A DSL for Corporate Wiki Initialization

Oscar Díaz and Gorka Puente

Onekin Research Group, University of the Basque Country, Spain
{oscar.diaz,gorka.puente}@ehu.es

Abstract. Some wikis support virtual communities that are built around the wiki itself (e.g., *Wikipedia*). By contrast, corporate wikis are not created in a vacuum since the community already exists. Documentation, organigrams, etc are all there by the time the wiki is created. The wiki should then be tuned to the existing information ecosystem. That is, wiki concerns (e.g., categories, permissions) are to be influenced by the corporate settings. So far, "all wikis are created equal": empty. This paper advocates for corporate wikis to be initialized with a "wiki scaffolding": a wiki installation where some categories, permissions, etc, are initialized to mimic the corporate settings. Such scaffolding is specified in terms of a Domain Specific Language (*DSL*). The DSL engine is then able to turn the DSL expression into a *MediaWiki* installation which is ready to be populated but now, along the company settings. The DSL is provided as a *FreeMind* plugin, and DSL expressions are denoted as mindmaps.

Keywords: wiki, dsl, MDE, information system.

1 Introduction

Wiki's pioneer, Ward Cunningham, defines wikis as "the simplest online database that could possibly work"[4]. Nowadays, wikis are becoming a favourite approach for collaborative knowledge formation and knowledge sharing [12]. So far, most studies are conducted for public-access wikis or wikis for supporting learning activities [13]. However, companies are increasingly realizing the benefits of wikis [3]. Indeed, the Intranet 2.0 Global Survey reports that around 47% of the respondent companies were somehow using wikis [10]. Based on these figures, we can expect an increasing adoption of wikis among companies.

As any other Information System, the interplay of technology, work practice, and organization is paramount to achieve successful wiki deployments. Therefore, we can expect differences when wikis are deployed to sustain open communities (e.g., *Wikipedia*), offered within a learning organization [13] or are deployed at a company [8]. The peculiarities of each organization will certainly percolate the wiki itself. Indeed, unlike other settings, companies provide an existing infrastructure that frames the wiki. Users, roles, permissions, terminology, documents, templates or project milestones are already there before the wiki is created. This is not the case (or at least not to the same extent) in open-access wikis (e.g., *Wikipedia*) where the community originates around the wiki itself.

H. Mouratidis and C. Rolland (Eds.): CAiSE 2011, LNCS 6741, pp. 237–251, 2011.

Educational settings sit in-between since they offer some pre-existing context but with less demanding constraints than companies.

Consequently, our premise is that, unlike other environments, corporations have all, personal organigrams, documentation practices and task schedules that frame both wiki users and wiki editing. The term **"wiki scaffolding"** is introduced to denote a wiki installation where some categories, templates, permissions, etc are initialized at the outset to mimic the corporate background. This includes structural concerns (e.g., how are wiki pages arranged along which categories), communication means (who is going to be notified of what), permission needs (e.g., who is allowed to do what), etc. So far, this background is patiently replicated by wiki users that, in some cases, are forced to go down to code.

Wiki scaffolding implies not only being knowledgeable about the wiki engine (e.g., *MediaWiki*) but also installing third-party extensions. This can certainly discourage users. Lawyer, architects, medical doctors are all profiting from wikis. Hence, *our aim is for wiki scaffolding to be made accessible to non technical people* ***(who)*** *that collaboratively agree (****how****) on a blueprint for the wiki (****what****).* To this end, we propose the use of a *Domain Specific Language* (DSL). DSLs are reckoned to enhance the quality, productivity, maintainability and portability while permitting domain experts understand, validate and develop the DSL programs themselves [7]. Additionally, collaboration and easy sharing can be promoted by using a *graphical DSL* (as opposed to a *textual DSL*). Specifically, the collaborative mandate suggests capitalizing on existing tools for supporting brainstorming. A common way of recording and expressing brainstorming sessions are *mind maps*. A mindmap is a diagram to express ideas around a central topic. Now, this central topic is "wiki scaffolding", and mindmaps constructs are reinterpreted to denote scaffolding concerns.

This paper presents the *Wiki Scaffolding Language (WSL)* (pronounced "whistle"). *WSL* is built on top of *FreeMind* [1], a popular, open source tool to create mindmaps. Hence, *WSL* expressions are mindmaps. Our bet is that users might already been exposed to mindmaps and even to *FreeMind*, hence reducing the learning curve for *WSL*. These maps (i.e., *WSL* expressions) are then compiled into a set of *MediaWiki* directives whose execution generates the wiki scaffold. *MediaWiki* is one of the most popular wiki engines [2].

This paper is organized along the design and use of *WSL*: *WSL* analysis (Section 2), *WSL* design (Section 3), *WSL* realization (Section 4), *WSL* verification (Section 5) and *WSL* enactment (Section 6). Discussion through related work is presented in Section 7. Some conclusions 8 end the paper.

2 WSL Analysis

This section identifies the scope and main abstractions behind *Wiki Scaffolding* (WS). The aim is to capture the company's work practice and settings as long as their impact on wiki operation. A main outcome of this analysis is a *feature diagram* that describes the commonalities and variabilities of domain concepts

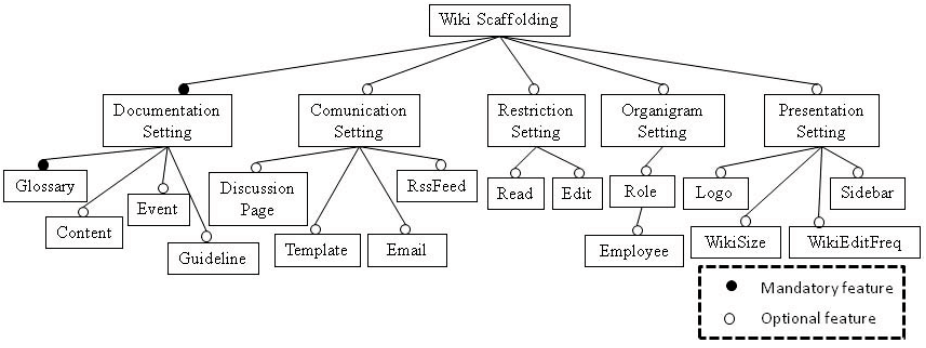

Fig. 1. Feature diagram

and their interdependencies [6]. Fig. 1 depicts the feature diagram for WS. The diagram states that a WS expression captures the company settings in terms of existing documentation practices, communication means, restrictions, the existing organigram and finally, presentation concerns. Next paragraphs delve into these notions.

Documentation Setting. A common problem for open communities is that of fixing a common terminology and understanding. This is easier in the case of corporate wikis where glossaries, documentation guidelines or even, some content might already exists. This setting needs to be captured in wiki terms. A basic classification of wiki pages is that of *"articles", "categories"* and *"templates"*. Articles stand for the *content* that is progressively and socially edited. Next, categories are commonly used as tags to easily locate, organize and navigate among articles. *Glossaries* can help to identify initial wiki categories. Finally, templates provide content to be embedded in other pages. Through parameterization, they permit to reuse and ensure a formatted content along distinct pages. Corporate *guidelines* can then be re-interpreted as wiki templates that guide article editing.

Fig. 1 depicts ***"glossary"***, ***"content"*** and ***"guideline"*** as three features of the company's documentation setting that can impact the wiki Moreover, wikis frequently support living projects where project milestones might need to be accounted for by the wiki. This does not apply to other settings where content is the result of free-willing participation and hence, contribution is not tight to pressing schedules. Wiki wise, this implies that ***"event"*** is a semantically meaningful piece of data, and so should it be markuped and rendered (e.g., through a calendar).

Communication Setting. Wikis are an effective mechanism to support knowledge building through collaboration. This implies the existence of coordination and conflict resolution strategies. When wikis are deployed in an existing organization, wikis become an additional means that should be integrated with existing communication channels. This poses a range of questions: who is going

to be notified of what? Does the existing organizational structure need to be mirrored in the wiki? How is currently achieved such communication? Is email/phone/chatting used for this purpose?

Wiki wise, communication can be internal or external. Internal communication is achieved within the wiki. At this respect, two mechanisms are considered: ***"discussion pages"*** *and* ***"templates"***. Discussion pages (a.k.a. *"talk"* pages in *MediaWiki*) can be used for discussion and communicating with other users. In this way, discussions are kept aside from the content of the associated page. Templates have also been identified as effective means to deliver fixed messages (e.g., warnings, to-do reminders, etc). On the other hand, external communication refers to the ability to notify wiki changes outside the wiki itself (e.g., through **"RSS feeds"** any *rss* client can be used).

Organigram Setting. Companies tend to be organized somehow, what is indicated with the distinct ***"roles"*** that the ***"employees"*** adopt in projects.

Restriction Setting. Unlike public-access wikis, corporate wikis normally limit access to employees. Permissions are counterintuitive in a wiki setting where openness is a hallmark. Indeed, *MediaWiki* natively supports a basic mechanism where the scope of permissions is the whole wiki: you can either edit the whole set of wiki pages or not (e.g., anonymous users cannot read pages). By default, wiki pages can be freely operated. However, permission demands are more stringent in a company setting. Indeed, a study on the use of wikis in the enterprise reports that "power relationships and competition between stakeholders created a need to read access in the ResearchWiki" [5]. For the time being, two permissions are considered: ***"read"*** and ***"edit"***. Additional permissions could be added in future releases if feedback so advises[1].

Presentation Setting. Most companies project a unified image in terms of rendering and presentation. Wikis resort to "skins"[2] for rendering. These skins are engine specific. However, we do not expect our target audience to know about skins. We should strive to capture presentation concerns in abstract terms, better said, through domain criteria that could later be used by the *DSL* engine to determine the most appropriate skin. Specifically, we consider ***"wikiSize"*** and ***"wikiEditFreq"***. Based on the expected size and edit frequency of the wiki, heuristics can make an educated guess about the wiki skin. In this way, the *DSL* engine frees stakeholders from being knowledgeable about presentation issues, offering good-enough outputs. Notice that the wiki administrator can latter change this automatically-selected skin. Additionally, the ***"logo"*** and ***"sidebar"*** features are introduced for customizing both headers and index panes which are available for speeding up wiki access.

[1] *MediaWiki* permissions include *"read", "edit", "createpage", "createtalk", "upload", "delete", "protect"* (i.e., allows locking a page to prevent edits and moves), etc.

[2] A skin is *"a preset package containing graphical appearance details"*, used to customise the look and feel of wiki pages.

3 WSL Design

The aforementioned concerns are now captured through a *DSL*. *DSL* design implies first to set the abstract syntax, and next, select one of the possible concrete syntaxes [11]. Based on the feature diagram, the **abstract syntax** describes the concepts of the language, the relationships among them, and the structuring rules that constrain the model elements and their combinations in order to respect the domain rules. This is expressed as the *DSL* metamodel. Fig. 2 depicts the abstract syntax for *WSL*. A *scaffolding* model includes four main model classes, namely:

(1) The **Content** class, which is a graph described along **Items** and **Links**. *Items* capture the different kinds of data existing in the company that need to be also available at wiki inception. As identified in section 2, this content includes glossary terms (*"category" itemType*), content ready to be available as a wiki article (*"article" itemType*), guides for content structure (*"template" itemType*) or events to capture scheduling milestones (*"event" itemType*). Next, **Links** relate these *Items* together. *Links* are also typed based on the type of the related *items*: *"relatedWith"* link (a general *item*-to-*item association)*; *"belongsTo"* link (to associate a *category* to an *item); "templatedBy"* link (to associate a *template* to an *item*); *"scheduledFor"* link (to associate an *event* to an *item*). Items also hold three boolean properties: **discussion** (to indicate whether this *item* is subject to discussion), **rssFeed** (to specify the availability of a feed subscription for this *item*) and **indexPaneEntry** (to capture that the *item* is to be indexed in the sidebar).

(2) The **Organigram** class, which captures a basic arrangement of **Employees** in terms of **Roles**.

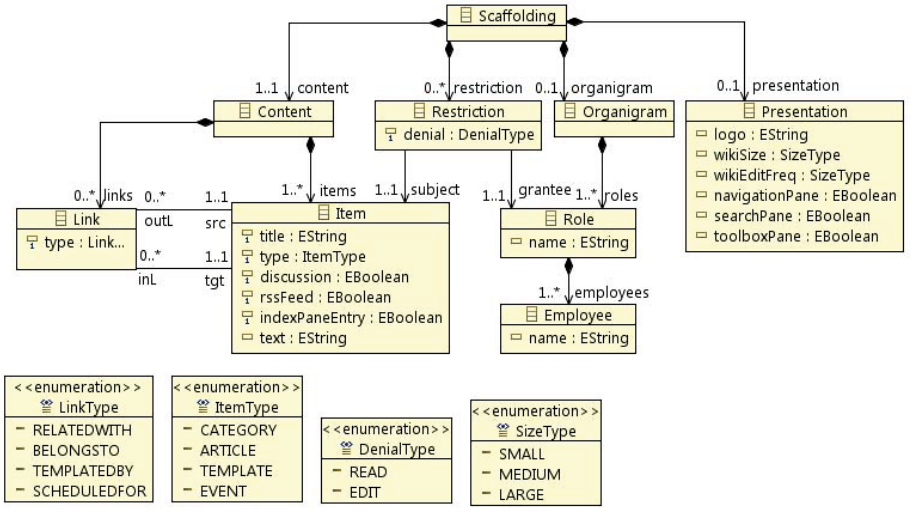

Fig. 2. *WSL* metamodel (abstract syntax)

(3) The **Restriction** class, which binds together three elements: a permission subject (i.e., an *Item*), a permission grantee (i.e., a *Role*) and a **denial** (i.e., *"read"* and *"edit"*).

(4) The **Presentation** class, which holds properties to guide the rendering of the wiki. Specifically, index requirements are captured through four common indexing schemas: *toolboxPane* (entries include *"what links here", "Upload file", "printable version"*, etc), *navigationPane* (entries include *"recent changes", "help" "main page"*, etc), *indexPane* (where entries are set by the designer through the *Item*'s *indexPaneEntry* attribute), and *searchPane* (as a search facility to locate articles based on content). The logo is also captured here.

Next, this abstract syntax is realized through a concrete syntax. This implies a mapping between the metamodel concepts and their textual or visual representation. Preliminary feedback indicates that a visual syntax would be more suitable. The user profile (i.e., domain experts) as well as the collaboratively way of obtaining the wiki blueprint, advise to go for a visual DSL. Rather than developing our own visual language, we decide to capitalize on an existing one: *FreeMind*. With over 6,000 daily downloads, *FreeMind* is one of the most popular tools for mindmap drawing. Fig. 3 shows a snapshot of a *FreeMind* map. The main advantage of this tool is the easiness to play around to capture your mental model (e.g., nodes, and their descendants, can be easily moved around; branches can be collapsed, etc). This decision not only speeds up development but, more importantly, it will hopefully facilitate *WSL* adoption among end users. Next section introduces *WSL* as a visual language on top of *FreeMind*.

4 WSL Realization

WSL is a visual language on top of *FreeMind*. That is, a *WSL* expression is a compliant *FreeMind* map. However, the opposite does not hold. Some maps might not deliver a compliant wiki scaffolding, where compliance is determined by the abstract syntax in fig. 2. Therefore, *WSL* maps are a subset of *FreeMind* maps. *FreeMind* maps are internally represented as *XML* files along an *XML* schema. On top of it, *WSL* imposes an additional set of constraints that ensures that maps account for compliant scaffoldings (i.e., conform to the *WSL* abstract syntax). Before delving into how *WSL* constructs are mapped into *FreeMind* elements, next subsection introduces an example.

4.1 WSL Example

Consider the use of wikis to support software projects. The scattering of stakeholders, the need for collaboration and tracking, and the iterative manners that characterize software projects make wikis an attractive platform [9]. Fig. 3 provides an example for the *Purchase Project* as a *WSL* mindmap.

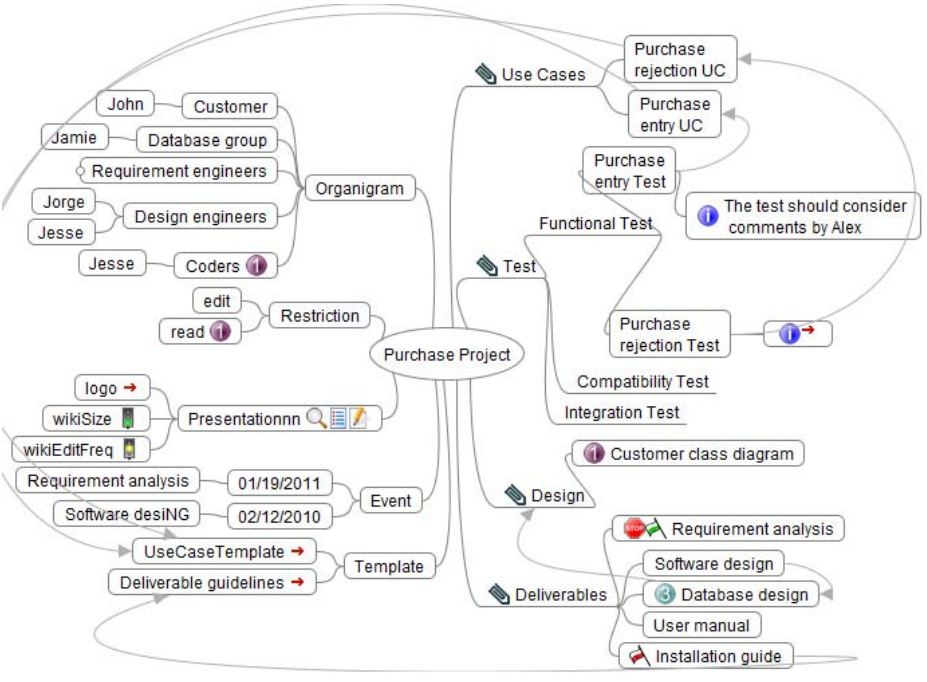

Fig. 3. *Purchase Project* Scaffolding

FreeMind depicts ideas and their relationships as nodes and edges that follow a radial distribution. In our example, the *Organigram* branch captures the existing roles as well as the employees assigned to these roles. The *Restriction* branch lists limitations in terms of wiki operations. The *Event* branch captures two milestones attached to pages *"Requirement analysis"* and *"Software desiNG"* at the onset. Next, the company already has some guidelines to capture use cases and document deliverables. Such practices should also be adhered to when in the wiki. The *Template* branch refers to two such guidelines through the *"UseCaseTemplate"* node and the *"DeliverableGuidelines"* node. The *Presentation* branch will impact on the rendering of the wiki based on the expected *"wikiSize"* and *"wikiEditingFreq"*. A "*traffic light*" icon is used to indicate the three possible values of these properties: large (red light), medium (yellow light) and small (green light). As for the sidebar, this node includes a navigation pane (denoted by the "*list*" icon) and a search pane (denote by the "*magnifier*" icon). The sidebar is finally completed with an index pane (denoted by the *"look here"* icon on categories *"Use Cases"*, *"Test"*, etc). Regarding to restrictions, *"priority"* icon sets a restriction whereby *"Coders"* (i.e., the role) are restricted from *reading* (i.e., the denial) the article *"Customer class diagram"* (i.e., the item).

As for the corporate glossary, common terms already in use include *"Use Cases", "Functional Test", "Compatibility Test"* etc. These terms find their way as wiki categories. Hierarchical relationships among categories are captured by describing a category as a child of the parent category (e.g., *"Test"* ← *"Functional Test"*). Wiki articles are denoted as *bubbled* nodes (e.g., *"Requirements analysis"* stands for an article which is categorised as *"Deliverable"*). The title of a node behaves as an identifier, so that two *FreeMind* nodes placed differently but with the very same title, stand for the same notion. This permits the *Content* graph to be flattened as a *FreeMind* tree.

It can look odd to introduce articles at wiki inception since wiki's *raison d'etre* is precisely collaborative article editing. Indeed, we do not expect too many articles to be introduced at scaffolding time. However, the need to come up with some articles might be known from the very beginning. The scaffolding permits so by introducing a node whose title becomes the title of the wiki article. For instance, the node *"Software design"* yields a wiki article with the namesake title. Even more, some relationships might be known at the outset. For instance, trace requirements made advisable to keep a hyperlink between the *"Purchase entry test"* and the *"Purchase entry UC"*. This is depicted as an arrow between the node counterparts.

Based on preliminary user feedback, we also consider article content to be known at scaffolding time. This is realized as a child of the given article (together with the *"info"* icon). Fig. 3 illustrates the two options. The content of *"Purchase entry test"* is explicitly provided as the text of its child node. By contrast, the content of *"Purchase rejection test"* is already available at the company as a *Word* document. *FreeMind* permits to introduce hyperlinks as node content (denoted through a small red arrow). This facility is used to our advantage to link *"Purchase rejection test"* to the external document holding its content. Likewise, corporate guidelines can find their way as wiki templates. So far, *WSL* only supports *Word* documents (exported as *XML*). At deployment time (i.e., when the *WSL* map is enacted), these external documents are turned into either, article content or wiki templates. Fig. 6 provides a screenshot of the main page as generated by the WSL engine. The rest of this section provides a detail account of *WSL* expressivity.

4.2 WSL Concrete Syntax

WSL abstract syntax is realized as a *graphical* concrete syntax. A mapping is then set between elements of the abstract syntax and their visual counterparts in *FreeMind*. These "visual counterparts" are set by the *FreeMind* metamodel. Therefore, a set of mappings between elements of the *WSL* metamodel and elements of the *FreeMind* metamodel is realized. Additionally, some constraints need to restrict the expressiveness of *FreeMind* to result in valid "scaffolding maps" (i.e., compliant with the *WSL* metamodel).

FreeMind* metamodel** (see fig. 4). A **Map** is a compound of **Nodes**. Nodes have a **title** and might hold a ***link to an external document (local or remote) as well as a set of properties mainly referring to rendering concerns. For instance,

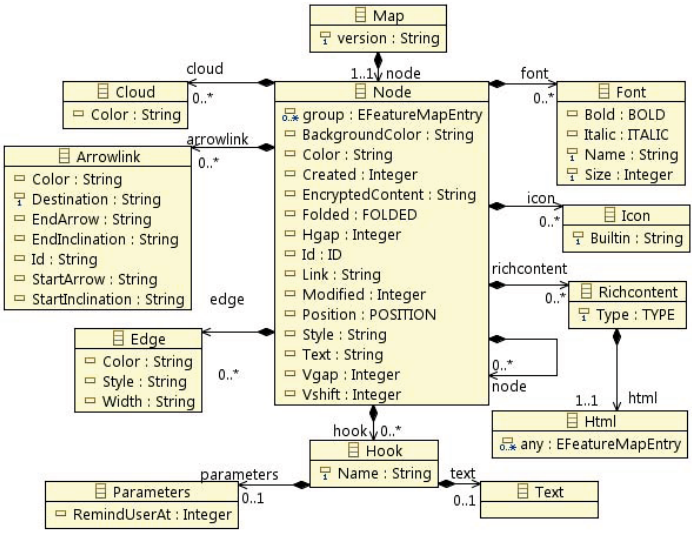

Fig. 4. *FreeMind* metamodel: primitives for mindmap drawing

the **Style** property can be ***fork*** and ***bubble*** and determines the look of the node as a tagged line tag or a bubble, respectively. Next, nodes are basically arranged in a tree-like way. A central node serves as the common root. Tree structures are constructed using **Edges**. An Edge is a connector that relates a node with its parent node. Additionally, **Arrowlinks** are also connectors but in this case, the connection is between two arbitrary nodes. Finally, **Icons**[3] and **Fonts** can be associated with nodes in an attempt to reflect the underlying semantics of the node. Of course, this semantics resides in the users' head.

WSL-to-FreeMind mapping (see Table 1, first two columns). Once *FreeMind* visual symbols are introduced, the next step is to indicate a mapping between the *WSL* abstract syntax and these symbols:

- *Scaffolding* class. The *root node* is the *FreeMind* counterpart of this class.
- *Organigram* class. A bubble node with title *"Organigram"* denotes the origin of the organigram hierarchy. Nodes having "*Organigram*" as parent denote roles. Likewise, nodes having *"Organigram"* as grandparent are interpreted as employees.
- *Presentation* class. A bubble node with title *"Presentation"* denotes this class. Boolean properties are captured as icons on *"Presentation"*. Value-based properties are represented as children nodes: *logo* (captured as a link to an image file), *wikiSize* and *wikiEditFreq*. The latter are decorated with traffic-light icons to account for their values.

[3] *Freemind* provides a fixed set of icons. In the last version, users can introduce their own icons, though it is not recommended for interoperability reasons.

Table 1. *WikiScaffolding*-to-*FreeMind* mapping & *FreeMind*-to-*MediaWiki* mapping

WSL	FreeMind	MediaWiki
Scaffolding	root node	main page[4]
Organigram	*"Organigram"* bubble node	n.a.
Role	child of *"Organigram"* node	wiki group
Employee	grandson of *"Organigram"* node	wiki user & user page
Presentation	*"Presentation"* bubble node	wiki skin[5]
logo	logo node	wiki logo
wikiSize	*wikiSize* node	wiki skin
wikiEditFreq	*wikiEditFreq* node	wiki skin
	with *"traffic light"* icons	
navigationPane	*"list"* icon	navigation in sidebar
searchPane	*"magnifier"* icon	search in sidebar
toolboxPane	*"refine"* icon	toolbox in sidebar
indexPane entry	*"look here"* icon at *Item*	element in the navigation bar
Restriction	*"Restriction"* bubble node and *"priority"* icons ..	blacklisted pages for groups[6]
denial	child of *"Restriction"* node	wiki permission.
Item		
title	node text	page title
category Item	fork node	category page
article Item	bubble node	article page
template Item	child of *"Template"* node	template page
event Item	child of *"Event"* node	calendar extension[7]
discussion	*"stop-sign"* icon	talk page for that page
RSSfeed	*"flag"* icons	RSS generator for that page[8]
text	child with *"info"* icon or linked files	page content
Link		
relatedWith Link	arrowLink connector	inter-page hyperlink *[[page]]* *[[:Category:parentCat]]*
belongsTo Link	edge connector	page-category hyperlink *[[Category:parentCat]]*
templatedBy Link	arrowLink connector	template-page hyperlink *{{template}}*
scheduledFor Link	edge connector	event-to-page link in the calendar widget

– *Restriction* class. A bubble node with title *"Restriction"* denotes this class. A restriction is a triplet: subject (i.e., an *Item* node), grantee (a *Role* node), and the denial type (i.e., *read* or *edit*). We resort to *priority* icons to denote those elements that conform to a restriction unit. That is, map nodes decorated with the same *priority* icon belong to the same *restriction*. Due to icon availability, *permissions* are limited to ten (*"priority"* icon ..).

[4] CategoryTree extension: www.mediawiki.org/wiki/Extension:CategoryTree
[5] MediaWiki skins include monobook (default), vector (e.g., used by Wikipedia), etc. WSL completes the offer with cavendish, rilpoint, guMax, guMaxDD and guMaxv.
[6] Blacklist extension at www.mediawiki.org/wiki/Extension:Blacklist
[7] Barrylb extension at www.mediawiki.org/wiki/Extension:Calendar_(Barrylb)
[8] WikiArticleFeeds extension at www.mediawiki.org/wiki/Extension:WikiArticleFeeds

- *Content* class. There is not a *FreeMind* counterpart for the *Content* class as such. Rather all nodes in the map except for *"Organigram"*, *"Presentation"*, *"Restriction"*, *"Event"* and *"Template"* nodes (and descendants) stand for *Content Items*. The node title behaves as an identifier, so that two *FreeMind* nodes placed differently but with the same title, stand for the same *Item*. This allows the *Content* graph to be flattened as a *FreeMind* tree.
- *Item* class. *Items* are typed as *"category", "article", "template"* and *"event"*. *Category Items* are denoted as fork nodes (i.e., nodes with the "fork" style). *Article Items* are captured as bubble nodes. Next, *Template Items* are children of the "*Template*" node. These nodes can either hold the page text content (i.e., text attribute) themselves as a child with the "*info*" icon or point to external documents from where the content is obtained at compile time (only txt and word as xml exported files in the current version) Finally, *Event Items* are children of the "*Event*" node. As for the boolean properties, *discussion, rssFeed and indexPaneEntry*, the affected *Items* (regardless of their type) are decorated with the "*stop sign*" icon, a "*flag*" icon and "*look here*" icon, respectively.
- *Link* class. *Links* are classified as *relatedWith*, *belongsTo*, *templatedBy* and *scheduledFor*. *FreeMind* offers two kinds of connectors: *Edges,* which are the default arcs connecting a node with its child, and *ArrowLinks*, which are arcs connecting two nodes anywhere in the map. *Edges* are interpreted as *belongsTo* links when they connect an *Item* to a *category Item* (e.g., fig. 3, arc from *"Database design"* to *"Deliverables"*) and as *scheduledFor* when they connect an *Item* to an *event Item* (e.g., fig. 3, edge from "*Requirement analysis*" to "*01/19/2011*"). As for *ArrowLinks*, they sustain (1) *RelatedWith* links when they relate an *Item* to another *Item* (e.g., fig. 3, arc from "*Software design*" to "*Database design*") and *(3) TemplatedBy* links when the ingoing node stands for a *template Item* (e.g., fig. 3, arc from "*Purchase entry UC*" to "*UseCaseTemplate*").

5 Verification of WSL Maps

WSL maps are a subset of *FreeMind* maps, i.e., *WSL* metamodel imposes additional constraints on top of the *FreeMind* metamodel. Such constraints can be verified on user request or at deployment time. Fig. 5 provides a snapshot of the *"Tools"* menu now extended to address *WSL* maps*: "WSL configuration"* permits to configure parameters for the *MediaWiki* installation; *"WSL deployment"* causes the generation of the wiki instance from the *WSL* specification; *"WSL Skeleton"* provides a *FreeMind* map with the basic *WSL* nodes (e.g., *Organigram*, *Restriction*, etc) so that misspells are prevented; and finally, *"WSL Map Checking"* triggers *WSL* map verification.

Fig. 5 depicts the verification outcome for our sample problem (see fig. 3). Messages can be warnings and errors. For our sample, two warnings are noted.

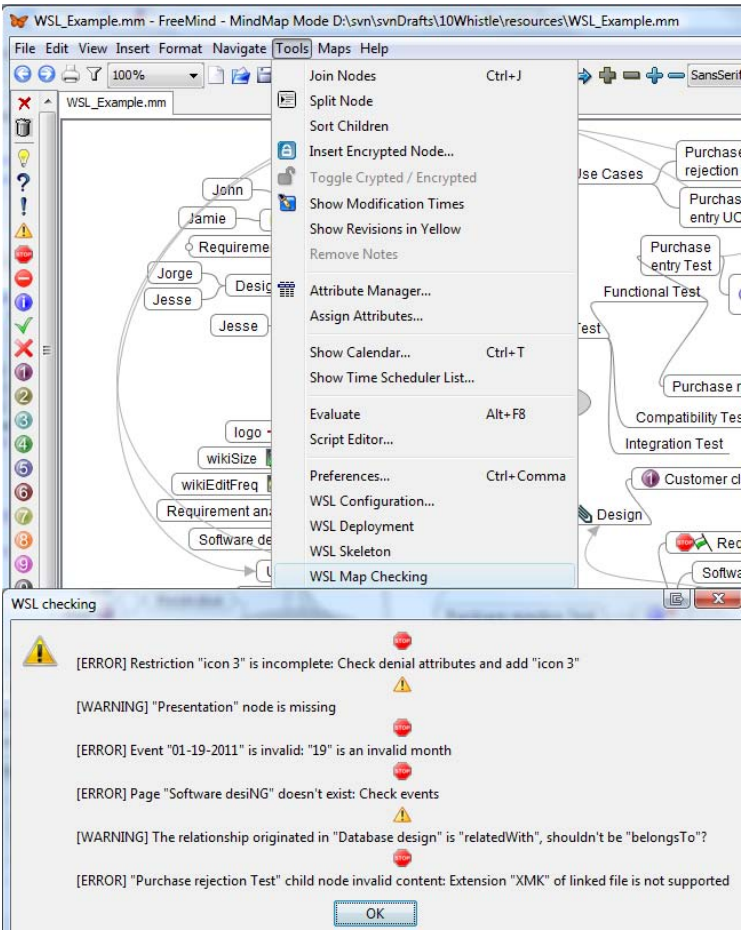

Fig. 5. Verifying the *WSL* map at fig. 3

One informs about the lack of the optional *Presentation* node which, in this example, is due to a misspelling (*"Presentationnn"*). The other warning notifies about a common mistake in wiki construction: setting a *relatedWith* relationship between an article and a category. This is an odd situation that could be mistaken with the *belongsTo* relationship, and so is it indicated. As for errors, they prevent the wiki from being generated. For our sample case, these errors include: a misspelling of an event date (e.g., *"01/19/2011"*); referring to a non-existent node (e.g., *"Software desiNG"*); partial definition of a restriction where either the denial, the employee or the article is missing (e.g., restriction 3); unsupported document extension (e.g., extension *"XMK"* is not supported; so far, only XML and TXT files can become page content).

6 Enactment of WSL Maps

By selecting the *"WSL deployment"* option of the *Tool* menu (see fig. 5), the current map is turned into a wiki installation in *MediaWiki*. This means that around 400 LOC (mainly *SQL* statements) are automatically generated for the current example. Figs. 6 and 7 provides three screenshots of the generated pages: the main page (illustrating the use of the *CategoryTree* and *Calendar* extensions), the *"Purchase rejection Test"* article page (which is obtained from a *Word XML* document) and the *"Purchase Rejection UC"* (which follows the *"UseCaseTemplate"* also externally obtained). Space limitations prevent us from giving a detail account of this generation process. For the purpose of this paper, it is enough to show the mapping between *FreeMind* constructs and *MediaWiki* primitives. The last two columns in Table 1 indicates such mappings.

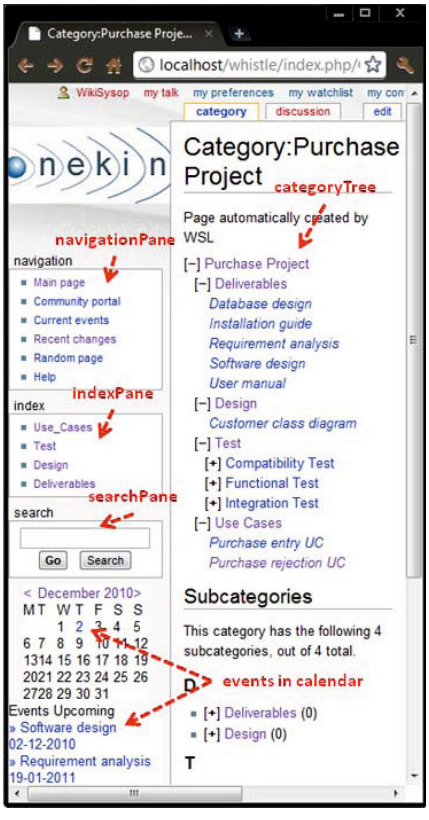

Fig. 6. *"Purchase Project"* wiki main page as generated by *WSL*

It is important to notice that some scaffolding features require additional *MediaWiki* extensions (e.g., *CategoryTree*). The *WSL* engine builds upon *MediaWiki* version 1.16 and the extensions have been tested against it. Such composition is provided as a unit by *WSL*. This raises the issue of platform evolution, i.e., new versions of *MediaWiki* (or its extensions) might impact the *WSL* engine. This is certainly true. But, how real is this threat? First, *MediaWiki* is a stable platform backed by thousands of installations. And second, wikis can be upgraded once deployed. That is, *WSL* can be used to generate the wiki scaffold, and next, the user can upgrade to the newest version (just two clicks away). This makes us confident about the lifespan of *WSL*.

7 Discussion through Related Work

Mindmaps have long been recognized as a useful technique for brainstorming. Recently, enhancements have been proposed to improve the efficacy of mind maps (e.g., use of pictorial stimuli [14]). Although benefits are reported, these extensions decrease simplicity, and jeopardize interoperability. There certainly

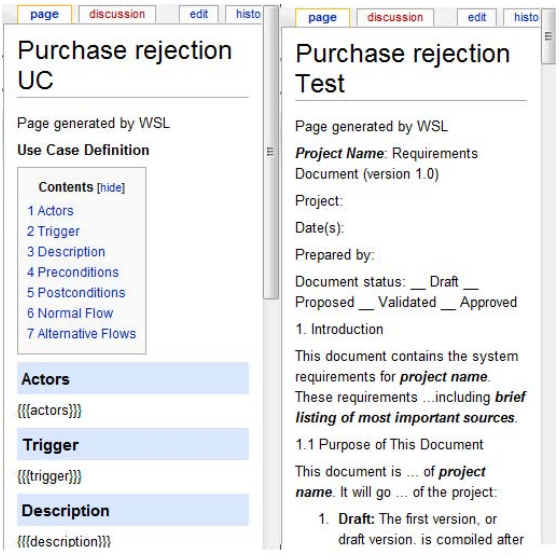

Fig. 7. Template and article pages as generated by *WSL*

exists more sophisticated tools for brainstorming than *FreeMind,* but we value popularity, simplicity and cost as main selection criteria.

Another important remark is that of scalability. Although it is not the aim of scaffolding to offer a complete wiki map but just a blueprint, large projects can require large scaffoldings. This can lead to cluttered *WSL* maps. Fortunately, *FreeMind* offer view-like mechanisms that permit to filter map nodes based on content and relationships. Testing stakeholders can filter those nodes based on containing the string *"test"*, whereas template-minded stakeholders can restrict the view to those nodes related with a template.

As for visual DSLs, they are still scarce compare with textual DSL ([11] for an overview). Our insight here is that the context where the DSL is to be deployed is generally overlooked in DSL publications. Our experience is that DSL success not only depends on finding the right abstractions but also on producing minimum disturbance to existing practices. *FreeMind* was chosen on these grounds.

8 Conclusions

We introduced the notion of "wiki scaffolding" as a way to capture the contextual setting for wikis deployed in an existing organization. While wikis for virtual communities create such setting as they go along, corporate wikis know this context at the onset. We introduced a *DSL* for wiki scaffolding that abstracts from the technicalities that go in setting those parameters down to wiki code. By capitalizing on *FreeMind* as the conduit for *WSL* concrete syntax, we expect non-technical communities to benefit from the scaffolding. Anecdotical evidences suggest that the benefits of the *DSL* go beyond speeding up wiki deployment

or promoting user participation. Knowledge retention is achieved by the *DSL* engine embedding good practices about both presentation and structure. This helps introducing wikis in organizations without wiki experience. As for the expressiveness of *WSL*, current constructs are based on a literature survey about the use of wikis in companies. Social conventions and incentives will emerge and evolve to guide contributors, resolve disputes and help manage wiki deployments in organizations. As these issues find support in wiki engines, *WSL* constructs will need to be extended.

Acknowledgments. This work is co-supported by the Spanish Ministry of Education, and the European Social Fund under contract TIN2008-06507-C02-01/TIN (MODELINE), and Conserjería de Educación y Ciencia of Castilla-La Mancha under contract PAC08-0160-6141 (IDONEO). Puente has a doctoral grant from the Spanish Ministry of Science & Education.

References

1. Freemind. Online, `http://freemind.sourceforge.net` (accessed November 25, 2010)
2. Mediawiki. Online, `http://www.mediawiki.org` (accessed November 25, 2010)
3. Carlin, D.: Corporate Wikis Go Viral. Online, `http://www.businessweek.com/technology/content/mar2007/tc20070312_476504.htm` (accessed November 25, 2010)
4. Cunningham, W.: What is a Wiki. Online, `http://www.wiki.org/wiki.cgi?WhatIsWiki` (accessed November 25, 2010)
5. Danis, C., Singer, D.: A Wiki Instance in the Enterprise: Opportunities, Concerns and Reality. In: Computer Supported Cooperative Work, CSCW (2008)
6. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (foda) feasibility study. Technical report, Carnegie-Mellon University Software Engineering Institute (1990)
7. Kelly, S., Tolvanen, J.-P.: Domain-Specific Modeling: Enabling Full Code Generation. Wiley-IEEE Computer Society (2008)
8. Lee, H., Bonk, C.: The Use of Wikis for Collaboration in Corporations: Perceptions and Implications for Future Research. In: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (2010)
9. Louridas, P.: Using wikis in software development. IEEE Software 23(2), 88–91 (2006)
10. Prescient Digital Media. Intranet 2.0 Global Survey. Online, `http://intranetblog.blogware.com/blog/_archives/2009/5/15/4187339.html` (accessed November 25, 2010)
11. Mernik, M., Heering, J., Sloane, A.M.: When and How to Develop Domain-Specific Languages. ACM Computing Surveys 37(4), 316–344 (2005)
12. Raman, M.: Wiki Technology as A "Free" Collaborative Tool within an Organizational Setting. IS Management 23, 59–66 (2006)
13. Toker, S., Moseley, J.L., Chow, A.T.: There a Wiki in Your Future?: Applications for Education, Instructional Design, and General Use. Educational Technology Magazine, 6 (2008)
14. Wang, H.-C., Cosley, D., Fussell, S.R.: Idea Expander: Supporting Group Brainstorming with Conversationally Triggered Visual Thinking Stimuli. In: Computer Supported Cooperative Work (CSCW), pp. 103–106 (2010)