

ACTIVE

Deliverable 1.3.2

Collaborative Articulation of Enterprise Knowledge (Demonstrator)

Editor:	Markus Krötzsch, Karlsruhe Institute of Technology (KIT)
Deliverable nature:	Prototype (P)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	Feb 28 2010
Actual delivery date:	Mar 5 2010
Suggested readers:	Knowledge Workspace Developers Case Study Partners
Version:	1.0
Total number of pages:	18
Keywords:	knowledge articulation, semantic content management system, user interface, Semantic MediaWiki, ontology editor, knowledge repair, LiveNetLife

Abstract

ACTIVE Task 1.3 investigates tools for the collaborative articulation of enterprise knowledge, in particular with respect to Web 2.0 paradigms of online cooperation. Semantic MediaWiki has been identified as a useful base application for this work, since it combines the successful wiki paradigm for collaboration with means of managing structured data that abounds in enterprise applications. A number of challenges have been identified in Deliverable 1.3.1 for addressing such application scenarios properly. This document reviews the development progress toward these challenges, both with respect to Semantic MediaWiki, and with respect to new extension modules that have been created to augment its features. In particular, we introduce a new Ontology Editor extension and the LiveNetLife communication tool.

Disclaimer

This document contains material, which is the copyright of certain ACTIVE consortium parties, and may not be reproduced or copied without permission.

All ACTIVE consortium parties have agreed to full publication of this document.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the ACTIVE consortium as a whole, nor a certain party of the ACTIVE consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

Impressum

[Full project title] ACTIVE – Knowledge-Powered Enterprise

[Short project title] ACTIVE

[Number and title of work-package] WP1: Enterprise Knowledge Structures

[Document title] Collaborative articulation of enterprise knowledge – Demonstrator

[Editor: Name, company] Markus Krötzsch, Karlsruhe Institute of Technology (KIT)

[Work-package leader: Name, company] Markus Krötzsch, Karlsruhe Institute of Technology (KIT)

[Estimation of PM spent on the Deliverable] 9

Copyright notice

©2008-2012 Participants in project ACTIVE

Executive summary

This document is the first formal deliverable for ACTIVE Task 1.3: Collaborative articulation of enterprise knowledge, and serves as a description of the implementation work undertaken so far. The overall goal of Task 1.3 is to investigate techniques for flexible and collaborative articulation of enterprise knowledge, in particular by considering paradigms from the “Web 2.0” community such as wikis and tagging. The task critically assesses these paradigms and leverages them to allow for articulating knowledge with greater expressivity where needed.

The platform Semantic MediaWiki (SMW) has been selected as a basis for developing this technology, and a number of challenges have been named and discussed in Deliverable 1.3.1. The present deliverable documents the progress that has been made toward achieving these goals. On the one hand, this includes a number of improvements in SMW and closely related extensions. On the other hand, some additional software modules have been created to augment this core functionality with further features. Namely, we describe a novel *Ontology Editor* extension that provides richer user interfaces and more specific support for common knowledge modelling tasks, and we outline *LiveNetLife* that enables real-time communication between knowledge workers when accessing online resources.

All software that is described herein has either been released as public software, or can be viewed and tested on specified demonstrator sites.

List of authors

Company	Author
University of Innsbruck	Tobias Bürger
Karlsruhe Institute of Technology (KIT)	Markus Krötzsch
University of Innsbruck	Michael Luger
Karlsruhe Institute of Technology (KIT)	Denny Vrandečić
University of Innsbruck	Stephan Wölger

Contents

Executive summary	3
List of authors	4
Abbreviations	7
1 Introduction	8
2 SMW Goals and Development Progress	9
2.1 Ease of Use	9
2.2 Desktop Knowledge Exchange	10
2.3 Higher-Level Knowledge Structures	11
2.4 Access Control	12
3 SMW Ontology Editor	12
3.1 Rich Interface	13
3.2 Knowledge Repair	14
3.3 Vocabulary Management	15
3.4 Knowledge Import and Export	16
4 LiveNetLife	16
5 Summary	17

List of Figures

1	Overview and navigation within ontology elements	13
2	Special page for category statistics	15

Abbreviations

GUI	Graphical User Interface
OE	Ontology Editor extension
OWL	Web Ontology Language
RDF	Resource Description Framework
SMW	Semantic MediaWiki
UI	User Interface

1 Introduction

The purpose of Task 1.3 is to provide software prototypes that enable the collaborative articulation of enterprise knowledge. Such articulation is mainly relevant for two distinct usage scenarios within ACTIVE. The first and possibly most obvious scenario is the use by knowledge workers as part of their daily work at the sites of the case study partners. This usage requires integration with desktop applications and other end-user components of the ACTIVE workspace to be feasible in practice. A second and slightly different application scenario is the articulation of knowledge that is required for ACTIVE components to be deployed in their concrete application domains. In spite of the heavy use of mining and pattern detection technologies in ACTIVE, it is expected that ACTIVE applications in each case study do also require a certain amount of higher-level knowledge to achieve maximal impact.

This deliverable focusses on software developments that have been undertaken for Task 1.3 in the second year of the ACTIVE project. Many of these efforts have led to updates and extensions of the development goals as outlined in Deliverable 1.3.1 [Dengler et al., 2009], especially with respect to the central knowledge management platform *SMW* that has been introduced therein. Moreover, some of the development targets have forked out into individual development projects that have not been discussed in detail in earlier deliverables. In particular, various knowledge articulation tasks are now supported by a dedicated *Ontology Editor* extension, and the online collaboration of knowledge workers is supported by a new real-time communication software *LiveNetLife*.

Recall from Deliverable 1.3.1 the main usage scenarios targeted by the software components produced in this task:

Specifying and managing process descriptions. Pattern detection can identify process-like structures in user behaviours (bottom-up recognition). In addition, however, companies typically use descriptions of business processes and standardised work-flows (top-down specification). It is clearly important to make these specifications accessible to ACTIVE components, e.g. since they provide a terminology for process steps that a bottom-up approach cannot discover. A concrete application is found in the Cadence case study, where formal process descriptions are available. Another situation is given in the Accenture case study where process descriptions are plenty but mostly informal.

Enterprise Knowledge Base. ACTIVE applications typically require a basic amount of background knowledge about concepts used in enterprise environments. A basic example are hierarchical relationships between core concepts that users employ for structuring or searching information. Suitable systems are needed to specify and manage this background knowledge. A typical application of such general enterprise knowledge in ACTIVE is the factbook as considered in the BT case study (see D 9.2.1).

Classifying Observed Processes. Even if formal process models have been specified, it is still a challenging task to relate those high-level descriptions to the observed process patterns of actual users. This task of mapping between low-level structures and high-level descriptions also requires support by human users who specify otherwise implicit knowledge. In this sense, an enterprise knowledge articulation system would be a component of a semantic mapping tool.

Shared Knowledge Resources. Instead of supporting the ACTIVE infrastructure in general, a knowledge articulation platform may also be used directly to collect and manage information in a group of knowledge workers.

For this deliverable, we assume the reader to be familiar with Semantic MediaWiki (SMW) as introduced in Deliverable 1.3.1 [Dengler et al., 2009] (see also [Krötzsch et al., 2007, Vrandečić and Krötzsch, 2009]), and we focus only on the changes since project month 12. The new components are described in greater detail.

In parallel to the development of new software, we continued the current strategy of public release and deployment so as to engage a maximal amount of users and contributors from outside the project. Besides a number of new public uses of ACTIVE technology such as the recent announcement of Pfizer's use of SMW in patent management [Walsh and Berridge, 2009], this has also been realised in an international gathering of SMW users in November 2009, attracting around 50 participants.¹ Further events of this kind are planned

¹http://smwforum.ontoprise.com/smwforum/index.php/SMW_Camp_09

for 2010, both in the US and in the EU. Moreover, LiveNetLife has become the basic technology for a new startup company.² This deliverable focusses on technical advances instead of practical exploitation, but we still consider dissemination and community building an important part even in the development process. Most importantly, it allows ACTIVE to build synergies with related users communities, which can only be accomplished if developers coordinate and align their efforts with user contributions.

2 SMW Goals and Development Progress

Deliverable 1.3.1 introduced a number of core challenges for the development of SMW – ease of use, desktop knowledge exchange, support for higher-level knowledge structures, community-driven conceptual modelling, and access control –, and formulated various development goals for each of them. In this section, we revisit the status of these goals, and augment them whenever new subtargets have emerged during the deployment of SMW that has happened so far. For more detailed descriptions of the rationale behind each feature, please refer to the corresponding section in Deliverable 1.3.1 [Dengler et al., 2009]. Goals for which no changes are to be reported since Deliverable 1.3.1 are omitted here; this is specifically the case for all goals that had been completed in SMW 1.4.2 already.

The most current version of SMW at the time of this writing is SMW 1.5.0. The current status of the SMW implementation can be accessed online³ at any time together with its full code documentation⁴. On-line information on deploying SMW can be found in [SMW Project, 2009a]. Some advanced functionalities related to SMW have been moved into extension packages that can be installed individually on wikis that run SMW. Major extension modules that are mentioned for individual features below are *Semantic Forms* [Koren, 2009] (current version 1.8.8), *Semantic Result Formats* [SMW Project, 2009b] (current version 1.4.5), and *Halo* [ontoprise GmbH, 2009] (current version 1.4.6). While only Semantic Result Formats is maintained primarily by ACTIVE project members, there is active coordination even with developers of other third party extensions to ensure the continued availability of those add-ons for ACTIVE.

2.1 Ease of Use

There are two main areas for this topic: ease of knowledge articulation/creation, and ease of knowledge access/retrieval. The first point has to address the current editing paradigm of wikis that requires users to formulate knowledge in wiki-syntax. While this wiki-syntax is reasonably easy to learn, it still creates a barrier for knowledge workers in enterprises who are not familiar with MediaWiki. There are various development goals to overcome this issue:

Form-based editing. The form-based input interface as provided by the Semantic Forms extension has seen a number of improvements in 2009, both in terms of stability and in terms of functionality. A detailed version history can be found online.⁵ Within ACTIVE, Semantic Forms are by now employed in all case studies. [Status: available in Semantic Forms extension, improved]

WYSIWYG editing. A complete WYSIWYG editor for SMW is not available yet, but various new approaches have been explored for enabling such editing in selected components. In particular, the Ontology Editor as described in Section 3 provides inline editing. These features have been completed only recently and are not employed in case studies yet. [Status: partly available in Ontology Editor and WYSIWYG extensions]

Annotation interfaces. The annotation features as provided by the Halo extension have not changed significantly. Tests with the Halo extensions have been conducted within the BT case study, but overall Semantic Forms appear to be more adequate for the structured data that occurs in ACTIVE applications. [Status: available in Halo extension]

²<http://livenetlife.com/>

³<http://svn.wikimedia.org/viewvc/mediawiki/trunk/extensions/SemanticMediaWiki/>

⁴<http://semantic-mediawiki.org/doc/>

⁵http://www.mediawiki.org/wiki/Extension:Semantic_Forms/Version_history

Data imports. A number of data import approaches has been explored in the last year since importing data is not a single problem but needs different solutions for each chosen source format. A prototype for importing tabular data (spreadsheet documents) is considered for usage in ACTIVE. The import of ontological vocabulary data is supported by the Ontology Editor. A new Web API for writing data to SMW has been developed for facilitating import tasks (see below). The use of office document imports is currently investigated in case studies and experiences are fed back to improve the available prototypes. [Status: available in prototypes; ontology import partly available in Ontology Editor]

The second relevant aspect is the ease of accessing knowledge in SMW. The aforementioned browsing and querying features leave room for improvements in terms of usability. Further related projects are given in the section on knowledge exchange below.

Faceted browsing. The Exhibit integration of SMW has further been improved in the previous year, and the code is now considered as a stable part of the Semantic Result Formats extension. Exhibit visualisation can thus seamlessly be used in all case studies, and is currently employed, e.g., at Accenture for displaying query results. Exhibit is also known to be used in enterprise wikis in companies outside the project, e.g. at Mayo Clinics (personal communication). [Status: available in Semantic Result Formats extension 1.4.3]

Process visualisation. The extension support for process visualisation has been further improved and extended to serve as a tool for handling knowledge processes in ACTIVE. Details about this activity can thus be found in Deliverable 3.2.2. This functionality plays an important role in ACTIVE, e.g. for visualising product methodologies in Cadence (see case study deliverable for details). [Status: available in extension developed in Task 3.2]

2.2 Desktop Knowledge Exchange

Exchanging knowledge with desktop applications is an important feature of any knowledge articulation platform that is to be used in enterprise contexts. This may be achieved directly, through application-specific data formats or interfaces, or indirectly, through the use mediator applications that can be accessed through standard data access protocols. The existing data export formats of SMW (iCalendar, vCard, CSV, RSS) continue to be available. The following additional features have been added recently:

JSON export. SMW now provides an export in JSON format that is stable since SMW 1.5.0. Based on this technology, it is possible to realise Web-based applications that re-use SMW content, e.g. by means of the Exhibit toolkit⁶ or the DataPress plugin to Wordpress. Integrating wikis and blogs has been an explicit goal of Task 1.3, but blogging does not occur in the context of the current case studies.⁷ [Status: available in SMW 1.5.0]

Desktop-based writing access. Besides the aforementioned prototype for spreadsheet import, new software that allows users to read and write annotations based on Microsoft Word and Outlook has been developed. These tools are currently at a prototype stage, and especially the plugin *WikiTags*⁸ is assessed and refined for deployment in ACTIVE. See Deliverable 3.1.2 for more information about WikiTags. Current usage of this feature in ACTIVE can be found in the Accepture case study, as described in Deliverable 10.1.2. [Status: prototype available]

One difficulty of developing extensions such as WikiTags is the fact that there is no easy way for writing semantic data to SMW in an automated fashion. Whereas SMW provides several means to export the knowledge within the wiki in a range of different machine-readable formats, there has previously been no straight-forward way to easily change the semantic annotations in the wiki. The MediaWiki API is a text-based, restful editing webservice API, i.e. it allows to access the wiki source of every page (that is accessible to the user based on their user rights) and it allows to write back changes to the wiki source (again, possibly restricted based on the user rights).

⁶see <http://beer.geekworks.de/> for an example

⁷<http://code.google.com/p/datapress/>

⁸http://wiking.vulcan.com/dev/index.php/Wiki_Office_add-in_release

We have extended the existing MediaWiki API with two new commands, *smwwrite* and *smwwritable*. They provide a web-based restful interface to change annotations directly within the wiki without having to bother with their exact representation within the text. This allows bots easily to update or add pieces of information to an SMW instance.

Due to the inherent complexity of the task, we call this API a *Best Effort* API. The task of updating arbitrary annotations within the wikis is extremely difficult due to the many features of MediaWiki that interact in creating the preprocessed wikitext that is eventually used by SMW to extract semantic data. For example, parts of the text can stem from templates which in turn may use extension functions to compute their output, so that a targeted modification of these parts of the wikitext may be impossible without unwanted ramifications throughout the wiki. Therefore we provide a stable interface for updating the data, but the wiki may answer that the given task is out of scope for the current implementation. We hope that we can extend the ability of the API to deal with an increasing share of situations without having to change the interface, thus being able to add more and more complex processing tasks in the background whereas the tools using the API can remain without change (but will simply receive less failure notices).

The functions provided by the API are as follows:

smwwritable has one parameter – *title* – that states the name of the page we are interested in. The API then examines the page and returns a list of all the property-value pairs that can be removed or changed through the API. Property values of a page that do not feature in this list can certainly not be modified through the API, but the presence of a property-value pair in the result does not guarantee that an attempt to change the property's value will be successful. Indeed, a page may include property-value pairs that appear to be editable, but are actually stated multiple times with one of these occurrences being immutable.

smwwrite has the following parameters:

- *title*: the page that is being changed
- *token*: an edit token (to prevent conflict edits)
- *add*: the list of annotations to add
- *remove*: the list of annotations to remove
- *flags*: a set of flags to guide the edit update (see below)
- *summary*: a rationale for the edit

The following flags are available:

- *atomic change*: perform the change only if the whole operation is expected to be successful
- *ignore constant*: in order for the whole operation to be considered successful, do not consider constant parts of the source
- *change text*: perform the changes within the text if possible, instead of using the *set* command

Due to its dependency on the (widely extended) *Page Object Model* extension (POM), and due to the significant amount of code accompanying this functionality it was decided that the functionality will not be part of SMW core but rather provided as an extension of its own. [Status: available in SMWWriter extension]

2.3 Higher-Level Knowledge Structures

Enterprise knowledge can be strongly structured not only on a data level, but also by means of general taxonomies or schematic rules and axioms. So far, SMW could represent complex terminological or schematic data only to a very limited amount, mainly restricted to supporting hierarchical organisation of categories (classes) and properties. Further types of ontological data should be supported to address more features of the light-weight modelling languages developed in Task 1.1:

Concept subsumption. The use of concept subsumption in SMW has been analysed to decide whether and how such functionality can be supported in the future. Answering queries in the presence of conceptual information requires additional inferencing operations that were found to be too costly for query-time

processing. The design that emerged from these considerations thus incorporates an independent asynchronous reasoning process that pre-computes and caches information that SMW requires at query time. As a first step toward realising this design, we have therefore developed an independent software component *Orel*⁹ that implements database-driven reasoning for the light-weight ontology languages OWL EL and OWL RL [Motik et al., 2009a]. It is planned to assess the integration of SMW and Orel for allowing an advanced level of ontological or rule-based knowledge to be used in SMW. [Status: under development]

The current progress in this complex task suggests that it could be feasible to support significantly more expressive modelling in SMW in the future, thus allowing to model more complex enterprise “knowledge” (instead of just enterprise “data”). This is supported by the availability of the light-weight modelling languages in OWL 2 which have been supported by the activities in Task 1.1, but the overall integration of all components remains a challenging endeavour.

A major motivation for such modelling approaches is to reduce overall redundancy by allowing the wiki to derive facts that necessarily follow from the content of the wiki, without users having to manually enter and update these facts. A simple example that arose in the context of a tagging model used by Accenture is the statement that every resource tagged with “SAP R3” should also be tagged with “SAP.” Now given a property “has tag” in SMW, this relationship cannot currently be articulated in SMW, and one has to add the according tags manually or embed the business logic in template calls.¹⁰ Clearly, it would be preferable to have the ability to articulate the underlying knowledge in a more explicit way than by programming a template which, in addition, would only work if all tags are set via this template. Enabling such modelling on a more general basis is the general goal of this effort, but a number of obstacles must be tackled before one can introduce such technologies into a wiki.

2.4 Access Control

High security access control cannot be realised in MediaWiki, and must be implemented on server level, e.g. by server-side access restrictions. For some medium amount of security, it would still be useful to have access control mechanisms within the wiki:

Write access restrictions. We have found that the namespace-based write access restrictions that new versions of MediaWiki provide are sufficient for many applications. A more complex option is provided as part of the extension described below. [Status: available in MediaWiki and extensions]

Read access restrictions. Restricting read access in a semantic wiki that is focussed on many forms of data exchange remains a challenging task. Yet, the new *Halo Access Control List* extension is now available for fine-grained access control in SMW, including read access control for semantic query results.¹¹ [Status: available in Halo Access Control List extension]

With the above improvements, we consider the question of access control to be sufficiently thoroughly covered in SMW. Access control is frequently requested for enterprise wiki solutions, and thus plays an important role for the general goals of ACTIVE. In the current case studies, however, access control is not vital and only basic control features are used.

3 SMW Ontology Editor

A new *Ontology Editor* (OE) extension to SMW has been developed at the University of Innsbruck. The SMW Ontology Editor is a platform for collaborative management of light-weight ontologies. Its rich user interface allows inexperienced users to easily create and maintain ontology elements. The system is built on top the Semantic MediaWiki platform. It provides a metamodel that distinguishes between vocabularies, categories,

⁹<http://code.google.com/p/orel/>

¹⁰Here we use tags as a specific example for cases where more powerful knowledge modelling in SMW would be useful; of course “tagging” occurs in more than one situation and may require different approaches elsewhere, such as the one described for the Ontology Editor below.

¹¹<http://smwforum.ontoprise.com/smwforum/index.php/Help:HaloAccessControlList>

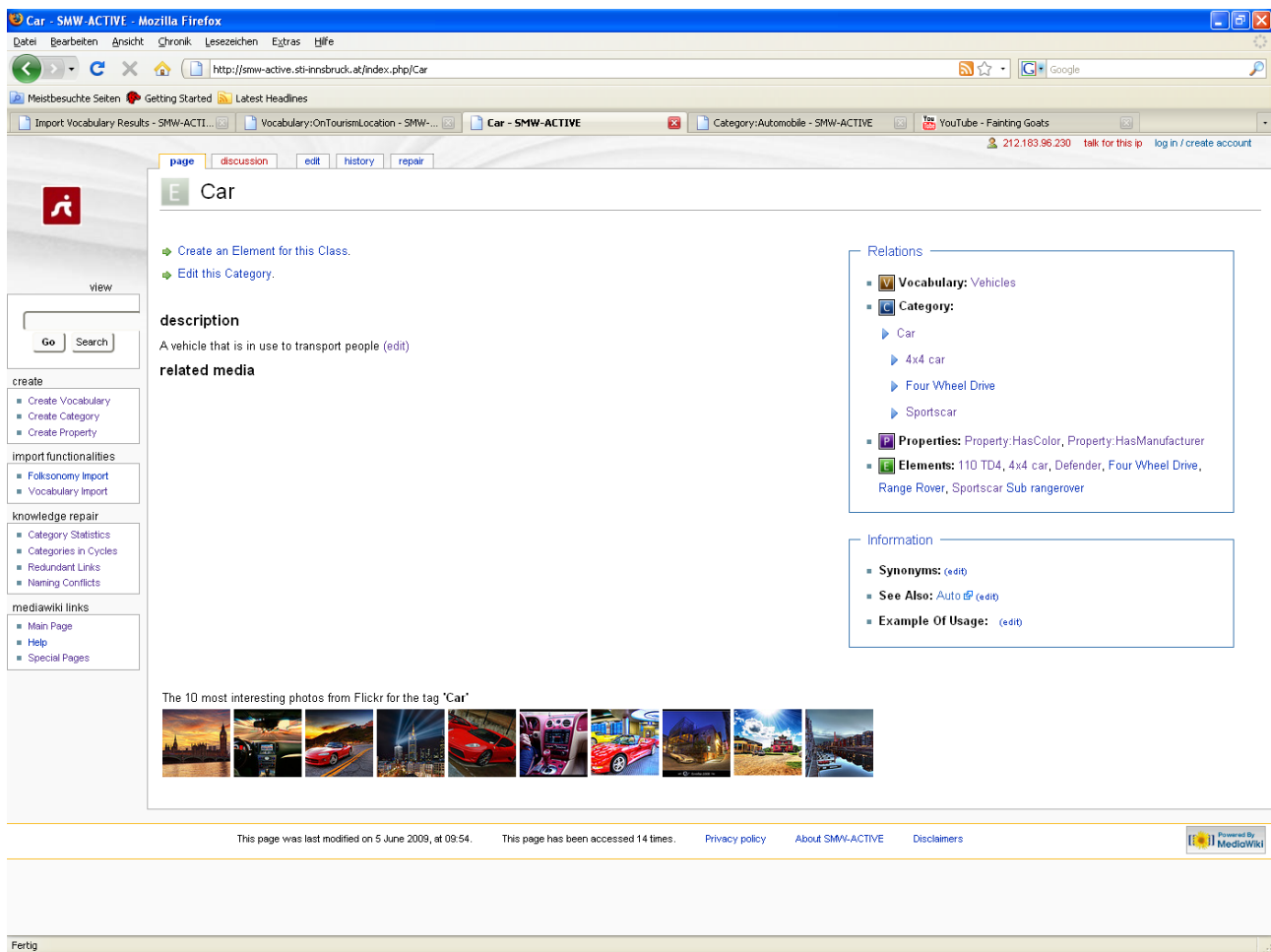


Figure 1: Overview and navigation within ontology elements

properties and elements. For the maintenance of evolving vocabularies, *knowledge repair* functionalities are provided that assist users with the discovery and mitigation of redundancies and inconsistencies within the knowledge base. The *Folksonomy Import* feature allows the import of external folksonomy data. Furthermore, the system supports the import and export of vocabularies and hierarchy information from OWL ontologies.

A public demonstrator of this software is found at <http://smw-active.sti-innsbruck.at/>. This site also provides the link to a screencast that shows some of the features that cannot be tested easily, such as folksonomy import.¹² The Ontology Editor extension has been released only recently, and is not used in ACTIVE case studies at the moment.

3.1 Rich Interface

The Ontology Editor provides a number of interface extensions to facilitate the editing of and access to information stored in SMW.

Form-based Management of Ontology Elements. Users are able to create and edit ontology elements using form-based interfaces. For example, during the process of creating a new category, one can supply information such as the desired description, synonyms, or related Web pages. The form allows the association of the affiliated vocabulary and parent-categories as well as new and existing properties through dropdown menus. In a similar fashion, an existing category can be revised using the same form. The form-functionality of the tool has been built on top of the Semantic Forms extension.

This feature makes it possible for non-expert users to create and maintain the knowledge base without having to familiarize with Wiki syntax or the syntax of SMW.

¹²<http://members.deri.at/~simonh/semanticMediaWiki/screencast.avi>

Tag Cloud. Individual pages are accompanied by a tagcloud that displays the group of associated elements. The elements of the tagcloud are displayed with different colors and font-sizes according to their significance. Elements that exhibit a higher number of page accesses and links to other pages are displayed with a larger font.

The main page of the wiki displays a tagcloud that takes the entire knowledge base into account.

Automatic Multimedia Annotations. When creating a category, the system invokes the Flickr API¹³ and searches for related images from Flickr.com by supplying the title of the category. The resulting images are associated and displayed as part of the category page.

Inline Editing. A category is described by meta-properties such as an informal description, a list of synonyms, related web pages etc. The values of those properties can be edited directly from the category's page in the form of popup forms. This is a convenient way for the user to make minor changes and to avoid the regular, richer edit form interface.

Tree-based Management of Category Hierarchies. SMW allows categories to be organised in a hierarchy. The OE visualises this hierarchy in a tree-based overview, based on the existing CategoryTree extension.¹⁴

Moreover, the category hierarchy can be edited in a dynamic fashion by performing drag & drop in the tree-based category view, thus extending the functionality of CategoryTree to allow direct write access.

Overview and Navigation within Ontology Elements. The building blocks of the SMW Ontology Editor are vocabularies, categories, properties, and individual elements. The structural relationships of those ontological elements is shown by providing links between the individual elements in a consistent fashion. An example of this can be seen in Fig. 1.

Tooltips and Auto Completion. During the process of managing ontology elements using the form-based interfaces, the user can get more information on a form element by selecting the context-dependent help menus.

Auto Completion supports the user in finding already existing pages (vocabularies, categories, ...) and can also be modified for integrating wikis or a thesaurus.

3.2 Knowledge Repair

OE provides a number of features for detecting potential modelling errors in the knowledge that is represented in SMW, and it provides suggestions for fixing these situations. The according functionality is referred to as *knowledge repair*.

Category Statistics. The special page “Category Statistics” provides an overview of all the categories within the system and statistical information in tabular layout, as shown in Fig. 2. This includes the number of associated categories, elements and properties, or statistics about the associated category-tree's structure. This page is helpful for the user to investigate the knowledge base and spot potential errors that are often not detected otherwise.

This information can also be accessed for individual categories by clicking on the “repair” tab on the top of each category page.

Category Hierarchy Cycles. Even if categories are not considered under strict “subclass of” semantics, it is generally not desirable to model cyclic category hierarchies. Yet this can easily happen in a collaborative editing environment due to incoherent understanding of a category's meaning among users. OE provides a special page for detecting such cycles so as to allow wiki contributors to take adequate actions for resolving the issues.

¹³<http://www.flickr.com/services/api/>

¹⁴<http://www.mediawiki.org/wiki/Extension:CategoryTree>

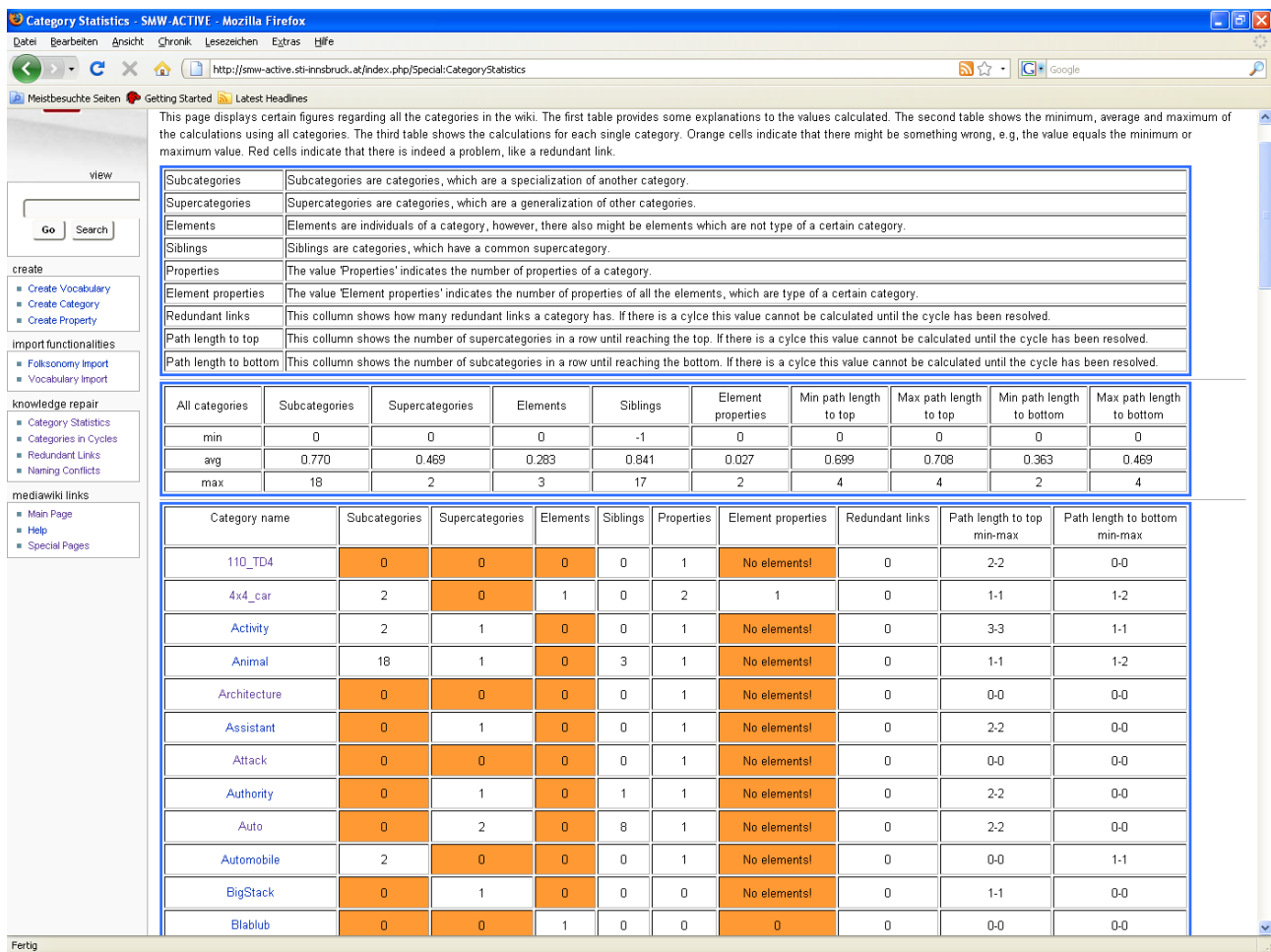


Figure 2: Special page for category statistics

Categories with Redundant Subclass Relations. Categories can be investigated for potential redundant links in the category-tree. Such a redundant link is detected if the category is a subcategory of two or more other categories which are on the same branch in the category tree. Links can be removed directly as part of this page.

Categories with Similar Property Sets. Redundancies in the set of categories can be found if categories share similar properties. If at least 50% of the associated properties match for two categories, OE considers that as candidates for being merged, and displays this suggestion to the user on the according special page.

Ontology Elements with Similar Names. Ontology elements can be checked for similar titles. This is a possible indicator for elements that should be merged. In particular, the vocabulary import features described further below may introduce derived names when encountering naming conflicts during import. This feature helps to find identify such cases for manual merging.

Editing Suggestions. The knowledge repair functionality is integrated as part of the main editor. During the process of introducing a new category into the system, as part of the form-page, it can be checked for redundancies, e.g. similar existing category-titles.

3.3 Vocabulary Management

The OE introduces *vocabularies* as an additional means of structuring knowledge in the wiki. A vocabulary is a group of various ontological elements (categories and properties) that relate to a common domain of application. For example, the vocabulary for describing knowledge processes is typically distinct from the vocabulary for

modelling organisational structures in an enterprise. OE allows this intention to be made explicit by assigning the respective schema elements to a vocabulary that is represented by a dedicated wiki page.

A special feature that is based on OE's vocabulary structure is the capability of versioning the members of some vocabulary as a whole. In contrast, MediaWiki provides versioning only on a page level, making it cumbersome and error-prone to view or restore a particular version of the wiki's knowledge model across many affected pages. With the "Versioning" special page it is possible to display and restore the history of changes of vocabularies and categories. It is possible to restore the structure of a vocabulary and the field information of categories independently. On the versioning page one can select a vocabulary or a category (by clicking on the picture beside the name in the list) to display a popup with detailed versioning information. On the left side of this popup it is possible to choose between *Vocabulary Structure Changes* and *Category Changes*. Inside of the selected box are the different version dates. Different versions can be selected and are displayed immediately using AJAX on the right side of the popup. A selected version can be restored by clicking the Restore Selected Version button.

3.4 Knowledge Import and Export

In addition to the data export facilities of SMW, OE provides a number of further features for importing and exporting structured data.

Folksonomy Import. An external folksonomy dataset can be mapped to an ontology representation and added into the system. The dataset should be configured to adhere to a given structure, and it can be uploaded through a form. The import of the tagging data into the system is divided into the following steps:

1. Determination of Levenshtein similarity metrics to find similar words.
2. Searching for the tags in Wordnet.
3. Searching for the tags in Wikipedia for the existence of page or redirect.
4. Spell checks and translations.
5. Grouping tags based on the results of steps 1–4.
6. Determination of co-occurrence and co-acting matrices.
7. Clustering of the tags.
8. Mapping to the SKOS ontology.
9. Mapping and insert according the SMW Ontology Editor metamodel.

Ontology Import. External OWL ontologies can be imported through a form-interface. The ontology elements are mapped to the Ontology Editor's metamodel. Users are notified when clashes occur, i.e. if an element with the same title already exists. The input format is OWL/XML as described in the OWL 2 standard [Motik et al., 2009b].

Ontology Export. The ontology import in OWL/XML is complemented by an according export function that complements the OWL/RDF export functions of SMW. Each vocabulary page contains an export tab that leads to an interface to export the vocabulary and its associated categories, properties and elements in the form of an ontology.

4 LiveNetLife

LiveNetLife is an application for contextualised real-time communication between the users of a web site. An interactive chat client is embedded into web pages, allowing users who currently view a page (or a set of related pages) to interact with each other. Besides the exchange of text messages, it is also possible to share the position of one's mouse pointer with other users. This allows users to discuss the content of web pages they are watching at the same time.

LiveNetLife has also been integrated with Semantic MediaWiki to include this functionality into wiki pages. LiveNetLife thus provides a simple framework for supporting knowledge workers that are active in the same context of a web-based knowledge resource, as illustrated in the following example:

Example. While Jure is preparing a deliverable for his project, he visits the internal project wiki that outlines all deliverables and tasks in the project. As he looks for information, an avatar pops up, showing that Denny, a person from another institution working on the same project, is also reading a related page on the wiki. Because of that Jure can be reasonably sure that he will not force a context switch on Denny, and so he asks him a question about the tasks they are both working on at that moment. Denny answers and asks Jure a question as well, all the while staying within the same context and proceeding with their tasks.

An important functionality that is used in this example is that user profiles of a wiki are used to personalise the activities of logged in users. If a user logs in to the wiki then he or she will no longer appear as anonymous to other users but will be associated with the respective name and image. To retrieve the latter, the LiveNetLife SMW extension queries for the value of the property *image* for the user page of the respective user. Thus users can customise the appearance of their avatar by editing their profile on the wiki. Note that this functionality is available to virtually all users of the LiveNetLife extension in ACTIVE, since the targeted scenarios require all readers and contributors to be logged in.

Technically, LiveNetLife is realised by means of a client-side JavaScript program that provides a simple user interface for sending messages and for viewing other users who view use the given web page. The communication is coordinated by a central server and brokered by the web site that integrates LiveNetLife. The infrastructure needed for this service is currently maintained by a dedicated company that has developed out of the ACTIVE consortium.¹⁵

The LiveNetLife SMW integration runs within both the BT and the Cadence case studies and is also used on the ACTIVE project wiki.

5 Summary

We have given an overview of a number of tools that have been developed or improved in Task 1.3 of the ACTIVE project within the second year. Most of the presented software is realised on the basis of the semantic content management system Semantic MediaWiki (SMW) either through direct extensions of this software or via dedicated extension modules that are available for it. A particularly comprehensive extension is the new *Ontology Editor* for SMW that improves the support for workflows in knowledge modelling. This component seamlessly integrates into the SMW platform while at the same time adding support for hitherto uncovered knowledge models such as folksonomies and vocabularies (as groups of schema elements). In addition, a number of new interactive user interfaces are provided to further facilitate editing and browsing. Independent from those predominantly server-based systems, the *LiveNetLife* application has been introduced as a light-weight solution for real-time communication among users of a web site.

Overall, the toolscape in Task 1.3 thus approaches the original goal of providing “Web 2.0” paradigms for enterprise knowledge management much more closely than it had been possible by a (semantic) wiki alone: rich UIs, asynchronous communication, and light-weight knowledge models such as folksonomies add to the overall utility and user experience. The implementation of the underlying software components is often a direct outcome of the work in Task 1.3 – but an even more significant result is the successful integration of these diverse features into a common platform that can be deployed in applications inside and outside the project.

¹⁵<http://www.livenetlife.com/>

References

- [Dengler et al., 2009] Dengler, F., Krötzsch, M., Siorpaes, K., and Vrandečić, D. (2009). Collaborative articulation of enterprise knowledge (early prototypes). Deliverable, ACTIVE. Available at http://www.active-project.eu/fileadmin/public_documents/d131-final.pdf.
- [Koren, 2009] Koren, Y. (Jan 2009). Semantic Forms online documentation. http://www.mediawiki.org/wiki/Extension:Semantic_Forms.
- [Krötzsch et al., 2007] Krötzsch, M., Vrandečić, D., Völkel, M., Haller, H., and Studer, R. (2007). Semantic Wikipedia. *Journal of Web Semantics*, 5:251–261.
- [Motik et al., 2009a] Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., and Lutz, C., editors (27 October 2009a). *OWL 2 Web Ontology Language: Profiles*. W3C Recommendation. Available at <http://www.w3.org/TR/owl2-profiles/>.
- [Motik et al., 2009b] Motik, B., Parsia, B., and Patel-Schneider, P. F., editors (27 October 2009b). *OWL 2 Web Ontology Language: XML Serialization*. W3C Recommendation. Available at <http://www.w3.org/TR/owl2-xml-serialization/>.
- [ontoprise GmbH, 2009] ontoprise GmbH (Jan 2009). Halo extension online documentation. http://smwforum.ontoprise.com/smwforum/index.php/Main_Page.
- [SMW Project, 2009a] SMW Project (Jan 2009a). Semantic MediaWiki online documentation. <http://semantic-mediawiki.org>.
- [SMW Project, 2009b] SMW Project (Jan 2009b). Semantic Result Formats online documentation. <http://semantic-mediawiki.org/wiki/SRF>.
- [Vrandečić and Krötzsch, 2009] Vrandečić, D. and Krötzsch, M. (2009). *Semantic Knowledge Management*, chapter 13: Semantic MediaWiki, pages 171–180. Springer.
- [Walsh and Berridge, 2009] Walsh, D. and Berridge, A. (2009). Pfizerpedia Patents – The who, what, when and why of patents. Presentation at International Chemical Information Conference, Vienna, Austria, 24–27 October 2009 (ICIC’09).