The organization of corporate wikis tends to deteriorate as time goes by. Rearranging categories, structuring articles and even moving sections among articles are cumbersome tasks in current wiki engines. This discourages the layman. But, it is the layman who writes the articles, knows the wiki content, and detects refactoring opportunities. Our goal is to improve the refactoring affordances of current wiki engines by providing an alternative front-end tuned to refactoring. This is achieved by (i) surfacing the structure of the wiki corpus as a mind map, and (ii) conducting refactoring as mind map reshaping. To this end, we introduce WikiWhirl, a domain-specific language for wiki refactoring. WikiWhirl is supported as an extension of FreeMind, a popular mind mapping tool. In this way, refactoring operations are intuitively conducted as actions upon mind map nodes. In a refactoring session a user imports the wiki structure as a FreeMind map; next, conducts the refactoring operations on the map, and finally, the effects are saved in the wiki database. The operational semantics of the WikiWhirl operations follow refactoring good practices (e.g., authorship preservation). Results from a controlled experiment suggest that WikiWhirl outperforms MediaWiki in three main affordance enablers: understandability, productivity and fulfillment of refactoring good practices.

**Keywords:** Corporate wikis, refactoring, affordance, mind map, FreeMind, MediaWiki
1. Introduction

Wikis are becoming a mainstream for knowledge formation and sharing (Raman 2006). Consubstantial to knowledge formation is exploration, tentative guessing and trial-and-error practices. That is, knowledge formation goes together with regular knowledge revision. In a wiki setting, this knowledge (i.e., the wiki content) and its structure evolve with the supporting community. Michel Buffa (Buffa and Gandon 2006) quotes the discussion of wiki creator Ward Cunningham along with Wikipedia founder Jimmy Wales, at the Wikisym'05 conference, who explained the “wiki way” philosophy:

“A wiki is like a garden; users... must take care of it. Start with some seeds and watch it grow, and the wiki will become moderated by its users’ community... Do not impose a rigid structure, users will refactor and structure the wiki as it grows...”

In practice, wikis often become highly disorganized, which in turn results either in lower user participation or in increased time and effort costs for manual maintenance by knowledgeable users. In both cases, this problem hinders wiki adoption in corporate processes. Despite the early identification of refactoring as part of the wiki’s modus operandi, most efforts have been directed to facilitate content editing while content refactoring has been largely overlooked. This paper addresses wiki refactoring.

The aim of wikis is an affordable approach to collaborative knowledge formation and sharing. Wiki refactoring is certainly part of the knowledge formation effort. For wiki refactoring to be effective, it has to be affordable. This is particularly so in a corporate setting where poorly structured wikis lead to important productivity losses. Hence, this paper addresses a twofold issue. First, it systematizes wiki refactoring (the “what”). Second, it assesses wiki refactoring affordances (i.e., a perceived opportunity for action) for current wiki engines (the “how”).

In software, refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behaviour (Fowler 1999). As for wikis, this definition raises two questions: (i) how the wiki structure can be altered i.e., the refactoring operations, and (ii) what this “external behaviour” is i.e., the invariant to be kept during wiki refactoring. Operations include categorize (i.e., characterizing the page content with a category), split (i.e., dividing the page content into two pages), merge (i.e., joining two pages into a single one), etc. As for the invariants, good practices advise to preserve wiki content and authorship. Wiki refactoring can change the wiki internal structure for the sake of navigability, accessibility or comprehension, but the content and its authorship should be kept immutable.

Unfortunately, these operations and invariants are frequently only in the mind of the user with little support from the wiki engines (e.g., MediaWiki). Wiki engines are primarily thought for spurring editions but overlook refactoring. As a result, wiki refactoring might become convoluted. For instance, merging/splitting two wiki articles requires of at least five interactions in MediaWiki. In other words, the semantics of refactoring is not natively supported by the wiki engine. The implications are twofold. First, refactoring is left to the user interpretation. Different users can face the same refactoring problem with different strategies: the wiki engine does not ensure coherence among the refactoring strategies used throughout the wiki lifespan. Second, the engine does not ensure refactoring reliability. Refactoring operations can be thought as database...
transactions in the sense that they might comprise a sequence of wiki interactions that 
(i) should be performed in an all-or-nothing manner, and (ii) should move the wiki to 
a consistent state (i.e., wiki content must be preserved). This operational semantics is
certainly not supported in current wiki engines but lives on the minds of the wiki users.

As a result, those that know what to refactor (e.g., the knowledge workers that know
which articles to merge) might ignore how to refactor (e.g., the operative that goes to
properly merge two articles). As pointed out in (Ford and Geiger 2012) when talking
about the wiki openness,

"this decentralized mode of governance is what has made it possible for the all-
volunteer wikipedian community to collectively build and maintain the project.
However, this mode is often impenetrable for new editors who lack the organizational
literacies required to interpret and author texts and traces."

While cosmetic editing might require minimal familiarization with the domain at hand,
structural refactoring expects deep knowledge about the wiki corpus and refactoring
principles. Means are needed for knowledge workers to refactor their own body of
knowledge by themselves. Based on this observation, we pose the following research
question: how to improve the refactoring affordances of current wiki engines.

We begin by introducing some refactoring scenarios (Section 2). These scenarios ground
the specifics of refactoring when in a wiki setting: the wiki structure, the refactoring
operations, the invariants and the refactoring notices are introduced in Section 3. Next,
we conduct a study about how these operations are supported in MediaWiki as to
evidencing the limitations of current wiki engines (Section 4). Next, we propose to
improve refactoring affordances through the introduction of a Domain-Specific Language
(DSL) for wiki refactoring, WikiWhirl (Section 5). DSLs are reckoned to enhance the
quality, productivity, maintainability and portability while permitting domain experts
understand, validate and develop the DSL expressions themselves (Kelly and Tolvanen
2008). For our purposes, DSLs permit to abstract the terms in which refactoring is
conducted from low-level MediaWiki interactions to refactoring-specific constructs. The
concrete syntax of these constructs (e.g., graphical versus textual) becomes a critical
decision since its adequacy to the target audience (i.e., knowledge workers) might finally
determine the acceptance of the whole approach. We choose mind maps as the concrete
syntax for WikiWhirl (Section 6). A mind map is a diagram used to capture and develop
ideas around a central topic in a radial way. Mind maps match our target audience
(i.e., knowledge workers) who might already be accustomed to apply mind maps during
brainstorming (Buzan and Griffiths 2010). Finally, WikiWhirl is supported as a plugin
for FreeMind (Müller et al. 2013). This popular mind mapping tool is now turned into
an alternative interface to MediaWiki. Users can now interact with MediaWiki through
FreeMind. Unlike the article-centric interface offered by MediaWiki, FreeMind moves the
wiki structure to the forefront. This permits to conceive wiki refactoring as mind map
reshaping: changing nodes of the mind map is interpreted as refactoring operations, and
so are they saved in the underlying wiki database.

Notice that WikiWhirl does not achieve anything that cannot be obtained by directly
interacting through the MediaWiki front-end. The difference stems from refactoring
affordance (who can understand/do the refactoring?), productivity (how long does it
take?), and consistency (i.e., whether refactoring is conducted in the same way, following
good practices, no matter the user). Results from a controlled experiment suggest that
WikiWhirl outperforms traditional wiki front-ends in three main affordance enablers:
global understandability, productivity and automatic compliance with refactoring good
practices (Section 8). WikiWhirl is available to download at http://www.onekin.org/wikiwhirl.

Therefore, this work contributes to the area of wiki refactoring by (i) formalizing a set of refactoring operations, (ii) evidencing the limitation of current approaches from an affordance perspective, (iii) proposing the use of mind maps as a suitable concrete syntax to express refactoring, and (iv) providing WikiWhirl as a proof-of-concept. The ideas presented in this paper originate in our previous work on wiki refactoring (Puente and Díaz 2012). We start by introducing some motivating scenarios.

2. Motivating Scenarios

Corporations are increasingly dependent on a reliable, complete and efficient access to knowledge. Wikis contribute to two major corporate knowledge management issues: knowledge codification and knowledge personalisation Lykourentzou et al. (2011a). Knowledge codification “denotes the gathering and organising of tacit and explicit knowledge that lies fragmented in various resources of the organisation”. Knowledge personalisation materialises by “disseminating and sharing organisational knowledge, through communication and cooperation processes that take place while performing practical business tasks”. Though refactoring issues arise in both scenarios, knowledge personalization is more susceptible to require refactoring amendments. Broadly, the less consolidated the knowledge is, the more likely refactoring would be. This section provides
an example of the evolution of a wiki inspired in a real example, the WikiVet\textsuperscript{1} wiki (Brown \textit{et al.} 2010).

WikiVet is a wiki in the veterinary domain. Figure 1 shows the traditional MediaWiki view of the article Coagulation Tests\textsuperscript{1}. This view is article-centric. However, wiki refactoring is about the structure of the wiki i.e., how content is scattered throughout the hierarchy of categories and articles. This structure is not apparent in Figure 1. You can dig into the article content, looking for links to related articles, or move to the bottom of the page to see how this article is categorized. But again, these insights are specific to the article at hand. If a global view about WikiVet content is sought, users are required to navigate (either by hyperlink or tabbing navigation) through different pages for making “the deep structure” emerge into their minds. This “deep structure” is the subject matter of refactoring.

WikiWhirl moves this structure at the forefront by providing an alternative way of interacting with the wiki: a mind map. Figure 2a depicts the deep structure for WikiVet (partial view), automatically obtained from the wiki database. Broadly, pages are turned into nodes while hyperlinks become edges. The icon of each node denotes its role: the “info” icon stands for article sections, the “edit” icon denotes articles, and the “folder” icon represents categories. Clicking on any of the nodes moves the user to the traditional wiki front-end for the page at hand (i.e., the rendering of Figure 1). On top of this structure, WikiWhirl introduces a set of refactoring operations. Next, we introduce three scenarios to highlight the importance of this alternative representation: wiki initialization, structure refactoring and content refactoring. As we go along, we characterize the refactoring endeavor.

2.1. Scenario 1: Wiki Initialization

Which problem this scenario addresses. Wiki stakeholders should envision a starting point in order to promote initial participation. The paralysis of facing an empty wiki and the lack of explicit statements about the wiki’s purpose might prevent grassroot initiatives from “getting off” the ground (Lykourentzou \textit{et al.} 2011b). This also applies to refactoring efforts.

How this problem is addressed. The above can be addressed by presenting the involved stakeholders with an overview of the already created structure (Díaz and Puente 2012). Figure 2a shows the initial WikiVet structure using the WikiWhirl front-end: a category WikiDrugs with subcategories Anaesthetic_Drugs and Sedatives_and_Transquilisers; a category WikiBlood with subcategories Anaemia, Cells, Immunology and Pressure; the subcategory Immunology has two subcategories Disorders and Flashcards; a category WikiEpi with a subcategory Education; a category WikiPath with subcategories General_Pathology and Clinical_Pathology.

What are the advantages brought by WikiWhirl. WikiWhirl’s mind maps facilitate understanding, and hence, easy consensus. Consensus is reached when all members of a team are willing to support a decision, even though a particular decision may not reflect an individual’s choice of action. Consensus benefits from a clear understanding of each other’s opinions and potential points of friction. Refactoring decisions are not an exception. The wiki content and structure are the result of a consensus among the stakeholders that emerge as different perspectives are considered, and the community

\textsuperscript{1}http://en.wikivet.net/Veterinary_Education Rahmen (accessed May 2013).

\textsuperscript{1}We have worked locally with a reproduction of this wiki.
becomes more acquaintance with the wiki corpus.

2.2. Scenario 2: Structure Refactoring

Which problem this scenario addresses. As time goes by, diverse perspectives begin to emerge. This growth pattern is characterized as “organic” in the sense that it is auto-regulated by the community itself (Cunningham 2006). This results in an increase in the number and size of the articles. Furthermore, the structure expands as new categories are introduced to realize complementary, overlapping or divergent classification criteria for page organization. Growth, and specifically organic growth, might cause disorientation, even more if this growth is not apparent but hidden in terms of pages and hyperlinks among the wiki pages.

How this problem is addressed. Mind maps make the wiki’s hidden structure
explicit. Back to the running example, the initial wiki structure has mutated to the one in Figure 3. Operations might include creation (e.g. creating the category Organisations with two sub-articles International_Intergovernmental and National_Governmental), categorization (e.g. the category Infectious_Agent with the category Anatomy_and_Physiology), uncategorization (e.g. the article Paracetamol from the category WikiDrugs and categorize it with the category Toxicology), renaming (e.g. the article Superficial_Anatomy to Superficial_Anatomy), dropping (e.g. the category Infectious_Agents and all its descendants). Mind maps offer a more intuitive representation than the mere listing of the wiki categories (see Figure 2b).

What are the advantages brought by WikiWhirl. WikiWhirl's mind maps bring focus, and this in turn improves productivity. Focus is obtained on both conceiving (i.e., understanding what to do) and doing refactoring. The former rests on mind maps. When comparing maps versus hyperlinks (i.e., the traditional MediaWiki front-end), evidences are reported about subjects who received the map had significantly less feeling of disorientation and more perception of the content structure than those in the hyperlink group (Su and Klein 2006). On the other hand, WikiWhirl facilitates the doing by providing refactoring-specific operators. For instance, category renaming is not a primitive operation in MediaWiki (i.e. it requires several clicks) while it is provided as a one-click option in the contextual menu of WikiWhirl when a category node is selected. This explains the difference between the 181 seconds that took to conduct the aforementioned refactoring using WikiWhirl, and the 364 seconds involved when resorting to MediaWiki.

2.3. Scenario 3: Content Refactoring

Which problem this scenario addresses. Refactoring is not only limited to categories and article links. Refactoring also impacts how content is arranged as sections within articles. Content refactoring might be due to the tangling and scattering of topics among distinct articles (rather than a one-to-one relationship between topics and articles). This is operationalized as splits and merges upon pages. Unfortunately, splits and merges are quite tortuous in MediaWiki. This jeopardizes both affordance and consistency. The former has to do with the number of artefacts, guidelines and interactions the user has to be aware of to refactor, keeping in mind that complexity undermines “openness” (i.e., who...
can conduct refactoring). On the other hand, consistency refers to whether the question “how refactoring is conducted” gets the same answer no matter the user.

**How this problem is addressed.** Splits and merges involve a set of low-level interactions over the *MediaWiki* front-end. We abstract those operations as a “transactions” upon the *MediaWiki* database, i.e. a set of database operations that account for a meaningful and atomic unit, i.e. the “split” unit and the “merge” unit.

**What are the advantages brought by WikiWhirl.** WikiWhirl supports split and merge as full-fledged operations. This improves coherences (i.e. all splits are conducted in the same way) while relieving the user for the tiny details of the *MediaWiki* interface. Figure 4 illustrates the resulting mind map after conducting different operations on the wiki in Figure 3. This includes splits (e.g. split the article *Steroids* into a new article *Non_Steroids* that will incorporate sections *Mechanism of Action* and *Actions*), merges (e.g. merge articles *Postgraduate_Courses* and *Short_Courses* into a new article *Courses*) and moves (e.g. move the section *Anatomy* of the article *Overview* to the article *Monocytes*). Time wise, previous operations took 14 minutes 13 seconds when conducted directly through the *MediaWiki* front-end. This figure dropped to 2 minutes 39 seconds when using WikiWhirl.

**3. Understanding Wiki Refactoring**

This section sets a common understanding about refactoring in wikis. Wikis support the lifecycle of knowledge construction from embryonic ideas to well-structured comprehensive documentation where distinct articles might be in different stages of their
lifecycle. Even though wikis can keep finalized articles, their added-value rests on the other stages of the article lifecycle: collaborative knowledge formation. Therefore, we regard as the essence of wikis not the support of a consolidated knowledge taxonomy, but rather the sustenance of evolving knowledge. On these grounds, users can and should be allowed to easily alter wiki structure.

Wiki structure is realized as wiki categories, basically tags in the sense that the reason of their use must be evident from the text of the categorized article. MediaWiki regards categories as browsable spaces in which the wiki's content can be partitioned and be dynamically re-arranged just by changing the category: "these tags create links at the bottom of the page that take you to the list of all pages in that category, which makes it easy to browse related articles." These browsable spaces can be disposed along membership relationships so that if logical membership of category C1 implies logical membership of category C2, then C1 should be made a subcategory of C2 (or C2 is the parent category of C1). Such arrangements are referred to as category hierarchies, and they constitute the cornerstone of wiki refactoring. The previous reference also makes an important observation: "Even though category hierarchies can be regarded as 'embryonic taxonomies', they lack any formal semantics. They limit themselves to facilitate location of articles through tag navigation. No inference power is built in."

3.1. Refactoring Operations

Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behaviour (Fowler 1999). This definition raises two questions: (i) how wikis can be restructured, and (ii) what this "external behaviour" is i.e., the invariant to be kept during wiki refactoring.

3.1.1. The Operations

There are three main reasons to include a refactoring operation. First, the operation is natively supported by most wiki engines. This includes:

- **create** i.e., the introduction of a new wiki page that plays the role of either an article or a category,
- **categorize** i.e., characterizing the content of a page through a tag (i.e., category),
- **uncategorize** i.e., removing a tag (i.e., category) from a page,
- **article rename** i.e., changing an article's title, which is used to singularized that article; it impacts in the article's URL,
- **drop** i.e., removing a page from the wiki.

Second, the operation as such is not supported by the wiki engine but is documented as part of Wikipedia's good practices:

- **Category rename**. Unlike articles, it is not possible to rename a category in MediaWiki. It is necessary to create a new category, change the category tag manually on every page, and redirect the old to the new category.
- **Split**. Split is a refactoring process documented by Wikipedia whereby part of the content of a page is migrated to a new page. The two main reasons for split are size

---

3When describing operations, we use "page" to refer to either an article or a category.
and content relevance. If either the page becomes too large or its content seems to diverge between different purposes, then it is considered a split. From an authorship perspective, it is a requirement of Wikipedia licensing that attribution is given to the original author(s), and deletion of that content should be avoided.

- **Merge.** Merge is a refactoring process documented by Wikipedia whereby the content of two pages is collapsed into a new one. Rationales include the unnecessary duplication of content, significant overlap with the topic of another article, and minimal content that could be covered in or requires the context of a page on a broader topic.

Third, the operation is introduced based on our own experience:

- **Section move.** At least during the inception, wikis are in constant evolution. Articles are created, merged, split or deleted as the community gains insights. During this process, we found that articles might be a too coarse-grained unit of rearrangement but rather, sections might better fit as the unit of exchange.

### 3.1.2. The Invariants

Refactoring can change the wiki’s internal structure for the sake of navigability, accessibility or comprehension, but the content (and its authorship) should be kept immutable. Seeking a database parallelism, logical independence is a solid principle of database operation whereby changes in the database schema should minimally disturb client applications. This notion of logical independence rises from any shared resource that evolves: let it be a database schema, a component or a software library. Wikis are shared resources. The question is then what can be affected by wikis’ evolution. While applications are impacted by changes in the database schema, wikis impact end users in their double role of readers and authors. Changing the wiki structure (as result of a refactoring) should cause minimal interference on these activities (i.e., reading and authoring). Hence, the aforementioned refactoring invariant is realized through two independence principles, namely:

- **Readership independence:** readers should be able to reach the same content after or before the refactoring. This principle preserves the content but not where the content is placed. Refactoring can rearrange the very same content along a different set of articles and categories. Such rearrangement should be traceable so that users can easily find the new location, and potentially, reverse malicious refactorings. In addition, since content might be subject to bookmarking, readership independence also preserves URL addresses upon changes on the articles’ title.

- **Authorship independence:** authors should keep their attribution no matter where the content is finally placed. Acknowledging the authorship has been reported as a main motivator of contributions (Arazy et al. 2010). It is also one of the Wikipedia’s good practices. Wiki refactoring must preserve authorship.

To support these principles, additional artefacts are introduced as part of the wiki realization: Talk artefacts (a.k.a. discussion pages) and Revision artefacts. The former hold discussions about the content of the associated page without interfering with content editing (e.g., talk pages might be used to publicize refactoring changes on the associated

---

*From this perspective, readership independence sustains one of the wiki hallmarks: observability. To counteract potential misbehaviour, the community should be able to detect and reverse malicious editions. Likewise, readership independence ensures refactoring changes to be traceable, and hence reversible.*

articles). On the other side, Revision artefacts keep a trace of the edits. In this way, users can monitor and review the work of other users, allowing mistake correction and vandalism reversion. These pages can also be used to trace refactoring changes so that the rest of the community is informed about who, when and how conducts the refactoring. The use of these artefacts to ensure both readership independence and authorship independence is discussed in Section 5.

3.2. The Process of Wiki Refactoring: Requirements

We advocate for wiki refactoring to follow main wiki hallmarks, namely: open, observable and collaborative. Among these requirements, we regard 'openness' as a pre-requisite for the other two. Spurring openness will bring collaboration, which in turn, rests on observability as a trust enabler.

**Open.** It implies lowering the barriers for layman participation. This tenet entails refactoring to be conducted with minimal disturbance (i.e., reducing “accidental complexity”) and in terms closer to the user. This calls for the introduction of Domain-Specific Languages (DSLs) that help users to conduct refactoring in high-level terms (Mernik et al. 2005).

**Observable.** It requires wikis to track changes as well as providing pervasive peer-review mechanisms. To counteract potential misbehaviour, the community can detect and revert malicious editions. Refactoring should also be observable. The appropriateness of a refactoring action (e.g., splitting an article) cannot be generally set by some formal verification but validated by the community. The test for wiki refactoring is whether the community backs the change. This introduces a double communication flow between the refactoring person and the community (observability) and vice versa, from the community to the refactoring person (noticeability). The former implies to keep track of refactoring changes as well as providing pervasive peer-review mechanisms. To this end, a refactoring system should support notices, i.e., announcements about a certain refactoring event (e.g., split notices, merge notices or move notices to inform the wiki community about the namesake operations). This complements revision notices (a.k.a. edit summaries) that are kept as part of the history of the artefacts to help others understand the intention of the edit.

**Collaborative.** Collaborative refactoring benefits from the community to inform about future refactoring actions. Notices can also be used to spot refactoring needs that should be eventually conducted by someone else. This approach is extensively used for articles by introducing boilerplate messages for various issues like copyright violation, neutrality disputes, etc., using a simple shortcut command realized as a “wiki template”¹. Some examples follow²:

- **Catneeded** notice. It suggests for the page to be categorized so that it can be listed with similar pages.
- **Catimprove** notice. It spotlights the need for additional or more specific categories.
- **Catdiffuse** notice. It indicates that any page added to this category should eventually be moved to the appropriate subcategories (diffuse) when sufficient information is available. If this subcategory does not exist yet, either create the subcategory or leave the page in the parent category for the time being.

Table 1. Refactoring affordances for MediaWiki in terms of time (#clicks) and cognitive load (#artefacts).

<table>
<thead>
<tr>
<th>Refactoring operation</th>
<th>Time load</th>
<th>Cognitive load</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># clicks</td>
<td>Revision notice</td>
</tr>
<tr>
<td>Create</td>
<td>3</td>
<td>✓</td>
</tr>
<tr>
<td>Categorize</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>Uncategorize</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>Art. rename</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>Drop</td>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>Cat. rename</td>
<td>5</td>
<td>✓</td>
</tr>
<tr>
<td>Split</td>
<td>6</td>
<td>✓</td>
</tr>
<tr>
<td>Merge</td>
<td>9</td>
<td>✓</td>
</tr>
<tr>
<td>Section move</td>
<td>5</td>
<td>✓</td>
</tr>
</tbody>
</table>

4. Perceived Affordance for Refactoring

This section analyzes refactoring affordances in current wiki engines. Perceived affordance is a sensed opportunity for action (Norman 2002). For example, in the context of a website, any digital native perceives that an underlined text indicates a hyperlink; and, consequently an opportunity to follow (click), and reach information that relates to the hyperlinked word or phrase. Affordability is consubstantial to the wiki’s open principle. Indeed, affordable edition is seen by many as the main breakthrough introduced by wikis. Likewise, refactoring will be affordable or will not be. However, affordance is not an absolute measure. Rather, affordances contextualize refactoring within a given representative user (i.e., the user dimension), framed by a certain socio-cultural environment (i.e., the social dimension) and conducted through a given tool (i.e., the technical dimension) (Mesgari and Faraj 2012, Vatrapu 2010). Specifically, this work takes knowledge workers as the wiki users, corporations (rather than open Internet) as the environment that hosts the wiki, and MediaWiki as the wiki engine to conduct the refactoring. Even though the analysis focuses on this popular engine, the outcomes are generalizable to other wiki engines.

4.1. About the Tool: MediaWiki

As most wiki engines, MediaWiki front-end favours article centricity rather than corpus centricity. That is, the MediaWiki front-end situates the article at the centre of edition, navigation and location. Rationales might be that MediaWiki was conceived for Wikipedia, where the wiki supports an encyclopaedia-like way of reading, arranging and locating information. Traditional encyclopaedias are mainly used for pinpointing a specific topic (i.e., the biography of a given person, the description of a city), and then, moving to entries that depart from this first article of interest. The wiki twist comes from making the encyclopaedia articles openly editable, and offering the means for collaborative edition. This “context of use” grounds the current MediaWiki front-end. However, this work moves the focus to a different “context of use” i.e., refactoring in corporate wikis.

We quantify refactoring affordances for MediaWiki in terms of (i) the number of interactions needed to conduct a refactoring operation, and (ii) the number of different artefacts that need to be manipulated to achieve a refactoring goal. Table 1 shows the outcome. As an example, Figure 5 outlines the different interactions and notices the user has to go through to fulfill a merge operation. The user (1) starts by creating a new page (e.g., Courses) where to move the content of the merging articles (e.g., Short_Courses and Postgraduate_Courses). To ensure readership independence, (2) a redirect notice should be introduced in the merging articles. To ensure authorship independence, (3) a merge
notice is introduced in the merged article to indicate its origin. To facilitate observability, (4) a revision notice (a.k.a. edit summary) is introduced and (5) the corresponding talk pages are accordingly updated for the community to be aware of the change.

The bottom line is that the current MediaWiki front-end makes refactoring convoluted. Users should not only invest time but also need to be aware of the process and notices involved. This jeopardizes affordability, and hence, openness, raising the barrier of wiki literacy. This issue is exacerbated in environments where the wiki structure either evolves frequently or the cost of disorganization is high. This can be the case of corporate wikis.

4.2. About the Environment: Organizations

Corporate wikis are catching on as lightweight knowledge management tools (for a survey of wikis in enterprise refer to (Lykourentzou et al. 2011a)). Indeed, the Intranet 2.0 Global Survey reports that around 67% of the respondent companies were somehow using wikis (Ward 2013). Corporate wikis tend to be smaller than open wikis. An estimate is for corporate wikis to contain an average of 1,500 articles (Stein and Blaschke 2009). This reduction in size however, hides that corporate users tend to be interested in a larger number of wiki pages than e.g., users of Wikipedia. Apart from cosmetic editing (e.g., typo corrections), wikipedians tend to contribute in few more than one article.

By contrast, the interest of corporate employees frequently expands along different wiki pages. This makes both clear structure and easy navigation more prominent in corporate wikis than in open wikis. Specifically, Lykourentzou et. al (Lykourentzou et al. 2011a)

---

state that “structure seems to also play an important role to the success or failure of a corporate wiki implementation. That is, poor structural support often seems to result to laborious information insertion and retrieval, navigation difficulties and information duplication”. In a working setting, these pitfalls impact the productivity of knowledge workers.

4.3. About the User: Knowledge Workers

Open wikis tend to be grounded in altruistic collaboration. By contrast, contribution in corporate wikis is not always a selfless activity at spare time but part of the duties at working time. This also implies that users’ time is more valuable for the organization. In this regard, the tool’s user-friendliness might improve the efficiency and efficacy of knowledge workers. Indeed, this feature is identified as a major technological enabler of the success of a corporate wiki (Lykourentzou et al. 2011a). In addition, “attribution of authorship seems to be the affordance that is most appropriate modification for corporate settings, and users believe that introducing an attribution mechanism will increase their involvement” (Yeo and Arazy 2012).

Figure 6 outlines the previous discussion along the three-dimensional affordance space. Technically, wiki engines incorporate facilities to categorize or rename as well as artefacts for hosting discussions (e.g., talk pages) and tracing within the wiki (e.g., history pages). However, some refactorings expand along different interactions throughout distinct wiki pages. As a result, users conducting refactoring are expected to have additional technical skills besides article editing. Good synthesis skills as well as a good knowledge about the wiki corpus are also recommended. Finally, these interactions are framed by a set of regulations or good practices about preserving authorship and readership. The company’s organigrams or confidentiality regulations might impose additional restrictions about who can refactor and how refactoring can be conducted. The wiki engine does not preclude some ‘socially’ incorrect refactoring (e.g., not preserving authorship). Social surveillance might then be needed to ensure socially acceptable refactoring.

We contend that the current MediaWiki front-end requires users to be literate about the intricacies of refactoring while leaving to the community the detection of socially-disturbing refactoring practices. Leveraging MediaWiki’s refactoring affordances may
be conceived as moving some duties from either “the user” dimension or “the social” dimension to “the technical” dimension (see Figure 6). To this end, we advocate for the use of DSLs (Mernik et al. 2005). Specifically, we introduce WikiWhirl, a DSL conceived as an alternative wiki front-end for refactoring. The construction of this DSL entails systematizing a set of refactoring operations (i.e., the abstract syntax) (Section 5), nailing these operations down into a concrete syntax (Section 6), and providing tool support (Section 7).

5. WikiWhirl: The Abstract Syntax

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 (Vallecillo 2010). This is expressed as the DSL metamodel. Figure 7 shows the WikiWhirl metamodel. WikiWhirl aims to specify a refactoring session. A WikiWhirl session is captured through a wiki model plus as a sequence of refactoring Operations. The Wiki model conceives the wiki as a composition of WikiResources, which are characterised by refactoring_notices. Refactoring_notices are boilerplate messages that spot some refactoring matter for the resource at hand. Subsection 3.2 introduced six such notices.

Resources are classified as RefactoringResources (i.e., the subject matter of the refactoring operations) and SupportingResources (i.e., those needed to support authorship and readership independence). Refactoring resources include Categories, Articles and Sections, where Categories and Articles are composed by Sections. Notice that wiki engines do not support sections as independent artefacts but as embedded inside the wikitext of pages. However, we promote sections as full-fledged classes as

---

1Only first level sections are considered (denoted as ‘== sectionName ==’ in wikitext).
Table 2. Refactoring operations in terms of pre and post conditions.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Pre-condition</th>
<th>Post-condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>create( newTitle ≠ null; Wiki.wikiResources. contains(newTitle) — false )</td>
<td>newTitle — true; Wiki.wikiResources. contains(newTitle) — true; aResource.title — newTitle</td>
<td></td>
</tr>
<tr>
<td>categorize( aResource ∈ RefactoringCompound; aResource ∈ RefactoringCompound; aResource ∈ RefactoringCompound)</td>
<td>contains(aCategory) — true</td>
<td></td>
</tr>
<tr>
<td>uncategorize( aResource ∈ RefactoringCompound; aResource ∈ RefactoringCompound; aResource ∈ RefactoringCompound)</td>
<td>contains(aCategory) — false</td>
<td></td>
</tr>
<tr>
<td>drop( aResource ∈ RefactoringCompound; Wiki.wikiResources. contains(aResource) — true; Wiki.wikiResources. drop(aResource) — false)</td>
<td>contains(RENAMENOTICE) — true; aResource.refactoring notice. contains(SPLITNOTICE) — true; newResource.refactoring notice. contains(SPLITNOTICE) — true;</td>
<td></td>
</tr>
<tr>
<td>split( aResource ∈ RefactoringCompound; Wiki.wikiResources. contains(aResource) — true; Wiki.wikiResources. contains(newResource) — false; Wiki.wikiResources. split(aResource) — true; Wiki.wikiResources. split(aResource) — false)</td>
<td>aResource.title — newTitle; aResource.refactoring notice. contains(SPLITNOTICE) — true; newResource.refactoring notice. contains(SPLITNOTICE) — true;</td>
<td></td>
</tr>
<tr>
<td>merge( resources ∈ RefactoringCompound; Wiki.wikiResources. contains(aResource1) — true; Wiki.wikiResources. contains(aResource2) — true; Wiki.wikiResources. merge(aResource1, aResource2) — true)</td>
<td>newResource.title — aResource1.title + aResource2.title; aResource1.refactoring notice. contains(MERGENOTICE) — true; aResource2.refactoring notice. contains(MERGENOTICE) — true; newResource.refactoring notice. contains(MERGENOTICE) — true;</td>
<td></td>
</tr>
<tr>
<td>move( aSection, sourceResource.sections. contains(aSection) — false; targetResource ≠ null; targetResource. contains(aSection) — true; sourceResource.refactoring notice. contains(MOVENOTICE) — true; targetResource.refactoring notice. contains(MOVENOTICE) — true)</td>
<td>sourceResource.sections. contains(aSection) — false; targetResource. contains(aSection) — true; sourceResource.refactoring notice. contains(MOVENOTICE) — true; targetResource.refactoring notice. contains(MOVENOTICE) — true;</td>
<td></td>
</tr>
</tbody>
</table>

representatives of the page structure. As for the SupportingResources, they include Talks and Revisions. On the other hand, a refactoring operation (Operation) acts upon existing RefactoringResources (i.e., ReferenceArg) or creates a new RefactoringResource (i.e., ResourceArg). RefactoringResources have a title, which is a user-given string that singularized the resource. Operation classes are those introduced in Subsection 3.1.1.

This abstract syntax permits to describe refactoring sessions. Broadly, a textual representation of one such session would be:

Split(Steroids, 'Steroids_split');
Rename(Steroids_split, 'Non-Steroids');
Move(Mechanism_Of_Action, Steroids, Non_Steroids);

where Steroids and Non_Steroids are Article resources, and Mechanism_Of_Action is
However, the important point is not so much about the syntax but how these primitives behave. This behaviour should respect the independence principles enunciated in subsection 3.1.2, namely: (1) readership independence: readers should be able to reach the same content after or before the refactoring, and (2) authorship independence: authors should keep their attribution no matter where the content is finally placed.

Using design-by-contract, we characterize these operations in terms of pre-conditions and post-conditions (see Table 2). The invariant is to keep the wiki content unalterable, i.e., sections (i.e., the artefacts that keep the content) can be re-arranged but not deleted.

These principles are to be engineered as part of the operational semantics of the refactoring operations. Figure 8 provides the operational semantics for $\text{Merge}(a\text{Resource1}, a\text{Resource2}, _\text{newResource})$:

- (lines 1-13) The $_\text{newResource}$ is created as either an Article or a Category. Its title is obtained by concatenating the titles of the merging resources (line 13). A refactoring notice with type MERGENOTICE is added (line 12).
- (lines 14-28) This $_\text{newResource}$ is associated with its first revision (line 28). This implies to first create a revision (line 21) whose content is that of the merging resources plus the merge notice (lines 23-24). Authorship independence advises that attribution is given to the original author. We follow this recommendation by adding the merge notice that indicates the origin of the resource (e.g., “The content of this page has been merged from [[aResource1]] and [[aResource2]] by Admin”) where Admin stands for the current user of WikiWhirl. Along wikitext syntax, square brackets denote the URL of the page at hand. Readership independence suggests that when a page is merged, an edit_summary should be left in the $_\text{newResource}$ (line 25) i.e., content that helps identify the merge operation in the page’s history (i.e., the mergeNotice string).
- (lines 29-30) The $_\text{newResource}$ is categorized along the lines of the merging resources.
- (lines 31-40) Collaborative refactoring implies refactoring decisions to be publically noted. This is achieved through the talk pages associated with $a\text{Resource1}$, $a\text{Resource2}$ and $_\text{newResource}$: the mergeNotice string is posted in the corresponding talk pages. In this way, WikiWhirl leaves a trace of how, when and who conducts the refactoring. This talk page can eventually frame discussions about this refactoring change. These lines create a talk page for the $_\text{newResource}$, and associates the so-created talk page with a revision describing the refactoring. Likewise, talk pages are associated with $a\text{Resource1}$ and $a\text{Resource2}$.
- (lines 41-48). Readership independence also calls for a mechanism to avoid dangling references. Merging two pages creates a redirect from the source pages to the target page. In this way, users’ bookmarked references to the merging pages do not break, but return the merged page.

We provide a brief description for the rest of the WikiWhirl operations (we omit simple operations such as create, rename article, drop, categorize, etc.):

- Rename ($a\text{Category}$, newTitle). Renaming $a\text{Category}$ implies to create a new category with title newTitle and content that of $a\text{Category}$. Previous links to $a\text{Category}$ (from both articles and categories) should be changed to the new category. Talk and revision pages are created to cater for readership independence and authorship independence.

- Split ($a\text{Resource}$, newResource). The title of the newResource results from concatenating the title of $a\text{Resource}$ with “_split” (e.g., “Steroids_split”). As for authorship and readership independence, a comment in the edit summary of the newResource must be made to note the split operation, as well as to create the corresponding discussions in the affected pages.
• **Move** (aSection, sourceResource, targetResource). Here, we follow the same recommendations that for split, i.e., introducing in the associated talk pages a note such as “Section aSection from sourceArticle was copied into newArticle at timestamp”. In addition, the recentChanges page of the sourceResource is to include a summary, noting the origin of this article (e.g., “Section aSection move to [targetArticle]”). Likewise, the recentChanges page of the targetResource should also include the summary “Section
The operational semantics serves to formalize our understanding of the refactoring operations. However, almost as important from an end-user point of view, is how these operations are to be represented. This is the aim of the concrete syntax.

6. WikiWhirl: The Concrete Syntax

There may be different concrete syntaxes for the same abstract syntax, where each concrete syntax might stand for the entirety or a subset of the abstract syntax. Here, we focus on the most complex part: the Wiki construct. A Wiki model is a compound of WikiResources. WikiResources can be RefactoringResources and SupportingResources. The latter are automatically obtained during the refactoring process. Basically, they keep traces and notices for the wiki community to reconstruct the refactoring process. Users only handle RefactoringResources whereas SupportingResources are the byproduct of this process. Therefore, we focus on how to specify Wiki models that contain RefactoringResources since these are the only resources the user has to interact with. Next, we advocate for the use of mind maps as an appropriate graphical notation for Wiki models.

While the abstract syntax addresses expressiveness, the concrete syntax cares for the DSL usability and understandability. Proposed alternatives for wiki visualization are influenced by their aims: visualizing author collaboration (Viégas et al. 2004), depicting article relationships (e.g., Annoki (Tansey and Stroulia 2010)), outlining the wiki structure (e.g., WikiNavMap (Ullman and Kay 2007)), etc. In addition, the expressiveness of these representations should be balanced against affordance for the target audience. Therefore, we need to look into (i) the characteristics of the object to be handled (e.g., its size), (ii) how this object is to be manipulated, and (iii) the context of use i.e., “the users, tasks, equipment, and the physical and social environments in which a product is used” (ISO/IEC 2001).

Wikis are graphs, where nodes stand for pages and edges denote relationships between those pages. For corporate wikis, the size is estimated an average of up to 1,500 pages (Stein and Blaschke 2009). Next, we consider manipulations i.e., the process of refactoring a wiki. Two approaches co-exist. In the bottom-up approach, the user knows the subject of refactoring (i.e., you know which article/category needs to be refactored), and next, a larger view might be required to set this subject into a larger context. By contrast, the top-down approach starts with a global view of the wiki, and next, the user looks for “bad smells” (e.g., too deep category hierarchies with few articles may indicate too much structure). This way of working calls for agile visualizations that permit to define “views” over existing wiki graphs as well as to collapse or extend these views as we gain understanding about the refactoring requirements. Finally, the context of use is that of knowledge workers who access the wiki within the boundaries of an organization.

Based on the previous observations, we advocate for the use of mind maps as the notation for WikiWhirl. A mind map is a diagram used to visually outline information. A mind map is created around a single word or text, placed in the center, to which associated ideas, words and concepts are added1. By making mind maps the concrete syntax of WikiWhirl, we are equating “ideas” to wiki resources, and “idea association”

to categorization. We believe this is reasonable as long as wiki articles tend to be work-in-progress whereas wiki categories (i.e., tags) are used to organize similar resources. However, wikis are graphs whereas mind maps tend to have a radial, hierarchical disposition of nodes. This implies that wiki graphs need to be converted into tree-like structures. This radial disposition of nodes facilitates 'the drilling down' and 'the rolling up' along the different radius of the map, which eases some refactoring processes.

However, a determinant factor for the selection of mind maps is popularity among our target audience. A survey performed to 334 respondents about the use of mind maps in organizations showed that they are mainly used for project planning, brainstorming, knowledge and project management or to-do lists (Frey 2010), all tasks related to knowledge workers. Interestingly enough, 21% of the respondents apply mind maps as a blueprint for the development of websites, i.e., to outline the website structure. Besides that, users perceive many benefits like the improvement of clarity of thinking, the management of information overload or the cleaner visualization of relationships. An evidence of this popularity is the number of tools for mind mapping1. As a result, we select mind maps as the notation for Wiki models.

6.1. Wiki Models as Mind Maps

We begin by introducing the different elements that conform a mind map, and next, we map these elements to the constructs of the Wiki model.

A Map is a compound of Nodes. Nodes have Text as their title and might hold a Link to an external resource (local or remote) as well as a set of properties mainly for rendering concerns. For instance, the Style attribute can be fork or bubble and determines the look of the node as a tagged line or a bubble, respectively. Next, nodes are basically arranged in a tree-like way. Tree structures are constructed using Edges. An edge is a graphical connector that relates a node with its immediate descendants. In addition, Arrowlinks are also connectors but in this case, the connection is between two arbitrary nodes (this enables mind maps to support graph-like structures). Finally, Icons and Fonts can be associated with nodes in an attempt to reflect the underlying semantics of the node (e.g., red font for important nodes). Of course, this semantics resides in the users' head.

Next, we map these elements to the Wiki constructs in Figure 7. Table 3 shows the output. WikiResources are classified as Categories, Articles and Sections. This categorization is denoted through icons: the “folder” icon ☺, the “edit” icon ☑ and the “info” icon ☑ denote category, article and section nodes, respectively. Next, we describe relationships. Since some relationships are M:N (e.g., parentCategories or parentCats in Figure 7), the first relationship instance is represented through an edge (the links that support the tree-like structure) while the rest of the relationship instances are denoted through arrowlinks. Figure 4 illustrates this situation. Category Blood_Changes belongs to two categories; the relationship with Clinical Pathology is denoted by an edge while that of WikiBlood is depicted as an arrowlink.

Finally “catneeded”, “catimprove” and “catdiffuse” notices are mapped to flags icons red ☻, blue ☑, and yellow ☐ respectively. The rest of the notices are not to be provided by the user but automatically generated as part of the operational semantics of the refactoring operations. Therefore, there is not a counterpart in the concrete syntax.

Table 3. From abstract constructs to their realization as mind map primitives.

<table>
<thead>
<tr>
<th>WikiWhirl primitives</th>
<th>Mind map primitives</th>
<th>FreeMind tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Wiki&quot; class</td>
<td>Root node</td>
<td>&lt;node/&gt;</td>
</tr>
<tr>
<td>&quot;title&quot; attribute</td>
<td>Node text</td>
<td>&lt;node text=&quot;title&quot;&gt;</td>
</tr>
<tr>
<td>&quot;URL&quot; attribute</td>
<td>Link</td>
<td>&lt;node link=&quot;url&quot;&gt;</td>
</tr>
<tr>
<td>&quot;Category&quot; class</td>
<td>Node with &quot;category&quot; icon</td>
<td>&lt;icon builtin=</td>
</tr>
<tr>
<td>references:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st &quot;parentCats&quot;</td>
<td>edge</td>
<td>&lt;edge/&gt;</td>
</tr>
<tr>
<td>rest &quot;parentCats&quot;</td>
<td>arrowlink</td>
<td>&lt;arrowlink/&gt;</td>
</tr>
<tr>
<td>&quot;itsPage&quot;</td>
<td>edge</td>
<td>&lt;edge/&gt;</td>
</tr>
<tr>
<td>&quot;Article&quot; class</td>
<td>Node with &quot;article&quot; icon</td>
<td>&lt;icon builtin=</td>
</tr>
<tr>
<td>references:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st &quot;parentCats&quot;</td>
<td>edge</td>
<td>&lt;edge/&gt;</td>
</tr>
<tr>
<td>rest &quot;parentCats&quot;</td>
<td>arrowlink</td>
<td>&lt;arrowlink/&gt;</td>
</tr>
<tr>
<td>&quot;itsPage&quot;</td>
<td>edge</td>
<td>&lt;edge/&gt;</td>
</tr>
<tr>
<td>&quot;Section&quot; class</td>
<td>Node with &quot;section&quot; icon</td>
<td>&lt;icon builtin=</td>
</tr>
<tr>
<td>&quot;catneeded&quot; notice</td>
<td>&quot;catneeded&quot; icon</td>
<td>&lt;icon builtin=&quot;✓&quot;&gt;</td>
</tr>
<tr>
<td>&quot;catimprove&quot; notice</td>
<td>&quot;catimprove&quot; icon</td>
<td>&lt;icon builtin=&quot;✓&quot;&gt;</td>
</tr>
<tr>
<td>&quot;catdiffuse&quot; notice</td>
<td>&quot;catdiffuse&quot; icon</td>
<td>&lt;icon builtin=&quot;✓&quot;&gt;</td>
</tr>
</tbody>
</table>

7. WikiWhirl: Tool Support

This section addresses the construction of a visual tool for WikiWhirl. Its importance should not be underestimated since the success in selecting the right visual metaphors will determine the final adoption of these ideas by the final users. A 3 minutes demo video and a presentation with the main highlights from a user perspective are available at [http://vimeo.com/62145661](http://vimeo.com/62145661) and [http://www.slideshare.net/GorkaPuente/intro-wikiwhirl](http://www.slideshare.net/GorkaPuente/intro-wikiwhirl), respectively. A brief about how to install WikiWhirl can be found at [http://www.onegin.org/wikiwhirl](http://www.onegin.org/wikiwhirl).

WikiWhirl visual editor taps into a mind map visualizer: FreeMind (Müller et al. 2013). With over 6,000 daily downloads, it is one of the most popular mind mapping tools. FreeMind supports easy edition and visualization of mind maps (e.g., nodes are easily moved around, branches collapsed, etc). Choosing an existing tool not only speeds up development, but increases the chances of adoption provided the target audience is already familiar with FreeMind (i.e., employees in corporate wikis).

Turning FreeMind into a wiki refactoring platform implies for FreeMind to become: (i) an editor of wiki maps, (ii) an enactor of refactoring operations, (iii) an interpreter of refactoring operations, and (iv), a workplace for refactoring sessions.

7.1. FreeMind as an Editor of Wiki Maps

Section 6 addresses the expressiveness of mind maps to capture Wiki models. Now, we focus on a particular implementation of mind maps. FreeMind uses XML Schema to specify maps. Figure 9 shows the metamodel counterpart of such schema. Therefore, the mapping between WikiWhirl and FreeMind is described in terms of this schema’s elements (i.e., the XML tags). Table 3 shows the result. For instance, "catneeded", "catimprove" and "catdiffuse" notices are mapped to the “flag-red” icon 🔴, “flag-blue” icon 🔵, and “flag-yellow” icon 🔴 available in FreeMind, respectively. It could have been possible to
introduce more mnemonic ad-hoc icons but this would have moved us outside FreeMind’s default icons\(^1\).

However, this mapping is not enough. A WikiWhirl’s wiki is a compliant FreeMind mind map. However, the opposite does not hold. Some FreeMind maps might not deliver a compliant Wiki model, where compliance is determined by the WikiWhirl’s abstract syntax. Therefore, WikiWhirl maps are a subset of the possible maps that can be drawn in FreeMind. Since FreeMind uses XML Schema to denote what is a valid map, we impose an additional set of XML constraints to restrict those mind maps that account for compliant Wiki models (i.e., conform to the WikiWhirl abstract syntax).

\(^1\)FreeMind provides a fixed set of icons. In the last version, users can introduce their own icons, although interoperability advice against this option.

Figure 9. FreeMind XML Schema for map drawing (represented as an Ecore metamodel).

Figure 10. Extending FreeMind interactions through the contextual right-mouse click.
Table 4. Mapping WikiWhirl operations to FreeMind interactions.

<table>
<thead>
<tr>
<th>WikiWhirl operation</th>
<th>FreeMind interaction</th>
<th>FreeMind Error message if precondition violated</th>
</tr>
</thead>
<tbody>
<tr>
<td>create(newTitle, Page)</td>
<td>click 'create node'</td>
<td>sections cannot be created; newTitle cannot be null; newTitle already exists</td>
</tr>
<tr>
<td>categorize(aResource, aCategory)</td>
<td>draw an arrowlink to a category node or drag a node to a category node</td>
<td>sections cannot be categorized; aCategory cannot categorize itself; aCategory must exist</td>
</tr>
<tr>
<td>un categorize(aResource, aCategory)</td>
<td>click 'remove' an arrowlink to a category node or move a node to the root node</td>
<td>sections cannot be uncategorized; aCategory must exist</td>
</tr>
<tr>
<td>drop(aResource)</td>
<td>click 'remove' a node</td>
<td>sections cannot be removed; aResource has section descendants</td>
</tr>
<tr>
<td>rename(aResource, newTitle)</td>
<td>edit the text of a node</td>
<td>sections cannot be renamed; newTitle cannot be null; newTitle is already taken</td>
</tr>
<tr>
<td>split(aResource, newResource)</td>
<td>select node + right-mouse click &quot;WikiWhirl split&quot;</td>
<td>sections cannot be split; newResource cannot be null; newResource already exist</td>
</tr>
<tr>
<td>merge(aResource1, aResource2, newResource)</td>
<td>select node aResource1 + right-mouse click &quot;WikiWhirl merge&quot;</td>
<td>sections cannot be merged; newResource cannot be null; newResource already exist</td>
</tr>
<tr>
<td>move(aSection, sourceResource, targetResource)</td>
<td>drag a Section node + drop a Section node to targetResource node</td>
<td>aSection must pertain to sourceResource; targetResource must exist</td>
</tr>
</tbody>
</table>

7.2. FreeMind as an Enactor of Refactoring Operations

The implications of FreeMind as an enactor of refactoring operations are two-fold. First, the FreeMind front-end should provide a way for the user to conduct the operation. For instance, creating a category implies to position the cursor in a node, right-click and select “new child node”, name the newly created node and, finally, add the “folder” icon 📁. Most of WikiWhirl operations have a direct mapping to FreeMind interactions (e.g., click, drag, drop, move) (see Table 4). Only Merge and Split do not have a direct counterpart. As a result, we extend the right-mouse-click menu with two options, WikiWhirl split and WikiWhirl merge (see Figure 10). Once a node (or two in the case of merge) is selected, the user can right-click on the mouse to perform the desired operation (i.e., merge or split).

However, some FreeMind interactions, although possible, might be invalid from a refactoring perspective: nodes that stand for sections can only be dragged but never removed; nodes that denote articles/categories can be deleted only if they do not contain sections, etc. That is, preconditions of WikiWhirl operations should be obeyed. Basically, preconditions are to front-end interactions as the metamodel’s structural constraints are to the shape of mind maps: they restrict FreeMind to conform to the WikiWhirl semantics. This leads us to the second implication.

FreeMind acts as the requestor of WikiWhirl operations on user’s behalf. Akin to the design by contract, the requestor should guarantee that preconditions are met before enacting the operation. Therefore, FreeMind, as the requestor, should ensure that user interactions comply with the preconditions. Hence, interactions are guarded by premises whose violation raises error messages (see Figure 4, last column). Figure 11 shows two examples where users are prevented from violating WikiWhirl semantics (although the interactions are totally valid FreeMind interactions). The error messages describe the
violation cause in terms of WikiWhirl semantics. As wisely noted in (Barisic et al. 2011), any user interface is a realization of a language with two directions: human-computer and computer-human, since the feedback from the computer needs also to be interpreted by the humans. Indeed, FreeMind speaks the WikiWhirl language: “Sorry, the content must be preserved” or “Sorry, a section cannot be merged” (as shown in the screenshots) are messages tuned to refactoring-aware users not just mind map draftsmen. In this way, WikiWhirl acts as a learning tool about good practices in refactoring.

7.3. FreeMind as an Interpreter of Refactoring Operations

FreeMind becomes an interpreter of refactoring operations. WikiWhirl’s operational semantics is now concretized for a given wiki engine. We choose MediaWiki. The constructs Article, Category, Talk, etc. find a specific realization in terms of the database schema of MediaWiki. Refactoring operations become transactions over the wiki database, and the operational semantics is realized as an SQL script upon the MediaWiki tables. Appendix A shows the SQL script for the merge operation as conducted in MediaWiki.

7.4. FreeMind as a Workplace for Refactoring Sessions

In this section, we look not at the definition of WikiWhirl but at using WikiWhirl. By experiencing different refactoring scenarios, we come up with additional requirements for FreeMind to smoothly support the process of refactoring. Requirements include: (i) loading of Wiki models, (ii) refactoring transactions, and (iii) FreeMind-MediaWiki roundtrips.

Loading of Wiki models. Most scenarios do not start with an empty wiki but with a wiki that already exists. This requires an “import” utility that obtains a wiki map out of an existing wiki. To this end, FreeMind is extended with an import utility that loads the wiki corpus from a MediaWiki installation. Figure 12a depicts the configuration menu for this utility along two parameters: the configuration service and the load mode. The former indicates the database service parameters (i.e., database name, db user login, db password, db host name). The latter, i.e., load mode, filters the type of nodes to be loaded for the sake of efficiency. Nodes can be filtered by either name or type. The former permits to focus on some part of the wiki based on the name of the category: the category and all its descendants are loaded. By contrast, type-based filtering permits to focus on either structural refactoring (e.g., type = “category”) or content refactoring (e.g., type = 
“content”) by loading only category nodes, or rather, loading all nodes (i.e., categories, articles and sections), respectively. **Refactoring transactions.** *FreeMind* sessions tend to look more like transactions where the user takes the decision about committing the session once he observes the resulting map. *FreeMind* has been extended with a **tracking window** (see Figure 12b) where user interactions are described in terms of *WikiWhirl* operations. This window offers three buttons: the **Commit** button that makes the changes to endure in the wiki database; the **Delete** button, that removes the highlighted operation from the trace; and the **Refresh wiki mind map** button, which restores a database dump, and regenerates the mind map. Workers can play around, exchange insights, and finally, commit to the resulting structure.

**FreeMind-MediaWiki roundtrips.** Users tend to move back and forth between the *FreeMind* view and the *MediaWiki* view. *FreeMind* permits a smooth transition between both views by turning node titles into URLs. In this way, the wiki map becomes “a site map” for *MediaWiki* installations i.e., the map accounts for a global view of the wiki where users are a click-away from the *HTML* page in the *MediaWiki* front-end. This serves two purposes. First, to access the full content of the article, should this matter for the refactoring purpose. Second, to check out the outcome of the refactoring operations. Once some refactoring operations are committed, users can move to the *MediaWiki* view to see the impact of such refactoring. This reinforces the perception of the *WikiWhirl* and the *MediaWiki* front-ends as complementary views of the wiki corpus.

### 7.5. Architecture of the WikiWhirl Plugin

The *WikiWhirl* plugin pivots around three main components (Figure 13):

- The **core** component, which confines *FreeMind* to the semantics of *WikiWhirl*. This includes the additional restrictions on the XML Schema for map description, the validation of the pre-conditions upon user interactions, and the tracking window.
The *importer* component, which extracts mind maps out of the wiki databases. For this purpose, we used *Schemol* (*Díaz et al.* 2013), a model injector for databases.

The *exporter* component, which uploads the *FreeMind* map to the *MediaWiki* database. This process is threefold. First, the refactoring trace (as specified in the tracking window) is mapped to a *WikiWhirl* model using the EMF persistence framework. Second, the *WikiWhirl* model is transformed into an SQL script. This model-to-text transformation is realized through *MOFScript*. Finally, this SQL script is enacted against the *MediaWiki* DBMS (i.e., *MySQL*).

Wiki engines other than *MediaWiki* can be used. This would require of new "database drivers" for both the *importer* and the *exporter*. Specifically, the *importer* would ask for new *Schemol* transformation to cope with the new database schema. The *exporter* would need a new *MOFScript* transformation to generate the required SQL statements for the new DBMS.

Architecturally, the *WikiWhirl* plugin is a traditional database application: it queries the *MediaWiki* database, processes the tuples (e.g., renders the mind map), and saves some tuples back. By large, the most costly part is querying the database. Specifically, the penalty goes with (i) extracting sections out of large articles (since all the raw text of articles needs to be parsed), and (ii), coping with large category hierarchies (due to the need of recursively traverse the category tree to deduce all the parent-child relationships, which involves more database tables). We conducted some tests to measure how *WikiWhirl* performs on four scenarios. The tests were carried out on a desktop PC Quad core 8GB RAM 64-bit connected to a remote database. Figure 14 depicts how *WikiWhirl* scales as the number of nodes increases:

- **Wiki initialization.** This occurs at the wiki onset when an initial scaffold has to be created. We estimate around 50 categories to be involved in this stage. Importing those 50 categories into *FreeMind* took 20 seconds.
- **Structure refactoring.** This is periodically conducted during the wiki lifecycle. If the structuring is focalized on a given category, we calculate an average of four subcategories with five articles each. This accounts for a total of 25 nodes (source category + 4 subcategories + 20 articles) with a load cost of 6 seconds.
- **Content refactoring.** This is similar to the previous case but now the 'section' nodes are also added. We calculate an average of 4 sections per article. In total, 105 nodes (5

---

Table 5. Performance test in four scenarios.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Nodes</th>
<th>Load time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Categories</td>
<td>Articles</td>
</tr>
<tr>
<td>Wiki initialization</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>Structure refactoring</td>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>Content refactoring</td>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>Refactoring complete view</td>
<td>70</td>
<td>350</td>
</tr>
</tbody>
</table>

categories + 20 articles + 80 sections) with a cost of 15 seconds (only 1.5MB of raw text).

- Refactoring *complete view*. In this case, the whole wiki is rendered. This might be occasionally conducted as a plenary session of the wiki stakeholders. Also, newcommers might benefit from having a global perspective about the wiki. Most frequently, this can be satisfied by rendering only categories and articles, without loading the sections. For this last scenario, we have used the wiki of our research group. Performance wise, 350 articles arranged along 70 categories took 39 minutes to load. This figure went up to almost 70 minutes when the 800 sections were also loaded (what in our wiki means 150MB of raw text!). The resulting mind map contains 1220 nodes. This is an extreme (and odd) case that jeopardizes the approach not so much because of performance but because of the handling of cluttered maps. Nevertheless, *FreeMind* provides a filter mechanism that helps in the duty, showing only those nodes needed in the refactoring session.

These figures make us conclude that *WikiWhirl* handles efficiently most of the common refactoring scenarios. Worth noticing, once tuples are loaded and the map displayed, the database session is closed. This implies that keeping the map on display has no impact on the wiki database. Hence, it is possible to use the very same *WikiWhirl* map for different refactoring sessions, then, saving the loading phase at the onset. This might cause the map to get outdated as other users update the wiki through the *MediaWiki* view (e.g., deleting a category or an article). This is certainly possible. Should this happen, a transaction error would arise at the time the user attempts to save the refactoring. The user would need to reload the map and start again. It is up to the user to estimate the likelihood of this error based on both the potential number of users and the scope of the refactoring.
8. Evaluation

WikiWhirl aims to increase refactoring affordance for knowledge workers. Affordance is a perceived opportunity for action and perception is the organization, identification and interpretation of sensory information in order to represent and understand the environment. Here, the environment is the wiki, and the action on this environment is refactoring. WikiWhirl seeks to increase the perceived opportunity for wiki refactoring (the action). This is achieved by (i) surfacing the structure of the wiki corpus through mind maps ("the sensory information"), and (ii) conducting refactoring as mind map reshaping. In this section, we evaluate to which extent this aim has been fulfilled.

We compare WikiWhirl with MediaWiki front-end (i.e., the baseline alternative) along three measures: understandability of the wiki structure, effectiveness in refactoring, and productivity (i.e., efficacy) in refactoring. The rationales are that "enhancing understanding will ease the detection of refactoring opportunities which, in turn, will impel users to refactor provided the means to do so in a effective and efficient way are available."

If either the wiki structure is difficult to grasp or the refactoring means are cumbersome to use, then, users will not be inclined to act. We use a controlled experiment to evaluate this insight. A controlled experiment permits to assess the existence of a cause-effect relationship between the use of WikiWhirl and an increased affordance. In software engineering, a controlled experiment can be defined as a randomized experiment (a.k.a. quasi-experiment) in which individuals or teams (the experimental units) conduct one or more software engineering tasks for the sake of comparing different populations, processes, techniques or tools (the treatments) (Sjoberg et al. 2005). In our case, the target audience are knowledge workers who are accustomed to article editing, but with no specific wiki refactoring background. Next, we delve into the details along the framework proposed in (Jedlitschka and Pfahl 2005).

8.1. Experimental Design

First, we state the goal of the experiment using the Goal/Question/Metric (GQM) method (Basili 1992):

- analyse the wiki refactoring affordance of knowledge workers working with WikiWhirl
- for the purpose of comparing it with a baseline alternative (MediaWiki)
- with respect to their effectiveness, global understandability and productivity
- from the point of view of a researcher trying to assess WikiWhirl
- in the context of a case study on selected representative WikiWhirl operations.

This goal introduces three dependent variables, namely:

1. Global understandability. It is defined as the effort required for the reading and correct interpretation of the artefact at hand (e.g., a wiki) (Patig 2008). Measuring understandability is still open to debate. Here, we follow the recommendations proposed in (Patig 2008). We introduce a set of multiple choice questions regarding the structure and the semantic content of a wiki. Both, the number of questions answered in the allocated time, and the number of such questions that were correctly answered are evaluated.

2. Effectiveness. It is defined as "the capability of the software product to enable
users to achieve specified goals with accuracy and completeness in a specified context of use” (ISO/IEC 2001). To measure accuracy and completeness, a point will be assigned to each action that follows the good practices in the course of a refactoring operation. For instance, 7 points can be earned for a merge if the user performs the following tasks: create new article, copy and paste content of each source article, add summary, create discussion, and create redirects from both source articles to the new one. The more points the participant obtains, the more complete and accurate the task outcome. In addition, we also measure whether the participants have been able to complete the task in the allocated time.

(3) **Productivity.** It is defined as “the capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use” (ISO/IEC 2001). In our context, productivity is used for its relation with time availability. Productivity will be measured using task completion time.

The **independent variables** include WikiWhirl as the tool to be measured, and MediaWiki as the baseline alternative. Hence, the experiment follows a unifactorial design where the rest of the parameters are controlled. These **controlled parameters** include:

1. **The participants’ previous exposure to MediaWiki.** We control this variable through a randomized block design where MediaWiki knowledge is the blocking variable (Spector 1993). That is, participants are divided into three groups depending on their previous MediaWiki skills, and only inside these groups are they randomly assigned to the WikiWhirl or MediaWiki groups. This ensures a comparable MediaWiki background in both experimental groups.
2. **The subjects’ familiarization with the domain used for the wiki in the experiment.** We ensure equal understandability by selecting a wiki from the veterinarian domain where the participants have a similar and scarce knowledge.
3. **The experimental setting.** The very same laboratory, experiment start time, observers and trainers will be used for both groups.

**Hypotheses.** The goal above leads to the following hypotheses:

- **H1null** Using WikiWhirl has no impact on **effectiveness** when contributors perform refactoring operations on a wiki.
- **H1alt** Using WikiWhirl has a significant impact on **effectiveness** when contributors perform refactoring operations on a wiki.
- **H2null** Using WikiWhirl has no impact on **global understandability** when contributors perform refactoring operations on a wiki.
- **H2alt** Using WikiWhirl has a significant impact on **global understandability** when contributors perform refactoring operations on a wiki.
- **H3null** Using WikiWhirl has no impact on **productivity** when contributors perform refactoring operations on a wiki.
- **H3alt** Using WikiWhirl has a significant impact on **productivity** when contributors perform refactoring operations on a wiki.

**Participants.** Participants will be recruited among the Computer Science Faculty and Ph.D. students of the University of the Basque Country. The only prerequisite is to have experience on wiki editing with MediaWiki. Participants from this sample can be assumed to have a similar background, while having an above average command of software and new technologies (see Subsection 8.4).

**Tasks.** Three tasks were designed:
(1) **Comprehension Task.** In order to measure global understandability, participants are confronted with a subset of the *WikiVet* wiki (see Figure 3), and a set of questions to assess to which extent they apprehend the deep structure of this wiki (see Appendix B). The questionnaire was prepared along the guidelines of (Patig 2008).

(2) **Structure-refactoring Task.** Based on this initial *WikiVet* wiki, participants will conduct distinct structure refactoring operations: create, categorize, uncategorize, rename and drop. The description of these tasks corresponds to the one introduced in Subsection 2.2.

(3) **Content-refactoring Task.** Based on the initial *WikiVet* wiki, participants are told to conduct distinct content refactoring operations: split, merge and move. The description of these tasks corresponds to the one introduced in Subsection 2.3.

Note that wiki initialization is left out of the experiment, as wiki initialization does not per se involve refactoring. It was excluded to shorten the experiment and avoid negative position effects (e.g., participants getting bored or tired) (Sarafino 2005).

**Instrumentation.** As aforementioned, a randomized block design is used using previous *MediaWiki* knowledge as the blocking variable. To this end, a questionnaire on *MediaWiki* knowledge and the frequency of its use by participants was designed (see Appendix B). A 50 minute training session was created for both groups. This training teaches how to perform refactoring operations with either *MediaWiki* or *WikiWhirl* depending on the group. For the experiment itself, handouts were written with textual instructions (e.g., what operation to perform on the wiki, when to take note of the time, etc). The slides used in the training (the same number of slides for both groups), will also be provided to participants. Two additional questionnaires were created, one to evaluate global understandability (see Appendix C) and another to gather the final results (see Appendix D).

**Data Collection Procedure.** Three different ways of data collection are envisioned. First, two observers will be present on each group to gather qualitative data on the execution. Second, execution time and questionnaire responses will be manually introduced by participants using the corresponding online questionnaire (see Appendix D). Third, authorship and readership errors will be manually collected by the observers after the execution of the experiment by recovering database dumps and logs from the computers used by participants. All this data will be collected anonymously. The data from the different sources (i.e., previous *MediaWiki* knowledge, understandability questionnaire, final questionnaire and computer data) for the same participant will be linked using a code.

**Analysis Procedure.** Descriptive statistics will be used to characterize the sample and to evaluate the participants’ experience. Moreover, *t*-test analyses will be performed to assess differences among the *MediaWiki* and *WikiWhirl* groups. In case of statistically significant differences, *Cohen’s d* will be used to calculate the size of the effect (Cohen 1988). IBM SPSS Statistics 20 for Windows¹ will be employed to perform the different analyses.

8.2. Execution

We summarize the execution of the aforementioned experiment conducted in May 2013. A call was issued to the faculty and Ph.D. students of the Faculty of Computer Science of the University of the Basque Country asking for volunteers to participate in the experiment. The call specified that wiki contributors were sought, so knowledge of how to edit an article was a must. It also stated that participants of the experiment would learn about wiki refactoring and enjoy a drink and a snack after completion. Thirteen people answered the call, three lecturers and ten Ph.D. students. Before the experiment they answered the questionnaire regarding MediaWiki knowledge. When processing their answers, an outlier who had a considerable experience in wiki refactoring was identified. Out of 34 possible points in the questionnaire, he obtained 27 while the second person with more MediaWiki experience obtained 16. Hence, the outlier was eliminated from the sample. As a result, a fairly homogeneous group of 12 participants was obtained, with an average of 10.33 points (out of 34) and a standard deviation of 2.57 in MediaWiki background. All participants indicated they knew how to edit an article and 75% knew how to create one, but 50% never actually perform the operation. Only two participants reported they knew how to create categories and none of them knew how to delete articles. None ever performed refactoring operations (i.e., reorganize existing articles and categories).

To ensure these 12 participants were evenly distributed among the MediaWiki group and the WikiWhirl group, a randomized block design was used. Participants were classified along three clusters of four people each based on the points obtained in the questionnaire: 7 to 9, 10 to 11 and 11 to 16. Within each cluster, participants were randomly assigned to either the MediaWiki group or the WikiWhirl group.

Hardware homogeneity was ensured by using the very same laboratory for both groups. All participants used computers with the same features (i.e., Intel Core 2 1.86 GHz, 3 GB RAM and Windows XP Professional SP3) and a clean installation of MediaWiki version 1.16.1. The individual MediaWiki installations are required to allow each participant to perform her own refactoring operations. On top of this, the WikiWhirl participants also had FreeMind 0.9.0 and WikiWhirl 0.3 installed. Each group was cited at the same hour (4:30 pm) of two different days. The same observers participated.

Before the experiment began, every participant signed an informed consent. Both groups received a 50' training on wiki refactoring. The same trainer gave a brief introduction on wikis, wiki evolution and wiki refactoring (10') and then explained the different operations that are performed during refactoring (40'). As the operations were being described, participants executed them in a local installation each had for learning purposes. This training installation is based on Evelopedia, a wiki on online games\(^1\). This training was followed by the three experimental tasks (see Subsection 8.1).

8.3. Analysis

In this section, we analyse the data collected from the experiment. Descriptive statistics are described first, then hypothesis testing is carried out followed by an analysis of other parameters that might have influenced the result.

\(^1\)http://wiki.eveonline.com/ (accessed May 2013).
Table 6. Participant Background along a Likert scale from 1 (none) to 5 (expert).

<table>
<thead>
<tr>
<th>Item</th>
<th>Mean</th>
<th>St. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming</td>
<td>4.00</td>
<td>0.74</td>
</tr>
<tr>
<td>Web Engineering</td>
<td>3.75</td>
<td>0.62</td>
</tr>
<tr>
<td>Web 2.0</td>
<td>3.33</td>
<td>0.89</td>
</tr>
<tr>
<td>Collaborative Systems</td>
<td>2.41</td>
<td>0.51</td>
</tr>
<tr>
<td>Mind Maps (WikiWhirl participants)</td>
<td>2.14</td>
<td>1.12</td>
</tr>
</tbody>
</table>

Table 7. Experimental measures.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Metric</th>
<th>MediaWiki</th>
<th>WikiWhirl</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>St. Dev</td>
<td>Mean</td>
</tr>
<tr>
<td>Global Understandability</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># Answers (out of 14)</td>
<td>6.00</td>
<td>1.09</td>
<td>14.00</td>
</tr>
<tr>
<td># Correct Answers</td>
<td>4.50</td>
<td>0.84</td>
<td>12.50</td>
</tr>
<tr>
<td>Effectiveness</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># Struct. Ref. Completed</td>
<td>0.33</td>
<td>0.52</td>
<td>1.00</td>
</tr>
<tr>
<td>(range: '1' fully, '0' none)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># Struct. Ref. Points</td>
<td>14.67</td>
<td>4.80</td>
<td>23.00</td>
</tr>
<tr>
<td>(range: '23' fully, '0' none)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># Cont. Ref. Completed</td>
<td>0.50</td>
<td>0.55</td>
<td>1.00</td>
</tr>
<tr>
<td>(range: '1' fully, '0' none)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># Cont. Ref. Points</td>
<td>20.83</td>
<td>6.46</td>
<td>36.00</td>
</tr>
<tr>
<td>(range: '36' fully, '0' none)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Productivity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>elapsed time for the comprehension task (minutes)</td>
<td>20.00</td>
<td>0.00</td>
<td>12.17</td>
</tr>
<tr>
<td>elapsed time for the structure-refactoring task (minutes)</td>
<td>38.17</td>
<td>3.61</td>
<td>9.67</td>
</tr>
<tr>
<td>elapsed time for the content-refactoring task (minutes)</td>
<td>17.17</td>
<td>3.19</td>
<td>9.71</td>
</tr>
</tbody>
</table>

**Descriptive Statistics**

**Participant background.** Twelve participants from the Faculty of Computer Science of the University of the Basque Country participated in the experiment (3 lecturers and 9 Ph.D. students). The majority of participants were male (58.3%). Regarding age, the average was 31.5 years with a standard deviation of 6.17. With respect to their background, participants had to rate themselves using a Likert scale from 1 (none) to 5 (expert). Results can be seen in Table 6. Note that participants working with WikiWhirl were asked their background with mind maps, as it could have an impact on their performance.

Dependent variables (see Table 7):

1. **Global understandability.** It is measured through the comprehension task. Specifically, we count the number of answers fulfilled in the allocated time (20’) and, out of them, the number of correct answers.

2. **Effectiveness.** It is measured through the refactoring tasks. Through a questionnaire, participants indicated to which extent they considered that the tasks have been completed in the allocated time (20’) where “1” indicates completion and “0” non completion. It is worth noticing that in the MediaWiki group only one third of the participants considered that they had completed the structure refactoring task. Moreover, we also measured to which extent the task was correctly achieved, i.e., preserving authorship and readership independence (e.g., including redirects, notices, etc). In this case, WikiWhirl participants obtained all the possible points, as these actions are transparently performed by WikiWhirl.

3. **Productivity.** We measured the time spent in each task: the comprehension task, the structure-refactoring task and the content-refactoring task.
Hypothesis Testing

In this section, we test the three hypotheses regarding global understandability, effectiveness and productivity. Hypotheses are evaluated using a t-test. The outcome is displayed using tables where the t and p columns present the t value in the t-test and the significance, respectively. In metrics where statistically significant differences were found, the effect size was calculated using Cohen’s d (Cohen 1988). The values are presented in the last column of these tables.

We begin by analyzing effectiveness. The hypothesis reads as follows:

- **H1null** Using WikiWhirl has no impact on effectiveness when contributors perform refactoring operations on a wiki.
- **H1alt** Using WikiWhirl has a significant impact on effectiveness when contributors perform refactoring operations on a wiki.

Table 8 summarizes the results. Regarding the structure refactoring task, the table shows statistically significant differences, i.e., WikiWhirl helps participants to fulfill timely (“Struct. Ref. Completed” metric) and completely and accurately (“# Struct. Ref. Points” metric) the task at hand. On the other hand, the content refactoring task exhibits no statistically significant difference for the “Cont. Ref. Completed” metric (i.e., p > 0.05), although such difference is found for the “# Cont. Ref. Points” metric. This is most interesting as it seems to suggest that MediaWiki participants thought they had fulfilled the task at hand but without following the good practices of preserving authorship and readership independence. For the metrics where statistically significant differences were found (i.e., p < 0.05), the effect size was calculated using Cohen’s d. A value of this parameter above 0.8 in absolute values denotes a large effect size as a result of the independent variable (i.e., the introduction of WikiWhirl) (Cohen 1988). Except for the “Cont. Ref. Completed” metric, which indicates a subjective perception of participants, the other metrics show considerable effect sizes and soundly sustain the case that the null hypothesis can be rejected, and hence,

WikiWhirl has a significant impact on effectiveness for wiki contributors

The second hypothesis is related to global understandability:

- **H2null** Using WikiWhirl has no impact on global understandability when contributors perform refactoring operations on a wiki.
- **H2alt** Using WikiWhirl has a significant impact on global understandability when contributors perform refactoring operations on a wiki.

Table 9 presents the results. Statistically significant differences were found in both the number of answered questions (“# Answers” metric) and the number of them that were correct (“# Correct Answers” metric). Cohen’s d shows large effect sizes for both metrics. Hence, the null hypothesis is rejected and we infer that using

---

### Table 8. Effectiveness test outcome.

<table>
<thead>
<tr>
<th>Metric</th>
<th>t</th>
<th>p</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Struct. Ref. Completed</td>
<td>-3.62</td>
<td>0.025</td>
<td>-1.825</td>
</tr>
<tr>
<td># Struct. Ref. Points</td>
<td>-4.250</td>
<td>0.008</td>
<td>-2.454</td>
</tr>
<tr>
<td>Cont. Ref. Completed</td>
<td>-2.236</td>
<td>0.076</td>
<td></td>
</tr>
<tr>
<td># Cont. Ref. Points</td>
<td>-5.748</td>
<td>0.002</td>
<td>-3.319</td>
</tr>
</tbody>
</table>

### Table 9. Global understandability test (H2).

<table>
<thead>
<tr>
<th>Metric</th>
<th>t</th>
<th>p</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td># Answers</td>
<td>-17.889</td>
<td>0.000</td>
<td>-10.328</td>
</tr>
<tr>
<td># Correct Answers</td>
<td>-14.606</td>
<td>0.000</td>
<td>-8.433</td>
</tr>
</tbody>
</table>
WikiWhirl has a significant impact on global understandability when contributors perform refactoring operations

The significant differences between MediaWiki (4.5 mean) and WikiWhirl (12.5 mean) on correctly answering the test in the 20’ frame (see Table 7) highlight the difficulty of easily grasping the wiki corpus with current wiki front-ends. As a conjecture, this situation might be due to wikis being initially thought for open encyclopedia-like wikis. Having a global understanding of the wiki corpus (even a subpart of it) was not a priority. Wikipedia is article minded rather than corpus minded. It sits the article at the center for editing, searching and navigating. However, corporate wikis look at wikis not only as archives of knowledge for later referral but also as enablers of knowledge formation. This might require to step back and look at the structure of the corpus before delving into a particular article. What is needed is a fluent mechanism that permits switch between both views.

The last hypothesis is related to productivity.

- *H3null* Using WikiWhirl has no impact on productivity when wiki contributors perform refactoring operations on a wiki.
- *H3alt* Using WikiWhirl has a significant impact on productivity when wiki contributors perform refactoring operations on a wiki.

Table 10 shows how statistically significant differences were found in the time participant spent in each task. Cohen’s $d$ again shows large effect sizes for the three metrics that were evaluated. Hence, the null hypothesis is also rejected and we infer that

WikiWhirl has a significant impact on productivity when contributors perform refactoring operations

**Parameter Influence**

We wanted to check whether previous background on mind mapping would favour the positive results obtained by the WikiWhirl group. Hence, we asked WikiWhirl users to rate their background in the matter using a Likert scale from 1 (none) to 5 (expert). The results from the four participants with low previous knowledge (2 or less in the scale) and the two ones with higher knowledge (3 or more) were compared.

Table 11 presents the results. In the cases of number of answers, completion of structure and content refactoring and the associated points results are exactly the same for both groups (hence, no t-test could be performed). This shows that there were no differences among both groups. For the rest of the metrics, no statistically significant differences were found between the results of both groups either.

Overall, the results support the hypotheses regarding the gains obtained by empowering knowledge workers with a tool to ease refactoring. We can conclude that the use of WikiWhirl significantly improves wiki refactoring affordance for wiki contributors by improving global understandability, by reducing refactoring procedure skills and by reducing the need for time availability as it increases productivity.

<p>| <strong>Table 10. Productivity test (H3).</strong> |</p>
<table>
<thead>
<tr>
<th><strong>Metric</strong></th>
<th><strong>t</strong></th>
<th><strong>p</strong></th>
<th><strong>d</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Understandability</td>
<td>5.248</td>
<td>0.003</td>
<td>3.030</td>
</tr>
<tr>
<td>Struct. Ref.</td>
<td>6.212</td>
<td>0.000</td>
<td>3.586</td>
</tr>
<tr>
<td>Cont. Ref.</td>
<td>4.389</td>
<td>0.001</td>
<td>2.534</td>
</tr>
</tbody>
</table>

...
Table 11. Mind map knowledge comparison.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Metric</th>
<th>Low Knowledge</th>
<th>High Knowledge</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>St. Dev</td>
<td>Mean</td>
<td>St. Dev</td>
</tr>
<tr>
<td>Global Understandability</td>
<td># Answers</td>
<td>14.00</td>
<td>0.00</td>
<td>14.00</td>
</tr>
<tr>
<td></td>
<td># Correct Answers</td>
<td>12.00</td>
<td>0.82</td>
<td>13.50</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>St. Dev</td>
<td>Mean</td>
<td>St. Dev</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>Struct. Ref. Completed</td>
<td>1.00</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td># Struct. Ref. Points</td>
<td>23.00</td>
<td>0.00</td>
<td>23.00</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>St. Dev</td>
<td>Mean</td>
<td>St. Dev</td>
</tr>
<tr>
<td>Productivity</td>
<td>Struct. Ref. Completed</td>
<td>1.00</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td># Cont. Ref. Points</td>
<td>36.00</td>
<td>0.00</td>
<td>36.00</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>St. Dev</td>
<td>Mean</td>
<td>St. Dev</td>
</tr>
<tr>
<td>Understandability</td>
<td>11.75</td>
<td>4.57</td>
<td>13.00</td>
<td>1.41</td>
</tr>
<tr>
<td></td>
<td>Struct. Ref. Completed</td>
<td>9.25</td>
<td>0.96</td>
<td>10.50</td>
</tr>
<tr>
<td></td>
<td>Cont. Ref. Completed</td>
<td>9.75</td>
<td>3.77</td>
<td>8.00</td>
</tr>
</tbody>
</table>

8.4. Threats to Validity

A main concern for internal validity in our case is the appropriateness of the sample. The first issue relates to sample size. Johnson et al. suggest six participants per group as the minimum required for a controlled experiment (Johnson 1992). Our experiment accounted precisely for six subjects in each group. Even though our results show statistically significant differences and large effect sizes, larger groups are needed to corroborate these findings.

A second issue involves the participants' background. So far, the experiment was conducted in a controlled environment, where participants were closely guided in the tasks. While the refactoring tasks as such tend to be similar no matter whether the setting is academic or business oriented, the user motivations and expertise might vary substantially. As for the skills, the participants have a strong background in Computer Science. This certainly facilitates the engagement with a software technology such as wikis. Indeed, the use of wikis for information system development is among the most thriving ones (Lykourentzou et al. 2011a). We hypothesise that the differences would have been larger between the two subject groups if the experiment were conducted among non savvy people.

Future work includes further experiments to quantify the benefits of WikiWhirl. Specifically, we reckon that authorship preservation would be most appreciated in a business setting. Some authors already recognized the importance of authorship recognition as an enabler of knowledge sharing (Wasko and Faraj 2005). Unlike bare MediaWiki, WikiWhirl ensures appropriate authorship tracing by construction. By rearranging articles and sections through WikiWhirl, the organization guarantees that appropriate cues are left to track back the authors.

9. Related Work

This work builds upon previous insights from two main areas: wiki visualization and wiki management. Wiki visualization plays a preponderant role on understandability, and hence, on affordance. The selection of the visualization method depends on the requirements to meet. Proposed alternatives for wiki visualization are influenced by their aims: explore wiki structure (e.g., Wiki Explorer\(^1\), WikiNavMap (Ullman and

\(^1\)http://www.kinf.wiai.uni-bamberg.de/mwstat (accessed May 2013).
Kay (2007)), visualize author collaboration (e.g., HistoryFlow (Viégas et al. 2004) or depict article relationships (e.g., Annoki (Tansey and Stroulia 2010)). Annoki (Tansey and Stroulia 2010) is a set of extensions for MediaWiki with two interesting inputs: WikiMap and wiEGO. The latter serves to derive the sections of a page from a user-drawn graph. As for WikiMap, it focuses a page at a time, and for the targeted page, shows different elements: linked pages, authors, etc. Of a similar flavour is WikiMindMap (Nyffenegger 2009). This plugin provides a page-centric hyperlink-driven visualization of Wikipedia. Using a mind map, WikiMindMap creates a node for both, each section and each link within the input article. The differences with WikiWhirl stems from (i) the aim (visualization vs. refactoring), (ii) the focus (sections and intra-links vs. category-based structure), and (iii), the scope (Wikipedia alone vs. any MediaWiki wiki). Finally, and besides categories, Wikipedia provides additional means for grouping articles: lists and navigation templates. Using pre-defined templates, a set of related articles can be referred to within a common structure. These mechanisms can be regarded as a kind of arrangement of set of articles to ease location. Unlike categories, no auto-linking is provided, i.e., including an article in a list does not automatically link back the article to this list. This showcases how MediaWiki supports a basic semantics of categories which separate them from mere hyperlinks between pages. In a corporate setting, we do not regard an extensive use of these mechanisms, mainly thought for bulky wikis where the sheer number of articles requires of additional organization mechanisms besides categories. Hence, WikiWhirl does not address lists nor navigation templates.

Most of the previous approaches pave the way towards the so-called “visual wikis” (Hirsch et al. 2010), i.e., a combination of a visual and a textual representation of a wiki. From this perspective, WikiWhirl offers a hybrid solution where FreeMind offers the visual view whereas MediaWiki accounts for the text. However, these tools look at visualization as an end in itself whereas for WikiWhirl visualization is a means towards refactoring. Hence, WikiWhirl does not offer the detailed revision trace of WikiTracer or HistoryFlow. Neither does it offer means to visualize inter-article relationships such as WikiExplorer or WikiNavMap. The aim is to outline the structure of the wiki corpus as the first step towards refactoring. This makes us depart from the mere visualization of wikis to the challenges of wiki management.

Wiki management includes a heterogeneous set of duties: wiki initialization, wiki refactoring, spotting typos, etc. No matter the task, a common effort is to attempt to automatize/assist as much as possible to keep loyal to the open and accessible wiki tenants. Some routine tasks such as mass edits or the checking of copyright violations are amenable to automatization. In this case, bots are a common mechanism. However, other tasks are more dubious, and automatic correction might be inappropriate since the back of the community might be needed. When user intervention is needed, DSLs might offer a compromise between accessibility and learnability. That is, users are empowered but the learning curve is still affordable. We were exploring this alternative for wiki initialization in (Díaz and Puente 2011). The aim was to depict a blueprint of the wiki as a mind map, and next, generate the wiki’s initial installation so that users do not have to cope with all the technicalities. For wiki refactoring, the problem is the other way around. WikiWhirl first extracts the wiki structure from an existing wiki; next, depicts the mind map counterpart, which can then be modified by the users; finally, the resulting mind map is saved back to the MediaWiki database. Only this last step keeps some resemblance

\footnote{See \url{http://en.wikipedia.org/wiki/Wikipedia:Categories,_lists,_and_navigation_templates} for a comparison (accessed May 2013).}

with our previous work on wiki initialization.

Back to refactoring, Rosenfeld et al. (Rosenfeld et al. 2010) propose a strategy for semantic wiki evolution based on software refactoring. They identify “bad smells” and the refactoring pattern counterparts. They introduce six semantic refactoring operations (e.g., move annotation: change the subject of an annotation) and four bad smells (e.g., a too-categorized concept, i.e., a concept that belongs to many categories). The differences with WikiWhirl are two-fold. First, the approach to sustain refactoring. Rosenfeld et al. use a template-based description whereas WikiWhirl bets for a graphical DSL. Second, the focus: semantic resources vs. wiki structure. We do not consider semantic resources (e.g., annotations along an ontology). Although this vision is fuelled by the proponents of the semantic wiki, its interest lines up with the linking open-data community project (Bizer et al. 2009). Here, the goal is to extend the Web with a data commons which is geared towards the automatic consumption of information by computers. This approach makes sense for open, encyclopaedia-like wikis like Wikipedia where the effort to come up with an ontology and to annotate articles along this ontology might pay off to facilitate search and consumption by third-party programs. However, this payoff is not so clear when wikis remain within the boundaries of an organization. So, WikiWhirl focuses on the plain structure of the wiki as captured by the wiki categories.

This work also relates to database refactoring giving support to schema evolution in databases (either for refactoring purposes (Ambler and Sadalage 2006) or simply, for schema enhancement (Papastefanatos et al. 2006)). Here, the challenge is to assist database administrators in assessing the predictability and the logical independence of the evolution process. That is, preserving the data and the applications developed on top of the evolving schema. A state-of-the-art tool is the PRISM workbench (Curino et al. 2008). A PRISM session is built upon the so-called Schema Modification Operators (SMOs), an operational language, which naturally captures the atomic operations to be used to evolve an existing schema. SMOs play a similar role to WikiWhirl operations, i.e., a platform-independent language in which to state the evolution needs. Next, the system automatically generates the corresponding SQL scripts that adapt the schema and migrate the data accordingly. Unlike databases, wikis are not thought for program but user consumption. Hence, applications’ logical independence is substituted by authorship and readership independence as means to keep refactoring transparent from the consumers. Therefore, WikiWhirl-generated SQL scripts preserve those properties.

10. Conclusions

This paper’s main contributions include (i) defining the semantics of common refactoring operations based on Wikipedia best practices, (ii) advocating for the use of mind maps as a visualization of wikis, and (iii) providing a proof of concept through WikiWhirl, a DSL for wiki refactoring built on top of FreeMind. From this perspective, WikiWhirl offers a refactoring-minded alternative to traditional editing-minded front-ends as available in current wiki engines.

We believe this research is important to increase the success of wikis in a corporate setting by addressing the main technological enablers as identified in (Lykourentzou et al. 2011a), namely: simplicity, user-friendliness and structural support. First, WikiWhirl makes refactoring simpler. Preliminary evaluations suggest that users find it intuitive and less demanding to conduct wiki refactoring in terms of node rearrangement in a mind map rather than through traditional MediaWiki interfaces. Learning wise, refactoring
good practices are engineered as part of WikiWhirl which can act as a tutor during the refactoring process. Second, user-friendliness is achieved in terms of a graphical display. Third, WikiWhirl has structural support as its raison d'être by empowering users to come up with better structured wikis by themselves. Wikis’ success partly rests on affordable editing. Likewise, WikiWhirl paves the way towards affordable refactoring. In this way, any user can potentially become a “wiki gardener” with the resulting effect on the engagement and satisfaction of the users with their wikis.

Acknowledgments We thank Laura Vozmediano for her help with the evaluation of WikiWhirl. This work is co-supported by the Spanish Ministry of Education, and the European Social Fund under contract TIN2011-23839 (Scriptongue). Puente has a doctoral grant from the Spanish Ministry of Science & Education.

Appendix A. MySQL Script for the Merge Operation

Figure B1 shows the script generated by WikiWhirl to perform a merge operation for the articles Postgraduate_Courses and Short_Courses:

- Line 2 starts the transaction.
- Line 3 inserts the corresponding metadata (e.g., namespace, title, timestamp, foreign key to revision table, etc.) in table page for the new created article Postgraduate_Courses_Short_Courses.
- Line 4 gets the id for the new page.
- Line 5 stores in the recentchanges table the necessary data to generate the “Recent changes” special page.
- Line 6 creates a new record in the revision table to indicate the new edit with a comment, similar to the recentchanges table.
- Line 7 inserts an edit summary in the text table, which is the initial content of the new article.
- Line 8 updates the user table to account for one more edit of the user performing the refactoring operation.
- Lines 9-11 get the three ids (those that correspond to the articles involved in the merge operation) to perform the following inserts.
- Lines 12-20 create the necessary records to indicate the merge operation in both source articles. In addition, redirects to the new merged article are created. Line 14 creates the content of the new article by concatenating the content of the source articles.
- Lines 21-22 modify the current category relationships to point to the new merged article.
- Lines 23-29 create a talk page for the new merged article.
- Line 30 commits the changes.

Appendix B. MediaWiki Background Questionnaire

We are performing a study to evaluate use of WikiWhirl, a tool to support wiki initialization and restructuring of wikis, which has been developed by the Onekin group of the University of the Basque Country (UPV/EHU). In this first questionnaire we want to evaluate your previous MediaWiki knowledge in order to create the experimental
groups\(^1\). All data will be stored anonymously. Thank you for your collaboration and the honesty of your answers.

(1) Write the first three letters of your mother’s name followed by the last four figures of your national ID number. Keeping your data anonymous, this code will be used to create homogeneous experimental groups. On the day of the experiment this code will be used again (e.g., Mary, ID 12345678F → Code MAR5678).

Basic Concepts

Indicate if the statements are true or false.

(1) Only the original author can modify a wiki
(2) HTML knowledge is required to create or edit articles of a wiki

\(^1\)As all wiki participants collaborate in are built using MediaWiki and in order to avoid confusion, in the rest of the questionnaire questions refer to wikis in general.
Figure B1. MySQL script automatically generated by WikiWhirl for the merge operation.

Wiki Knowledge

Indicate if the statements are true or false.

1. I know how to edit an article in a wiki
2. I know how to create an article in a wiki
3. I know how to delete an article in a wiki
4. I know how to create a category in a wiki
5. To rename a category, it is enough to click the corresponding button.
(6) To change an article’s category (categorize), it is enough to edit it and to change categoryName in [[Category:categoryName]]

**Wiki Use**

We want to know the frequency with which you perform the following tasks.

1. I add content to existing articles
   a) Never
   b) Between 1 and 5 times a WEEK
   c) Between 6 and 10 times a WEEK
   d) More than 10 times a WEEK

2. I create new articles
   a) Never
   b) Between 1 and 5 times a YEAR
   c) Between 6 and 10 times a YEAR
   d) More than 10 times a YEAR

3. I create new categories
   a) Never
   b) Between 1 and 5 times a YEAR
   c) Between 6 and 10 times a YEAR
   d) More than 10 times a YEAR

4. I reorganize existing article and category groups (e.g., assign articles to new categories, change article names, etc.)
   a) Never
   b) Between 1 and 5 times a YEAR
   c) Between 6 and 10 times a YEAR
   d) More than 10 times a YEAR

**Others**

1. Describe your relationship with Wikipedia
   a) I consult it
   b) I consult it and I have an account
   c) I am an active user
   d) None of the above

**Appendix C. Global Understandability Questionnaire**

In this questionnaire we want to evaluate how easy it is to understand the structure and content of the wiki. All data will be stored anonymously. Thank you for your collaboration.

1. Write the first three letters of your mother’s name followed by the last four figures of your national ID number. Keeping your data anonymous, this code will only be used to link the answers of all questionnaires (e.g., Mary, ID 12345678F -> Code MAR5678).
WikiVet Questions

(1) Could you name two parent categories of Blood_Changes?
   a) A category can only have one parent category, in this case Clinical_Pathology.
   b) WikiBlood and Clinical_Pathology.
   c) General Pathology and WikiEpi.

(2) Mycoplasmas are characterized as Infectious_Agents?
   a) Yes.
   b) No, they are characterized as a pathology (WikiPath).
   c) No, they are characterized as a virus (Viruses).

(3) Which statement indicates MORE PRECISELY the relation between articles Anaplasmosis and Coagulation_Tests?
   a) Both are characterized as general pathologies.
   b) Both are characterized as pathologies (WikiPath).
   c) Both are characterized as degenerative pathologies (Degenerations).

(4) Fungi are characterized as...
   a) Mycoses
   b) Infectious_Agents
   c) Bacteria

(5) How many articles are characterized as epidemiology (WikiEpi)?
   a) 1
   b) 3
   c) 4

(6) In total, how many articles and categories are characterized as pathologies (WikiPath)?
   a) 2
   b) 9
   c) 10

(7) How many categories are defined in the WikiVet wiki?
   a) 5
   b) 24
   c) 25

(8) With the data available in the wiki, would it be correct to characterize the Anaplasmosis as a Clinical_Pathology?
   a) No, because it has nothing to do with pathologies.
   b) No, because even if it characterized as a pathology, it is characterized as a General_Pathology.
   c) Yes, because it is related to Clinical_Pathology.

(9) With the data available in the wiki, what do you think the Short_Courses article is about?
   a) About short courses in epidemiology (WikiEpi).
   b) About short courses in education.
   c) About short courses in veterinary (WikiVet).

(10) With the data available in the wiki, what have cells, la anaemia and haematology changes in common?
    a) They are clinical pathologies.
b) They are all related to blood.
c) They characterize immunology and pressure.

(11) With the data available in the wiki, do you believe that an haematology change can indicate a clinical pathology?
a) No, an haematology change does not indicate a clinical pathology.
b) Yes, both concepts are related in the wiki.
c) No, both concepts are not related in the wiki.

(12) With the data available in the wiki, do you believe that anaplasmosis is caused by a parasite?
a) No, it is caused by a bacteria.
b) Yes, both concepts are related in the wiki.
c) No, both concepts are not related in the wiki.

(13) With the data available in the wiki, do you believe that parasites can be...
   a) Fungy and bacteria.
   b) Mycoses y mycoplasmas.
   c) None of the above.

(14) With the data available in the wiki, do you believe that paracetamol is prescribed in...
   a) Degenerative pathologies.
   b) Clinical pathologies.
   c) No answer can be deduced from the wiki.

Appendix D. Final Questionnaire

We are performing a study to evaluate use of WikiWhirl, a tool to support wiki initialization and restructuration of wikis, which has been developed by the Onekin group of the University of the Basque Country (UPV/EHU). This last questionnaire has two short sections. All data will be stored anonymously. Thank you for your collaboration and the honesty of your answers.

General

(1) Write the first three letters of your mother’s name followed by the last four figures of your national ID number. Keeping your data anonymous, this code will only be used to link the answers of all questionnaires (e.g., Mary, ID 12345678F -> Code MAR5678).

(2) Gender
   a) Female
   b) Male

(3) Age

(4) Rate your programming background (1-none, 5-expert)
(5) Rate your Web engineering background (1-none, 5-expert)
(6) Rate your Web 2.0 background (1-none, 5-expert)
(7) Rate your collaborative system background (1-none, 5-expert)
(8) Rate your mind mapping background (1-none, 5-expert)
Execution

In this section the performed tasks and the invested time are collected. For each task, we ask you to write whether you have finished its parts and how long it took. Remember that it is normal not to have finished some of the tasks in the allocated time.

FIRST TASK: Understand Content

(1) I have finished the questionnaire on how understandable the content and structure of the wiki is.
(2) If you have finished the task, write down the time it took (minutes).

SECOND TASK: Refactor the Wiki Structure

(1) I have finished the CREATE operations in the handout.
(2) If you have finished the CREATE operations, write down the time it took (minutes).
(3) I have finished the CATEGORIZE operations in the handout.
(4) If you have finished the CATEGORIZE operations, write down the time it took (minutes).
(5) I have finished the UNCATEGORIZE operations in the handout.
(6) If you have finished the UNCATEGORIZE operations, write down the time it took (minutes).
(7) I have finished the RENAME operations in the handout.
(8) If you have finished the RENAME operations, write down the time it took (minutes).
(9) I have finished the DROP operations in the handout.
(10) If you have finished the DROP operations, write down the time it took (minutes).

THIRD TASK: Refactor the Wiki Content

(1) I have finished the SPLIT operations in the handout.
(2) If you have finished the SPLIT operations, write down the time it took (minutes).
(3) I have finished the MERGE operations in the handout.
(4) If you have finished the MERGE operations, write down the time it took (minutes).
(5) I have finished the MOVE operations in the handout.
(6) If you have finished the MOVE operations, write down the time it took (minutes).

References


REFERENCES


Müller, J., et al., 2013. FreeMind. [online] [accessed May 2013].

REFERENCES


