D6.3

Knowledge Model: Project Knowledge Management

Project title:	**Knowledge in a Wiki**
Project acronym:	KIWI
Project number:	ICT-2007.4.2-211932
Project instrument:	EU FP7 Small and Medium-Scale Focused Research Project (STREP)
Project thematic priority:	Information and Communication Technologies (ICT)
Document type:	D (deliverable)
Nature of document:	R (report)
Dissemination level:	PU (public)
Document number:	ICT211932/AAU/D6.3/R/PU
Responsible editors:	Peter Dolog
Contributing authors:	Peter Dolog (AAU), Fred Durao (AAU), Daniel Grolin (Logica), Karsten Jahn (AAU), Peter Axel Nielsen (AAU), Andreas Munk-Madsen (AAU), Keld Pedersen (Logica)
Reviewers:	Henry Story (Sun Microsystems), Jakub Kotowski (LMU)
Contributing participants:	AAU, Logica
Contributing workpackages:	WP6
Contractual delivery:	28 February 2009
Actual delivery:	1 April 2009

Abstract:

This document provides the basic ideas behind the selected features and the overall design of the knowledge model for software project management.

Keyword List:

Project Management, Knowledge Management,

Project funded by the European Commission within the Seventh Framework Programme © KIWI 2008

Table of Contents

Foreword: Purpose of this Report

The Knowledge model for project management serves several goals:

- Introducing relevant concepts of project management area for software development (Section 1).
- Reviewing and understanding the real case requirements from the industrial perspective. (Section 2).
- Giving some preliminary suggestions for usage in KIWI system (Sections 3).

This document is intended for technological partners to understand how for example the software development concepts can be applied to a semantic wiki framework.

1. Introduction

Software development is a highly collaborative activity that is usually performed by one or more teams, in a longer period of time, and in several stages. In every stage, team members need to share knowledge about actions they have performed, solutions to the problems they have introduced, and so on. Adoption of tools and techniques intended to support project development requires knowledge and skills so that these tools can save time of personnel and reduce costs during the software development process. Software applications supposed to assist members in the organization must comprise knowledge about *designing management and development processes*, *planning and monitoring projects*, *managing requirements*, *quality assurance*, *configuration management*, *and measurement and analysis* 7. This knowledge is necessary to overcome traditional obstacles evidenced over the years in software development environments such as lack of collaboration, technical deficiency, and communication problems. In addition, usually it is tricky to find all the relevant information when solving a specific task, sometimes even not knowing that relevant knowledge exists 7. Successful software development projects require commitments and intense collaboration of all parties involved so that tasks can be realized in parallel and synchronized with the project expectations.

We have reviewed a representative work in knowledge modelling for software project management in the KIWI deliverable D5.3 7. We have pointed to the work dealing with CMMI such as an ontology modelling CMMI processes and practice 7 or with focus on quality assurance 7. We have also pointed to several other ontologies dealing with some partial aspects of software processes. We have identified, that the main problem is that most of the analyzed literature does not point to the concrete electronic sources of the proposed OWL ontologies so we cannot use them directly with the one exception for the CMMI ontology mentioned above. The literature and the ontologies however inspired us in building the ontology in this report.

Wiki systems appear as a viable solution to support collaborative work since its main feature is to provide collaborative space for knowledge sharing and wide participation. **Wiki** is a new kind of collaborative system, which has widely spread since its introduction to the work group of “design patterns” by Ward Cunningham in 1995. There are many different wikis available these days with many different features and functionality. Even the scopes of them are very different. If Wikis are systematically utilized in enterprise environments, they can support the organization members and stakeholders to follow the process by designing process plans, assigning tasks to users, recommending previously solved problems with lessons learned, and so on 7.

In this report we explore the knowledge areas of project management and suggest applications which can be supported by KiWi. In section 2, an overview about knowledge areas for software project management is introduced. Section 3 presents proposed knowledge model for software project management and describes the classes created in the ontology and finally Section 4 shows applications which can be supported by KiWi.

2. Knowledge Areas for Software Project Management

The software project management practice requires knowledge and skills necessary for project organisation, planning, and control to ensure the timely and cost-effective production of all the end-products, to maintain acceptable standards of quality, and to achieve (for the enterprise) the benefit for which the investment in the project has been made. The overall knowledge however is spread over distinct areas such as *requirement management, project planning, process management, quality assurance, configuration management and*

measurement and analysis. Not incidentally, these areas correspond to the project management related work process areas of software development. The software development practice and technical knowledge of each area is crucial to the achievement of the goals expected during the software project management. The assumption is that if the knowledge about how to perform certain project management and development activities is shared among enterprise employees, the higher quality of software end product will be achieved.

2.1.Requirement Management

Gathering and managing requirements are important challenges in project management. Understanding of requirements is crucial to succeed with any projects. Many projects failed due to poorly understood requirements and their management throughout the project lifecycle. The continuously evolving baseline of requirements needs to be managed effectively. Project managers need to assess and understand the uniqueness of the requirements gathering process for their individual projects. In particular, requirement management is concerned with validating, approving, controlling, tracking and changing requirements. Requirements matrix is a concept used to maintain traceability relations between the requirements and their realization in one or more work products (e.g., relations between requirements, who are responsible customer representatives and project participants for particular entries, which test or review validates the requirements, which program structure implements the requirements, and so on) 7.

2.2.Project Planning

Project planning is a discipline for predicting and estimating how a project will be accomplished with certain resources available (such as personnel, timeframe, budget). It describes in detail what are the resources and the commitments to be achieved by all the involved participants and stakeholders. The project planning process in a project results in a project plan. The project plan shows the list of tasks, deliverables, and deadlines each of those. In particular, the name of individuals and organisations with a leading role in the project are associated with the correspondent activities. Estimations and methods utilized in each activity planned are established as well 7.

In project planning, each team member identifies and ranks risks associated with the records management initiative. Examples of risk include employee willingness to adapt to changes associated with new processes, the IT infrastructure created by new records or document management software, relationships between outsourced vendor services, complexity of the initiative, and availability of staff to participate. The risks are analysed and documented in a risk management plan. Risk is observed and managed according to s risk management process throughout the life of the project.

2.3.Process Management

Process management is concerned with the management of all knowledge items that constitutes a process description to be used by the development projects. Each process in software development is described by a model to serve as a guideline for a project team member who will be assigned to perform it. A well designed process assists organization to effectively solve complex organization tasks, communication problems, activity instantiation and execution and closure and how that should be documented 7.

Each activity envisioned and executed in every project can follow such a process model. Process managers require considerable participation and feedback from the project managers and software developers working in the projects to document deviations of executed activities from the process model. The deviations serve for updates of the processes for the whole enterprise.

2.4.Configuration Management

Configuration Management (CM) involves the development and application of procedures and standards to manage an evolving software product. CM may be seen as part of a more general quality management process. When released to CM, software systems are sometimes called baselines as they are a starting point for further development 7. Configuration management deals usually with changes requested to software and associated work products requested either by a customer or due to identified bug or issue.

2.5.Quality assurance

Quality assurance and control ensures that work performed by the project team will meet the quality requirements defined within the vision and scope statements. Quality assurance deals with testing processes and defects and issues in software products as well as processes. This process also guides the project teams in defining performance and defect metrics that can be tracked throughout the life of the project. A feedback loop is established that enables continuous improvement of the project deliverables the team is producing. For example, if the project team is developing an educational module that will be used to train employees on the basics of records management, then a quality consideration would be to pilot the training with a sampling of employees. This pilot could include validating that training objectives have been achieved, that the length of the training was adequate and that the delivery mechanism was effective 7.

2.6.Measurement and Analysis

Measurement is critical for the successful management of software projects. It is the basis of project planning because it is used to establish achievable project targets. Measurement helps to monitor the progress of a project. The project manager has to utilize techniques for sizing software, understand and interpret software metrics for use in measuring and comparing productivity and quality. The data on the status of activities, resource utilization, and the technical quality of the work done is required as an input for this activity. The data is compared with the plans to measure the deviations from the plan and to identify the areas that need corrective actions. Effective estimation of software development effort, productivity, schedule, and cost early in the development life-cycle is necessary for proper project planning and risk management. It prevents the project from lack of resources and eventual deviations from the early planned activities. The numbers collected by measurement of software project activities show a progress made by all participants involved 7.

2.7.Methodology for Ontology Development

The ontology has been created as a result of analysis of work processes in Logica. Of course CMMI as a generic standard helped in there but there are several specifics which are always different from company to company. Each work process creates one or more work products. The products are stored in a format relevant to the work produced, for example word document for requirements document, source code for a program module, issue for a requirements change or a bug, Microsoft project plan for plans, and so on. We analyzed those products as well as the common attributes necessary to describe them which are always company specific. The work product types resulted in classes and attributes in properties of the ontology. Furthermore, there are some dependencies for example between requirements change and a plan change. These have been further reflected in the ontology as attributes.

3. KIWI application for Knowledge Management Problems in Logica

The KIWI application should help reducing the following problems in Logica:

1. Lack of collaboration and knowledge sharing during the design of processes (new management and development processes) – It's very difficult to align all the various requirements to the processes, difficult for people to contribute, and difficult to reach acceptable and useable results. Therefore, it seems, that collaborative process guidelines editing tool such as KIWI could be useful for the phase when the knowledge about the processes is just emerging.
2. Large gap between process related knowledge and the actual process use as it unfolds in real projects – The process guidelines are very often too abstract and generic as they need to cover all the instances of development and management processes in an organization. It's often very difficult to exploit general process related knowledge in a practical project context. Even though a substantial amount of resources is used to create general process related knowledge to support project managers when doing their job, the actual exploitation and practical use of this knowledge is too limited. Therefore, it seems that the linking and recommendation of the right knowledge item as planned in the KIWI system could be of interest.
3. Not sufficient awareness about particular knowledge items – it is very often difficult to find all the relevant information when solving a specific task. Sometimes people don't even know that relevant knowledge exists. Therefore, it seems that the KIWI system can improve this aspect by providing recommendation to knowledge items relevant for a problem solving activity.
4. Lack of support for navigation in relations between various kinds of knowledge in project management – project management tasks involve significant portion of decision making. Decisions need input on various relations between evidence collected throughout the whole lifecycle of the project. There's too little support when dealing with a large number of complex relationships between the various kinds of project management knowledge. Therefore, it seems that KIWI system could help with its support for navigating between linked content items.
5. Isolated islands of information – Too much time is used manually moving and transforming information around different medias and programs, and between structured (e.g. risk lists in spreadsheets) and unstructured (e.g. emails) formats. KIWI system has a potential to unify access to the information.

Based on above analysis, the goals for the Logica KIWI applications are:

1. *Collaborative process design* – The application should make it easier for all the stakeholders to get involved and contribute with their knowledge when management and development processes are designed. The goal is to get better processes, to make sure that they comply with the various requirements, and to make it easier to implement processes because they have been designed in a collaborative way.
2. *Process deployment and implementation* – When designed it should be easier to use a specific process, than not using the process – it should be instantly available on the project managers or developers desktop, easily integrated into project plans and supported by tools and knowledge resources. It should also be quite easy to tailor processes to specific projects characteristics.
3. *Information pushing* – Information is not something you should look for when in trouble – when you start on a task all the relevant information should be presented for you in the

right order: If you don't get it automatically, you don't have to look for it, because it's probably not there.

4. *Complexity management* – Project managers should not used time or intellectual resources on checking relationships among various kinds of project management data (e.g. relationships between risks and plans), the system should do it.
5. *Information integration* – It should be easy to move back and forth between using data in unstructured and structured formats (e.g. planning notes or meeting minutes and the structured activity plan for a project), and information should be easy to reuse and relate across the various project management domains (e.g. if a defect/issue is registered (quality assurance) it shouldn't be necessary to manually create a task (planning) to deal with the defect.

Basic idea behind the KIWI application for Logica is to close the gap between wiki systems and ERP like systems, and use the advantages of the two different ways of dealing with knowledge. The scenarios described later share the common property: working with information, or knowledge, that gradually becomes more and more structured, and moves from wiki pages with little or no structure, to very structured forms were information can be manipulated using all the known technologies from the enterprise resource planning.

The KIWI application for Logica should support the members in the organization in:

- Designing management and development processes (describing, negotiating, and agreeing on ways and organization is working).
- Planning and monitoring projects (e.g., conducting a risk analysis, observing the performance of programming such as lines of codes, observing number of changes, and so on).
- Managing requirements (e.g., documenting and sharing information about requirements, and changes to them, and so on).
- Quality assurance (e.g., arranging and conducting reviews, tests and audits, and so on).
- Configuration management (e.g., baselining products and controlling changes).
- Measurement and analysis (e.g., collecting, analyzing and discussing metrics).

Some of the process areas may be omitted due to time or resource constraints.
The primary users are:

- Process engineers that participates in designing management and developing processes in the organization (e.g., a group of project managers and project management experts that design a new process for planning development projects or a new process for doing risk analysis).
- Project managers and systems developers that use these processes when managing development projects or doing systems development in the organization.

There are other kinds of users, but these two groups are the most important. The application contains information – structured as well as unstructured – about key concepts related to:

- Project management (e.g., plans, activities, estimates, risks, status, and so on).
- Requirement management (e.g., requirements, change requests, and so on).
- Quality assurance (e.g., reviews, defects, and so on).
- Configuration management (e.g., products, baselines, change control boards, and so on).
- Measurement and analysis (e.g., metrics, analysis results, and so on).
- Process management (e.g., the process descriptions, checklists, templates that support the other processes, and so on).

All these concepts are characterized by constant evolution until they are preserved as refined structured information. From the perspective of an ordinary project manager or systems

developer the basic application will consist of three parts:

1. *Process guidance* – this part of the application contains access to information about how to do a specific project management or development task, e.g., how to do a risk analysis. It also contains information about a specific instance of the process (e.g., the specific activities needed when performing risk analysis in a concrete situation). It can be perceived as a simple, non-rigid workflow (e.g., identify all risks, categorize them, evaluate likelihood and possible impact, and so on), rules that must be followed (e.g., all risk with a specific combination of likelihood and impact must be reviewed with management), access to various knowledge resources (e.g., a checklist with common risks, wiki pages with risk analysis best practices, risk analysis done in the past, and so on).
2. *Notes and work in progress or descriptions of a final work product* – this part consists of ordinary wiki pages were all the un-structured, such as, notes, discussions, viewpoints, comments, and ideas concerning the current task, e.g., doing risk analysis, are noted. For instance, during a risk analysis meeting, different stakeholders meet to evaluate the risks related to an upcoming project and come up with ideas for dealing with these risks which are noted in the associated wiki page and perhaps labeled with additional tags or annotations. This part of the application can be perceived as a "virtual" blackboard or a shared note book. The part can also be used to describe and document a final work product.
3. *Work products structured information* – The final work products are based on the information documented in the part of the application called "Notes and work in progress". These work products represent the structured part of the final result – e.g., the "final" risk analysis in a structured format. A functionality for transition from unstructured to structured information is very desirable.

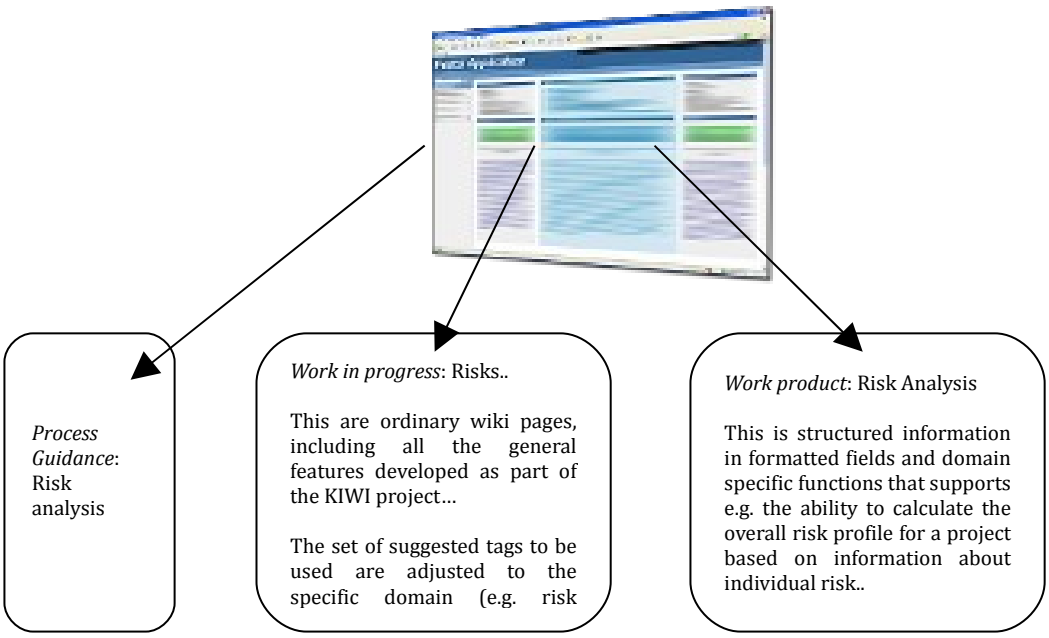


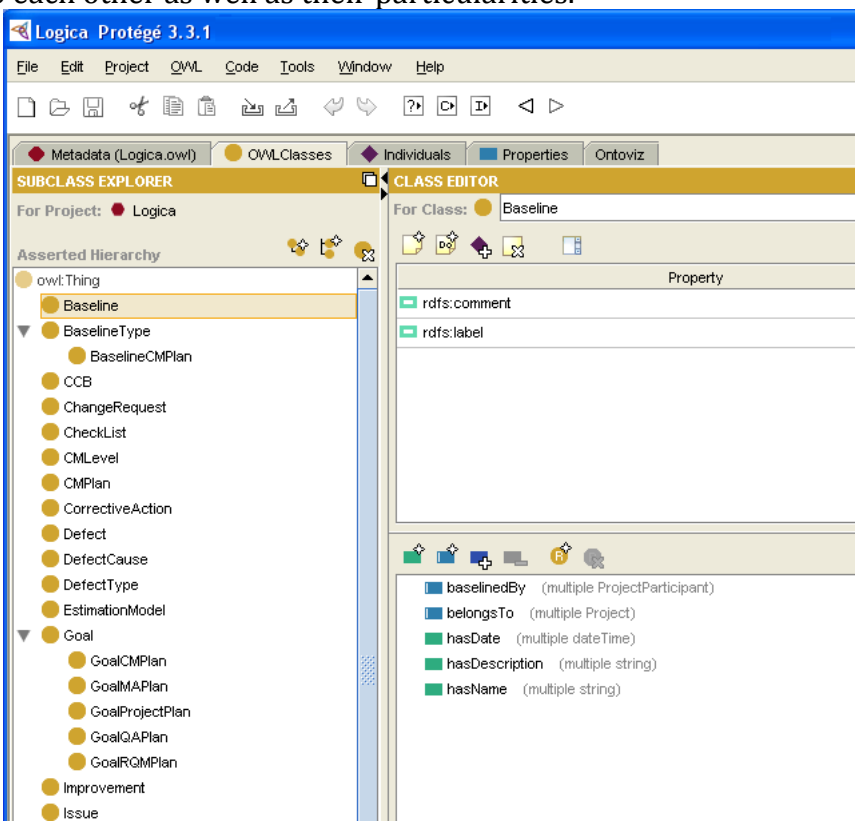
Fig. 1: A prototypical sketch for KIWI application for Logica

Fig. 1 depicts a prototypical sketch for KIWI application for Logica sketching the three aforementioned parts. Let us make an example of use. When a specific process has been selected in the left part, the wiki page in the middle (the Work in progress part) is automatically customized to this specific purpose, e.g. by adjusting the set of readily available tags and annotations to risk related concepts or by formatting the page in a way suited for the specific purpose, and the last part of the application is automatically changed to a structured form that accept, validates, calculates, stores structured information, for example about risks.

The “Work in progress” and the “Work product” part is also related. One of the goals is to support the transition between the structured and the unstructured information. Based on the tagging or more formal annotations of content in the “Work in progress part” and a shared knowledge model, the transition of information from “Work in progress” to the “Work product part” is partially automatic. When a specific function on the structured part (e.g., “Import risk analysis notes”) is activated, the tagged or annotated information in the wiki part is placed in the related form fields in the “Work product part”. The two different “information containers” are interlinked. This links are utilized to configure right content in all three parts when user navigates through the information preserved as content items.

4. Knowledge Model for Software Project Management

We have already mentioned the knowledge model above when talking about different user tasks. We have created an OWL ontology based on the above assumption for a KIWI application which will be used as a shared knowledge model in Logica application. The ontology was created with the purpose of representing the software project management and development scenarios. The semantic model can be used later by software applications which use such knowledge as input for their tasks. The ontology illustrates how different entities communicate to each other as well as their particularities.



The ontology was build with *Protégé*[1], an ontology editor tool which provides facilities for creating classes as entities and establishing relationship between them though semantic properties. The current version of the ontology comprises 57 classes with 103 properties and can be reached at https://svn.salzburgresearch.at/svn/kiwi/KiWi/trunk/extensions/ontologies/resources/logica/logica.owl. The most important entities of the ontology are described in this document, and a more detailed overview can be accessed at

[1] http://protege.stanford.edu/

http://iwis.cs.aau.dk/KIWIproject/Logica_Knowledge_Model/. The proposed ontology still can be enhanced by inheriting concepts from FOAF ontology, which describes the characteristics (name, full name, email, etc) and relationships amongst friends of friends, and their friends, and the stories that they tell. The idea of using FOAF vocabulary is to provide the social network support which is usual in software companies. In the following, the knowledge concepts in the proposed ontology are grouped according to the knowledge areas described above.

4.1.Requirement Management

For requirement management the classes created in the ontology are:
User Requirements – This class describes the user requirements, define the properties to determine their priority, complexity, relation to other requirements, uncertainty, cost to implement them, possible risks, status, and pre-conditions for their instantiation.
Functional Requirements – This class defines a list of functional requirements that will be implemented as system functionalities.
Non Functional Requirements – This class defines a list of nonfunctional requirements that may be issued by the requirements such as performance, usability, testability, modularity.
Product Requirements – Similarly to user requirements, this class describes the product requirements, associates them to a particular product, defines properties to determine their priority, complexity, relation to other requirements, uncertainty, cost to implement them, possible risks, status, and pre-conditions for their instantiation.
Requirements List – This class represents a collection of requirement that be associated to a particular project in execution.

4.2.Project Planning

For project planning the classes created in the ontology are:
Project – This class describes the project and its description.
Project Phase – This class defines the phases existing in the projects. It defines the beginning and ending dates and associates them with a particular project plan.
Project Role – This class defines the responsibility and the degree of authority of each project participant.
Project Participant – This class identifies the member who is working in the project. Besides his/her identification, it states the initial and final date of the member in a given project planning as well as the working hours.
Project Plan – This class defines the project scope, the project goal, the life cycle methodology model adopted, the estimation model utilized, the resources allocated, the skill necessary, the stakeholder involved, project tracking, and a complete tracking frequency for skills, budget and schedule.
Project Resources – This class defines the resources utilized by a project and associates them to a project plan. In addition, it states when the resources have been ordered and who the responsible for their usage is.
Estimation Model – This class defines the estimation model utilized for estimating the resources needed to complete the project. In particular, it estimates the minimal and maximal hours for the realization of a given task.
Issue – This class describes the issue related to a work product, its priority, its current status, the date which was identified and resolved and associate with the project participant employed to solve it.
Life Cycle Model – This class describes the possible life cycle models which can be used by forthcoming projects. It also defines when to instantiate a model and how to customize it according to the project resources and needs.

Milestone Type – This class defines the milestones to be reached during the project. It describes the type of results expected, type of tasks to be completed and relates it to a particular project phase.
Task – This class defines the tasks to be executed during a given project phase. It states the start and ending date, the current status, and associates it with the project participant responsible for its realization.
Risk – This class describes eventual risks during the project executions. It defines when the risk was identified, its impact, the strategy for reducing the impact, the risk mitigation strategy (likelihood reduction), the project participant responsible to solve it and the risk remarks.
Skill Category – This class describes skill necessary for a given task. The project participants are associated with the skills based on their previous experiences. This class also provides slots for describing techniques, methods and specific knowledge for a particular skill category.
Stakeholder – This class identifies a stakeholder related to a project, defines the role, the authority, tasks to be solved, the own requirements, the resources to be provided and which information he/she needs to be informed about. A stakeholder not necessarily has to be a project participant, but anyone who is collaborating with the project as an investor, auditor.
Work Product – This class describes the artefact or deliverable under development. It defines who is the responsible for it, its size and complexity and its category (e.g., document, diagram, and code).

4.3.Process Management

For process management the classes created in the ontology are:
Process Definition – This class defines the process to be adopted by a project as well as its inputs and outputs. It also states the entry and exit criteria, the owner of the process and related area.
Process Area – This class specifies the process and its customization according to different areas such as project management, quality assurance and configuration management. It also defines the owner of the process and the adopted policies.
Tool – This class keeps information about the tools which can be utilized during the project. It defines who is using it and describes how to use it.
Methods and Techniques – This class describes methods and techniques that can be applied in a project for a particular task.
Lessons Learned – This class maintain the valuable information from previous projects. It specifies the utilized tools, techniques and methodologies, the main concerns and during the project and in which phase the successful or improper activities where undertaken.
Goal – This class defines the major goals of the projects and the metrics utilized to measure if the goal was achieved.
Improvement – This class defines the possible improvement to a given product. After formal inspection, a project participant can claim for improvements and justify his/her decision about it.

4.4.Quality Assurance

For quality assurance the classes created in the ontology are:
Quality Assurance Plan – This class describes the plan for quality assurance for a given project. It also states the purpose, the scope, and associate with the expected goals.
Defect – This class describes the defects identified in a product. It also states the current version of the product, when the defect was identified, the possible cause, who is the project participant which will fix it and the solution which will be adopted.
Review – This class defines the revision for products during its life cycle. It also describe how

the review will process will take place, who is the responsible for this activity and the conclusion remarks after concluding the revision.
Reviewer – This class indentifies the project participant who is assigned to the revision task. It also describes the reviewer responsibilities and the personal conclusion remarks after concluding the revision task.
Non Compliance – This class states the non compliance in work products verified after formal inspections or test phase. Is defines also the corrective actions, the root cause and further documentation to assisting the stakeholder during the correction tasks.

4.5.Configuration Management

For configuration management the classes created in the ontology are:
Configuration Management Plan – This class describes the purpose, goals, scope, and tools utilized during the configuration management of a project. In addition, it defines when audits must be performed and dates for the baselines.
Baseline – This class states the baseline on products during the life cycle in the project. It also identifies how to approve the baseline and keep revisions observations.
Configuration Control Board (CCB) – This class defines the purpose, role, authority of the CCB and their members.
Change Request – This class describes the change requests. In addition it states who the participants responsible for tackling the changes are.

4.6.Measurement and Analysis

For measurement analysis the classes created in the ontology are:
Measure – This class describes what will be measured during the project. It determines which data sources will be utilized, the storage process, the data analysis methods and the metrics to be used for measurements. It also defines which tools will be utilized to measure the activities. For instance: productivity is a measure that can be calculated during the execution of the project.
Metric – This class describes the metric to be utilized during a project as well as the measurement objective. In addition, it details the information needs for collecting metrics. It also defines project participants responsible for it. For instance: man-hour is a metric of productivity that one person produces in one hour's time.

4.7.The Ontology Classes and the Semantic Properties

The classes addressed in the ontology are described by its attributes and semantic properties. The attributes specify the content addressed by the classes and the semantic properties are employed to establish relationships between different classes.

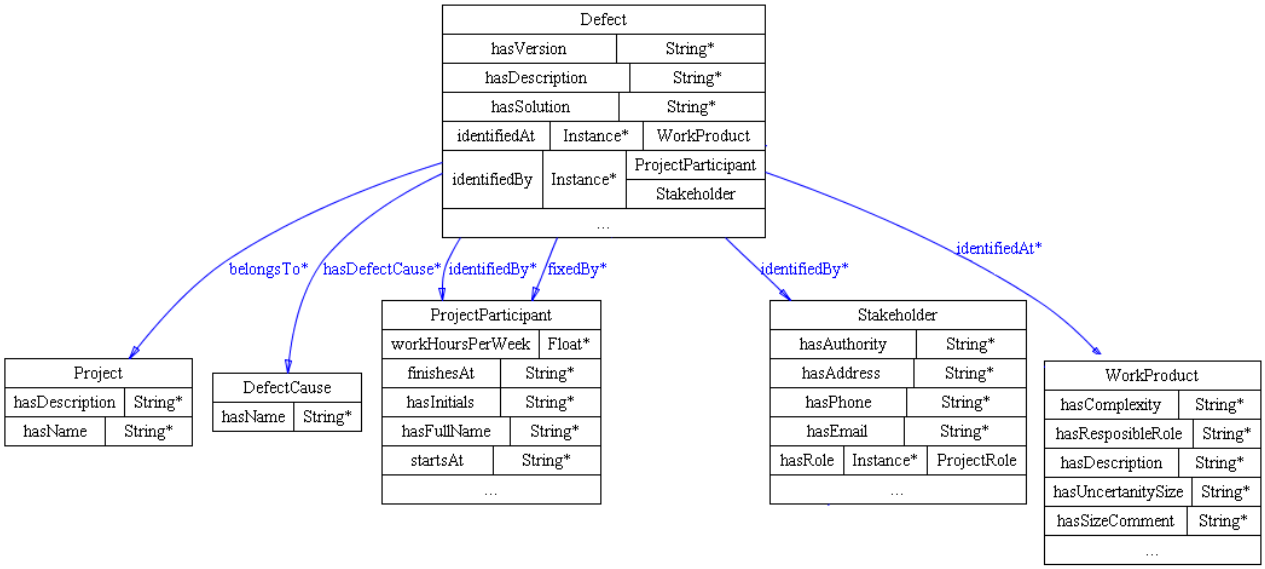


Fig. 2: Attributes and semantic properties from Defect class.

According to Fig. 2, the class **Defect** contains "*hasDescription*", "*hasSolution*" and "*hasVersion*" attributes and "*belongsTo*", "*hasDefectCause*", "*identifiedBy*", "*fixedBy*", "*fixedBy*" and "*identifiedAt*" as semantic properties. Other semantic properties in the ontology have been created similarly and you can find more details about them in the ontology.

5. Ontology Application in KIWI System

The shared knowledge model can be utilized to support various tasks and can be of use for improving an operation of enabling technologies to be developed in the KIWI project.

5.1. Text extraction

Text extraction can be particularly useful technology for extracting the tags or more formal annotation from the text. These can serve as context and work free tags which are directly referring to the content and semantics of the text depicted in a content item or on a wiki page. These can be useful in particular, when recommending, inferring and searching for domain specific information such as concrete topics of a project such as an ERP system for a municipality, particular technologies the risk analysis page talks about, the requirements of customer such as a dialog supporting registration of cases, and so on. The developed ontology serves here as a reference model and a source for tag terms or as a type system for semantic annotations.

5.2. Rule based inferencing

Rules could be used in all most parts of the application: Process related rules could be used to guide the work process (e.g. "if the risk level are above a certain level the risk analysis should be approved by a certain manager", or "the process is not finished before a work product is approved") and they could be used to validate the provided information (e.g., "all risk with a impact rating > 0.5 must have a strategy for reducing the impact"). The developed ontology can be used here to facilitate constants, function symbols, and predicate labels.

5.3. Adaptation and Personalization

The tags from knowledge models or instantiated annotations can be used to recommend relevant or new information entered into the wiki. The relevance can be calculated by relating to the task the user is currently performing, his role, as well as other tags he tagged in the currently displayed page. Those tags are collected and used to find similar tags in the KIWI repository. The pages which are tagged with the similar tags and are from different users are recommended – pushed to the user (see information push requirement mentioned erlier). This feature can be envisioned for almost any process in project management. For example, tagging could be used for the individual steps in process descriptions. The tags that represent the knowledge which is particularly relevant when doing specific tasks can indicate in which activity such knowledge should be used.

The knowledge model developed can further improve the personalization. By using the knowledge model for tagging, we collect more knowledge and the recommendations and adaptation activities will become more precise. For example, through the project plan (the activity list in the scenarios) we know what the users are supposed to work on at any given time (tasks and products), and through the project role descriptions linked to each project participant we also know which responsibilities they have. The likelihood that the extracted information is relevant and focused is therefore quite high. There are several possibilities for providing relevant information that most likely need to be combined:

1. Based on the assignments people have in their project plan, and based on their roles (e.g. being project manager or business analyst), it can be deduced what information they need (e.g., if we know that a programmer is assigned to a specific programming task, he probably needs skill that relates to this specific task).
2. By collecting historical data about the information people in specific roles look for when solving specific tasks of a specific type (history of submitted queries).
3. By collecting historical data about what information a specific user used last time when he / she solved specific task, or type of task (history of used pages/information items).

It's necessary to distinguish between two types of information: General information related to types of tasks/processes or types of products and specific information related to a specific instance of a process/a specific task. The risk analyses process and the general knowledge resources (e.g., a general risk checklist) are relevant for all members of the organization doing a risk analysis, but notes about a single risk is most relevant for a specific project, and not for other projects. Participants assigned to a specific task or a specific risk in the risk analysis process are especially interested in general information related to this specific task, but also in specific information about this particular risk.

5.4.Transforming between structured and unstructured information

Knowledge models and tagging can also be utilized for transition between unstructured information (notes, work in progress, comments) and more structured formats, and the two different parts of the application share a common knowledge model. The part of the knowledge model that's easy available for a user at any time is determined by:

1. The domain of the process selected in the process part of the application: If a risk analysis process is chosen, the information in the "unstructured wiki part" is most likely about risks so risk related concepts are available for tagging.
2. The specific object chosen in the right side (e.g. the risk form) part of the application. If a specific object (a specific risk) is selected, this object should be available as a tag in the "unstructured wiki part" of the application. Furthermore, the form fields can enhance the tagging of unstructured information and point out which of the information should be placed in which field.

5.5.Workflows

The example heavily relies on the workflow component in the system to implement specific processes. The workflows are non-rigid in the sense that they might be changed, they are strongly related to the various kind of knowledge presented for the user, quantitative date (productivity measures) are collected about the workflows (processes), and these data are used as metrics in the estimation and general planning process. The workflows are goven by the process steps descriptions as described above. The particular order in which the activities and steps are preformed is not enforced and depend on the project. The way which the activities are actually performed can be recorded and observed from the historical user activities in the system and can serve as further information on deviations the process team can utilize.

5.6.Examples of Usage

Quality Assurance and Defects. A Wiki page can be created describing a defect identified in a work product to be fixed by a certain project participant. Using the ontology concepts as seen in Fig. 3, as soon as the page is annotated with a proper semantic type, e.g., "*Defect*", the related semantic properties will be required to be filled out such as "*identifiedAt", "injectedDuring" and "hasDefectCause".* KiWi system will assist the defect description by requiring (additional) relevant information according to the knowledge model. In addition, the entries can be validated so that inconsistent data is not stored.

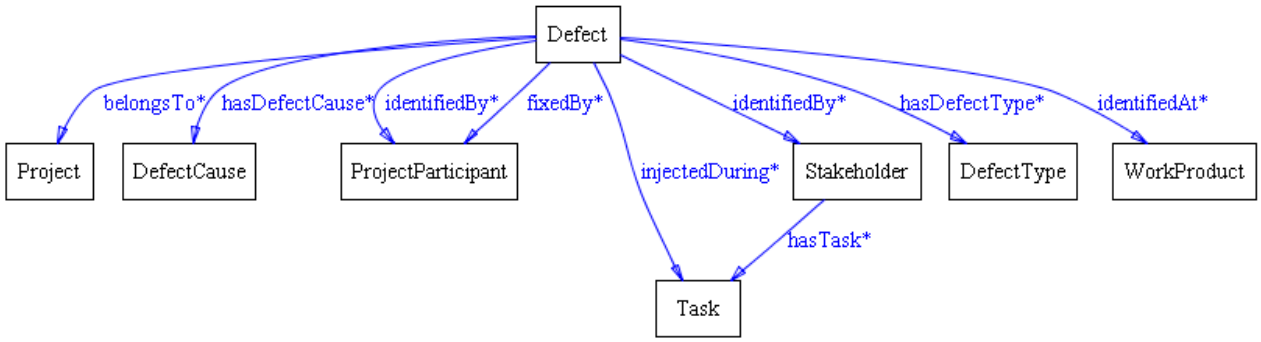


Fig. 3: Semantic Relationships and Properties of Defect

KIWI can further assist with notifications. As soon as the required properties are filled and page is saved, the responsible receives a notification about his new issue with proper information which facilitates the defect resolution.

Semantic Recommendations from Legacy Content. A project manager can report in a Wiki page a task to a given project participant. Following the ontology model as seen Fig. 4, as soon as the page is annotated as a "*Task*", the Kiwi system looks for pages annotated as "*Lesson Learned*" with similar "*Task Definitions*" and in the same "*Process Area*".

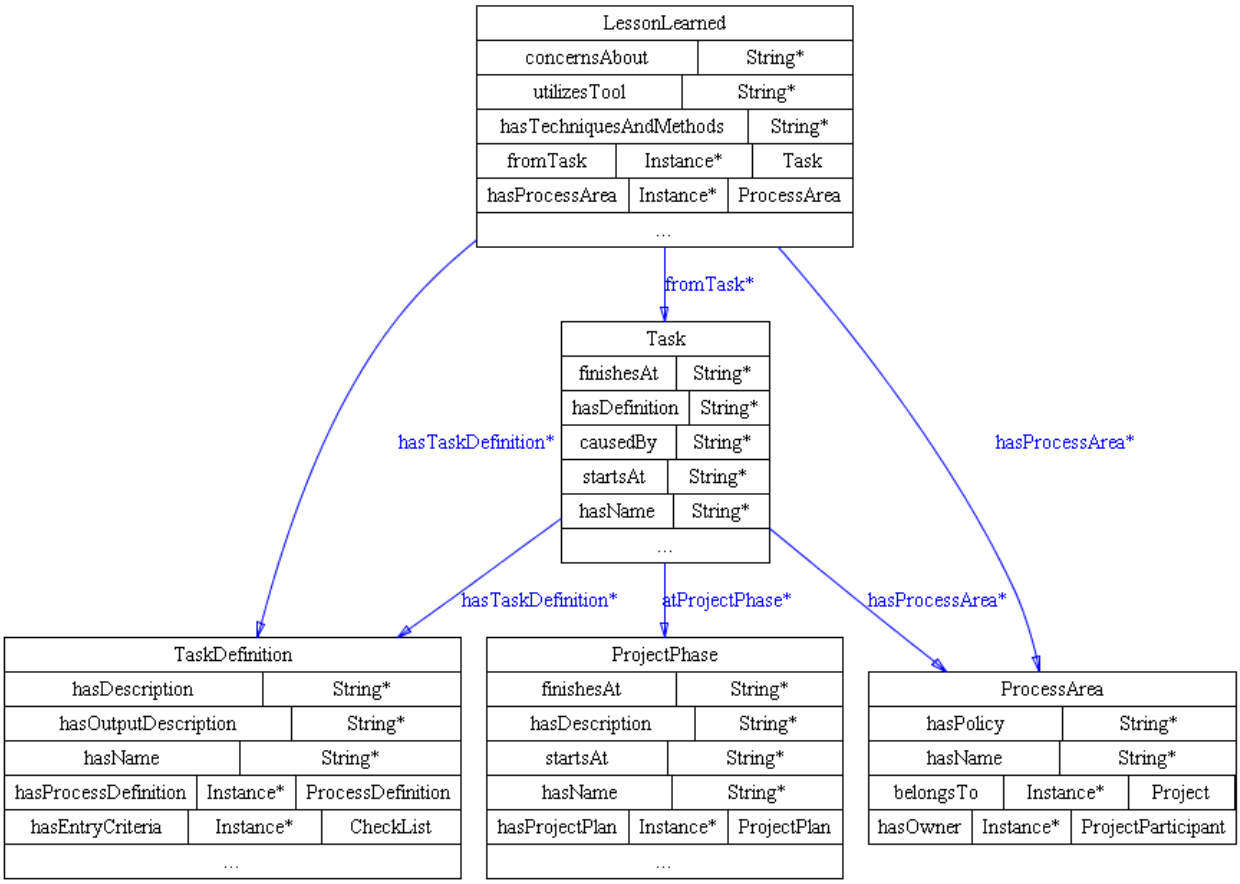


Fig. 4: Semantic modelling for Task.

The outcome is a list of recommendations about concerns, techniques and tools that can be utilized for the task. Although this is typically a data retrieval problem, the semantic annotations narrow the search by filtering out those items which are determined by ontology properties such as "*hasTaskDefinition*" and "*hasProcessArea*".

Semantic Reasoning for Project Planning. When a project plan is being built by a project manager, the KiWi system may reason over the plan needs and the available resources to suggest which members have the required skills to assume planned tasks. According to ontology model as seen in Fig. 5, once the Wiki page is annotated as "project plan", KiWi system may also look for available project resources which are not allocated or will not be utilized in the forthcoming days. In addition, KIWI system can infer critical risks for the project based on its description and analysis of previous projects.

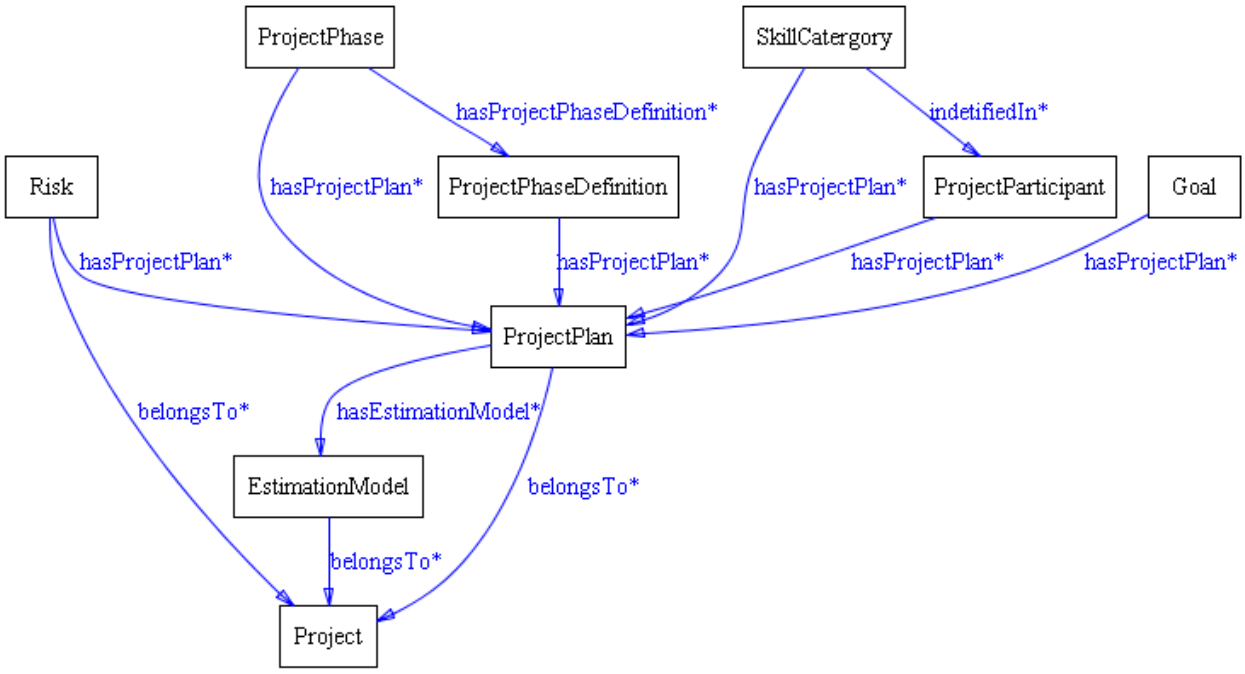


Fig. 5: Project Planning and the conceptual relationships

The project planning assistance provided by KIWI can be an effective mechanism to drive software process in the best way so that risks are quickly identified and the resources are intelligently allocated.

6. Conclusions

In this report we described the knowledge model for project management to be adopted in KiWi. The report reviews relevant areas of software development and utilizes this knowledge to develop a domain ontology covering the major concepts studied and establishing semantic relationship between them. Furthermore, analysis of relevant practices and concepts in Logica organization provided also a significant input to the construction of the ontology. The use of the ontology within KiWi system is also discussed in terms of possible applications that support project management. We show examples how semantic annotations from ontology model can be applied for assisting project members during the software development.

As future works, the semantic model can be expanded to uncovered areas which bring relevant information for the workflow. In addition, the proposed model can be tailored to particular applications so that the process and culture of organizations are respected. The proposed ontology still can be enhanced by inheriting and reusing concepts from external ontologies such as FOAF ontology[2], SIOC ontology[3], W3C time ontology[4], and parts of the Beatle software ontology model for bugs for example[5]. FOAF for example describes the characteristics (name, full name, email, etc) and relationships amongst friends of friends, and their friends, and the stories that they tell. The idea of using FOAF vocabulary is to provide the social network support which is usual in software companies. In the following, the knowledge concepts in the proposed ontology are grouped according to the knowledge areas described above. Further work is needed on this. Further work is also needed to review the constraints on the relations designed in this ontology.

[2] http://www.foaf-project.org/
[3] http://rdfs.org/sioc/spec/
[4] http://www.w3.org/TR/owl-time/
[5] http://baetle.googlecode.com/svn/evoont/trunk/evoont.png

7. References

[1] Bower, P.: Risk Management Options. http://www.projectsmart.co.uk/risk-management-options.html (March 2009)

[2] Dolog, P., Grolin D., Jahn K., Munk-Madsen A., Nielsen, P., Pedersen, K.: Requirements Project Knowledge Management. KiWi WP5 Deliverable. http://wiki.kiwi-project.eu/index.jsp?title=kiwi-pub:KiWi_D5.4_final.pdf

[3] Haughey, D. Requirement Gathering http://www.projectsmart.co.uk/requirements-gathering.html (March 2009)

[4] Haughey, D.: Project Planning: A Step by Step Guide. http://www.projectsmart.co.uk/project-planning.html (March 2009)

[5] Nielsen, P. and Dolog, P.: State-of-the-Art on Software Project Management. KiWi Deliverable WP5. http://wiki.kiwi-project.eu/index.jsp?title=kiwi-pub:KiWi_D5.1_final.pdf

[6] Pedersen, K., Dolog, P.: KIWI Features The Logica Use Case

[7] Schaffert, S.: IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp 388-396, 2006.

[8] Software Engineering Body of Knowledge (SWEBOK): Software Configuration Management http://www.swebok.org/ch7.html#Ref1.3.3(March 2009)

[9] Software Engineering Body of Knowledge (SWEBOK): Software Quality Management Processes http://www.swebok.org/ch11.html#Ref2 (March 2009)

[10] Zubrow, D: About Software Engineering Measurement and Analysis (SEMA), Software Engineering Institute. http://www.sei.cmu.edu/sema/about.html (March 2009)

[11] Liao, L., Y. Qu, & H. K. N. Leung, (2005). A Software Process Ontology and Its Application. *ISWC2005 Workshop on Semantic Web Enabled Software Engineering*.

[12] Wang, M. H. & C. S. Lee, (2007). An Intelligent Fuzzy Agent based on PPQA Ontology for Supporting CMMI Assessment. *Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International*, : 1–6.