt, M. Nygard: Different roles and mutual dependencies of data, information and knowledge – an AI perspective on their integration. In: *Data and Knowledge Engineering*, 16:191–222, 1995.

2 V.R. Basili, G. Caldiera, D. Rombach: Experience Factory; In J.J. Marciniak (ed.), *Encyclopedia of Software Engineering*, vol 1, 469–476; John Wiley & Sons; 1994.

3 M. Broy, S. Hartkopf, K. Kohler, D. Rombach: Germany: Combining Software and Application Competencies. In *IEEE Software*, 18(4), July/August 2001.

4 CeBase: NSF Center for Empirically Based Software Engineering. Homepage @ http://www.cebase.org. visited May 2002.

5 M. Gnatz, F. Marschall, G. Popp, A. Rausch, W. Schwerin: *Modular Process Patterns Supporting an Evolutionary Software Develeopment Process*. In: F. Bomarius, S. Komi-Sirviö (eds.), Product Focused Software Process Improvement. 3rd International Conference, PROFES 2001, Kaiserslautern, Germany, September 2001, Lecture Notes in Computer Science (LNCS) No. 2188, Springer, 2001, 327–340.

6 R.L. Feldmann: On Developing a Repository Structure Tailored for Reuse with Improvement. In: G. Ruhe, F. Bomarius (eds.), Learning Software Organizations: Methodology and Applications, Lecture Notes in Computer Science (LNCS) No. 1756, Springer, 2000, 51–71.

7 R.L. Feldmann, I. John, M. Pizka: ViSEK Repository Schema. ViSEK report 003/E, 2002, Virtuelles Software Engineering Kompetenzzentrum. On-line @ http://visek.de

8 S. Henninger: Supporting the Construction and Evolution of Component Repositories. In: *Proc. of the Eighteenth Int. Conference on Software Engineering*, 279–288. IEEE Computer Society Press, March 1996.

9 F. Lehner: How do Companies Learn? Selected Application from the IT-Sector. Keynote given at the 3rd International LSO Workshop, University of Kaiserslautern, Germany, September 2001. (On-line @ http://wwwagse.informatik.uni-kl.de/LSO2001/Lehner.pdf).

10 M. Lindvall, M. Frey, P. Costa, R. Tesoriero: Lessons Learned about Structuring and Describing Experience for Three Experience Bases. In: K.-D. Althoff, et. al. (eds.), Advances in Learning Software Organizations, Lecture Notes in Computer Science (LNCS) No. 2176, Springer, 2001, 106–119.

11 C. Tautz: Customizing Software Engineering Experience Management Systems to Organizational Needs. PhD thesis, Department of Computer Sciences, University of Kaiserslautern, March 2000.

12 ViSEK: Virtuelles Software Engineering Kompetenzzentrum. Homepage @ http://visek.de, visited May 2002.