A Spatial Hypertext Wiki for Architectural Knowledge Management

Carlos Solís
ShyWiki Organization
Mexico City, Mexico
csolis@shywiki.org

Nour Ali, Muhammad Ali Babar
Lero, University of Limerick,
Limerick-Ireland
{Nour.Ali, Muhammad.AliBabar}@lero.ie

Abstract

The absence of a disciplined approach for capturing and managing architectural knowledge causes the loss of substantial knowledge generated during the software architecture process. This paper describes the use of a Spatial Hypertext Wiki (ShyWiki) as a tool for Architectural Knowledge Management (AKM) support. Specifically, we demonstrate that ShyWiki can be used for implementing lightweight knowledge sharing workspaces, which includes AKM tools, decision support facilities, and activity awareness features. We also describe how distributed stakeholders involved in the software architecting process can share knowledge and manage their tasks by various features provided by ShyWiki.

Keywords: architectural knowledge, spatial hypertext, wiki, knowledge management

1. Introduction

Software architecture is a discipline that focuses on the design and specification of overall structure of a software-intensive system. It bridges the gap between the requirements and the detailed design phases of the software development lifecycle [1] and is the first step for designing a system which needs to fulfill a collection of desired quality attributes [2].

Although significant progress has been made to support the architecture process over the last decade, relatively little effort has been spent on developing techniques and tools for effectively managing knowledge pertaining to software architecture. Architectural knowledge (AK) can mainly be classified in two categories, namely contextual and technical. The former is called Design Rationale (DR) [3], [4] and provides the answers to questions about a certain design choice or the process followed to make that choice [5],[6]. If it is not captured, knowledge concerning the domain analysis, patterns used, evaluated design options, and made decisions is lost and unavailable for supporting subsequent decisions [7],[8],[9]. The other type of knowledge is technical (such as patterns, styles, tactics, and analysis models) [10]. Such knowledge is required to design and evaluate architectures.

Recently, various researchers [11], [12] have proposed different ways for capturing contextual knowledge underpinning design decisions. An essential requirement of all these approaches is to describe architecture in terms of design decisions and the DR surrounding them. However, it has been reported that design decisions and their rationale are usually not rigorously captured and managed. One of the main reasons reported for this situation is the lack of suitable methodological and tool support for effectively and efficiently capturing and managing AK [13].

Recently, Wikis have emerged as an effective technology for enabling organizations and individuals to capture and share knowledge. Researchers have also demonstrated that Wikis can be used for Architectural Knowledge Management (AKM) [14]. The advantage of using Wikis for AKM is that architecting tasks can be performed in a collaborative, distributed and a reusable way.

In this work, we propose the use of Spatial Hypertext Wiki (ShyWiki) [15] as a tool for AKM. The ShyWiki provides the advantages of a Wiki for AKM as well as its Spatial Hypertext characteristics. The content of a ShyWiki page includes a set of notes which a user can manipulate (add, remove, move, etc). In this way, ShyWiki allows distributed stakeholders to collaborate in the software architecting process for making different decisions (such as choice of quality attributes, use of design patterns, and tradeoff analysis) in an easy and flexible way. In the presented research, we consider the ShyWiki as a mechanism for implementing knowledge sharing workspaces, which includes AKM tools, decisions support facilities, and awareness features.

978-1-4244-3742-9/09/$25.00

This paper is structured as follows: Section 2 presents previous work performed in AKM. Section 3 motivates the use of wikis for knowledge management. Section 4 gives an overview of ShyWikis features. Section 5 presents ShyWiki as an AK Sharing Workspace. Finally, section 6 presents conclusions and further work.

2. Architecture Knowledge Management

The major objective of knowledge management is to improve business processes and practices by using individual and organizational knowledge resources. These include skills, capabilities, experiences, routines, cultural norms, and technologies [16]. The software architecture process needs and generates both explicit and implicit knowledge. These are mutually complementary entities that interact with each other in various creative activities [17]. In the context of this paper, we define architecture knowledge management as an approach that improves the software architecture process outcomes by introducing various processes, and practices for identifying, and capturing both architectural knowledge and expertise in order to make them available, transferable, and reusable across projects in organizations.

The architecture process aims to solve a mixture of ill- and well-defined problems, which involve processing a significant amount of knowledge. Architects require topic knowledge (learnt from text books and courses) and episodic knowledge (experience with the knowledge) [18]. One of the main problems in the software architecture process is the lack of facilities for capturing and accessing knowledge underpinning the design decisions, and the processes and activities leading to these decisions [8], [10]. Examples of this kind of knowledge involve the rationale behind choosing a certain middleware for communication mechanisms between different tiers, choosing an API instead of a wrapper, or knowing the appropriate person to contact with for discussing the performance of different architectural choices. Much of this knowledge is episodic and usually not documented [19]. The absence of a disciplined approach for capturing and managing architectural knowledge causes the loss of substantial knowledge generated during the software architecture process. Thus, depriving organizations of a valuable resource, e.g., losing key personnel may mean loss of knowledge [19], [20].

One of the main reasons for not capturing and managing design decisions and their rationale is that suitable methodologies and tools are unavailable [13]. As a result, several researchers have developed various tools [21],[22],[23],[24] for Architectural Knowledge Management (AKM) and others have identified requirements with the intention of providing such tools. However, most of these efforts overlook the fact that increasingly working teams in today's organizations are distributed in the context of global software development. Moreover, software development teams are increasingly adopting lightweight mechanisms for documenting and sharing software development information and knowledge.

3. Wikis and Knowledge Management

Wikis have been used for Knowledge Management (KM). In this section, we present previous experiences of using Wikis in KM in order to discuss the benefits of using Wikis.

A survey performed on 168 corporate Wikis shows that the successful implantation of a Wiki can help in improving the work processes and collaboration in an organization [26]. The survey also reports that Wikis help in capturing and sharing knowledge specially for solving tasks that require novel solutions. The sustainability of a Wiki is based on the length of the Wiki, the number of contributors, the number of passive users (lurkers), and the frequency of accesses.

Xu reports the experience of using a Wiki for a project management course in computer science [27]. The students used the Wiki as a tool for collective e-learning and knowledge sharing. The conclusion of this experience was that the students were able to share the solutions they found to the problems they experienced using the Wiki.

Desouza reports that users usually get frustrated by the difficulties they encounter in contributing to a knowledge repository [28]. Also, the categorization of the knowledge is inflexible and hard. On the other hand, Wikis can be easily used. For example, Chau and Maurer report the experience of using a Wiki in a software engineering company where all the different actors in the company were able to share knowledge, and that the content was organized and managed by the collective efforts of the users [29]. This is due to the fact that Wikis have an open nature, which avoids the hierarchical barriers in an organization, as well as the set of action permissions encountered in other knowledge management tools. In addition, users in wikis can use hyperlinks to associate and categorize knowledge.

Empirical evidence indicates that a Wiki needs initial bulk upload of knowledge for attracting users and the knowledge sharing should be part of the

corporative culture [30]. Researchers have also reported the use of Wikis for knowledge sharing across organizational boundaries. Critical factors were encountered due to the lack of Wikis integration into existing work practices and project governance. Also, problems related to the work culture [30] such as resistance to be known as an expert, lack of incentives for contributing, or inexistence of dialog about knowledge can be found in Wikis adoption. Despite these difficulties, the mentioned factors affect any KM system and not only Wikis.

In the following, knowledge management issues that are expected to be addressed by Wikis are explained:

- **Usability**. Wikis are easy to use as users do not need to spend effort and time in learning how to use them.
- **Access.** Wikis have low entry barriers for individuals and organizations. People from different areas of an organization can contribute to any part of a Wiki on a topic that interests them.
- **Diversity**. Wikis promote a diversity of knowledge sources. Software documenting tools can generate html documents to be uploaded onto a Wiki.
- **Content Search.** Wikis can be indexed using an inverted index and metadata.
- **Content Versioning.** Most wikis have a version control mechanism to track the changes made to a document. This feature provides changes to be traceable without destroying previous versions.
- **Content Transclusion.** A section of a document can be included inside another without modifying the original source.
- **Content Organization.** End users participate in the organization and structure of the content.
- **Distributed Access.** Wikis provide a centralized knowledge repository to geographically distributed end users of the knowledge repository.

4. An Overview of Spatial Hypertext Wiki

ShyWiki [15] is a wiki which uses spatial hypertext [31] for representing its content. ShyWiki defines visual/spatial relations among the elements on a wiki page. The content of the hypertext documents can be divided into logical sections of information. Each of these sections can contain elements of formatted text, images or other types of media. On a Shywiki page, each element is similar to an adhesive note (see Figure 1), i.e., a spatial hypertext document can be seen as a set of notes. Properties of notes such as their colors, or positions are used for relating them. A user can perform the following actions to capture and manage knowledge with ShyWiki:

- **Create pages.** When a wiki link is navigated for the first time, a new document in ShyWiki is created. For example, Figure 1 indicates that the name of the created wiki page is called *ShyWiki*.

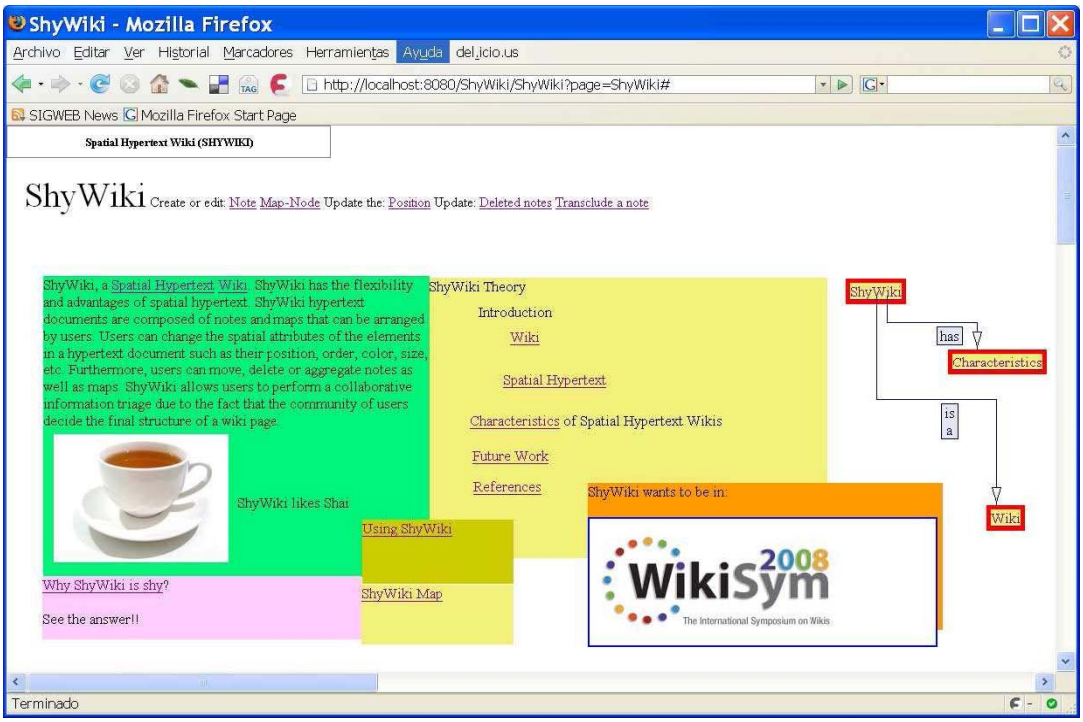

Figure 1. A screenshot of ShyWiki

- **Move notes.** Notes can be moved freely in the interface by dragging and dropping them. In this way, the user can accommodate the information as she/he believes is more convenient.
- **Group notes**. The user can group the notes to create aggregations. In this way, a note can be dragged and dropped inside another note, and the note becomes an internal part of an external one. Once notes are grouped, a user can manipulate a set of notes together. For example, the left hand note in Figure 1 is a group note which includes a note containing an image.
- **Transclude notes.** Transclusion is the inclusion of notes already defined in other documents by reference, i.e., without duplicating them in other notes. Users can transclude a note inside another wiki page by indicating the source document and the note identifier. The position of the transcluded note can be changed, but the other properties can only be modified, if the original note is edited.
- **Label Links.** The hyperlinks in ShyWiki can be labelled. A label indicates the meaning of the association represented by the link. There is a special kind of note called *map node,* it represents a WikiPage or concept and it is used to draw graphs. For example, in Figure 1 the *map nodes ShyWiki*, *Characteristics, and Wiki* were added to the page. The arrows labelled "*has"* and "*is_a"* were drawn automatically because the links *Characteristics* and *Wiki* have these labels. The meaning of the graph can be straightforward known: *ShyWiki has Characteristics* and *ShyWiki is a Wiki.*

5. Knowledge-Sharing Workspace (KSW)

The use of the Knowledge-Sharing Workspace (KSW) paradigm for AKM has been proposed in [32]. The KSW for AKM usually consists of AKM tools, a decision making tool, collaborative services, a workflow engine, and roles all integrated in a single interface as shown in Figure 2.

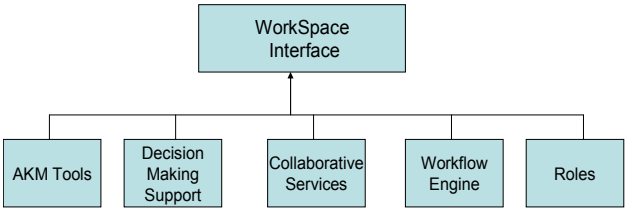

Figure 2. Knowledge-Sharing Workspace Structure

Wikis provide a lightweight mechanism for implementing a KSW paradigm. This paper proposes the use of ShyWiki as a workspace interface. In this way, the advantages of using a Wiki are gained as well as the special characteristics of spatial hypertext. Figure 3 shows the structure of KSW based on the ShyWiki. ShyWiki can be used to provide AKM facilities and decision making support during the software architecture process. However, the ShyWiki does not provide a workflow engine, collaborative services, and roles as shown in Figure 3.

There are Wikis that include workflow engines which can be used to implement KSW with workflow capabilities. However, the hypertext characteristics can be used to create an activity awareness environment that can replace a workflow engine [33], [34]. ShyWiki currently does not integrate collaborative services such as blogs and chat rooms. However, these can be complemented. Additionally, a characteristic of most Wikis is that all users have the same privileges, i.e., there is no need to control what users can or cannot do. Therefore, there is no "Roles" component in ShyWiki. When ShyWiki has to be used in an enterprise which requires priviliges of users to be controlled, ShyWiki has to be extended.

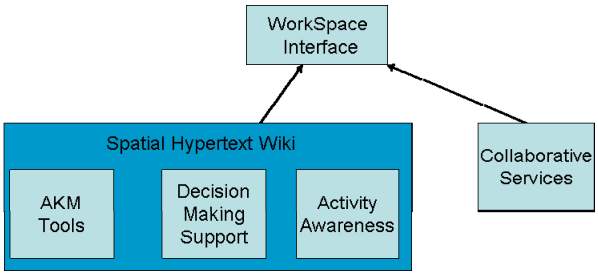

Figure 3. ShyWiki based Knowledge-Sharing Workspace structure

In the following, the different components of KSW are explained from ShyWiki's perspective. The software architecture design process and rationale performed for developing ShyWiki is going to be used as an example for illustrating the features of the ShyWiki as a support mechanism for managing architectural knowledge.

5.1 Architecture Knowledge Management Tools

ShyWiki provides facilities for describing architectural design decisions during modelling and an AKM repository. Currently, ShyWiki does not include any modelling tool for architecture; nor does it propose any specific notation. In this way, ShyWiki is a lightweight mechanism of managing AK and is

independent of any tool or notation. However, ShyWiki can be adapted for specific organizations by including specific tools and notations. Instead, images of architectural models can be imported and can be annotated by using the ShyWiki notes actions. The knowledge repository of ShyWiki can include reusable architectural artifacts such as general scenarios, patterns, styles, general design decisions and tactics, which aid in the architecture design.

Figure 4 shows a diagram with the design of ShyWiki architecture. It is composed of five components: the *Web Browser*, the *Client JavaScript*, the *ShyWiki Servlet*, the *JSON Services* and the *EMF Persistence* component. This diagram has been imported as an image to the ShyWiki page (using ShyWiki's upload image service). The notes added on the image include information about the architecture model that is shared among the wiki users. For example, the note near the ShyWiki Servlet component is added in order to describe its functionality.

In the following, we list some of the ways ShyWiki can support AK sharing during the software architecting process:

- Colour of Notes: The colour of notes can be used as an intuitive way of knowing who created the knowledge, to associate related knowledge, or to represent the state of things. Figure 4 shows that the ShyWiki architecture has been annotated by two users. Each user using notes with different colours.
- Spatial position of Notes: A note can be included at different places on or around a model of architecture to indicate the knowledge about the different architectural elements. For example, a note included on top of a component can be used to describe an architectural decision for creating this component, or a note added away from any architecture concept can describe architecture descriptions. Figure 4 shows that notes near the components describe the decisions of having these components, while the top notes outside the diagram are notes that describe the architecture in general.
- Group of Notes: Notes about the related architectural decisions can be grouped to cluster design decisions based on a specific criteria. For example, each architect who includes a new description related to a previous architecture decision can be included in a group of notes about a particular design decision. Figure 4 shows a group of notes written by two different users in order to share knowledge about the *EMF Persistence* component.

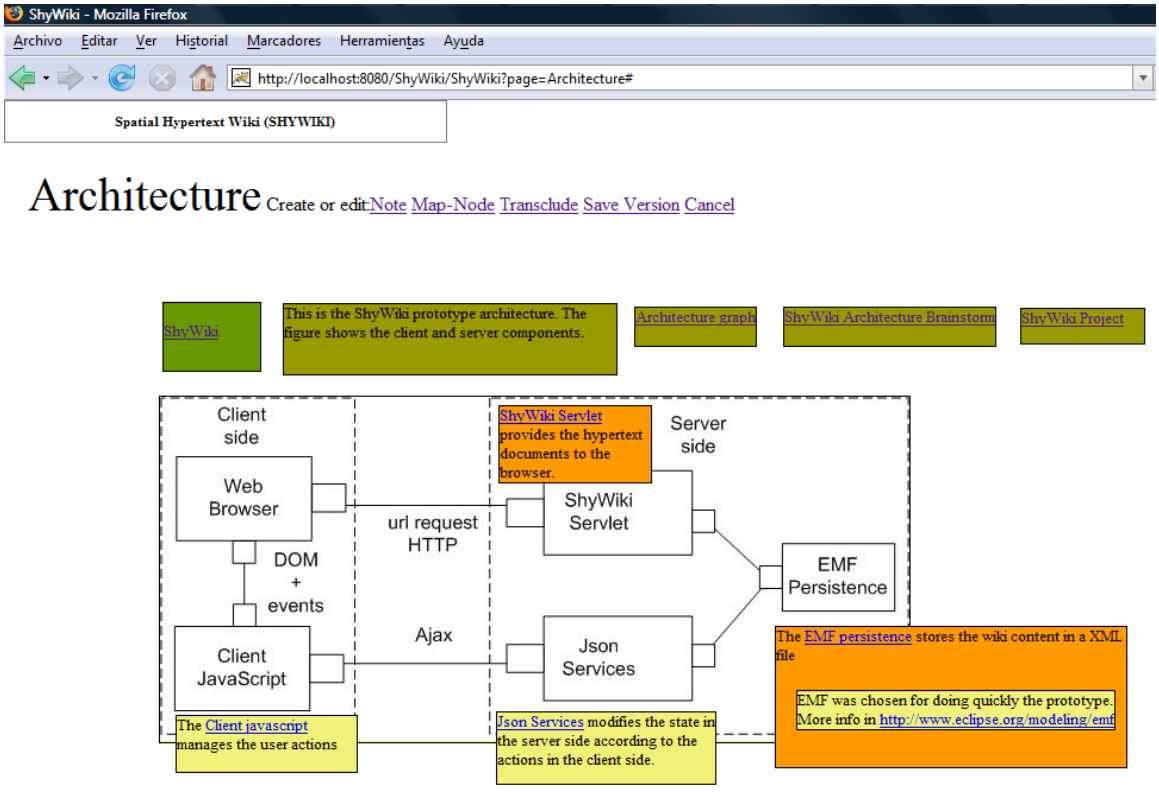

Figure 4. The annotated ShyWiki architecture

- Transclusion of Notes: In many cases, an architect usually needs to cross reference different design decisions and related artefacts. An architect can exploit the note transclusion facility to link related notes placed on different pages of a Wiki.
- Links inside Notes: A note can be added with a link. This link can direct architects to a page which includes more detailed abstractions of an architecture. For example, the image would only show a component. However, that image may not represent the detailed complexity of that component. A note with a link can be included in order to show the detailed architecture of that complex component in another image. Figure 4 shows many notes that can be expanded by following the links.
- Labeled links: The same page can have a knowledge graph which represents the associations which a page has with other pages. For example, an architecture model which applies an architecture pattern can be represented. For example, Figure 5 shows a graph built from the labelled links in Figure 4.

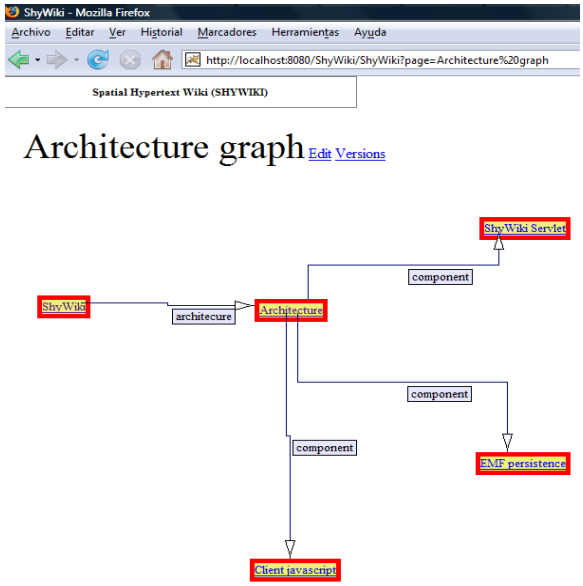

Figure 5. ShyWiki architecture graph

5.2 Decision Making Support

Software architects have to look for new and better ways for satisfying stakeholders' requirements. Software architects usually leverage their tacit and formal knowledge to make appropriate design decision to transform the requirements into reliable systems. The process of finding a solution requires knowledge and creativity. Creativity can make knowledge workers to take more effective and innovative decisions [35].

Idea processing systems deal with the creation and synthesis of ideas by organizing and visualizing them [36]. Idea processing is used to solve problems that do not have an evident solution, and that are not solved using a formula or algorithm. Instead, ideas are useful for obtaining keys where the problem requires knowledge and experience. Software architecture design decisions falls into this kind of problem.

Collaborative creativity is usually performed by group thinking techniques, such as brainstorming. This technique is used extensively in agile programming particularly in the pair programming practice of extreme programming (XP) [37]. It is also used to discuss design decisions between XP partners: the driver and the navigator. Brainstorming is usually done by writing the ideas on a blackboard. This method is frequently replaced by a method called brain-writing, where the group members write the ideas on post-it notes. A brainstorming session of three phases is called Kawakita Jiro method, better known as KJ method [38]. In the first phase, ideas are generated and written on post-it notes without any evaluation or critique. In the second phase, ideas are grouped. For example, when an idea on a post-it note describes a better solution than the others, then it is put over the other notes (prioritization). If the ideas are related, they create a spatial group of ideas. A group of ideas is created by moving related notes to the same space. The goal of the third phase is to create a consensus about the solution to be adopted. In this phase the groups of ideas are categorized, i.e., ideas of a group are considered to be the best solution, then the second, and so on.

ShyWiki is also a kind of idea processing systems due to the fact that elements of ShyWiki are like post-it notes. In this way, a brain-writing session is easily supported. In addition, ShyWiki has many advantages over traditional brainstorming and brain-writing systems without loosing their simplicity such as providing distributed team work. With the ShyWiki, there is no need for using a room with a blackboard or even to use paper based post-it notes. The session participants can be geographically distributed. A ShyWiki can support different brainstorming sessions in parallel; one for each team working on a particular problem.

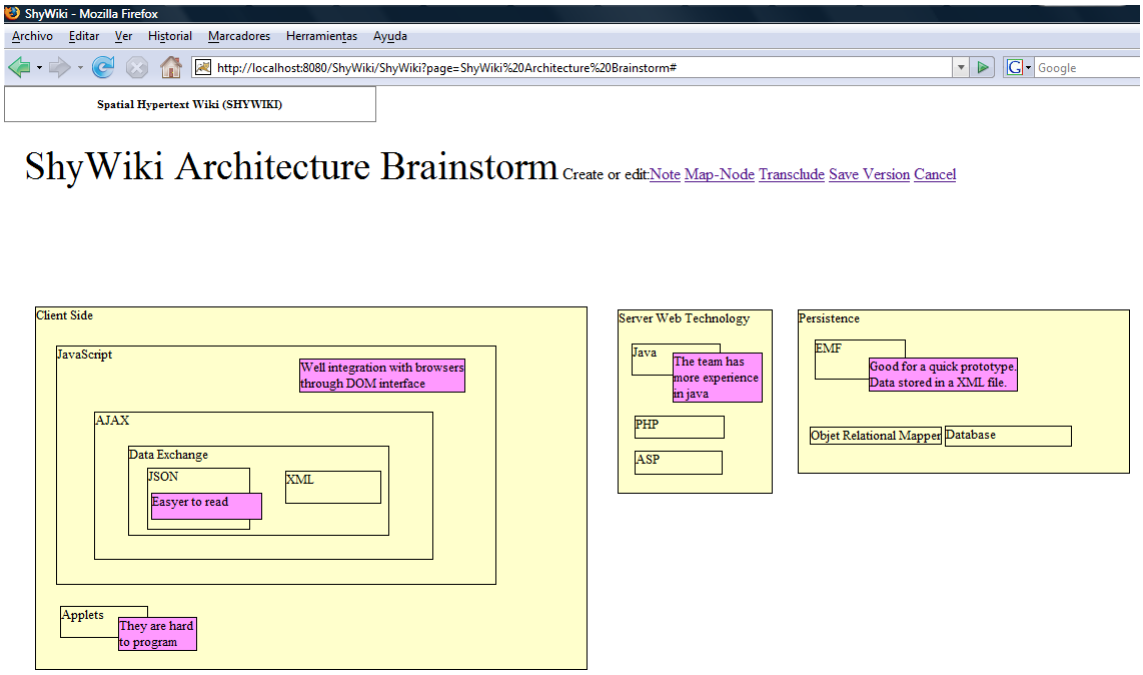

Figure 6. ShyWiki brainstorming session.

The three phases of the KJ method can be developed inside a ShyWiki. The participants can easily enrich other notes, by adding new content to the original ones, or by adding annotations to them. The drag and drop facilities aid in grouping ideas by using relations of proximity (spatial properties) between notes, or covering one note with another to make stacks. The grouping of ideas can also be done by creating big container notes. In addition, the ideas can be classified using background or border colours. As a result, the relations that can be expressed among ShyWikis' notes are richer and easier to manipulate than the blackboard or paper based techniques. A ShyWiki can automatically preserve a brainstorming session, which can be reconstructed if required later on. Furthermore, the use of the versioning facility of a ShyWiki page can enable a user to track the path followed by a team to make a certain design decision. The decision making process can be reconstructed along with the rationale for different steps taken during the process.

Figure 6 shows the brainstorming session for designing ShyWiki's architecture. There are three groups of notes, each for brainstorming an aspect of the ShyWiki architecture: the web server technology, the client side technology, and the persistence. For the web server technology PHP, ASP, and Java were considered. As it can be noticed from Figure 6 that Java was finally chosen because the development team had more experience in this language. In the discussion about the alternatives for the client technology, the goal was to have an interactive interface. This goal can be achieved in web browsers using either applets or JavaScript. JavaScript was selected because it is better integrated with the browsers, and it easier to program than applets. The JavaScript note groups a note for representing AJAX, which is needed to send the client state to the server. AJAX data exchange formats are JavaScript Object Notation (JSON) and XML. The JSON format was used because it is easier to read than XML. For the persistence aspect the options taken into account were to use a Database, an Object Relational Mapper, or to use EMF which provides persistency using XML files. The decision taken was to use EMF since there were time restrictions for developing an initial prototype.

In ShyWiki, the hypertext dimension opens up new ways for using creative group decision making techniques. The post-it notes are not linear. Rather they are hypertextual notes, which can be navigated by means of hyperlinks. A post-it note can contain complex ideas due to the fact that a hyperlink can be included in a note which allows a new wiki page to be opened with further details. Parallel and nested brain-writing sessions can also be supported. Any Wiki page may represent a brain-writing session, and every note can be expanded into another Wiki page. As a result, the discussion about a particular idea in a session can be performed using a nested brain-writing session. A ShyWiki note content is not limited to text as it can

have images or hyperlinks to internal resources (other Wiki pages or specific notes) or external resources.

5.3 Activity Awareness

The KSW provides a virtual space where people can collaborate, share information and knowledge in order to perform tasks. The KSWs relate what to do, who does it, and which information is needed. The information corresponds to software artefacts, requirements, models, meeting summaries, etc, that should be part of a knowledge repository.

The dependencies among the tasks in a software project are not described using fixed workflow models, because such workflows are flexible. The roles in a business process are not significant in the interaction with a Wiki, the access to the content and information is straightforward, and the authentication is enough to control the access to a Wiki in a company. The content is protected by the automatic versioning facility.

The task assignments are described using check lists and task completion notifications. The people working on a project must be aware of others activities, because there can be dependencies among various activities. The project leaders need the information to perform their work. The tasks can be in one of the following states: waiting, enabled, in execution, and completed. The cooperative performance of tasks can be synchronous or asynchronous [33]. In pure hypertext Wikis (such as ShyWiki), the cooperation is asynchronous, i.e., there are no active notifications of the other users' actions.

The asynchronous cooperation in a Wiki is provided by two mechanisms: blocking and versioning. When a user is editing a wiki page, this wiki page is blocked in order not to allow another user to edit it at the same time. Additionally, one user can know exactly the actions of other users through a Wiki page's versioning because all the actions performed are logged. In the case of ShyWiki, the creation, edition, movement, resize, and other actions are logged. The other mechanism of asynchronous cooperation is using a Wiki to represent the state of the tasks, and their dependencies. In this type of asynchronous cooperation ShyWiki can be used more effectively than other Wikis, because it uses spatial hypertext.

The state of notes can be represented mainly by their colors, adorns (small images), borders and positions. In the first case, the users have to assign a color, border, or adorn for each state in which a note can be. In the second case, the position of notes is used. For example, if there are containers of objects (a group of notes), a movement of a note to another group of notes changes its state.

When the task state is represented by colors and borders then the following steps can be done to represent a shared workspace: Define a Wiki page per project. The activities to be performed in the project are listed. The activities that span in many sub-activities can be represented as a complex note or the sub-activities can be represented in the Wiki page of the spanned task. Group activities that are composed of individual activities are represented in such a way. Each activity has a Wiki page and the Wiki pages that are relevant to an activity must be linked (such as diagrams, project, brainstorming, and the peoples responsible).

Each person has a Wiki page, which is her/his personal space. The activities that have to be performed by a person are represented in the personal space by transcluding the activity notes of the project activities. Some kind of graphic state representation of the activities has to be chosen for a given project. When an activity is finished, a user has to change the state of the activity on the projects' Wiki page. In this way, the transcluded notes are updated. The activities in the personal space would also change. The project manager can have a transcluded note in her/his personal workspace, and know about the state of each activity.

When a task's state is represented by the position of a note, then that task is in a state X if the note about that task is located in some area X. If that task is moved to another area, it moves into a different state. This can be supported by having different containers for pending tasks, another for tasks in execution, and another for finished tasks. If a user needs to be aware that a task has been finished, then she/he has to view the tasks containers. However, a better solution is to have a special kind of note which varies its content in function of a spatial query in order to be aware of changes. A typical query would be as the following: *check if there are notes that has been moved inside note Y,* if the query is true then a note shows content CX else content CY. This kind of behaviour is not implemented in the current version of ShyWiki, however the information needed for the query is already stored.

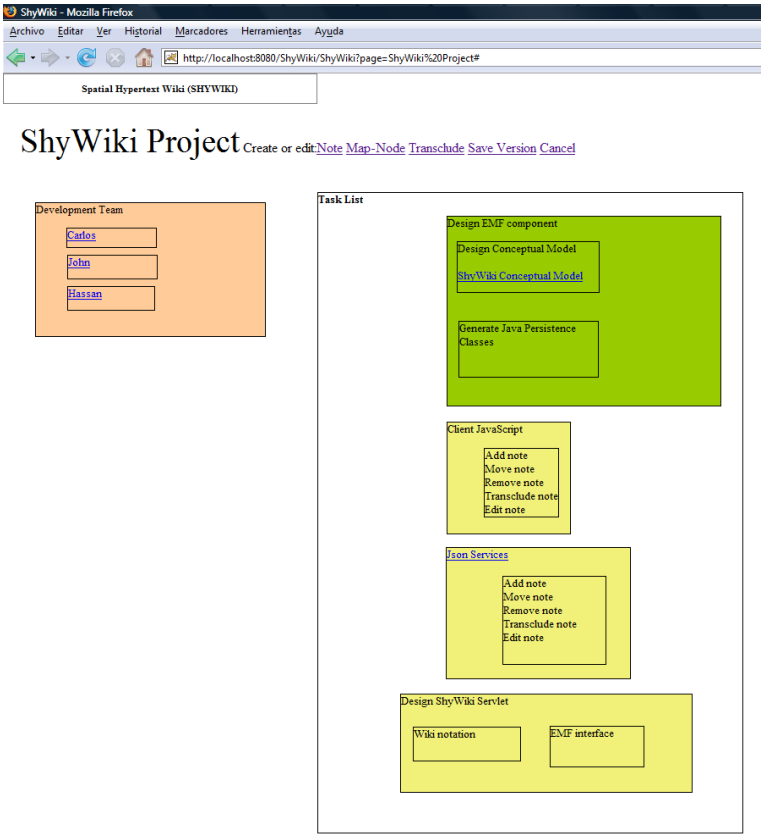

Figure 7. ShyWiki development tasks.

The ShyWiki versioning module has all the information about the changes on a single note, or on a whole Wiki page. Such information can be used to create notes that inform about the changes happened on the Wiki. The basic question that could be answered is: Has the Wiki page Y or Note X changed? In this way, the personal spaces can include notes to Wiki pages that a user is interested in, and can be aware of changes that are the result of a collaborative work. This behaviour that can be easily added to ShyWiki but it is not implemented yet.

The workspace representing the team and the project activities related to the development of ShyWiki is presented in Figure 7. In the left hand side of the figure there is a list with the team members. Each one has more specific information about their tasks in their personal wiki pages. The right hand side of the figure shows the tasks. The tasks in green are the ones that have already been finished, while the tasks in yellow are the ones that are under execution. The wiki page that represents each task has the person responsible of the software artefacts.

6. Conclusions and further work

This work has presented the use of Spatial Hypertext Wiki (ShyWiki) as a tool for AKM. Our implementation of ShyWiki provides a lightweight knowledge sharing workspaces, which include AKM tools, decision support facilities, and activity awareness features. The advantages of using ShyWiki as a wiki for AKM is that it provides a flexible environment where stakeholders can be distributed. In addition, the spatial hypertext nature provides an easy way for sharing knowledge and relating elements using visual/spatial properties. The stakeholders involved in the software architecting process can interact together by manipulating notes in a ShyWiki page. Each note includes knowledge that can be represented as text, hyperlinks, images, etc. The properties of notes determined by users such as their colour, size, or position are used to classify and categorize the kind of knowledge shared.

The software architecting work activities are supported by capturing the rationale and processes and activities leading to a certain architecture decision. The rationale behind software architecture decisions is

stored in the brainstorming sessions. The trace of the rationale can be performed directly by the versioning facility and indirectly by visualizing the properties of notes. An example of using the property of a note is by using its colour for indicating to an actor the expert person he/she needs to contact with for a specific design choice.

In addition, the brainstorming and the activity awareness facilities allow the management of software architecture projects. The brainstorming session shows which people have actually have participated in a decision. The activity awareness can be used to know how many tasks are left or how many tasks have been done.

In the near future, the usability of the current version of ShyWiki is going to be evaluated. Specifically, case studies in particular to the field of global software engineering focusing in architectural design are going to be used for the evaluation. This evaluation is going to allow us to detect further improvements. In addition, we are going to extend ShyWiki with a query language that allows the users to find objects, attributes and the spatial/visual relations stored in the knowledge repository.

7. Acknowledgments

This work is partially supported by Science Foundation Ireland under grant number 03/CE2/I303-1.

8. References

[1] L. Bass, P. Clements, R. Kazman, "Software Architecture in Practice", Second Edition, Addison Wesley, 2003.

[2] M. Shaw, and D. Garlan, "Software Architecture: Perspectives on an Emerging Discipline", Prentice Hall, April, 1996.

[3] C. Potts and G. Bruns, Recording the Reasons for Design Decisions, *10th Int'l Conf. on Software Eng.*, 1988.

[4] J. Lee and K.-Y. Lai, What's in Design Rationale?, *Human-Computer Interaction*, 1991. 6(3-4): pp. 251-280.

[5] A.H. Dutoit and B. Paech, Rationale Management in Software Engineering, in Handbook of Software Engineering and Knowledge Engineering, S. Change, Editor. 2001, World Scientific Publishing, Singapore. pp. 1-29.

[6] T. Gruber and D. Russell, Design Knowledge and Design Rationale: A Framework for Representation, Capture, and Use, *Tech Report KSL 90-45*, Knowledge Laboratory, Stanford University, Standford, United States, 1991.

[7] J. Tyree and A. Akerman, Architecture Decisions: Demystifying Architecture, *IEEE Software*, 2005. **22**(2): pp. 19-27.

[8] J. Bosch, Software Architecture: The Next Step, *European Workshop on Software Architecture*, 2004.

[9] F. Pena-Mora and S. Vadhavkar, Augmenting design patterns with design rationale, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 1997. 11: pp. 93-108.

[10] M. Ali-Babar, I. Gorton, and B. Kitchenham, A Framework for Supporting Architecture Knowledge and Rationale Management, in Rationale Management in Software Engineering, A.H. Dutoit, et al., Editors. 2006, Springer. pp. 237-254.

[11] A. Jansen and J. Bosch, Software Architecture as a Set of Architectural Design Decisions, *Proc. of the 5th Working IEEE/IFIP Conference on Software Architecture*, 2005.

[12] P. Kruchten, P. Lago, and H.V. Vliet, Building up and Reasoning about Architecture Knowledge, *Proc. of the 2nd International Conference on Quality of Software Architectures*, 2006.

[13] A. Tang, M. Ali-Babar, I. Gorton, and J. Han, A Survey of Architecture Design Rationale, *Journal of Systems and Software*, 2006. **79**(12): pp. 1792-1804.

[14] R. Farenhorst, H. van Vliet., Experiences with a Wiki to Support Architectural Knowledge Sharing, In *3rd Workshop on Wikis for Software Engineering (Wikis4SE)*, 2008.

[15] C. Solís and N. Ali. ShyWiki-a spatial hypertext wiki. In *WikiSym '08: Proc. of the 2008 International Symposium on Wikis*, 2008.

[16] G.J.B. Probst. Practical Knowledge Management: A Model That Works. Last accessed on 14th March, 2005, Available from: http://know.unige.ch/publications/Prismartikel.PDF.

[17] I. Nonaka and H. Takeuchi, The Knowledge-Creating Company. 1995: Oxford University Press.

[18] P.N. Robillard, The role of knolwedge in software development, *Communications of the ACM*, 1991. 42(1): pp. 87-92.

[19] L.G. Terveen, P.G. Selfridge, and M.D. Long, Living Design Memory: Framework, Implementation, Lessons Learned, *Human-Computer Interaction*, 1995. 10(1): pp. 1-37.

[20] A.P.J. Jarczyk, P. Loffler, and F.M.S. III, Design Rationale for Software Engineering: A Survey, *Proc. 25th Hawaii Int'l. Conf. on System Sciences*, 1992.

[21] R. Capilla, F. Nava, S. Perez, and J.D. Duenas, A Web-based Tool for Managing Architectural Design Decisions, *Proc. of the 1st Workshop on Sharing and Reusing Architectural Knowledge*, 2006.

[22] A. Tang, Y. Yin, and J. Han, A Rationale-based Architecture Model for Design Traceability and Reasoning, *Journal of Systems and Software*, 2007. 80(6): pp. 918-934.

[23] M. Ali-Babar and I. Gorton, A Tool for Managing Software Architecture Knowledge, Proc. of the 2nd Workshop on SHAring and Reusing architectural knowledge - Architecture, rationale, and Design Intent (SHARK/ADI 2007), Collocated with ICSE 2007., 2007.

[24] A. Jansen, J.v.d. Ven, P. Avgeriou, and D.K. Hammer, Tool Support for Architectural Decisions, *Proc. of the 6th Working IEEE/IFIP Conference on Software Architecture*, 2007.

[25] R. Farenhorst, P. Lago, and H.v. Vliet, Effective Tool Support for Architectural Knowledge Sharing, *Proc. of the First European Conference on Software Architecture*, 2007.

[26] A. Majchrzak, C. Wagner, and D. Yates. Corporate wiki users: results of a survey. In *WikiSym '06: Proc. of the 2006 international symposium on Wikis*, pp. 99–104, 2006.

[27] L. Xu. Project the wiki way: using wiki for computer science course project management. *J. Comput. Small Coll.*, 22(6):109–116, 2007.

[28] K. C. Desouza. Barriers to effective use of knowledge management systems in software engineering. *Communications of the ACM*, 46(1):99–101, 2003.

[29] T. Chau and F. Maurer. A case study of wiki-based experience repository at a medium-sized software company. In *K-CAP '05: Proc. of the 3rd international conference on Knowledge capture*, pp. 185–186, 2005.

[30] R. Giordano. An investigation of the use of a wiki to support knowledge exchange in public health. In *GROUP '07: Proc. of the 2007 international ACM conference on Supporting group work*, pp. 269–272, 2007.

[31] C. Marshall, and F.M. Shipman, Spatial hypertext: designing for change. *Communications of the ACM,* 1995 38(8): pp. 88-97.

[32] M. Ali-Babar. The application of knowledge-sharing workspace paradigm for software architecture processes. In *SHARK '08: Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge*, ACM, Leipzig, Germany, 2008, p 45– 48.

[33] T. Nomura, K. Hayashi, T. Hazama, and S. Gudmundson. Interlocus: workspace configuration mechanisms for activity awareness. In *CSCW '98: Proc. of the 1998 ACM conference on Computer supported cooperative work*, pp. 19–28, 1998.

[34] J. Rubart, J. M. Haake, D. A. Tietze, and W. Wang. Organizing shared enterprise workspaces using component-based cooperative hypermedia. In *Proc. of the 12th ACM conference on Hypertext and Hypermedia*, pp. 73–82, 2001.

[35] G. Forgionne and J. Newman. An experiment on the effectiveness of creativity enhancing decision-making support systems. *Decis. Support Syst.*, 42(4):2126–2136, 2007.

[36] L. F. Young. Knowledge-based systems for idea processing support. *SIGMIS Database*, 21(2-3):27–33, 1990.

[37] K. Beck. *Extreme Programming*. Addison Wesley-Verlag, 12 2003.

[38] J. Kawakita. *The original KJ-method*. Kawakita Research Institute, Tokyo, Japan, 1982.