An infrastructure for planning, monitoring and reusing capstone projects with the help of semantic wikis

Jean-Hugues Belpois, Vincent Ribaud, Philippe Saliou

An infrastructure for planning, monitoring and reusing capstone projects with the help of semantic wikis

Jean-Hugues Belpois, Vincent Ribaud, Philippe Saliou
University of Brest, Computer Science Department, C.S. 93837, 29238 Brest Cedex 3
{Jean-Hugues.Belpois, Vincent.Ribaud, Philippe.Saliou}@univ-brest.fr

Abstract

The capstone project provides students, working in groups, with a significant project experience. Students should deliver one or several iterations of a software system, along with all artifacts appropriate to the process model they are using. A system based on reference and organizational models and powered by three semantic wikis (using SMW) is used to help the drive of capstone projects. This paper describes aspects raised by challenges of capstone projects management and presents the infrastructure of the solution we built.

1. Introduction

The Software Engineering 2004 Volume [1] states that a capstone project course is essential in a software engineering degree program. The capstone course provides students with the opportunity to undertake a significant software engineering project, in which they will deepen their knowledge of many software engineering areas. At Brest University, the final year of the Software Engineering Master's programme includes a 4-months project, performed by a 6-students team within a virtual company and tutored by an experimented software engineer.

Planning, monitoring and reporting these capstone projects involve different kinds of stakeholders (faculty, coaches, students) communicating and collaborating with the help of office software suites, collaborative tools and distributed systems.

Since 2002, we made several attempts to build a dedicated information system intended to manage the SE master's reference framework, project planning, periodic reporting and assessment support. Requirements were changing over the time and the system suffered of brittleness. Instead of this feature-rich system, we moved to an "as simple as possible" system using semantic wikis.

Section 2 drafts some challenges of the drive of capstone projects, mainly: reuse of parts of past projects, project planning with regularly updates, project monitoring. Section 3 relates wiki (and semantic wiki) use in software engineering, hence providing a description of related work. Section 4 describes the structure of the three semantic wikis used and the inter-wiki links. Section 5 states the technical problems: multilingual issues; copyright, access control and rights management; electronic and real resources management; reasoning capabilities. We conclude with some perspectives and final remarks.

2. Challenges of capstone projects' drive

2.1. Our capstone project structure

In most capstone project courses, students must manage the project themselves, following all appropriate project management techniques. Also, students should be expected to deliver one or several iterations of a software system, along with all artifacts appropriate to the process model they had selected [1].

But, in our system, neither are the students managing the project, nor may they choose the process model and appropriate artifacts to be delivered. The process reference model is adapted and simplified from ISO/IEC 12207; we are using 13 processes organized with 3 process groups: Development Engineering, Project Management and Development Support. A list of processes is given at the beginning of section 4.

We use a Y-shaped life cycle that separates resolution of technical issues from resolution of feature issues [15]. First, the cycle is divided into two branches (tracks): a functional track and a technical track. Then these two tracks amalgamate for the realization of the system.

The Y-shaped life cycle is temporally organized in 9 stages of 2 weeks each. Each stage is divided in several scenes that carry on activities belonging to the three

process groups. Each stage is named from the main activity occurring at this stage. The temporal organization is: Stage 0: Warm-up; Stage 1: Project set-up; Stage 2: Requirement capture; Stage 3: Technical architecture; Stage 4: Requirement analysis; Stage 5: Design; Stage 6: Realization; Stage 7: Tests - Integration; Stage 8: Verification &Validation.

In software studios or capstone projects, the instructor's role is more of a mentor, or, better a coach [18] . We are very closed to Tomayko's work and most observations pointed out in his "Software Development Studio 5-year Retrospective" apply to our system: "The use of a well-established development process, a matrix organization, and one-to-one mentoring give the highest return on investment [18]". However, our system adds several dimensions to the coach's role because organizing and supervising the stage/scene cycle is devoted to the coach together with the individual monitoring of each student's competency development.

2.2. A model of small software project

The two latter authors practised software engineering as project managers for nearly ten years in a small department of a software company. The organization was typical of a Very Small Enterprise (VSE): one manager acting sometimes as a project manager, three software project managers, between 6 and 12 software developers. The only difference with a VSE was the existence of a corporate baseline – but far unknown of most department employees.

In such a small organization, there are two views of projects that we may call "external" and "internal". The external view sees a project as a black box: starting and stopping dates, inputs and outputs work products, resources needed or provided, cost, milestone … Some of its elements are represented in the upper-level of figure 1. The internal view has to see the project as a white box and deals with all aspects related to the 5 W + 2 H: Who, What, When, Where and Why plus How and How much. This view is generally structured with the help of a WBS (Working Breakdown Structure: "*a deliverable-oriented hierarchical decomposition of the work to be executed by the project team to accomplish the project objectives and create the required deliverables. It organizes and defines the total scope of the project*" [5]. Elements of this internal view are represented in the lower-level of figure 1.

This paper presents a set of interlinked semantic wikis intended to sustain the progress of capstone projects. A wiki is basically a set of pages interconnected with links and a semantic wiki allows us to give different meanings to links and to value a set of properties associated with each page (typically called metadata).

Our capstone project is intended to simulate as far as possible a small software project. According to the simplified model presented at the beginning of this section, we may distinguish two levels of concerns, each associated with a set of stakeholders.

The external level may be called the "programme level" and main stakeholders are academic staff. Its concerns are two-fold: about Master programme objectives, structure and assessment and about programme activation with a set of different projects, each of them associated with its proper processes, stages, people, tools and obviously the statement of work of the project. Creating and updating information at this level is under the responsibility of the academic staff of coaches. Searching, retrieving and consulting information is for everyone, but the main interest is for project team mates.

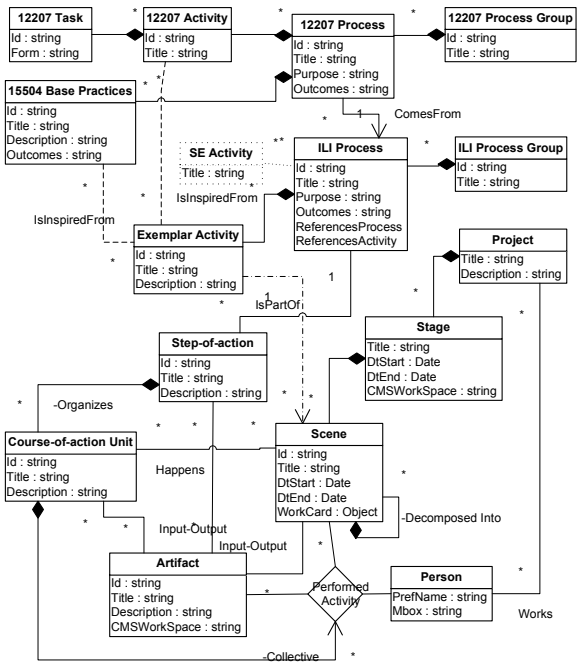

Figure 1. The upper half depicts reference models (issued from 12207 and 15504 standards) and the external view of a project. The lower half represents the internal view of an enacted project. Input and output work product (artifact) are associated to the scene, the activity and the process. Students reports on the course of their activity through a structure of steps of course-of-action units.

The internal level may be called the "enactment level" whom main stakeholders are the project team

(students) and its coach (faculty). This level is concerned with the project execution, its day-to-day life, inputs, outputs, events and processes. Creating a first shape of information at this level is under the responsibility of the team coach because it corresponds to the information that may be found in an initial WBS. Then information of this level are created, updated, retrieved and used by the whole software team.

The rest of this section presents challenges and important outcomes in reuse, planning and monitoring together with requirements for supporting these practices.

2.3. Reuse of parts of past capstone projects

B. Meyer states that "*a long-term project should involve the reuse, understanding, modification, and extension of existing software. The best way to achieve this goal is to imagine the project running over several years, with each new class taking over the result of the preceding one and developing it further*" [10].

In order to achieve this goal, coaches have to establish a reuse policy and to systematically exploit reuse opportunities. As an example, an important feature for the coach is that a new group can redo part of a preceding work and that previous results can be used in order to help them.

An asset is an item of interest that is stored in a reuse library or any other unit of information of potential value to a re-user. An Asset Management Process should establish an asset classification scheme, provide an asset storage and retrieval and record asset changes [3, p.78]. Keeping in mind the scope of our system (helping the progress of capstone projects), we state that our reuse challenges are about classifying, recording, locating, and sharing assets.

2.4. Capstone project planning

In our system, before a capstone project starts, the coach has to define the new project in which the students will be immersed; to prepare the logistics necessary for the smooth running of the service; to tailor the learning process in order to fit with project's constraints and learning objectives.

In capstone projects, the coach aims to achieve learning objectives rather than project objectives. In our system, the coach establishes a general planning of the development cycle and provides a structured breakdown of work in small units. Each unit comprises: the definition of work to be undertaken, the expected products, and pedagogical resources (e.g., guidelines, examples). This WBS is similar to a real WBS but also different inasmuch as it incorporates needs induced by our learning system. Once the project is running, the whole team has to update the structure with the enacted events and this activity is information management rather project management. Challenges are about the activities people perform in order to produce, organize, maintain, retrieve and use information items.

2.5. Capstone project monitoring

The Merriam-Webster Online Dictionary defines the verb *monitor* as "to watch, keep track of, or check usually for a special purpose". What we want first is to monitor – to gather information and to report it. Reporting may serve various purposes that are, for the moment, out of scope of this paper.

Capstone project management is primarily concerned with learning objectives and specific constraints such as students' abilities development or coaches' help that have to scaffold students' activity when required and gradually decrease help (un-scaffold). Hence, the monitoring should be focused on these topics. The best source of information comes from the students themselves reporting on what they did and why. Thus, monitoring is primarily concerned with features that help to capture, secure, distribute and archive information.

3. Wiki use in software engineering

In this section, we relate wiki (and semantic wiki) use in software engineering. Rauschmayer reports that wikis are generally used for: collecting data or knowledge; coordination, planning, project management; web site, light-weight content-management system; document editing; discussion, forum; whiteboard [13]. In the field of software engineering, wikis use helps to manage software projects, software lifecycle documentation, code reference information, bug tracking, and supports communication around the project members.

Regarding challenges of section 2 and usages listed above, we relate several uses of wikis with our objectives: reuse is addressed through documentation management, planning through publication-oriented management and monitoring through experience management. In order to facilitate the reach of these challenges, we propose a very simple architecture based on the use of several inter-linked semantic wikis. The use of a Content Management System (CMS) was envisaged but was not necessary.

Lanier argued that "*software as we know it is a brittle substance. It breaks before it bends*" [7]. The antidote to brittle software systems might be to start from (and keep) the simpler option. Maxwell [9] pointed out that "*Our experience with and reflection on using wiki as a platform suggests that there is much to be gained from an approach which builds up from simple foundations rather than attempting to customize an already-complex architecture* [9]". We agree with this point of view and we recommend starting a content-based project with wikis, "*a system which makes almost no expectations and which poses almost no restrictions on its users* [9]".

Reuse. Challenges of using wikis to tackle reuse in software projects are related in [14] with recording, reusing, locating and sharing information. Difficulties related to locating reusable components which satisfy a given requirement, assessing their relevance, and adapting them to the current task are still high. We agree with [11] which believe "that the retrieval difficulty is related to the crucial problem of interaction between component providers and users". As related in §2.3, coaches wish primarily to reuse previous documentation artifacts and we propose to use documentation management principles as a basis to facilitate this reuse process. Documentation management aims to establish standards and requirements for documents, to identify, develop, check, distribute and maintain documents [4, Documentation Base Practices, p. 50].

Wikis are often used for software documentation management. Strengths come from built-in mechanisms to maintain control over documentation artifacts. When there is a need for an editorial workflow, a CMS may be preferable. But in our case, we need to maintain a Web-based remote repository, to manage versions of documents with an access to previous versions, to define relationships between various documents, to have flexibility to adapt the structure and relationships after the course of projects. The two former points are achieved with the use of traditional wikis, the two latter requires adding semantic to wikis. As an example, students may have to perform a retro-design of a piece of software built in previous projects and should have an access to the successive versions of the code documentation (linked to the code in a configuration tool). This is a traditional wiki use that can be enhanced with semantic annotations such as authors, dates, notes about different versions and semantic links between a given version and the version of the requirements document used. Semantic wikis combine a static classification based on the internal structure of the repository and a dynamic classification when it is needed. It means, for example, that coaches may be able to re-build a "knowledge path" through existing artifacts, highlighting with the help of semantic properties the important information useful for new students.

Planning. As mentioned in §2.4, project planning is intended to prepare the execution of the project. It shall contain description of associated activities and tasks and identification of the software products (artifacts) that will be provided (or needed) [3, Project Planning Process, p. 32]. These descriptions and artifacts have to be kept up-to-date as the project goes along. These issues are closed to those of a publication-oriented system. As we wish to keep this kind of publication system "as simple as possible" and to rely on wikis rather on a dedicated (and probably complex) system, we believe as mentioned in [9] that "*the strategy of building larger-scale editorial structures on the thinnest of technological foundations seems to also mean that the work of organizing and managing the content in the system can remain an editorial process, rather than a technical administration task*". In other words, it means that the organization of content, design of site navigation, and the ongoing management of contributions and changes should be handled by a staff of authors (students and coach), not by a technical one. We have to define the editorial process: what information has to be delivered or gathered and reported, when it should happen and which role is assigned to different information management tasks. The wiki literature has a humoristic folklore about the roles played in a wiki, but at least three roles are indispensable: *gardener* – one who spends time attending to the quality, flow, and overall polish of content on a wiki; *champion* – able to generate interest, give the training, monitor growth and fix problems; *maintainer* – a person assigned or self-assigned to a page, space or section of a wiki who accountably takes responsibility for the quality of some of the content [17] , [22]. Coaches act as champions, students naturally as maintainers and gardener's role should be specifically entrusted to team members. For example, at the beginning of each stage (2 weeks), the coach creates a set of wikis pages corresponding to the envisaged work scenes and estimated attributes such as workload, resources and link pages to assigned students' pages. At the stage end, students update pages, attributes and links to reflect the enacted stage. A gardener is assigned to each stage in order to keep it as sound as possible.

Monitoring. According to [12], "*Capstone projects have shown to support self-directed and experiential learning, where students reflect and interpret their experiences to build abstractions […], which are applied and tested in new situations and which provide the foundation for having new experiences*".

Reflective skills [16] are skills to be learned by the students, in addition to the technical skills of software engineering. In [12], Ras and al. present SOP, an adapted Wiki for information and experience management in software projects. SOP serves as a means to capture observations and share all kinds of information relevant to the project such as roles and process descriptions, documentation templates and guidelines, observations and experiences on software engineering (SE) technologies, etc. This kind of information is recorded at the programme level (cf. § 2.2) but we wish to enhance this experience management with the recording of day-to-day events of projects. We believe that only a preliminary structure can be fixed at the beginning of the project and the structure of the content, navigation paths and contributions management will – and should – change during the project, incrementally and often. Wiki content may be not reflect a sound and complete system but it corresponds to the life of a project and provides the required simplicity and flexibility. This experience gathering is intended to stimulate a reflective thought but discussion about this goal is out the scope of this paper. Interested readers may refer to [12], [16], [20].

4. Semantic wikis for capstone projects

4.1. A simple model of capstone project

Before the description of the role of wikis and the inter-wiki links, we need to explain how the learning process of our capstone project fits with the project organization.

From the 43 processes of ISO/IEC 12207:2008, we concentrate on those related to the software development cycle, that is: 6.2.2 Infrastructure Management, 6.3.1 Project Planning, 6.4.1 Stakeholder Requirements Definition, 6.4.4 Implementation Process replaced by 7.1.1 Software (SW) Implementation Process and its 6 sub-processes, 7.2.1 SW Documentation Management, 7.2.2 SW Configuration Management, 7.2.3 SW Quality Assurance, 7.2.4 and 7.2.5 SW Verification & Validation.

Each process above can be analyzed from different points of view: expected knowledge and skills; stakeholders' roles; input and output deliverables; required tools and resources. At work, what makes sense for these multiple point of views is their articulation within the activity situation (the cohesion of the work situation). In an apprenticeship project, it is also the (learning) situation which leads to an understanding of the multi-dimensional nature of activity. The capstone project is a theatre play where the different actors learn to play their roles. We carefully designed a set of learning scenes which are the reference context where part of the play happens. The scene aims at a unity of place, time and action; the scene is together a situation where students learn and do, a scenario of actions, a role distribution, an area mobilizing resources and means. As an example, consider the "Design tailoring" scene, students have to write a usage guide from the design activity as it is proposed in the Unified Process, then retro-engineering the code of a project performed last year in order to produce a design document for this project.

Several scenes happen simultaneously. During the same period of time, students work in subgroups on different scenes belonging to different processes. As stated at the beginning of section 2, the complete cycle of scenes is temporally organized into stages. As an example, let us see what generally happen during the stage 5 "Design". Typical scenes are listed in table 1 with the name of the scene in column 1, the process where the activity fits in column 2, the reference to the 12207 process in column 3. In order to materialize the group process dimension, the table 1 is divided in 3 sub-tables – one per group process.

Scene	**Process**	**12207**
Group process : *Project Management*		
Configuration tool set-up	SW configuration management	7.2.2
Scene	**Process**	**12207**
Group process : *Development Engineering*		
Requirements update	Requirements capture	7.1.2
Design tailoring	Design	7.1.4
Database design SW Design	Design Design	7.1.4 7.1.4
Software Test Plan Elaboration	Integration-Qualification	7.1.7
Scene	**Process**	**12207**
Group process : *Development Support*		
System and networking support	System and networking support	6.2.2
Modelling and development rules	Methods and tools support	7.3.1

Table 1. Typical scenes at stage 5.

4.2. 12207 – life cycle processes wiki

A process reference model, issued from ISO/IEC 12207 standard [3] is represented in the *12207* wiki and provides the description of software life cycle processes, activities and tasks for software used as a stand-alone entity, or an embedded or integral part of the total system. The processes in the 12207 standard form a comprehensive set. An organization, depending on its purpose, can select an appropriate subset to fulfill that purpose. The 12207 was used and filled for the first time this academic year. This wiki (http://oysterz.univ-brest.fr/12207) is a hypertext reference of the ISO/IEC 12207:2008 for the process level: title, purpose, list of outcomes and process decomposition in activities and tasks; it may be reused "as it is" for the next projects.

4.3. 24765 – vocabulary wiki

The 24765 wiki (http://oysterz.univ-brest.fr/24765) contains a glossary of technical terms used in the capstone projects. This wiki was pre-populated with the terms used in the ISO/IEC 12207 (56 terms), and then augmented with terms used in the Software Engineering Master Programme and inside capstone projects. The 24765 standard provides a descriptive vocabulary in English of the terms as they are currently defined in approved software and systems engineering standards of ISO/IEC SC7 (sub-committee for software engineering) and the IEEE Computer Society. The descriptive vocabulary may contain several definitions of the same term as found in the existing standards. This wiki is actually under reengineering but ISO provides on-line Software Engineering Vocabulary (SEVOCAB) at http://pascal.computer.org/sev_display.

4.4. Capstone projects wiki

In our capstone projects, there are two hierarchical decompositions that are linked; the former is an activity model adapted from the ISO/IEC 12207: process group, process, activity and task; the latter is an organizational model: project, stage, scene, and step. A scene is related to a single process but aims to mobilize several practices of the process. Conversely, a practice can be undertaken within several scenes of the same process.

The third wiki is used to manage information on project progress. We can distinguish several levels in the structure concerning different stakeholders:

- An upper-level structure of group process / process (reference framework) and project / stage (year's organization), under the control of the programme manager.
- A mid-level structure acting as a simple but realistic model of a project: breakdown of the project stages into work scenes; membership of project and allocation of persons to scenes; expected inputs and outputs. Each coach of a capstone project fills this structure with instances (wiki pages) corresponding to his/her project plan and has to update it regularly.
- The lower-level structure deals with the activity performed by project's members (students in this case): who, what, when, and how they did the tasks required by the work scene together with delivered inputs and outputs; this reporting is written by each member at the end of each stage (typically each 2 weeks).

Many inter-wiki links facilitate navigation and information retrieval.

5. Technical problems

This section presents technical problems raised by challenges of capstone projects management, listed in section 2.

We use Semantic MediaWiki (SMW) [7], an extension to MediaWiki. Semantic MediaWiki enables wiki-users to semantically annotate wiki pages. Thanks to semantic annotations (metadata), the wiki contents can be browsed, searched, and reused in novel ways [7]. Every semantic annotation within SMW, such as for instance Vince Turtle is author of the Requirement document, is a RDF triple - of the form <subject, predicate, object> (<Vince Turtle, author, Requirement document>) - connecting through a property (here the property author) the subject's page (here, the page related to the user Vince Turtle) to another page (here the page depicting and providing access to the Requirement document) or a data value (for instance, a date). In MediaWiki, categories classify articles according to their content; hence SMW categories are considered as OWL classes.

5.1. Multilingual issues

Multilingual issues are required in 12207 and 24765 wikis. Standard are usually published in English, and translations are built on an identical structure (chapter, section …) than the English version.

In the ISO/IEC 12207, parts of standards are identified by hierarchical identifiers (e.g. 7.1.4 – Design) and an identifier (together with a language code) can be used to identify the same part in different

translations (e.g. 7.1.4/fr – *Conception* – in French). MediaWiki provides a language management extension that is used to map parallel structures between translations.

In the case of the 24765 vocabulary, we follow the approach prone by IFLA [21] of a symmetrical thesaurus. All different language versions of a multilingual thesaurus are identical and symmetrical; each descriptor has one and only one equivalent in every language and is related in the same way to other descriptors in the given language. The 24765 wiki is structured with the skos vocabulary [20] and the transitive property skos:exactMatch is used to link each descriptor in every language to the English descriptor.

5.2. Copyright, access control and rights management

MediaWiki is not designed to be a CMS. To the contrary, it was designed to be as open as possible but it supports enough protection of private content for our purposes.

- Wiki 12207 reproduces parts of an ISO/IEC standard that is subject to copyright. The copy licensed to the University of Brest does not allow us to distribute any part of the standard outside of the University and access to this wiki needs to be restricted to coaches and students (registered users).
- For the ISO/IEC 24765, the primary tool for maintaining this vocabulary is a database that is modified in a controlled fashion. Hosted by the IEEE Computer Society, the SEVOCAB (Systems and software engineering vocabulary) database is publicly accessible at www.computer.org/sevocab. ISO/IEC 24765 is issued periodically as a formal, published International Standard reflecting a "snapshot" of the database. The copyright notice provided with the database permits users to copy definitions from the database as long as the source of the definition is cited [5]. Hence, Wiki 24765 can reproduce a subset of the vocabulary with its source and let access to anyone.
- The wiki dedicated to project management has three levels composed of several spaces of articles with different requirements for access control and rights management. We use namespace to separate the different spaces. The programme manager, each coach and each group of students is associated with its own namespace. Pages are stored within a given namespace and access right is controlled for each namespace and each group through the use of a MediaWiki extension.

5.3. Electronic and real resources management

MediaWiki does not provide a good way to handle download and upload of electronic resources. It is possible to upload them using the image uploading feature, but it has limits. Moreover, software project artifacts have different types (documents, web pages, code files …) and it requires a dedicated management system. We use the file system to handle project artifacts excepted for code files where we use the Subversion software as a version control system. This allows to link wiki pages with URL referencing folders in the file system or spaces handled in Subversion, providing required version management. However, it may be not sufficient to refer electronic resources by URL because additional information on the resource is needed (such as the language, the format …). Moreover, we may refer to real resources such as books or persons. Using URN to identify resources gives us the ability to annotate resources with additional metadata. Hence, each resource is associated with a wiki page and the URL of that page is used as an URI for the resource. This associated page works as a metadata record and domain-independent metadata are controlled by the Dublin Core metadata terms vocabulary (http://dublincore.org/documents/dcmi-terms). Other types of resource (e.g. a Person) may use popular vocabularies over the internet (e.g. foaf). It requires a vocabulary management inside the semantic wikis and abilities to import and export RDF, features that SMW provide.

5.4. Reasoning capabilities

One way to organize the organic growth of Wiki content is to add structure by enriching Wiki-pages with additional metadata. It is often enough to conceive the proposed “semantic approach” as extend your Wiki with RDF [3]. Providing RDF metadata requires additional effort and the main benefit is an application of reasoning mechanisms. Except for sub-properties, RDF reasoning capabilities in SMW are poor and three crucial features are missing: inverse properties (e.g. dc:hasPart/dc:isPartOf), symmetric properties (e.g. foaf:knows), transitive properties (e.g. skos:exactMatch). The main problem is that these kinds of properties violate the Wiki principle of locality: changes made to a page content are not allowed to affect other pages’ content. But, as these kinds of properties involve two pages, annotating a page should lead to annotate the other page consistently. We solve this problem by using an annotation in only one of the two pages and using inline queries in the other one. For instance, activities (requirement gathering, requirement elicitation ...) are belonging to a single process

(requirement process), we link with a property each activity to its corresponding process and we use a query retrieving all activity for a given value of process to build the inverse property. Semantic MediaWiki includes a simple RDF query language for semantic search, so that users can directly request certain information from the wiki. So it is possible to rebuild the other part of inverse, symmetric or transitive properties by using queries (it needs to republish the page in order to force the system to compute up-to-date values).

6. Conclusion

Three semantic wikis are used as an infrastructure intended to facilitate the drive of capstone projects: planning, monitoring and reuse. Our capstone project uses a 13-process reference model adapted of the ISO/IEC 12207 standard. The project organization is decomposed in stages and scenes and follows a Y-shaped lifecycle. Based on this reference model and this organization, our wiki-based infrastructure addresses reuse through documentation management, planning through publication-oriented management and monitoring through experience management. An important requirement is to keep the technical solution 'as simple as possible'. Solid structure and simple architecture let the system be easy-to-use and easy-to-evolve.

7. Acknowledgement

The authors wish to thank the anonymous referees for the constructive and helpful comments that led to restructure part of the manuscript and to improve the final result.

8. References

[1] ACM and IEEE, *Software Engineering 2004*, http://sites.computer.org/ccse (last accessed February, 2008).

[2] B. Decker, E. Ras, J. Rech, B. Klein, and C. Hoecht, "Self-organized Reuse of Software Engineering Knowledge Supported by Semantic Wikis", *IESE-Report, 115.05/E*, Fraunhofer Institute, Kaiserslautern, 2005.

[3] ISO/IEC 12207:2008, "*Information technology -- Software life cycle processes*", International Organization for Standardization (ISO), Geneva, 2008.

[4] ISO/IEC 15504:2004, "*Information technology -- Process assessment*". International Organization for Standardization (ISO), Geneva, 2004.

[5] ISO/IEC FCD 24765, "*Systems and software engineering -- Vocabulary*", International Organization for Standardization (ISO), Geneva, 2008.

[6] M. Krötzsch, D. Vrandecic, M. Völkel, H. Haller, and R. Studer, "Semantic Wikipedia", *Journal of Web Semantics 5/2007*, Elsevier, Amsterdam, 2007, pp. 251–261.

[7] J. Lanier, "The Gory Antigora: Illusions of Capitalism and Computers", Cato Unbound, http://www.catounbound. org/2006/01/09/jaron-lanier/the-gory-antigora/ , 2006

[8] M. e Maqsood, T. Javed, "Practicum in software project management: an endeavor to effective and pragmatic software project management education", *Proceedings of the the 6th joint meeting of ESEC/FSE on The foundations of software engineering*, ACM, New York, 2007, pp.471-479

[9] J. W. Maxwell, "Using Wiki as a Multi-Mode Publishing Platform", *Proceedings of the 25th annual ACM international conference on Design of communication*, ACM, New York, 2001, pp.196-200

[10] B. Meyer, "Software Engineering in the Academy", *IEEE Computer, 34*, IEEE, Piscataway NJ, 2001, pp. 28- 35.

[11] P. Ramadour, C. Cauvet, "An Ontology-based Support for Asset Design and Reuse", *ENC'08 Mexican International Conference on Computer Science*, Mexico, 2008

[12] E. Ras, R. Carbon, B. Decker, J. Rech, "Experience Management Wikis for Reflective Practice in Software Capstone Projects", *IEEE Transactions on Education, Volume 50, Issue, 4*, IEEE, Piscataway, 2007, pp. 312-320

[13] A. Rauschmayer, "Next-Generation Wikis: What Users Expect; How RDF Helps", *Third Semantic Wiki Workshop. at ESWC*, Redaktion Sun SITE, Aachen, 2009, poster.

[14] J. Rech, C. Bogner, and V. Haas, "Using Wikis to Tackle Reuse in Software Projects", *IEEE Software, Volume 24, Issue 6*, IEEE, Piscataway, Nov.-Dec. 2007, pp.99-104.

[15] P. Roques, F. Vallée, *UML en action*, Eyrolles, Paris, 2002

[16] D. Schön, *Educating the Reflective Practitioner*, Jossey-Bass, San Francisco, 1987.

[17] N. Serakiotou and al., "(Wiki + ResTechs) = (Fresh documentation + Organic Knowledge Management + Training Materials + Good, Cheap Technical Writers)", *Proceedings of the 36th ACM SIGUCCS conference on User services conference*, ACM, New York, 2008, pp.173-179

[18] J. E. Tomayko, "Carnegie Mellon's software development studio: a five year retrospective", *Proceedings of the 9th Conference on Software Engineering Education*, IEEE, Piscataway NJ, 1996, pp.119-129.

[19] J.E. Tomayko and O. Hazzan,, “Reflection processes in the teaching and learning of human aspects of software engineering”, in Proceedings of 17th Conference on Software Engineering Education and Training, New York: IEEE Press, pp. 32- 38, 2004

[20] W3C, *SKOS Simple Knowledge Organization System Reference*, ERCIM, Sophia-Antipolis, 2008.

[21] Working Group on Guidelines for Multilingual Thesauri, *Guidelines for Multilingual Thesauri*, IFLA, The Hague, 2005.

[22] Wiki patterns, http://www.wikipatterns.com/ (last accessed January 29th, 2009).