D8.5

The KiWi Vision: Collaborative Knowledge Management, powered by the Semantic Web

<table>
<thead>
<tr>
<th>Project title:</th>
<th>Knowledge in a Wiki</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project acronym:</td>
<td>KIWI</td>
</tr>
<tr>
<td>Project number:</td>
<td>ICT-2007.4.2-211932</td>
</tr>
<tr>
<td>Project instrument:</td>
<td>EU FP7 Small and Medium-Scale Focused Research Project (STREP)</td>
</tr>
<tr>
<td>Project thematic priority:</td>
<td>Information and Communication Technologies (ICT)</td>
</tr>
<tr>
<td>Document type:</td>
<td>D (deliverable)</td>
</tr>
<tr>
<td>Nature of document:</td>
<td>R (report)</td>
</tr>
<tr>
<td>Dissemination level:</td>
<td>PU (public)</td>
</tr>
<tr>
<td>Document number:</td>
<td>ICT211932/SRFG/D8.5/D/PU/final</td>
</tr>
<tr>
<td>Responsible editors:</td>
<td>Sebastian Schaffert</td>
</tr>
<tr>
<td>Contributing authors:</td>
<td>François Bry, Peter Dolog, Julia Eder, Szaby Grünwald, Jana Herwig, Josef Holy, Peter-Axel Nielsen, Pavel Smrz</td>
</tr>
<tr>
<td>Reviewers:</td>
<td>Peter-Axel Nielsen, Jana Herwig, Norbert Eisinger, François Bry</td>
</tr>
<tr>
<td>Contributing participants:</td>
<td>SRFG, SWC</td>
</tr>
<tr>
<td>Contributing work packages:</td>
<td>WP8</td>
</tr>
<tr>
<td>Contractual delivery:</td>
<td>31 August 2008</td>
</tr>
<tr>
<td>Actual delivery:</td>
<td>28 August 2008</td>
</tr>
</tbody>
</table>

**Abstract:**
This deliverable describes the common vision of the KiWi project, ranging from motivation over use cases and usage scenarios to user interaction, system architecture and technologies, and the research that is performed as part of the project. The deliverable is intended for a wide audience to give an overview over the project. Parts are deliberately written in journalistic, non-technical style to allow reuse in other dissemination material.

**Keyword List:**
KiWi, Vision, Use Cases, Technology, Architecture, User Interface, Dissemination
Foreword: Purpose of this Report

1. Introduction: What is KiWi about?

2. The KiWi Use Cases: What KiWi Can Do!

3. User Interface and Usage Model: What Does KiWi Feel and Look Like?

4. The KiWi Technology: How will it work?

5. Enabling Technologies: What kind of research will KiWi do?

6. Perspectives and Conclusion
Foreword: Purpose of this Report

The “KiWi Vision” document is one of the first deliverables of the KiWi project and is part of the “Dissemination” work package. This report serves several goals:

- Providing sufficient input, both non-technical and technical, for the dissemination activities to start early in the project (Section 1).
- Develop already early on in the project scenarios for use of the KiWi system that go beyond the project itself, so that dissemination activities can provide interested persons with concrete examples how the generic KiWi system can be used for a wide variety of purposes (Section 2).
- Developing an early consensus among the project partners about the concrete tasks and outcomes that the KiWi project will deliver. This is necessary to ensure that the project participants work on a common goal and come up with a common system (Sections 3-5).

This document is therefore structured in several parts that are different in style and audience.

- The section “Introduction” (Section 1) is written in a very journalistic (and enthusiastic) style that is meant to describe even for non-technical people the vision for knowledge management in the “future” Web that we have. To avoid a too technical language, it deliberately does not describe in detail the outcomes of the project.
- The section “The KiWi Use Cases” (Section 2) summarises the user requirements of the two KiWi use cases, software knowledge management (Sun Microsystems) and project knowledge management (Logica). Since Sun and Logica also have quite different company philosophies, the use cases also cover two rather different, yet complementary, approaches to using the KiWi system. Section 2 also sketches a public showcase.
- The section “User Interface and Usage Model” (Section 3) describes the main aspects of the user interface and user interaction within the KiWi system. A particular emphasis is given to the “usage model”, i.e. the core system concepts the user is exposed to.
- The section “The KiWi Technology” (Section 4) gives an overview over the KiWi system architecture and the technologies used to implement the KiWi system. KiWi builds upon the popular Java Enterprise Edition framework and some extended frameworks.
- The section “Enabling Technologies” (Section 5) introduces the research objectives of the KiWi project. It describes the challenges in “reasoning”, “reason maintenance”, “information extraction” and “personalisation”, how we plan to overcome them, and how this fits into the KiWi system.
- The section “Perspectives and Conclusion” (Section 6) summarises the deliverable and provides a short outlook into what is still to come in the project, and describes possible further application areas where the KiWi system could be used to improve web-based platforms beyond what is actually covered by the project itself.

It should be noted that this deliverable constitutes a vision in its proper sense and does not define requirements for the KiWi project. Some of the properties described here might not be part of the actual project outcome.
1. Introduction: What is KiWi about?

1.1. The KiWi Approach to Knowledge Management

Although the promise of effective knowledge management has had the industry abuzz for well over a decade, the reality of available systems fails to meet the expectations. The EU-funded project KiWi – Knowledge in a Wiki project sets out to combine the wiki method of collaborative content creation with the technologies of the Semantic Web to bring knowledge management to the next level. This section explains the core principles of ‘Knowledge Management 3.0’.

Have you ever wondered how people managed to get things done in their jobs in the days of the pen and the typewriter? Ever since in the 1980s computers entered the individual’s workplace on a large scale, and even more so since they got hooked up to the Internet in the 1990s, working without computers has become almost unthinkable.

As a side effect, knowledge has become “measurable”: by the amount of data in- and output, by the number of words, figures, documents, media and log files generated and processed by the employees, who were suddenly known as knowledge workers. Parallel to the expansion of the so-called knowledge-intensive industries, the fear arose that knowledge might be lost through mismanagement: Data that are merely stored, but not used, are dead – only data that are used, exploited and circulated can transform into knowledge.

The KiWi – Knowledge in a Wiki project proposes to approach knowledge management from a fresh perspective by combining it with, on the one hand, the wiki philosophy and, on the other, the methods of the Semantic Web. The one has revolutionized our notion of knowledge (and of who controls it) over the past five years, the other one is going to fundamentally transform our practice of knowledge sharing across all platforms in the years to come.

To put it in a nutshell: KiWi aims for a vision and a first glimpse of the future of collaborative knowledge management by developing a prototype of a running knowledge management system that is as easy to use as a wiki, and backed by the intelligence of the Semantic Web. Sounds like a good idea – but how is this to be achieved?

Please follow us on a short excursion through the present challenges of knowledge management, the solutions the wiki way of collaboration has to offer and through the promises of the Semantic Web.

The Rules of Good Knowledge Management: People and Collaboration First, Technology Last

For well over a decade, ‘knowledge management’ has been one of the buzzwords of the industry. In spite of the hype and its promise of yet another paradigm shift: Take a closer look at the way that knowledge management is put into practice, and you will find that companies and institutions often resort to the traditional methods of the trade when it comes to an actual implementation. As a result, knowledge management (KM) is often part of a company’s quality management (QM) efforts – with dramatic consequences for those individuals whose data and information are to be managed:

If KM is carried out with a QM approach, employees are often obliged to fill in endless fields and forms and enter seemingly arbitrary metadata – as dictated by ‘the system’. Instead of
reflecting the meaning and significance that a given set of information has to the employee, the defined workflows are often mainly committed to the technological needs of the present QM system, and/or replicate rigid hierarchies within the institution.

This ‘dictate of the system’ also reveals a popular misunderstanding of the benefits of knowledge management: The key impetus for companies to implement a KM strategy is often the fear of losing knowledge if an employee becomes ill or even leaves the company. But KM systems which are designed with the intention to extract knowledge from employees are often not well accepted: Implicitly, they give rise to the fear among employees that they might be made redundant once they have shared all their wisdom with the company.

But knowledge management done correctly does not aim to extract wisdom: It aims to facilitate knowledge sharing and collaboration among the members of a given group, be it employees, managers or researchers. Rather than forcing people to adapt to the system’s standards, a good KM system is centred around the needs, preferences and the structure of knowledge the people bring in. Instead of “knowledge is power”, the organisational culture of the future thus has to switch to “sharing is power”.

The Wiki Way: Not a Technology, but a Philosophy

When one thinks of wikis, the layout of Wikipedia inevitably comes to mind: grey and blue on white, navigation bar and tool box to the left, a few tabs situated above a rather text-heavy page. Immediately, people recognize a wiki by this appearance that has been moulded by the free encyclopedia that anyone can edit.

At the same time, one will inevitably fail to understand the essence of wikis if one thinks of it as a generic piece of software: It was not the software, but the wiki principles that have transformed and revolutionized the sphere of content creation and organisation.

- **Wikis allow anyone to edit**: The core principle is that there are no access restrictions or strict hierarchies on the content of a wiki. Anyone can easily contribute his or her own knowledge, his or her own ideas, and his or her own content.
- **Wikis are easy to use**: Anyone who is sufficiently familiar with the basic functionalities of word processing software (write, delete, save) has all the skills required to edit, correct and expand a wiki.
- **Wiki content is linkable**: By allowing users to create links between words and as such between concepts, wikis also allow for the creation of semantic relations, i.e. of meaning.
- **Wikis support versioning**: Never does information disappear on a wiki. If a page is edited, the previous version is still stored somewhere. This has an important psychological effect as it takes away the **wiki writer’s block**: the fear that something might get lost through editing.
- **Wikis support all media**: Wikis are web-based. So whichever type of content you have, be it text, images, audio, spreadsheets, documents – anything that can be displayed in a web browser can be displayed in a wiki. And even if a file of a rare format cannot be displayed in the browser itself, it can still be downloaded.

Within the wiki world, no user is above or below another with regard to one principle: Anyone can edit. And as the wiki philosophy did away with the gatekeepers (traditionally: journalists, editors, publishers), it also opened its gates to the forces of chaos. To keep those forces under control, the wiki world relies on the wisdom of many. Or, as the saying named Linus Torvalds’ law goes: “Given enough eyeballs, all bugs are shallow.”
While Wikipedia is the outstanding example that this is indeed true, a knowledge management strategy within a company can impossibly recruit as many active users as Wikipedia. In the era of the attention economy, one of the biggest motivations for users to contribute to social media is to receive attention and recognition for what they are doing – a force that can and must be harnessed for one’s own knowledge management efforts.

Integrating a social component is therefore one of the aims of the **KiWi** project. Communal knowledge that is generated collaboratively by a group of people with shared tasks and similar interests is always going to be superior to any isolated expert’s advice when it comes to reflecting the topics, challenges, questions and answers that are relevant to this particular group. And unlike many corporate KM systems, a wiki is not bound by the restraints of a too complex or hierarchical workflow design: Anyone can ‘be the boss’ and have the last say – if only for a fleeting moment, until the horizon of shared knowledge expands and the wiki is edited again.

“**There is no white raven... right here and now**”:

**Keeping structures flexible on the Semantic Web**

A wiki is like a blank sheet of paper: One can use it to write love letters or business reports, print on it, draw on it or cover it with doodles – and while this might be appreciated by some, its lack of structure may also appear threatening to others.

Structures, if dictated by the system and not questioned for their actual benefit, can present a hindrance to effective knowledge management – but structures are also invaluable where they support the user in carrying out tasks. Who would be able to file a tax declaration in a timely manner on just a blank sheet of paper, i.e. without the instructions presented by the fields, their labels and order on the form sheet?

Providing a flexible set of structures that are applicable to different (and yet unnumbered) types of content is the main objective of the concepts and technologies, which together form the Semantic Web. The main principles on which these technologies rely are, firstly, the fact that both structures and their relationships can be made explicit and that, secondly, these structures and relationships can be processed further applying the methods of information technology (e.g. algorithms, or so-called “reasoning”).

The peculiar aspect about Semantic Web technologies – and the point in which they differ from, for instance, traditional database methods – is that they are based on a so called *open world assumption*. Its opposite, the closed world assumption, is what our thoughts and perceptions of the world rely on most of the time.

A closed world assumption works like a time table: If it says in the timetable that trains are running at 07:16, 07:37 and 07:58, I will ‘naturally’ assume that there will be no train service at 07:25. Databases, for instance, rely on closed-world assumptions.

An open world assumption neither rules out the existence of the mentioned 07:25 train, nor the existence of the legendary white raven¹ – here, just because something is absent doesn’t mean that it does not exist.

KiWi takes into account both approaches: while relying on a closed world assumption for reasoning purposes (“if the timetable contains no train at 07:25, there is no such train”), the KiWi system is still open to easily add new information and also to integrate with external

¹ the “white raven” refers to a phenomenon also called “Hempel’s ravens” or “Hempel’s paradox”, and reveals the problem of induction: the fact that all ravens that I have seen have been black does not necessarily mean that all ravens are black, as it does not rule out that there might be a white raven somewhere
services.

And other than, for instance, a database where an empty field is going to produce an error, the Semantic Web is semi-structured: It does not enforce the use of certain structures, but checks whether the available information is consistent with a given model of knowledge representation, called ontology.

An ontology represents the concepts and the relationships between those concepts within a given domain. To cite just two examples of existing ontologies: Dublin Core is a simple ontology used for documents and publishing; FOAF (Friend of a Friend) is an ontology describing persons, their activities and their relations to other people and objects.

Unlike a database, ontologies are designed in such a way that they remain flexible for future expansion. Ontologies can also be used as a ‘translation layer’ to interpret and open the content of a database – but if a consistency check fails, this does not result in a fatal error: At best, one can work around an inconsistency, at worst, the ontology cannot be used for the given purpose.

Within the context of collaborative media (e.g. Wikis), Semantic Web technologies can be instrumental in helping ward off the ever-imminent threat of chaos by backing up user annotations with ontology-based structures that can be used for more efficient search and navigation, and furthermore help moulding social interaction, e.g. by enhancing relationships between users with Friend of a Friend data. Semantic Web technologies also promise to support the user e.g. by personalizing the appearance of the system based on background and interests of the user, as well as on the context of the content currently displayed.

**Knowledge Management**

+ Wiki Philosophy
+ Semantic Web

= KiWi

As the equation suggests, the vision of KiWi is to make these three areas merge: KiWi aims to make a form of knowledge management possible where knowledge is generated because knowledge is shared. The KiWi system is therefore going to be highly usable: Its contents are linkable to allow for the creation of meaning and open for semantic (e.g. ontology-based) enhancement of meaning.

Furthermore, KiWi supports versioning to promote trust and confidence in the system and remains open and flexible for all types of contents. The KiWi system offers appropriate structures and an adaptive and modifiable interface that support the user in a way that corresponds to his or her mode of creating content.

As a user-centred system, it is vital that KiWi considers all of the following five aspects:

1. **Role**: KiWi reflects the role of the user, i.e. the position within a team (company, organisation, ...), and the tasks for which the user accepts responsibility as well as the knowledge that she has, as this, too, is decisive for the roles she is able to accept.

2. **Domain**: KiWi is reflective of different domains, i.e. different fields of knowledge and of the concepts and relationships of the concepts that are relevant to a given domain. KiWi does, however, not limit itself to categories and concepts, but is also designed in a way that it is able to adapt to the styles of knowledge presentation and communication within a given domain. Lab reports, for instance, are a highly structured type of
documentation used in chemistry and mechatronics; executive summaries rely on prose and rhetoric rather than on obvious structure and are typically used in business.

3. **Context:** Context is king in any form of successful communication. **KiWi** is therefore able to reflect at least central aspects of the context that shapes a particular task or presentation: What is it about? What does one want to create it or use it for? Who is going to use the results of this task or make use of this document?

4. **Experience:** A usable interface considers the experience of a user. The **KiWi** interface is designed in such a way that it does not ward off users by confronting them with arbitrary, system-immanent requirements that the user is not familiar with. While **KiWi** does not ask too much of the inexperienced user, it must also not ask to little of the experienced user: Not all users will be able, for instance, to enter metadata reliably – but the skills of those who are must be harnessed in a way to benefit the entire system.

5. **Preference:** User preference and the motivation and readiness to use a system are closely related. The usability of a system increases if users are given the opportunity to select their preferred presentation (and thus: interaction) style. This includes control over background colours and layout, but is ever more vital when it comes to choosing between, for instance, Gantt-diagrams or Navision reports as a preferred style of information.

**KiWi** is not ‘just another wiki system’ – as a matter of fact, it does not even have to look like a wiki if, for instance, a certain group of users are more likely to accept and work with it if it looks like cost management software. At its heart, **KiWi** seeks to transfer the wiki philosophy of collaborative content creation into the sphere of knowledge management and marry it there with the notion of flexible semantic structures that support the user in carrying out his or her knowledge management tasks.

**KiWi** does not simply manage knowledge or content: It reflects and enhances community relationships and the structure of collaborative knowledge and puts it at the service of knowledge management. The main outcome of **KiWi** will thus be a completely new kind of (semantic) content management system based on the Wiki philosophy, rather than just the revision of already existing technology. This system can then be tailored towards the knowledge management needs of the particular organisation or community it is deployed in.

The viability of this vision is going to be put to the test in two different use cases, both carried out together with companies from the knowledge-intensive industries: One has its focus on knowledge management in software development (conducted in collaboration with SUN Microsystems), the other one on knowledge management in project management (in collaboration with Logica). In both these and all future application scenarios, the **KiWi** system is going to have to be adapted to the respective needs.

Throughout the history of human civilization, the relationship between human activity and the impact of technology has been a complex one. The question who of the two is actually in the driver’s seat has been much debated, and never resolved – yet strength and advantage of having a vision has beyond argument. The KiWi vision in as much reflects the challenges of the present as it looks into the future – stay tuned with us on [www.kiwi-project.eu](http://www.kiwi-project.eu) if you would like to learn more.
1.2. The KiWi System: Breaking System and Information Boundaries

The “Wiki Philosophy” described above does not only apply to wikis as a technology alone. Most other social software systems follow the Wiki philosophy as well: a blog, photo sharing site, or social networking platforms all share the same principles of everyone contributing, ease of use, easy linking of information, versioning, and web-based media. Being similar in philosophy, these systems are in most parts identical when it comes to technology.

Based on this insight, the KiWi Vision is to develop a system that can serve as a platform for implementing and integrating many different kinds of social software services, thus *breaking system boundaries* between these services, which are justified merely by technology and not by concept.\(^2\) Beyond this, the KiWi system will allow users not only to participate and collaborate in the creation of *content*, but also in the development of *the platform itself*, giving them the opportunity to develop completely new kinds of social software. It will do so by treating user interface components (“widgets”) like normal content that can be edited and stored in the system and shared with other services. Content in the KiWi system will be integrated in ways that go beyond mere user interface level linking, by employing semantic technologies for annotating and connecting information. The KiWi system is thus also *breaking information boundaries*, allowing users to connect content in many different new ways that are not possible with current systems, which mostly prescribe the possible connections.

Overall, KiWi thus takes the Wiki idea to the next level: the KiWi system will be a completely new kind of social software platform that allows users to share knowledge more easily, to integrate more naturally and tightly, and to adapt content and functionalities to their personal requirements.

In the following, we briefly summarise the central aspects of the KiWi system: breaking system boundaries, breaking information boundaries, and putting the user in the centre.

1.2.1. Breaking System Boundaries

The heart of the KiWi system will be a reusable platform that provides the shared functionalities that are necessary for many or even most different kinds of social software services. The KiWi system thus allows social software developers to easily build and adapt new services as they are required, e.g. within enterprises or on public social software sites. Core functionalities required by social software systems are:

- representation of many different content formats, e.g. text, images, etc.
- easy linking between content items
- versioning and change tracking of content items
- annotation of content with meta-data, e.g. in the form of tags
- rating and commenting of content by users
- integration of different content items using machine-readable connections (e.g. RDF)
- user management, user profiles, and social networking
- personalisation and context adaptation

\(^2\) it is noteworthy that this is essentially the same insight that caused the tremendous success of the World Wide Web in the 1990s: the WWW did “nothing more than” breaking technologically justified system boundaries.
• reputation and incentive system (using Sun’s Community Equity)
• information extraction
• reasoning and recommendation system (e.g. for tags)

In the course of the project, KiWi will implement all or at least most of these core functionalities. KiWi could thus simply be seen as a framework that allows software developers to more easily build social software applications for knowledge sharing and knowledge management. However, KiWi goes further, and this is where the boundaries between systems are breaking: the KiWi system will also allow to build different kinds of social software services in the same instance using the same content. For example, the same photos used in a photo sharing service like Flickr\(^3\) could be used in a service for sharing travel stories like on Tripwolf\(^4\) and in a learning community blog and forum like the Digital Photography School\(^5\).

Furthermore, every user of the KiWi system can become a developer of the platform itself, because KiWi treats functionality and user interface components (“widgets”) like any other content item that can collaboratively be created, edited, and used. In this way, the community together can build innovative and new social software services by combining different functionalities created by users. This functionality will be realised by integrating the Zembly\(^6\) system, which allows users to collaboratively create widgets that can be reused in different platforms. Zembly is currently under development by project partner Sun Microsystems.

The KiWi system is thus “breaking system boundaries”.

1.2.2. Breaking Information Boundaries

What is lacking in the current social software world is a kind of integration of the different kinds of content that goes beyond mere linking. Integration in this context means to be able to connect content in ways that carry more meaning than just linking using e.g. hyperlinks or user interface level inclusion using e.g. HTML img tags or “mashups”.

For example, traditional linking means that a user is able to link from a blog post in the "Digital Photography School (DPS)" to an image uploaded onto Flickr by means of e.g. an HTML img tag, meaning that the image is included in the blog post, but only on the user interface level in the user’s browser. In contrast, integration would mean that the photo on Flickr becomes an integrated part of the DPS blog system with all information available, e.g. EXIF metadata about shutter speed, aperture, and photographer, tags added by users, annotations, associated comments, etc. Instead of being limited to the system where it has been added, context information about the content is available anywhere the content is used.

Note that this does not necessarily rule out that the content still resides in another system. Although not foreseen in the KiWi project, content integration could also be realised by providing appropriate adapters to external content that access e.g. a Web Service to retrieve content and associated information from other sites (which must of course support this).

Beyond the integration of existing content as described above, the KiWi system also supports the representation of content whose shape and schema is still emerging or even currently not planned at all. For example, a user profile might currently consist of name, age, email, address and phone, but in a later stage of the system, it might turn out to be interesting to also add

\(^3\) http://www.flickr.com/
\(^4\) http://www.tripwolf.at/
\(^5\) http://digital-photography-school.com/blog/
\(^6\) http://www.zembly.com/
coordinates for the current location of the user to display him on a map (as e.g. in Plazes7). Based on semi-structured Semantic Web technologies like RDF and OWL, representing such kinds of “evolving knowledge” is supported by the KiWi system without requiring to completely redesign the application’s database schemas, user interface, or APIs.

The KiWi system is thus also “breaking information boundaries”.

1.2.3. Putting the User in the Centre

Everyone is different, and software systems should take this into account. By “putting the user in the centre”, we mean that the KiWi system allows to tailor the presentation and functionality of the platform to the information needs and experience of the user, and to make the use of the platform as easy as possible. “Information needs” materialise in the explicit or implicit preferences and interests of the user regarding e.g. user interface or topic, in the context of the currently presented content, and in the current context of the user, e.g. “at home”, “at work”, “travelling”.

In KiWi, preferences will be represented by appropriate user models that may even be generated automatically by observing user behaviour (see Section 5.3). The context of the currently presented content is made explicit to the system by appropriate annotations and tagging, (and the current context of the user by allowing users to switch between different roles). For instance, the system could observe that a user is frequently accessing content that is concerned with “portrait photography” while she is in the role “at home” and derive from this that this is a topic of interest for her. For portrait photos, the system could offer a specific presentation that is different from other photos, e.g. by displaying a user profile box of the depicted model. While “at home”, the user could then get a list of recent “portrait photos”, whereas she would get a list of recent posts to the company blog while “at work”.

User interface adaptation will be supported by allowing advanced users to expand the platform by custom widgets, and by defining custom layouts that allow arranging widgets in context dependent ways in the browser window (see Section 3.3). For example, there could be a custom layout for portrait photos as described above, and the user could have a personalised start page displaying widgets with information that is relevant for him or her only.

In addition to personalisation and adaptation, the KiWi system as a platform will provide generic support for common tasks to make using the platform as easy as possible for the user. Widgets are not only suitable for the presentation of information, but also for interaction with the system. Developers and advanced users could thus easily provide so-called “Semantic Forms” (as e.g. in Semantic MediaWiki®) that make it easy to enter content in a structured way. For example, an editor widget for “portrait photos” could offer a field to either provide a link to an existing profile or to immediately enter information about the depicted person. Information extraction will support the user in efficient searching and browsing as well as in tag recommendation. For example, the information extraction component in KiWi could automatically detect that the user is entering “Paris” and ask whether he means “Paris, capital of France” or “Paris, first name”, and tag the content appropriately. Furthermore, a rule-based reasoning system will allow deriving implicit information from what is given explicitly. Such reasoning could also be used to make recommendations, e.g. of the form “content tagged with jaguar, the animal, should also be tagged with animal”.

7 http://www.plazes.com/
1.3. KiWi in the Enterprise: Acknowledging that Corporate Settings are Different

A central aspect of the KiWi project is that it specifically addresses the use of wiki systems in the enterprise. While there are numerous examples of successful use of wikis in public systems, the adoption in corporate settings very often fails. The reason for this is that corporate settings are quite different in several regards:

- The amount of content and the number of contributors is usually lower. As a consequence, corporate wikis often tend to contain not enough content to be interesting enough to use. Furthermore, content is often out-dated and not maintained properly, and the content is insufficiently structured. In public wikis, this problem is avoided by having many contributors, some of which are mostly concerned with maintaining content and structure of the system (so-called “wiki gardeners” or “wiki gnomes”\(^9\)).

- There is no “intrinsic” motivation to maintain and update content. In public wikis, every visitor can potentially see poorly maintained and out-dated content, so there is often a personal motivation to maintain the content properly. In corporate settings, such a motivation is often significantly smaller.

- The motivation to share knowledge is often low. In the enterprise, the saying “knowledge is power” mostly prevails: having certain knowledge is a means to secure one’s job, to gain more income, or to gain more power. However, the switch to “sharing is power” approach (as proclaimed by Vint Cerf, the “father of the Internet”) in organisational cultures would benefit most organisations. The KiWi technology is able to support this shift as easily as possible.

- Additional effort without immediate benefit will encumber the adoption of new systems by users. Many systems (e.g. for document management, content management, or quality management) come with significant additional effort for the employees in the company without providing immediate benefit. In order to motivate the participation of employees, some sort of incentive mechanism is necessary.

- Textual content and Wiki mark-up is insufficient. In many cases, the content that is shared in enterprise settings contains more complex content like tables, images, or layouted text. The KiWi system will be able to provide an easy-to-use support for this. Also, it is quite frequently necessary to share documents in non-wiki office formats like PowerPoint presentations or Word / OpenOffice text documents.

- Enterprise settings need permission management and different access rights. Unlike in public systems, the protection of corporate knowledge is usually a necessity. For example, temporary employees or interns are not supposed to get access to critical financial or strategic information. Permission management must also differentiate between read and write permissions: some content might be of official nature and must not be changed except by a few persons.

- Corporate settings need workflow support. Following a certain workflow for certain kinds of tasks is standard in most enterprises, particularly when processes are formally described as part of quality management procedures and certification. The KiWi system is going to provide at least basic support for workflows.

---

KiWi will address all these issues by integrating technologies that are not found in other kinds of wikis. Particularly, KiWi will provide a sophisticated permission and identity management system that allows configuring fine-grained access rights to different kinds of content. It will also include support for managing content-related workflows. Both functionalities are already provided by the Seam framework\(^{10}\) KiWi builds upon: access rights and permission management in Seam allow authentication, roles, and rule-based security checks, and workflow support is realised using the jBPM business process modelling language.

In order to encourage the participation of users and the active sharing of knowledge, KiWi aims to integrate the “Community Equity 2.0” system (now called “SunSpace”) developed by the KiWi partner Sun Microsystems. At the core, Community Equity is concerned with measuring the value of one’s contribution to the community. The “personal equity” is calculated by taking into account the “contribution equity” (the value of the content that I contributed to the system), the “participation equity” (how did I participate in the community, e.g. by rating content of others, by tagging, or by commenting), the “skills equity” (what did others think about the quality of my contribution), and the “reputation equity” (a combination of the former three values). Likewise, each content item contributed to the system is associated with a so-called “information equity” indicating the value of this content. The information equity is determined based on the number of views, downloads, reuse, and ratings of the content item, and degrades over time, i.e. more up-to-date content has a higher information equity.

The KiWi system will also support rich text editing with advanced text layouting capabilities for tables and the inclusion of media content. Additionally, uploaded multimedia or office content will be analysed for full text search and meta-data extraction as far as possible with existing technology. This will enable office users to easily search also through office documents.

Note that, while these features are primarily targeted at enterprise environments, they can likewise be useful in “open” social software applications. Indeed, features like workflow management and community equity would significantly enhance the value of many currently existing social software platforms.

\(^{10}\) [http://www.seamframework.org/](http://www.seamframework.org/)
2. The KiWi Use Cases: What KiWi Can Do!

Besides developing the core system, the KiWi project investigates two enterprise knowledge management use cases at its industrial partners Sun Microsystems and Logica Denmark. In addition, KiWi aims to develop a public showcase that illustrates the features of the KiWi system in a pure social software setting, without requiring the rather complex context of the enterprise knowledge management. The two use cases and the public showcase are described in the following sections.

2.1. The Logica Use Case: Managing Process Knowledge in Projects

In software development projects there is an ongoing struggle with keeping track of knowledge, with exploring and exploiting the knowledge we already have, and with creating and validating new knowledge gained through software development projects.

In current technologies, this is supported by the use of simple document types written in standard productivity tools like Word and Excel. The contents of these documents are created and maintained by a few managers whose responsibilities cover the contents of the documents. For example, the software project manager is responsible for the project plan and it is her responsibility to ensure that the project plan is realistic and that all developers, customers, users, and other stakeholders are committed to the plan or to modify the plan accordingly. The software process engineer, as another example, is responsible for the quality of the descriptions of processes to be performed by development projects and the process descriptions should have a high quality reflecting the company's policies and their best experiences. The problem with the current technologies is that, even though the documents are in a shared directory, the writing of these documents is centralised and rests on the shoulders of the responsible manager. Document quality is ensured by systematic and organised reviewing where knowledgeable colleagues comment and eventually approve the documents. While approval is needed for core documents, it also makes the change of documents less flexible than desired.
The documentation of software processes and their performance in projects is a complex undertaking (see figure). The documents contain knowledge that is difficult to collect, write, and commit to. The documents change (sometimes even radically) as a project progresses and events, ideas, and actions unfold during the project. To improve the quality and usefulness of the documented software processes and their performance in projects, the vision for Logica is to employ a wiki technology for writing, storing, reading and retrieving the knowledge which is currently in long and complicated documents. The knowledge will be split into several pages and the complex relationships will be created and maintained as links and with tagging of pages’ contents and links. This will allow for a much more collaborative and decentralised writing and maintenance of contents. It will also allow for a more flexible change and approval of contents.

Some contents on pages will be text written in wiki style with easy linking to other pages and content items. Some contents on pages will be written into tables with equally easy writing of cell contents and linking. The linking will for a major part be used to hold the complex structure of the many pages. The pages will also contain information of change history, dependency history, and approval history. The change history makes it possible to trace all changes back in time. The dependency history makes it possible to see changes to all pages which a page depends on or is a dependent of. The approval history makes it possible to see the status of a page (and its dependent pages) in the formal management approval process.

Many content items, links and tags come directly from templates. A template is more than just a single page; it is a set of pages forming a complete framework for a software process. This could for example be for a project plan covering issues such as schedule, work breakdown structure, resources, commitments, etc.

The structure of links and tags is so complex that the users will be supported in linking to the right content and tagging with the right tags. The user will not have to remember and recall names of content items and tags, but will have a context-dependent set to choose from. The context-dependent form of interface extends to many other areas of interaction between the collaborative writers and readers as the vision for the KiWi system at Logica is that which pages, links, tags get displayed depends on who the user is, her project history and the history of interaction. The user may be a project manager for a particular project with a certain track record on several past projects. The interaction history should reflect how the user at a micro-level has interacted with the system lately, e.g., in terms of common searches, common pages read, common pages written, linked, tagged, etc.

Retrieving history of many kinds will altogether be very useful for Logica and it does not exist in the current technology. Retrieving history based on tags will be a significant advancement. The tag structure can be elaborate and should cover all knowledge domains including process management, project management, requirements management, quality assurance and similar process areas. It should also cover customers’ domains like accounting, administration, and other domains for which Logica delivers software solutions. It should for example be possible to retrieve pages concerned with similar projects, technologies, or conditions, etc.

The vision is that pages will be dynamic far beyond the current documents. The ambition for Logica is first that much more knowledge will find its way into the system by increasing the affordance for all involved software managers and developers. Second, the complex relationships between pages and contents will be used to increase the awareness of changes, dependencies and approvals. Any relevant change to a page should either propagate directly to related pages or the writers should be alerted that further changes are necessary and what the approval status is.

All in all, the expected benefits for Logica will be (1) a much more open and distributed experience of writing, searching and reading pages, (2) a better overview of the network of
knowledge items, (3) elaborate tagging with many relevant domain concepts and labels, (4) intelligent searching with a useful limitation of search results, (5) a significantly more dynamic body of knowledge, (6) propagation and alerts on changes and dependencies, and (7) better contents to search and read and hence to contribute for all software managers and developers.

### 2.2. The Sun Use Case: Managing Software Development Knowledge

The vast majority of software created by Sun Microsystems is developed in an open source manner. This puts an enormous pressure on building and maintaining various communities of practice and interest, spanning over the actual corporate boundaries. Knowledge management in such communities is a challenge, solvable by the effective utilization of a traditional wiki model, in combination with the latest advancements in the web development (social networking, Semantic Web). Sun has already put great effort into the deployment and utilization of various wiki systems allowing various open communities to effectively create, share and collaborate.

NetBeans as a product is an Integrated Development Environment (IDE), with a history spanning over more than 10 years of continuous development. Development of the NetBeans IDE is a relatively complex process, involving many acting parties, various types of knowledge and different roles of participants (Figure 1).

![Figure 1: The various roles of participants in the NetBeans development process at Sun Microsystems. Roles in the darker and more central parts of the diagram are more important than those placed in lighter ones.](image)

The complexity of the current development process is further enhanced by the rich array of different knowledge management and document storage systems used by the developers. These encompass classical Wiki systems, Bug-tracking systems, FTP repositories, versioning systems, e-mail repositories and many other types of systems.

The KiWi project will provide the means to integrate the information that is currently housed in many heterogeneous data sources, thereby increasing efficiency of the development process and overall organisation at the NetBeans development group at Sun Microsystems.
2.3. A Public KiWi Showcase: Photo Stories

Besides the two concrete use cases described in the previous sections, we also strive to develop a public showcase that allows us to demonstrate the capabilities of the KiWi system as a platform for social software applications to a more general audience that is not familiar with the specificities of project management and software engineering.

The scenario of the showcase is to develop a social software platform allowing the sharing of photos and travel stories that are integrated with photos ("photo stories"). This platform is not unlike the sites Flickr\textsuperscript{11} and TripWolf\textsuperscript{12}, but distinguishes itself by providing a richer integration of content, support for different user roles, different perspectives on the content (photographer, traveller), customisability and personalisation (personal homepage, custom layouts and widgets), advanced search, etc.

In the scenario, we envision the following different roles, described in form of characters. Note that we will not necessarily support all these roles and functionalities in the final showcase implementation, the description just serves as a visionary setting.

\textit{Szaby} is going to travel to the Philippines. Still at home, he starts a new “trip” in the KiWi system. While he being on his trip, he takes many pictures with his digital camera. In some of the cities, he visits an Internet Café and directly uploads his photos onto the KiWi system. He also specifies the location where the pictures have been taken, quickly writes together the stories associated with the pictures, similar to a blog, and tags these items with tags that he considers relevant. The KiWi system automatically makes suggestions based on the content of the text and based on taxonomies or ontologies stored in the knowledge base. Using the locations of the photos, the KiWi system can also automatically create a trip route that Szaby can share with his friends or with other travellers. He is also able to describe hotels and restaurants and rate them for other users. After the trip, Szaby returns home and wants to create a photo book out of this information. For this purpose, he refines the texts he has written during the trip and extracts the relevant content for a photo book that he assembles using a custom tool provided by the platform. He can order his photo book directly from the webpage.

\textit{Julia} is going on a trip to the Philippines later this year, and wants to plan her route based on information on the KiWi system about things that are of interest to her. Since Szaby is on her list of friends in the KiWi system and has a quite high community equity ranking, the system automatically shows her Szaby’s recorded travel route, which she can copy into her own plan and modify according to her own interests. In addition, the system automatically proposes additional places of interest in the area based on the tags that have been associated with pictures and stories. The system also provides a flexible, faceted search interface that allows her to add and remove search criteria that are suggested by the system, or manually add keywords. When Julia is finished with planning her route, she is able to let the system generate a printable personal travel guide for the route. She can also share her route with her friends and let them comment on it.

\textit{Sebastian} is interested in photography, so instead of the role “traveller”, he accesses the system mainly in the role “photographer”. In this role, he is primarily interested in pictures and information about them, and not so much in travel route specifics, accommodation, etc. This time, he is looking for nice pictures of sunsets for a presentation he has to do. Using the search interface, he finds many pictures of sunsets taken by travellers and other users of the

\textsuperscript{11} http://www.flickr.com/
\textsuperscript{12} http://www.tripwolf.com/
system, because they are tagged appropriately. Since he has both Szaby and Julia on his “friends” list, their pictures have a higher community equity for him and are thus ranked higher in the search results.

Obviously, this scenario is very ambitious and goes significantly beyond what current social software platforms do. While it is not the aim of the KiWi project to develop a fully-working travel and photo platform, we will as part of the showcase implement at least part of the described functionalities and demonstrate how they could be extended to actually reach the vision.
3. User Interface and Usage Model: What Does KiWi Feel and Look Like?

The “look and feel” of the KiWi system consists of two parts: the main usage concepts that users will be exposed to, and the user interface of the system itself. While the usage concepts are mainly concerned with what kinds of content are represented inside the KiWi system and what can be done with them, the user interface describes how the user interacts with this content.

3.1. Core Usage Concepts: Content Items, Tagging & Annotations, Users

The KiWi system comprises only few core concepts that are exposed to the user. Central is the content item, which represents a “unit of knowledge” and can be seen as a generalisation of Wiki pages. Each content item has a URI\(^\text{13}\) that allows annotating it with RDF metadata for providing machine-readable knowledge, and a human readable content in form of text or multimedia data. URLs of content items are not arbitrary but can be used for directly accessing the content item in the browser or in open linked data\(^\text{14}\). Annotation of content items is possible using an advanced tagging mechanism and also directly using RDF triples. Central to the KiWi system are also revisions, allowing to revert any modifications that have been applied to the knowledge base.

In addition to the content-related concepts, the KiWi system also knows about users and roles. A user is a person (or other agent) interacting with the system. A role represents the environment the user is interacting in, e.g. “at home” or “at work”. Information about users is represented using special kinds of content items – user information is thus also identified by a URI and can be annotated. The KiWi system also provides support for social networking functionalities, and implements Sun’s community equity\(^\text{15}\) system for modelling the value of participation and representing reputation within the community.

The current content item, current user, and current role together form the current context of the KiWi system. The context determines what the user will see when accessing the KiWi system and how he can interact with it. The presentation will change when the context changes, e.g. when navigating to another content item or when switching role.

These core concepts are discussed in more detail in the following sections.

3.1.1. Content Items

Content Items are the core usage concept of the KiWi system. They represent a “unit of information” in KiWi, e.g. a page about a certain topic, a user profile, etc. When a user accesses the KiWi system, he is always interacting with exactly one (primary) content item, the context content item. The context content item can be viewed, modified, and annotated by the user. Though changes might also affect other content items, the context content item is always the primary content item.

Each content item has both, a machine readable symbolic representation and a human readable textual or multimedia representation. This correspondence is illustrated in the following figure:

\(^{13}\) Uniform Resource Identifier, http://www.w3.org/Addressing/
\(^{14}\) http://linkeddata.org/
\(^{15}\) http://blogs.sun.com/peterreiser/entry/community_equity_specification
Figure 2: Each content item has both, a machine-readable and a human-readable representation.

**Unique Identifiers.** One of the core assumptions of the KiWi usage model is that there is a 1:1 correspondence between a resource (URI) and a content item. This means that every content item can be uniquely addressed, e.g. for annotation or linking, and that every URI associated with the current instance of the KiWi system can be presented to the user as a content item. URIs also serve as machine-readable symbolic representations of a content item, to be used in *extended triples* (see below) in the RDF-based metadata store.

**Content.** Each content item is also related to human-readable content. Content can be either textual content or multimedia content (images, audio, video, PDF documents, ...). Textual content takes the form of a wiki page and can be edited easily, supports easy linking, and versioning. Multimedia content in KiWi is treated as a “black box”, i.e. it can be uploaded and versioned, but it is not intended as part of the project to provide editing of multimedia content from within the browser.

**Content Item Composition.** A content item can be presented to the user as a composition of several other content items. For example, a wiki page containing an image can be seen conceptually as a composed content item. Technically, this kind of composition is solved using an *inclusion mechanism*, using a special kind of link (include-link). Each of the included content items is still independent and has its own URI (and can therefore also presented as a wiki page of its own), although the including content items can be seen as context of the content item.

**Content Item Fragments.** Similarly, it is also possible to identify fragments of a content item, for example to annotate a sentence or a single word occurring in a text. Content Item fragments have their own URI, but are – unlike included content items – not independent of the containing content item (and can therefore not be presented as wiki pages of their own; if a content item fragment URI is the current context, then the containing content item is presented instead, perhaps highlighting the fragment). Fragments may overlap, as this is a property often found in textual documents.

**Special Types of Content Items.** The KiWi system recognises certain special types of content items, namely those representing users, roles, widgets, layouts, and rules. For these content items, the KiWi system provides specific presentation and editing widgets. Depending on the domain, specific instances of the KiWi system might provide additional kinds of special content items, e.g. for representing OWL classes and properties.

For example, consider a content item representing the “elephant” (see Figure 3). The content
item might have a URI of http://www.kiwi-project.eu/example/Elephant and a textual description for humans to read. In addition, there might be annotations saying that the elephant is an animal, has grey colour, and big ears. The presentation of the content item might include an image of an elephant (which is also its own content item) and a taxonomy box that was automatically generated from the annotations of the content item. The main content item would be the text describing the elephant, which includes the image as included content item; the taxonomy box is generated automatically.

Figure 3: Content Item about the "elephant" (from Wikipedia); contains a textual description and an automatically generated taxonomy box, and includes a content item representing the image of an elephant.

3.1.2. Tags and Folksonomy

Tagging is one of two ways of annotating content items in the KiWi system. In KiWi, tags serve many different purposes, for example associating content items with certain topics or grouping content items in “knowledge spaces”. There are two kinds of tags: explicit tags are explicitly added to a content item by a user; implicit tags are created by the system, e.g. based on automatic mechanisms like information extraction from text or reasoning on existing tags.

Conceptually, an explicit tag is a 3-ary (ternary, threefold...) relation between two content items (the tagged content item and the tagging content item) and a user (the tagging user or tagger). An implicit tag is a binary relation between two content items, a tagged and a tagging one. Tagging content items are identified using one or more labels that are available for annotating content items. In case of ambiguous tag labels (i.e. the same tag label for different content items), the KiWi system asks the user to choose the appropriate content item. If the user enters a new label that is not yet used elsewhere, it is displayed like a wiki-link to a non-
existing page; when the user clicks on it, he is given the choice to either associate the label with an existing content item or to create a new content item explaining this tag label. Internally, a tag is furthermore given maintenance information like creation time and date and a URI for uniquely identifying a tag.

For example, the content item that describes the fictitious elephant “Benjamin Blümchen” could be tagged with the label “elephant”, thereby associating it with the content item describing “Elephant”. The tagged content item would be “Benjamin Blümchen”, the tagging content item would be “Elephant”, and the tag label used for tagging would be “elephant”, which is a tag label of the content item “Elephant”.

Inside the system, a tag is mapped to an RDF structure that can be used for deriving additional RDF metadata by means of reasoning. Also, tags can be “lifted” to taxonomy or ontology concepts by advanced users (e.g. by using the “meaning of a tag” ontology\(^\text{16}\)), in which case more information about the meaning and context of a tag becomes available, e.g. for reasoning or querying. For example, the tag associating “Benjamin Blümchen” with “Elephant” could be lifted to an “is-a” relationship, and the tagging content item “Elephant” could be lifted to an ontology concept.

Tags can be used by the KiWi system for many different purposes. For example, tags can help with searching by offering a faceted search interface or by offering tag clouds. Furthermore, it is possible to derive user preferences from the tags she has used or to identify users with similar interests via clustering. Similarly, tags can also be used for grouping related content items, e.g. for defining group work spaces or for clustering thematically related items. Beyond that, the way how tags are used is left to the application developers and users that implement a specific instance of the KiWi system.

### 3.1.3. RDF Annotations

At the core of the KiWi system, all metadata is stored in RDF (Resource Description Framework)\(^\text{17}\). While RDF has earned a lot of criticism, it is a very flexible language and almost ideally suited for representing the relations between Web sites and thus content items. Using RDF at the core also allows integrating KiWi with the multitude of other existing Semantic Web applications, e.g. ontology editors, visualisation tools, semantic search engines, or other Semantic Wikis.

**Page and Link Types.** The second way to annotate content items in the KiWi system is thus by directly adding RDF metadata, a functionality already found in existing Semantic Wiki systems like IkeWiki. Since each content item represents an RDF resource, it can be associated with a type (OWL or RDFS class). In IkeWiki, this is done by a dialogue that is opened if a user clicks the “+” icon behind the title of the page. Likewise, it is also possible to annotate a hyperlink between the two content items, thus creating a new triple in the RDF store. In IkeWiki, this can be achieved by clicking the “+” icon behind a hyperlink occurring in the text. And third, new triples can be added explicitly by providing an appropriate editor (as the “add relation” dialogue in IkeWiki).

**Extended Triples.** Machine-readable meta-data in KiWi is represented using what we call “extended triples”. An extended triple conceptually stores a triple like in RDF, but contains additional “maintenance information” that can be used internally by the KiWi system for various tasks like versioning, transactions, associating a triple with a certain workspace, user, or group, or for reason maintenance (i.e. storing why a certain triple has been asserted). In principle, an “extended triple” can thus be seen as a “triple with attributes”.

\(^\text{16}\) [http://moat-project.org/ontology](http://moat-project.org/ontology)

\(^\text{17}\) [http://www.w3.org/RDF/](http://www.w3.org/RDF/)
Note that these attributes could also be represented using RDF triples and reification. However, such a representation has several disadvantages compared to the extended triples proposed for KiWi:

- it requires reification, meaning that the original triple, which seemingly provides the most interesting information, is broken up into parts that have to be reassembled
- it mixes up several levels of abstraction, which is not only inconvenient for machines and reasoning, but also for the user, as it clutters the knowledge space
- it is difficult to filter out the information that is used for internal purposes and that is not supposed to be exchanged with external systems

The implementation of extended triples is straightforward and fits in easily with already existing tools and standards without disguising the original meaning. Triple attributes containing maintenance information are only represented programatically inside the system. To the outside world, extended triples look like ordinary triples and can be exported into the usual Semantic Web formats like RDF.

**Hiding Complexity.** Annotating with RDF metadata does not necessarily mean that the user has to use an RDF editor or even know RDF. RDF annotations can also be added using e.g. “Semantic Forms” that completely hide RDF from the user (see below). The user then only sees an ordinary form-based editor where she can fill in certain fields. Internally, the system decides which values are stored as RDF and which as human-readable content.

### 3.1.4. Users

**User Profiles.** As the KiWi system is supposed to serve as a platform for social software, the concept of a user also plays an important role. At the core, a KiWi user has at least a login, a password, first and last name, and an email address. Each user has a unique URI and is thus represented in the system by a resource and content item. The content item serves as user profile page, which can be rendered and edited in a structured (e.g. tabular) fashion by providing appropriate Semantic Forms. Deployments of the KiWi system in a specific application area might choose to add additional information about e.g. address, skills, etc. Since the KiWi system is based on semi-structured RDF and XML data, no schema is prescribed. The KiWi system is thus easily extensible to new kinds of user data.

**Social Networking.** In addition to profiles, the KiWi system provides support to model social relations (e.g. by using Friend of a Friend\(^\text{18}\)) between users to provide basic social networking capabilities. In order to allow integration with other social networking platforms, the KiWi system strives to implement the Google OpenSocial API\(^\text{19}\) and will also expose its social networking data as RDF for other Semantic Web applications to use.

**Reputation System.** One of the most important traits of a community platform is to motivate

---

\(^{18}\) [http://www.foaf-project.org/](http://www.foaf-project.org/)

\(^{19}\) [http://code.google.com/apis/opensocial/](http://code.google.com/apis/opensocial/)
users to participate. Most of the successful social software platforms therefore have some kind of (usually implicit) reputation system: users who contribute frequently and with high quality get a higher reputation inside the community. While implicit reputation works well in open communities like Wikipedia, communities within enterprises have quite different characteristics, because they are smaller and because employees are less excited about sharing because “knowledge is power” (to which one should hold on) is still a frequent conviction. The KiWi system therefore aims to implement an explicit reputation system based on the Community Equity\textsuperscript{20} project developed by KiWi partner Sun Microsystems. With community equity, the value of participation, contribution, skills, and reputation within a community is explicitly calculated. Users participating and contributing more and of higher quality get a higher reputation. Quality is determined by peer-to-peer rating mechanisms. The value of contributions also decreases with time, reflecting the fact that information becomes less relevant when it is old and constantly motivating users to continue with contributing. The level of integration of Community Equity is still not decided and depends on licensing issues and available project resources.

**User Models and Personalisation.** The KiWi system will also offer a personalisation and adaptation component for tailoring the presentation of content to the interests and needs of the user. For this purpose, the KiWi system also automatically and manually maintains user models for each user. While the exact properties of the user models are not yet fixed, they will very likely allow to explicitly model user interests. Also, it is conceivable that interests are automatically derived and community memberships based on the tags a user has given and based on the content items he most frequently visits.

### 3.1.5. Roles

In every social interaction, a person can have many different roles and resulting different behaviour. For example, a person might have considerably different interests while in the role “at home” as compared to the role “at work”. The problem of a missing role concept is often found in Web applications: for example, when one uses Amazon to buy books both for private and for professional use, or both for one’s children and for oneself. The recommendations made by the system then often become inappropriate.

KiWi aims to reflect such different roles by an explicit role concept. Technically, roles can be seen as means to separate different user models for the same user. When a user switches to another role, the system uses a different user model as a base for adapting the presentation and editing of content. The user can configure roles explicitly by e.g. setting different preferences. Note that for simplicity reasons, the user profile as the identity of the user that is visible to other users stays the same in all roles.

This very straightforward role concept is also very flexible and can be used for many different purposes. The separation of “at home” and “at work” described above is one possibility. Another possibility could be system roles: a user could switch between “registered user” and “administrator” and get completely different options for working with the system. The role system could therefore also be used for access management.

### 3.1.6. Context

The currently presented content item, the current user, and the currently selected role together make up the context of the user’s interaction with the KiWi system. The context determines what content is displayed, how it is displayed, and in which ways the user can interact with it.

\textsuperscript{20} \url{http://blogs.sun.com/peterreiser/entry/community_equity_specification}
For example, in the photo stories showcase, the user “Szaby” (the current user) could browse to a photograph displaying a beach on the Philippines (the current content item). Since he is in the role “traveller” (the current role), the photograph is displayed next to a map that shows the travel route that the creator had taken when taking the photograph. When switching to the role “photographer”, the display could change and instead display metadata about the image (aperture, shutter speed, ISO settings) and give a list of related photographs.

3.1.7. Revisions

The last of the core concepts are revisions. A revision is a logical update to the KiWi system’s knowledge base, e.g. adding a photo, editing the text of a content item, or changing a user profile. A logical update usually involves updates in many different parts of the knowledge base involving both human-readable content and meta-data. For example, adding a photo to the KiWi system might involve the extraction of EXIF meta-data that is stored as RDF data. Similarly, editing the textual content of a content item might lead to a link being deleted, thus requiring to also remove a corresponding RDF relation.

This kind of revision system is much more complex than the revision systems found in normal wikis: whereas changes in normal wikis are always “local” in the sense that there is only one page affected, revisions in the KiWi system can involve many different content items. Revisions are therefore always attached to all content items involved. In this way, users can easily track the history of changes for each content item.

In KiWi, revisions are always explicit and under the control of the user. A revision starts when the user starts modifying the system, and it ends when the user explicitly clicks on “commit” or “save”. In cases involving several manual updates by the user, the KiWi system could support her by indicating (e.g. via a red status bar) that there are “pending commits” as soon as the first update is carried out.

Note that this revision concept might raise the need to develop a sophisticated transaction model to cover all situations (e.g. complex dependencies between revisions). This is, however, beyond the scope of KiWi. In KiWi, we strive to apply a pragmatic and simple solution for revision management.

3.2. Advanced Usage Concepts: Rules, Structured Tags, Endorsers & Rejecters

Beyond the core concepts described in the previous section, the KiWi system also features some more advanced usage concepts, which we will describe in the following. Advanced usage concepts are not necessarily available to the user in every KiWi installation.

3.2.1. Rules

A rule has, or is specified through, a “rule content item”, i.e. a content item of a specific form described below. A rule content item includes:

- a “rule code content item” (mandatory, nonempty, specified by users),
- textual content intended as explanation of the code (optional, specified by users)

The code of a rule is compelled to a certain rule syntax (to be defined with the KiWi query language).

Rules consist of a construction and of a query. A rule construction may specify:

- a content item
- the assignment of a tag to a content item
• the construction of an RDF subgraph to be added to the triple store

Rules cannot construct users, groups, or other rules.

A rule query can select content items depending on the data they carry and retrieve data from content items. Data means here anything from a content item: text or pictures within content items, tags assigned to content items, links from or to content items, RDF metadata associated with the content item, etc.

The tags assigned (by users or by the system) to the content item specifying a rule serve several purposes:

• they might contribute, together with the rule explanation, to explain the rule,

• they might serve to specify rule sets or reasoning contexts as follows: all rules with a same tag form a rule set and reasoning can be restricted to specified rule sets.

For simplicity reasons, no additional access right “executable” is possible for a rule. Indeed, this would make the reasoning extremely complicated and hardly tractable – both for humans and software.

3.2.2. Structured Tags

A structured tag resembles a record of a programming language, a possibly nested tuple of a non-first-normal-form relational database or an atomic formula of first-order logic. A structured tag is build up from atomic tags. More precisely, a structured tag consists of:

• a structured tag name which is an atomic tag,

• one or several attribute name-value pairs of the form attribute names - attribute values, each attribute name and each attribute value being an atomic tag.

Examples of structured tags are as follows:

| hotel(3stars downtown) |
| hotel(location(downtown)) |
| hotel(comfortable) |

Note that it should be possible to tag (a same) one content item with several of the above tags as well as with the atomic tag “hotel”. Note also that the tag syntax used above is used for convenience. A two-dimensional syntax such as in

| hotel |
| 3stars |
| downtown |

or

| hotel: 3stars, downtown. |

could be preferable for rendering.

Note that the structured tag name can be seen as a relation name. However, such “relations” do not have a fixed arity, i.e. the same structured tag name can be used in some cases as an atomic tag (for example “hotel”), in other cases as name of a structured tag with, say, two attributes (for example “hotel(3stars downtown)”), yet in other cases as name of a structured tag with, say, one attribute (for example “hotel(location(downtown))”).
3.3. User Interface: Widgets & Layouts

KiWi aims to follow a user-centred software development approach. Therefore, we begin with a “layman” description of what we think that the KiWi system might look like.

**Freely Arrangeable Layout.** The KiWi user interface will be composed of flexible and configurable “widgets” that can be tailored and composed according to the specific needs of the context, e.g. by graphically rearranging the layout as in iGoogle\(^1\) or by customising the layout based on the user preferences and user model. For example, a page showing biology content could automatically insert a taxonomy box widget in the top right corner of the text (see Figure 4), render the page in larger font because the user has “bad vision”, or provide an interactive widget for tagging biology resources.

![Figure 4: Taxonomy Widget showing additional information for biology content](image)

KiWi widgets are supposed to not only support presentation of content but also interaction with the system. An “editor widget” could thus provide the Wiki-style editing capability, a “tagging widget” the possibility to annotate content with free-text tags, and a “socialise widget” could integrate the KiWi system with other Social Software systems like Digg or Magnolia.

For this purpose, KiWi will provide:

- a *layouting language* (similar in spirit to CSS or similar languages; probably XML-based) that allows to define how widgets are arranged on a page;
- a *widget description language* that allows to describe visual and interactive properties of components; the language will either be used or build on the W3C working draft “Widgets 1.0”\(^2\)
- a *graphical, browser-based layouting tool* that lets users rearrange the layout of components using drag and drop or similar techniques (as e.g. in iGoogle, see Figure 5); such layouts will then be stored internally using the layouting language.

\(^1\) http://www.google.com/ig
\(^2\) http://www.w3.org/TR/2008/WD-widgets-20080414/
In the layouting language, conflicts will be inevitable. To resolve these conflicts, a simple resolution strategy will be selected, e.g. by assigning precedences to layouts.

Customisable Widgets. The KiWi system will come with a set of standard widgets similar to the UI components that are currently available in the IkeWiki system, i.e. an article view, an editor view, a box for outgoing and incoming references, etc (see Figure 6 below). In addition, developers and even users will be able to define custom widgets for presentation as well as interaction with the KiWi system.

- **presentation widgets** display some sort of additional information about the content, e.g. a biology taxonomy box, the references box, a navigation box, etc.
- **interaction widgets** allow the user to interact with the system, e.g. for editing content, for annotating content, etc.

All kinds of widgets will be stored as part of the knowledge base and are considered a special kind of content item. Users will be able to define their own widgets using a set of different languages:

- **simple HTML widgets** contain mostly static content and JavaScript; they can be used for basic functionalities like navigation boxes etc.
- **template-based widgets** are similar to HTML widgets but allow filling certain positions with values from the knowledge base; candidate languages are e.g. FTL, Java ServerFaces, or similar; such languages (e.g. Java ServerFaces) often also allow a certain level of interaction with the system
- **rule-based widgets** make use of the rule-based reasoning and query language developed as part of the KiWi system; users will be able to define a widget as a set of...
rules that generates the desired HTML output using content from the knowledge base

- **free-form widgets** allow to use “real” programming/scripting languages, e.g. Groovy, JSP, or JRuby, to define widgets; this gives experienced users the full flexibility of programming languages for providing more complex functionalities

---

**Figure 6:** customised references widget, implemented in different languages

For all kinds of widgets, the KiWi system will provide appropriate APIs (e.g. in the form of Java Beans for template-based widgets) for users and developers to access the content of the knowledge base. To support the user in creating widgets, the KiWi user interface will also provide editors for at least the most simple kinds of widgets (see “Zembly” below).

**Zembly.** Zembly\(^{25}\) is a new social software platform supported by KiWi partner Sun Microsystems. With Zembly, users can collaboratively and easily create widgets that can be used in a number of different social software applications, e.g. Facebook, Google OpenSocial, iGoogle, or even on Apple’s iPhone. In principle, Zembly is thus a “wiki for widgets”.

“Applications consist of widgets, which are reusable pieces of user interface, and services, which are reusable server-side logic that ties everything together” (Zembly website). Widgets are hosted at Zembly and primarily executed in the user’s browser using HTML, CSS, and JavaScript. Zembly provides an easy to use, browser-based editor for developing widgets, which supports the user by features like “preview” and “code completion”. Server-side services can be used to implement adapters to external services, like DBpedia\(^{26}\).

---

\(^{25}\) [http://www.zembly.com](http://www.zembly.com)

\(^{26}\) [http://dbpedia.org/](http://dbpedia.org/)
The KiWi system will integrate and support Zembly in order to provide users with a well designed and functional editor for creating their custom KiWi widgets. The level of integration with KiWi depends on the evolution and licensing of Zembly, which is yet unclear. In any case, we will provide adapters (services) for Zembly widgets to access KiWi systems, and we will allow Zembly widgets to run as KiWi widgets. However, widgets themselves will then still be hosted on the Zembly site. If the Zembly technology is released under a sufficiently liberal license within the project lifetime, we also strive to integrate Zembly directly into the KiWi system, so that users can use the same technology for collaboratively developing native KiWi widgets.
4. The KiWi Technology: How will it work?

The KiWi system will be implemented using the Seam Framework\(^{27}\), which is itself based on Java Enterprise Edition (Java EE 5) platform. According to the website, “Seam is a powerful open source development platform for building rich Internet applications in Java. Seam integrates technologies such as Asynchronous JavaScript and XML (AJAX), Java Server Faces (JSF), Java Persistence (JPA), Enterprise Java Beans (EJB 3.0) and Business Process Management (BPM) into a unified full-stack solution, complete with sophisticated tooling.”

The Seam Framework is a representative of a new kind of frameworks for developing web applications that follow the model-view-controller (MVC) paradigm, where data access (model), business logic (controller) and user interface (view) are strictly separated. Consequently, the KiWi architecture also follows a layered MVC architecture as shown in the following figure:

![KiWi Technology Architecture Diagram](image)

We introduce the different layers of the KiWi architecture in the following, together with a description of the technologies we are likely to use.

\(^{27}\) http://www.seamframework.org/
4.1. Model: Persistence & Data Model

The KiWi system will provide as part of the framework a storage backend as well as an advanced programming interface (API). The purpose of the storage backend is to provide unified access to several different kinds of data: content, meta-data, rules, as well as layout and widget definitions. The purpose of the API is to provide a clean programming model that allows to access and work with the different kinds of objects that are stored in the storage backend, e.g. for giving access to resources and extended triples to widget developers.

The KiWi system will provide a data model (implemented as Java EE entity beans) that provides a convenient API to concepts that are frequently found in social software systems. While following an object-oriented programming model the KiWi API still provides access to content and meta-data in their native forms (XML and Triples) and via the querying and reasoning language developed in the project. More specifically, the API will define interfaces for the following kinds of data:

- **content items;** the content item class connects an RDF resource with textual or media content and gives access to properties like ‘author of the current revision’, ‘date’, ‘title’, and ‘language’
- **resource descriptions,** providing access to URI, textual descriptions, as well as additional basic information; following the “high cohesion” software design pattern, resource descriptions will also offer access to meta-data that is relevant for the resource (e.g. type, incoming and outgoing triples, ...)
- **textual descriptions,** giving access to the XML tree in appropriate representations (e.g. DOM/XOM and string representation), different revisions of the same content, different language versions, authors, etc.
- **tags;** each tag is represented by a separate instance of the tag class, which gives easy access to the tagged and the tagging content item as well as the tagging user; algorithms for counting and clustering allow to perform statistical analysis over tags
- **extended triples** (including the graph structure defined by them), allowing to navigate through the relations in the knowledge base; in many cases, it might also be appropriate to define higher level constructs for representing compound concepts like classes/types, n-ary relations requiring intermediate blank nodes, etc.
- **users and roles,** providing an interface to identity management, permissions, user models and preferences, personal/group knowledge spaces (see below), etc.
- **layout definitions,** making it possible to easily access the defined widgets and render the defined layout into a format suitable for the browser (i.e., HTML + JavaScript)
- **widget definitions,** providing means to access, store, and layout the widgets properly, as well as execute them to perform their respective functionalities;
- **rule definitions,** making it possible to access and define the properties of individual rules as well as sets of rules that belong together, execute rules and enable/disable them on a per-rule and per-user basis, etc.
- **community equity;** the KiWi system also provides for each user and content item the community equity associated with it
- **revisions;** for each content item, it is possible to access the list of previous revisions that updated it; revisions themselves are logical containers consisting of content or metadata updates
- **search and querying,** allowing to directly execute queries on the currently relevant
state of the knowledge base using full-text search as well as the query- and reasoning language defined in KiWi.

While some of these items might seem too specific (they could be handled by other means of storage), defining them provides a convenient way of accessing such data by widget developers and other developers modifying the system.

Additionally, the KiWi storage backend will provide indexes as required for efficient operation (e.g. full text or specialised indexes for accessing extended triples). For technical and performance reasons, the different kinds of resources might be stored in different database systems or specialised means of data representation (e.g. custom triple store). The storage backend provides sufficient abstraction so that (1) the (e.g. widget) developer who needs access to the system does not need to care about the means of storage, and (2) it is possible to switch the specific storage system without needing to also modify the API.

Persisting in the relational database works through the Java Persistency API and thus supports many different databases easily. Whereas the development environment runs on an embedded HSQL database\(^{28}\), productive installations will support both, the MySQL\(^{29}\) and the PostgreSQL\(^{30}\) system. RDF triples are stored using the Sesame 2 triple store system\(^{31}\). Sesame 2 provides a very flexible API that can be extended e.g. by custom reasoners, and the data can be persisted either in simple files or using a separate triple server that can be accessed by the KiWi system as a client. The triple store is preloaded with a set of RDFS or OWL data models (ontologies) that represent a specific configuration of the system.

The decision which information is stored in a relational database and which is stored in the triple store is based on the following rule: if the kind and structure of the information is relevant throughout the complete lifetime of the system or even in all systems, then it is persisted in the relational database; if the structure of the information is a “configuration issue”, it is stored in the triple store. For example, the concepts “user login” and “user password” are applicable in all systems; therefore, this information is stored in the relational database. On the other hand, the relation “knows” to other users in the social network is a configuration issue and thus stored in the triple store.

4.2. Controller: Seam, EJB, and the KiWi API

The KiWi API implements the logic of the system and provides a unified access to the data model, so that developers do not need to care about how KiWi entities like content items and users are managed. Typical functionalities provided by the KiWi API are e.g. creating, accessing, and storing content items, tagging, logging in users, and searching and querying. Since KiWi is using the Seam Framework, the KiWi API also directly performs access and permission management on a per-method level. The controller part of the KiWi system will be implemented using Enterprise Java Beans (EJB) and Seam Components. This provides a lot of flexibility, e.g. for running the KiWi system in distributed or load-balanced environments using Java EE servers.

**Service Endpoint.** The *service endpoint* is KiWi’s interface to other systems. The purpose of the service endpoint is that an installation of the KiWi system is able to exchange data with other tools on the (Semantic) Web without sacrificing significant parts of the information. The service endpoint will be realised by the following functionalities:

- a Web Service interface that exposes most of the functionalities of the KiWi API not

---

\(^{28}\) [http://www.hsqldb.org/](http://www.hsqldb.org/)

\(^{29}\) [http://www.mysql.com/](http://www.mysql.com/)

\(^{30}\) [http://www.postgresql.org/](http://www.postgresql.org/)

\(^{31}\) [http://www.openrdf.org/](http://www.openrdf.org/)
only to the user interface but also to external applications, using the functionality already provided by the Seam Framework.

- the integration of external content via Web Services, e.g. for accessing the document server of a company. KiWi provides an extensible framework for accessing and integrating external services.
- an RDF/OWL export/import interface that allows to export/import the knowledge base as a whole or as clearly defined parts in RDF/OWL format so that the content of the knowledge base can be further processed using Semantic Web tools like Protégé or similar.
- an XML-based representation format (“wiki interchange format”) that allows to exchange wiki pages, including semantic annotations, with other Semantic Wiki systems (e.g. Semantic MediaWiki as further developed in the project ACTIVE, or even other ordinary Wiki systems)
- a SPARQL Web Service endpoint that allows to directly access the content of the KiWi knowledge base by issuing SPARQL queries (including updates).

**Widget Execution Environment.** The widget execution environment provides for each of the widget languages a natural interface to the KiWi system. In the first stages, this will be achieved for the languages JSF (Java Server Faces), Groovy, and JavaScript using appropriate extensions provided by the Seam Framework. In later stages, we might add execution environments for additional languages, e.g. the rule-based reasoning language provided by the enabling technologies.

**Personalisation.** The KiWi API integrates the personalisation and adaptation component so that it is available to all applications that build on top of the KiWi system. Central aspects of the personalisation component are the management of a user model for each user interacting with the KiWi system and the automatic selection of appropriate layouts and content visualisations based on this user model.

**Information Extraction.** The information extraction component analyses different kinds of content stored in the KiWi system to extract metadata from text (e.g. by entity recognition) and media (e.g. EXIF data from images). This metadata will either be stored directly in the KiWi knowledge base or proposed to the user as a possible annotation. Since information extraction might involve several different steps for different kinds of content, the information extraction component will be realised as a pipeline that allows interaction with the user.

**Reasoning.** A further important aspect of the KiWi storage backend and API is the reasoning component, as reasoning provides the means to process and recombine formal knowledge and thus to “intelligently” support the user. The main properties of the reasoning component are the following:

- KiWi reasoning is rule-based. The rationale behind this is that rules are more natural to express by humans than other kinds of “implicit” reasoning, that rules are flexible to use (they can serve many different reasoning purposes), and that rules can provably be evaluated efficiently.
- KiWi reasoning has a simple, intuitive (procedural as well as declarative) semantics that is easy to follow by humans and sufficiently easy to compute by machines. This might mean that expressiveness could be sacrificed for the sake of simplicity and ease-of-use (e.g. only limited forms of recursion).
- KiWi reasoning provides access to both, content and meta-data. The rule-based language can thus be used to combine (structured) content with meta-data stored in triples. The reason for this is that reasoning is meant to provide support to the user, which requires accessing content and meta-data alike.
• KiWi reasoning will not provide specific support for OWL or RDFS reasoning. The most significant parts of the OWL and RDFS semantics will be modelled using appropriate rules that can be enabled/disabled as required.

• KiWi reasoning is modular and flexible in the sense that users can enable/disable rules or rule sets either for the whole system or only for their personal view on the system. For example, when a user is annoyed by a certain layout decision that is the consequence of a rule, he must be able to disable this rule for his personal view, but possibly leave it on for others.

• KiWi reasoning is transparent to users as well as developers. When using the API, it is possible to access the derived (extended) triples as well as the base (extended) triples and the maintenance information associated with each triple. Note that this does not necessarily mean that the reasoning language itself has access to such maintenance information.

4.3. View: the KiWi User Interface

The KiWi user interface is implemented as a specific configuration of layouts and widgets. As mentioned in previous sections, widgets can be implemented in various different languages. The core implementation language will be Java Server Faces / Facelets in combination with the user interface libraries already provided by the Seam Framework (RichFaces and Ajax4JSF). Additional widgets might be implemented in languages like Groovy, JavaScript, or the rule-based KiWi reasoning language together with HTML.

The plain KiWi system will come with a set of core widgets and a default layout configuration that provides the basic wiki functionality. The default layout will resemble the IkeWiki layout, which is similar to the layout found in many other wiki systems. The following widgets will be provided as part of the core system:

• **view/edit content item**: this widget implements the main display and editing of content items; the editing part provides an easy-to-use rich text editor that supports the user in layouting and automatically analyses the text for possible links and annotations

• **history**: enables support for listing, comparing, and restoring previous versions of a content item

• **rating**: provides the functionality to users to rate content items; rating will be backed by the community equity system implemented in KiWi

• **tagging**: allows to tag a content item and to display tags in a tag cloud; other widgets in the system might use the tagging functionality to query or display information

• **annotation**: allows to annotate content items and links between content items with RDF types as in other Semantic Wikis

• **login/registration/logout**: implements the basic user access functionalities

• **user profile**: allows users to configure their personal profile; part of this widget are also the social networking capabilities, e.g. to add friends or roles

• **search**: provides full-text search capabilities and a facettted browsing interface

• **navigation**: provides a simple to configure table of contents for the KiWi wiki

• **layout configuration**: allows users and administrators to reconfigure the page layout using a graphical layout editor that allows adding, removing and rearranging widgets on a page
• *widget editor*; allows users and administrators to edit or upload custom widgets, e.g. implemented using Zembly or JSF

• *management interface*; provides options for configuring the system, e.g. permission and user management, ontology management, etc.

In addition to these core widgets, the two use cases and the showcase will implement additional widgets and layouts as needed by the respective environments.
5. Enabling Technologies: What kind of research will KiWi do?

Motivating Scenario. Consider using a wiki in a software project for the collaborative management of the project itself, of the software specifications and of related tasks such as scheduling meetings. On the one hand, well-specified practices - for example the accounting rules of the company - will most likely have to be followed. On the other hand, the wiki will have to be used even before the core concepts and structure of the software to develop are specified. It is therefore a first core objective of the KiWi wiki to provide support for working with both: well-designed ontologies that are already available and emerging ontologies that are not yet satisfactorily designed.

Supporting existing ontologies. The approach chosen for the KiWi wiki to support existing ontologies is to make the wiki aware of RDF and of a limited amount of OWL concepts. This choice has been made for pragmatic reasons: RDF is relatively easy to understand and use and becomes more and more widespread.

Assistance in the design of ontologies. The KiWi wiki approach towards providing assistance in the design of ontologies is first to support it with social tagging, second to enhance tagging with reasoning and information extraction. Social tagging is a well-established approach for annotating content that does not require any predetermined methodology. Informal tags can, as the collaborative work progresses, be turned into more formal annotations that, eventually, can be turned into RDF statements.

It is not a project objective to develop advanced technological solutions for an automatic evolution from informal tags to well formalized ontologies. Instead, it is a conscious choice of the project to leave the control of this evolution with the (human) users. Thus, the project only aims at providing a platform for the tracking of this evolution.

Tagging will be enhanced in the KiWi wiki in two complementary manners: First through reasoning, second through information extraction.

Enhancing tagging through reasoning. It is often useful to specify simple semantic relationships between tags. For example, a tag "bug" or a tag "revise" assigned to a specification can imply a tag "todo". Such simple rules are especially useful in a professional context. The KiWi wiki will make it possible on the one hand to specify such implications using rules (like, for example, "bug --> todo" and "revise --> todo"), on the other hand to automatically generate so called implicit tags (such as "todo") for tags (such as "bug" or "revise") explicitly specified by human users.

Enhancing tagging through information extraction. Often, the tags that can be assigned to a wiki page can be derived from the content of that page. "Information extraction" is the name given to a field building upon natural language processing techniques. It can be seen as a process which involves semantic classification of certain pieces of information stated in one or more documents. Information extraction can, in its full potential, be considered as a light form of text understanding. Information extraction will be deployed in the KiWi wiki in order to suggest to the (human) users tags that could be added to wiki pages. Once again, it is a conscious choice of the project not to strive for a fully automated tagging but instead to leave the control to the (human) users.
As an example, consider the following sentence (assumed to appear on a wiki page):

"Alice will travel to Salzburg on August 21st in order to inspect the collections of the Mozart Museum."

The system can suggest that "Alice" is a person, "August 21st" is a date and "Mozart Museum" is a place. Furthermore, the system may suggest that Alice is in a relation "travel to" with "Salzburg" and "August 21st" and in a relation "inspect" with "Mozart Museum". Thus, the tags are offered by the information extraction systems. If they are not correct, then the user is able to amend them. Provided that such content is annotated properly, an event such as as mentioned in the example sentence can be automatically added to a calendar, which could be an additional component of the wiki. Another tool added to the wiki could aggregate all places "Alice" has visited. A further additional tool could show such places on a map.

5.1. Reasoning and Querying.

In the KiWi wiki, reasoning will not be limited to generating implicit tags. Indeed, reasoning is useful for generating views on the wiki content. Views are useful for restructuring contents and for personalization/adaptation. Consider for example wiki pages describing various tasks in a software project. These pages might contain both technical information and budget information. A view might be desirable that aggregates all budget information for a project. Another view might collect contact data of the persons involved in a task or in the whole project.

Views are conveniently expressed by rules building upon a query language. A ruler thus has the form: "Query --> NewPage". Thus, selecting a user-friendly query language is an issue of concern for the KiWi wiki. Indeed, even though the KiWi wiki users might have programming skills, it is desirable to keep the level of technical skills needed for using the KiWi wiki as low as possible.

If reasoning is used, then consistency is an issue, i.e. a policy is needed in case contradictory information is derived. The KiWi wiki will track inconsistencies so as to present them to the (human) user as along with the reason for their occurrence (i.e. the pages, tags and rules that led to inconsistencies). For the following two reasons, the KiWi wiki will not apply a technological approach to avoiding inconsistencies: First, the (human) users should be in full control of the data and meta-data: second, in collaborative work as well as in work in progress, inconsistencies must be expected. It is a conscious choice of the KiWi project to let the (human) users handle inconsistencies. In other words, in the KiWi project, inconsistencies are not considered harmful. Instead, they are considered a fully normal step in both collaboration and work in progress that the users, not the system, have to deal with.

5.2. Reason Maintenance.

Reason maintenance is firstly about tracking why some conclusions have been derived, e.g. why the system has added an implicit tag "todo" to a web page (for example, because the web page has a tag "bug" and there is a rule "bug --> todo"). Reason maintenance is secondly about undoing reasoning that no longer holds after an update, e.g. the removal of the implicit tag "todo" after a user has removed the tag "bug". Reason maintenance is necessary because reasoning, even in its simple form, might involve many data and rules, thus becoming difficult to follow. Reason maintenance is also needed as the versioning technology of reasoning systems. Reason maintenance has been studied in the past but rarely applied to practical systems. Recently, a few researchers have begun to investigate its potential for social media in general, and wikis in particular.
5.3. Personalization and Adaptation.

Being based on the wiki philosophy, the KiWi system adopts a “publish – then filter” approach. The filtering approach can be looked at from two perspectives: users’ interests and users’ needs imposed by their work context. Consider again a wiki deployed for the management of a software project and used by managers and software analysts producing software specifications and similar tasks. Some users will not always be interested in all kinds of contents. A human resource manager, for example, might be interested in seeing who has contributed to what tasks over a period of time. An accountant might be interested in the delivery dates of software components and in the hours spent by programmers on these software components. In general, such information will be distributed over various wiki spaces and pages.

One can look at this problem from a different perspective: The data and knowledge maintained in the wiki grows over time and at some point will become even undesirable and it will become impossible for one person to grasp all the data and knowledge available. Personalization aims at helping from this perspective, too, as it filters out information, which could confuse, overload or otherwise distract a user in a task.

Personalization is in general a decision based on what is most useful for a particular user or a community of collaborating users in specific situations. Therefore, personalization should rely on the content’s semantics. One might expect, though, that the information is recognizable - either automatically using information extraction techniques, or semi-automatically using explicit tags (assigned to contents by (human) users) and implicit tags (assigned to contents by the reasoner). Using specific rules, it is thus possible to present this information to (human) users that fit to their interests and/or roles. The common name for such a data presentation is “personalization”, even though “adaption” to tasks and roles would often be a better name. The semantics of information will bring knowledge necessary to decide on:

- which links should be included or suppressed on the user interface or in the content in particular situations
- which facets are relevant for particular situations
- what are suitable configurations of all those

We expect the links, facets, and their compositions to have semantic annotations for example maintained in tags on their purpose in domains’ installed on the KiWi by means of the flexible ontologies. In the KiWi project, the relevant domains are the domain of software project management and software development. Every page, every fragment, every knowledge item is contributed and evolved by people. Contributions as the main semantic creation activity actually carry another meaning, the meaning of who is interested in what, who likes to contribute more, who is corrector rather than a creator, who has which style and so on. Furthermore, collaborations are established on particular islands of knowledge in KiWi either explicitly or implicitly by interest indicated group formations. This forms the semantics on users and groups which is another aspect which is taken into account in personalization. In fact, the above motioned aspects of knowledge are compared with the knowledge about other resources to identify the most suited configuration of facets, recommended links and so on.

The personalization decisions are in fact additional annotations in knowledge space, or other inferred facts related to recommendation or personalization task. Therefore, the personalization is based on the capabilities of a reasoning language (described above/below). Furthermore, explanations or reason maintenance will help to indicate if some personalization assumptions have been change. For example, if there is a new knowledge about person’s contribution to an area the KiWi system did not know about that may change the conclusions of the personalization reasoning. To not confuse a user, the KiWi system will
signal these changes and explain them that user or the other group members know the reasons behind the reorganization of the user interface.

The main aim of the personalization task is to ease exploration and search of growing knowledge in the wiki. The KiWi system will support, e.g., that the source code modules relevant for a feature a programmer is working on while also looking on other solutions. The software architects will be supported by facets pointing to requirements and design parts relevant to currently discussed items and plans. Project managers will be notified about other items in project document type hierarchy when changing for example risk items or changing requirements. Every change added to the system will get linked into the other work products or an issue for creating such a work product. KiWi will be able to point to different relevant experience contributions about planned solutions either relevant to proposals, requirements, architecture, or similar. Personalization will thus contribute to just in time access to information necessary to the performed task. Furthermore, as the facets arrangements are personalized on the user interface, the arrangements itself will be computed according to personal preferences and requirements on the task that user can access quickly what is necessary and can even react and contribute effectively with own experience.

5.4. Information Extraction

Information extraction is defined as the process of selectively structuring and combining data that are explicitly stated or implied in one or more documents. This process involves semantic classification of certain pieces of information and is considered as a light form of text understanding. In the context of KiWi, the structured information is to be used as a basis for reasoning and personalization/adaptation of the content. It will allow sophisticated querying to the content. Such queries will help to localize and summarize precise information, rather than just to retrieve a list of the most relevant documents.

The problem of transforming unstructured information into structured one can be solved by assigning special tags, called annotations, to certain pieces of the unstructured data. Annotations generally provide metadata describing the content of individual entities in a text.

IE systems of the past used to be based on a set of handmade rules to identify about which entities should be annotated and which type of annotation should be assigned to them. Though such systems can provide satisfactory results in particular applications, they are not suitable to be generally applied in KiWi. In particular, the extraction rules are specific for the target application and cannot be easily ported to different domains. Moreover, the creation as well as “tuning” of these rules presents tedious work, which needs cooperation of domain experts and knowledge engineers. All these drawbacks led us to the decision to build extraction system able to learn extraction rules semi- or fully automatically in KiWi. This will require the system to be given a set of annotated examples based on which it will learn what and how to annotate in the “unseen” text.

Even though the current IE technology cannot provide 100%-correct annotation results, even the systems without a near-perfect precision can significantly improve user experience in many real applications. KiWi will provide easy-to-use interfaces enabling users to validate the annotations suggested or automatically generated by the system.

The output structures of the KiWi IE components also need to reflect the requirements on the smooth integratability and portability. That is why we will prefer ontology-based

---

descriptions over the traditional rigid template-based specifications of the output.\textsuperscript{34} The ontology nodes represent the conceptualizations and describe the annotation types that should be assigned to specific entities found in the text. Furthermore, when the ontology is populated with annotated entities found in texts, the reasoning engine can take the new facts in the ontological structure into account and derive new knowledge based on them.

In addition to the annotation component sketched in the next subsection, KiWi will enhance the system with classification and clustering modules. Classification will be used to categorize newly created pages; clustering will enable similarity-based search of KiWi resources.

\subsection{5.4.1. Annotation component}

One of the challenges in KiWi is to develop an easy-to-use tool that will integrate the standard text editing with the manual annotation and the validation of the annotations suggested by the system. The user should be able to decide how the annotations to be validated will be presented and what is the lowest level of precision she accepts for the fully automatic annotation without her intervention.

The annotations can cover generic “is-a” or “part-of” relations as well as complex, in-domain relations describing, for example, that person $A$ uses a tool $T$. In general, the classes can be hierarchically structured and connected by various nontrivial relations.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{ontology_diagram.png}
\caption{An example of the information extraction task}
\end{figure}

During the annotation process, the entities and relations of interest will be associated with nodes and edges of an ontology. This will allow users to query for various semantic entities and relations. It will also enable the reasoning engine to produce summarized answers. Moreover, the entities associated with the nodes and edges in the ontology will serve as examples for the rule acquisition techniques used in the heart of the information extraction system.

\textsuperscript{34} A. Maedche, G. Neumann, and S. Staab. \textit{Bootstrapping an ontology-based information extraction system}, 2002.
Consider now the example in Figure 8. There is a query for employees that are experts on Java ME. The personal pages of the developers have been processed by the information extraction component. The reasoning takes these annotations into account and infers that Bill Rodgers has Java ME programming skill, even though this information has not been explicitly stated in the knowledge base.

5.4.2. How to make the IE subsystem easy to use

From a conceptual point of view, information extraction systems distinguish two phases: the training phase and the deployment phase. In the training phase the system acquires a model that covers a given set of annotation examples. In the deployment phase, the system identifies and classifies relevant semantic information in new texts, i.e., texts that were not included in the training set.

Machine learning plays a central role in the current information extraction paradigm. A predominant approach is to annotate large texts with the information to be extracted, and then use a learning procedure to extract some characteristics from the annotated texts. Unfortunately, this approach usually requires a large tagged data set. It is unrealistic to expect that KiWi users will provide this kind of data to make the system “semantics-aware”.

Therefore, we will explore combinations of the standard supervised algorithms with the methods that are able to learn from untagged texts. It will take advantage of bootstrapping, which refers to the technique that starts from a small initial effort and gradually grows into something larger and more significant. One of the methods that rely on this principle is the expansion. A small seed of extraction rules (learned from few examples) is applied to the unannotated corpora in order to discover new extraction rules. Newly discovered entities that are considered sufficiently similar to other members of the training set are added and the process is iterated until no other rules are discovered.

Another approach we are going to apply in KiWi is active learning. In active learning, the system itself decides what are the best candidates for annotation in order to maximize the speed of the learning process. A user is then asked to annotate these instances only. The idea of active learning perfectly fits the wiki philosophy that every user can annotate every page for which she has sufficient rights. All changes are naturally reported and there is no problem to come back to a previous version in case somebody made inappropriate annotations.

5.4.3. Summary on the role of information extraction in KiWi

Information extraction is one of the enabling technologies which will allow KiWi users to easily “semantify” the content by structuring it and adding various annotations. A key feature in this respect is the annotation component, which will integrate text editing with semantic tagging. KiWi will also provide similarity-based search and semi-automatic classification of wiki pages.

---

6. Perspectives and Conclusion

6.1. Conclusion

This deliverable summarises what we call the “KiWi Vision”: the goals of the project, possible usage scenarios, core concepts of the KiWi system, how we will implement the system and which technologies we will use, and finally which research will be undertaken by the project. Together, this should give a comprehensive summary of the project and provide different kinds of readers with the information about the project that is relevant to them.

One of the goals of this deliverable is furthermore to use it as a base for further dissemination material like information brochures, fact sheets, posters, etc. It should also help participants in the project to represent the project at conferences, exhibitions, trade fairs, etc. and be able to speak about the overall goals of the project in sufficient detail.

Finally, the deliverable also aims to provide a common understanding of the project to the individual project partners, so that the likelihood of different project strands developing in different directions is reduced.

6.2. Beyond KiWi: Further Scenarios Where KiWi Could be Useful

Beyond the use cases and showcase considered in KiWi, the KiWi system will be applicable in and adaptable to a multitude of different applications. In the following, we briefly sketch several such areas. Some of the described systems already exist (e.g. tagIT) but we believe that solving them with the KiWi system might result in a better system. Other systems are imaginary but nonetheless in our view of the KiWi system realistic.

6.2.1. tagIT: The Youth Atlas of Salzburg

“Tag” your own Salzburg – this is the motto of the project tagIT, calling upon youths to tag their favourite locations on the Web. The best skate parks, the best opportunities for climbing, the best cafés, and many other locations in Salzburg are just awaiting to be tagged on the tagIT website.

A “tag” in tagIT is basically a piece of information that is associated with a location on an online map based on Google Maps. Such information can consist of a textual description, multimedia files (images, audio, video), a description how to get there, and additional comments by other users. Additionally, such tags are associated with one or more categories in a simple hierarchy and can be searched by multiple different means, including full text, category, and location. And most importantly: Anyone can add his own new tags, comment on tags made by others, and upload additional media files. First experiments with a mobile phone based client have even shown that the real world and the Web can be combined to allow new, interesting forms of communication. The current tagIT user interface is depicted in Figure 9.

36 http://tagit.salzburgresearch.at
While looking completely different from a Wiki at first glance, tagIT actually pretty closely follows the Wiki philosophy described in Section 1: a tag is in principle just like a Wiki page, just presented differently as a location on the map; everyone can submit a tag, everyone can contribute to a tag, everyone can attach/link to comments and multimedia files, etc. Additionally, through the association of tags with categories and locations, a tag is also annotated with quite some semantic annotations, although this information is in the current implementation not available as explicit annotations using Semantic Web formats.

Consequently, the KiWi system as a combination of Wiki philosophy and Semantic Web technologies would be an ideal platform for implementing a “tagIT 2” system, where tags are even further connected and many of the deficiencies of the current implementation could be addressed due to the use of standards for representing knowledge and due to the advanced functionalities provided by KiWi. For example, tagIT would immediately profit from versioning and the representation of categories as ontologies, and it would be comparably easy to add social networking capabilities (relations to friends) and personalisation (highlight tags that have been added by my friends). In addition, the coupling of tagIT with a real wiki could make it easy to add and link to any kind of background information that might be relevant but is not directly associated with a location, e.g. about persons, associations, etc.

6.2.2. Semantic Blogging: Linking People, Blog Posts, and Wiki Articles

A weblog (short: blog) is a software system that allows users to publish online articles in form of a personal, chronological diary. Blogs are nowadays very widespread and serve as a means to communicate with friends, within communities of interest and practice (e.g. photoblogging), or even as a new medium for online journalism. The basic trait of every blog
is still the reverse chronological presentation of content items, and – on a more social level – the personal, subjective writing style.

A semantic blog is based on the idea that blogs already contain a lot of structured information. For example, articles usually refer to certain well-defined topics, articles are usually associated with one or several categories, articles are tagged with free-form tags, articles are linked with each other and in between several blogs, etc. In a semantic blog, this information is represented explicitly in the form of Semantic Web data (e.g. using the ontologies FOAF and SIOC), which can be exported to other systems or used for improved search and navigation.

Even given the differences described above, wikis and blogs share many commonalities on a technical level. Both allow rather free editing, both consist of textual content with a specific mark-up language, both provide versioning and linking, allow updating, etc. The difference is mainly in how the technology is used and not how it is realised. Given these commonalities, it is comparably easy to see how the KiWi system could be used to power a semantic blog, as this is primarily a matter of presenting the data in the right fashion (reverse chronological order, widgets for “typical” blog services like blog roll, category list, ...). In addition, a semantic blog realised with KiWi could allow for much richer kinds of blogging by combining blog posts with background information that follows a wiki style presentation, and by including certain social networking features based on users and roles in KiWi. And finally, semantic features like personalisation, information extraction, and semantically supported search and navigation are beneficial for a blog like for any other content-based Web system. (Hier könnte man auf das blogplugin zemanta verweisen, da ja auch auf zembly früher verwiesen wurde)

6.2.3. The Newspaper of the Future: Citizen Journalism meets Professional Journalism

As part of its activities in the “Salzburg NewMediaLab”, the project coordinator Salzburg Research is cooperating with the local newspaper and publisher “Salzburger Nachrichten” in developing the “Newspaper of the Future”. The goal of this project is to develop a new kind of news platform where professional journalists and interested users can together build a better, more interesting news site. The motivation behind this is that (1) articles written by ordinary readers are perceived to be more authentic by other readers than “professional” articles, (2) ordinary readers are often much faster at the location of an interesting event, and (3) mobile phones and widespread Internet access now make it possible and – in fact – easy to participate in the production of news content. For example, Salzburger Nachrichten would profit considerably from photos and stories collected by witnesses in case of a flooding in a remote mountain valley where it is difficult to bring journalists.

To further this aim, the project “Future Content Platforms” tries to integrate several different kinds of content: professional news articles, journalist blogs, reader blogs (reader stories), photo galleries, a wiki (background information), and an event calendar. All kinds of content are supposed to be connected with an “event” usually characterised at least by location and time. Obviously, this information is easily encoded using Semantic Web technologies.

In addition to that, the project investigates new kinds of navigation, search, and access to an online news site. For this purpose, each event for which one or more news contributions exist is supposed to be annotated according to the LATCH principle: by location, by alphabet, by time, by category (kind of event), and by hierarchy (topic). Users then have these 5 different dimensions as means of retrieving and navigating through information.
A major obstacle in the realisation of the project to date is the lack of a sufficient community-based framework or system for managing news content that takes into account both, the community aspects as well as the semantic features underlying intelligent search and navigation. Existing solutions are not only expensive, they are also not flexible enough to cope with community aspects, different kinds of content, as well as semantic technologies.

With the KiWi system, it would be comparably easy to build up such a system. The main challenges would then be to (1) develop a (mostly social) concept for realising the vision, to (2) implement this concept in the form of appropriate KiWi UI widgets and layout, and (3) to implement appropriate knowledge models (ontologies and rules) to model the LATCH principles.

6.2.4. Historical Archives: Collaboratively Managing and Creating Knowledge from Rich Cultural Assets

Much of the world’s history is up till now collected in libraries and archives throughout the world in the form of textual documents, drawings, photos, and nowadays also video and audio material. Libraries and archives are also more and more striving to digitize their content in order to make it more easily accessible. Even then, most of this content only represents small traces of history and does not really tell much about what really happened, i.e. the stories behind the artefacts that are collected is not told.

In order to make history “living”, it is thus necessary to involve the people of the world in collecting history by telling “their” stories. For example, the photo archive of Daimler AG consists of thousands of photos of different car models that have been built throughout the history of the company. While technical information about these models is available, people are more likely interested in the stories associated with the photos and models (“this is the car I drove to Italy with in my summer vacation 1959”). Beyond stories, people are also capable of contributing to archives with other media like their own photos (e.g. from their grandparents’ photo archive), films, etc. The following figure (from the SNML project “Smart Media Archives”) demonstrates what such archives could look like:
Figure 10: Mock-up of historic car photo archive; the mock-up demonstrates how users can contribute their own stories to pictures and how they can annotate images with interesting features (taken from SNML project “Smart Media Archives”).

The KiWi system could be the ideal platform for implementing such a historical archive, both for its collaborative features and for its semantic capabilities. Collaborative features like wiki-style editing could make it possible for users to contribute to archives with their own content. Semantic capabilities could provide for easy categorisation of artefacts and linking of different but related artefacts that are represented in the archive. In this way, the user is supported in browsing and – more importantly – discussing the artefacts.

In order to realise this vision, it is however necessary to strengthen the multimedia capabilities of the KiWi system, which is not considered a part of the KiWi project. Most importantly, it would be beneficial to annotate, link and discuss also parts of images, video, and audio material.