Integrated Project

Priority 2.4.7

Semantic based knowledge systems

The Social Semantic Desktop

Mandriva Community Case Study
2nd Social Semantic Help Desk Prototype

Deliverable D11.3

Version 1.0

31.01.2009

Dissemination level: PU

Nature	P
Due date	31.12.2008
Lead contractor	EDGE-IT S.A.R.L
Start date of project	01.01.2006
Duration	36 months

Authors

Eugeni Dodonov, EDGE-IT S.A.R.L
Arnaud Laprévote, EDGE-IT S.A.R.L
Stéphane Laurière, XWiki
Fabio Mancinelli, XWiki
Sebastian Trüg, EDGE-IT S.A.R.L

Mentors

Ansgar Bernardi, DFKI GmbH

Project Co-ordinator

Dr. Ansgar Bernardi
German Research Center for Artificial Intelligence (DFKI) GmbH
Trippstadter Str. 122
D 67663 Kaiserslautern
Germany
Email: bernardi@dfki.uni-kl.de, phone: +49 631 205 75 105

Partners

DEUTSCHES FORSCHUNGSZENTRUM F. KUENSTLICHE INTELLIGENZ GMBH
IBM IRELAND PRODUCT DISTRIBUTION LIMITED
SAP AG
HEWLETT PACKARD GALWAY LTD
THALES S.A.
PRC GROUP - THE MANAGEMENT HOUSE S.A.
EDGE-IT S.A.R.L
COGNIUM SYSTEMS S.A.
NATIONAL UNIVERSITY OF IRELAND, GALWAY
ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE
FORSCHUNGSZENTRUM INFORMATIK AN DER UNIVERSITAET KARLSRUHE
UNIVERSITAET HANNOVER
INSTITUTE OF COMMUNICATION AND COMPUTER SYSTEMS
KUNGLIGA TEKNISKA HOEGSKOLAN
UNIVERSITA DELLA SVIZZERA ITALIANA
IRION MANAGEMENT CONSULTING GMBH

Copyright: Nepomuk Consortium 2008
Copyright on template: Irion Management Consulting GmbH 2008

Versions

Version	Date	Reason
0.1	31.12.2008	First draft by the authors
0.2	21.01.2009	Features' description addition
1.0	31.01.2009	Remarks from the mentor taken into account

Explanations of abbreviations on front page

Nature
R: Report
P: Prototype
R/P: Report and Prototype
O: Other

Dissemination level
PU: Public
PP: Restricted to other FP6 participants
RE: Restricted to specified group
CO: Confidential, only for Nepomuk partners

Executive summary

"Open source software is creating a global software space, with dependencies between projects, is meshing software from many different sources. But we are not meshing the data about the software!" [1]

The objective of the NEPOMUK Mandriva case study is the adoption, application and validation of the services provided by the NEPOMUK platform in the context of the on-line community of Mandriva Linux users. The team involved designed a prototype providing the members of this community with a new generation tool for sharing knowledge related to the Mandriva Linux project. It features in particular collaborative semantic annotation at the desktop or at the Web levels using a dedicated vocabulary, and semantic search across a set of peers. The expected user benefit provided by the NEPOMUK possibilities is the capability to interlink a personal semantic Web with a public semantic Web maintained by a large community, for improved learning and issue solving processes.

The second version of the NEPOMUK Mandriva help desk is named "SWIM", which stands for "Semantic Web enabled Issue Manager". The prototype lets users annotate resources locally or remotely using the NEPOMUK-KDE infrastructure and share their annotation with their trusted peers. Users can for instance link a question and the set of related answers with local files, related software or hardware, existing bug entries, other discussions on the Web etc. The established semantic graph is then meant to let them find again information efficiently when dealing with similar issues, and to contribute to the community shared knowledge. This approach could be widened to all processes involved in the collective design and usage of open-source software by allowing contributors to annotate all information artefacts of the related software forge: source code, specifications, mailing-lists etc.

SWIM has been brought on-line as an additional service of the Mandriva Club at the following URL: `http://club.mandriva.com/swim/`. It features a tight integration with the KDE semantic desktop implementation: connecting a new USB peripheral will for instance present the user a set of links pointing at relevant information such as related drivers, tutorials, articles, bug entries and discussions.

The evaluation of SWIM has been carried out by Mandriva Linux users through an online questionnaire after experimenting the system through evaluation scenarios. User answers to the questionnaire show a real interest in the approach: users consider that SWIM improves the efficiency of the community for bringing answers collectively to technical questions indeed. The users also submitted suggestions for improvements, which will be taken into account in the future implementations of SWIM.

The effort consisting in harnessing the Semantic Web principles for improving community based software engineering is still at its infancy. SWIM counts among the first industry steps toward Semantic Web based software engineering. Exploitation of the prototype will first consist in globally improving the connection between Linux help desks, issue trackers, discussions and people. SWIM is likely to continue its evolution as an open-source project now open to a wider community through the dedicated development site at `http://nepomuk.forge.mandriva.com`. The commercial exploitation of the prototype will consist in packaging SWIM as an enterprise collaborative annotation system, around which Mandriva considers offering a range of professional services.

[1] Henry Story, Sun Semantic Web evangelist

Table of contents

1 Introduction

> *"Open source software is creating a global software space, with dependencies between projects, is meshing software from many different sources. But we are not meshing the data about the software!"* [2]

This report presents a documented and evaluated prototype of a social help desk for the Mandriva Linux community, including lessons learned, methodological guidelines, and a roadmap for future implementations. The prototype puts into practice the NEPOMUK platform in the context of the Mandriva Linux open community and shows how collaborative semantic annotation at the desktop or at the Web levels increases the efficiency of the community for using and producing knowledge. The expected user benefit provided by the NEPOMUK possibilities is the capability to interlink a personal semantic Web with a public semantic Web maintained by a large community, for better learning and issue solving processes.

The second version of the prototype is named "SWIM", standing for "Semantic Web enabled Issue Manager". It brings many enhancements and new features. Its functionalities were widened and tightly integrated in the desktop by introducing a dedicated KDE application relying on the NEPOMUK-KDE infrastructure [3] allowing to annotate resources locally or remotely, and to share annotations with trusted peers.

SWIM has been brought on-line as an additional service of the Mandriva Club at the following URL: `http://club.mandriva.com/swim/`. Its evaluation has been carried out qualitatively by Mandriva Linux users through an online questionnaire after experimenting the system through evaluation scenarios. User answers to the questionnaire show a real interest in the approach and proves that the expected user benefit is partially or entirely fulfilled, depending on user profiles. Users globally consider that the prototype improves the efficiency of the community for bringing answers collectively to technical questions indeed. Suggestions submitted by the users will be taken into account for the future versions of SWIM.

The document comprises the following sections:

- Section 2 is a note on the work progress,
- Section 3 presents a typical SWIM scenario and the associated demo script,
- Section 4 outlines the SWIM architecture,
- Section 5 presents the evaluation's methodology and its results,
- Section 6 depicts the exploitation paths that are being considered.

[2] Henry Story, Sun Semantic Web evangelist
[3] `http://nepomuk.kde.org`

2 Work progress

The assessment report following the final project's review outlined the fact that, by the time of the review, it was

> *"not clear what the actual outcome of the workpackage will be: tangible results could not be presented at the review meeting", and that the project had to "redress this situation, taking into account the short time that is still available for the project".*

During the last months of the project, the team involved in the implementation, the packaging and the assessment of SWIM was broadened for fulfilling the requirements expressed in the help desk specification document [1]. This permitted to assemble the various components that had been worked on since since the first release of SWIM (named XWITS) into an integrated, stable and efficient software, SWIM v2.0, released in December 2008. This second version could be tested by a panel of Mandriva Linux users in January 2009. The users who took part in the experiment globally agreed on the added value brought by SWIM in their daily Linux-related information search and learning processes, as described more deeply in the assessment section of this report.

SWIM was packaged as an installable software for Mandriva Linux 2009.0. The installation process is described in the scenario's demonstration section 3.3.

The Mandriva SWIM team started to disseminate the prototype to external communities by blogging about it [4], by opening up the dedicated project's site to the open-source community at `http://nepomuk.forge.mandriva.com`, and by taking part in active discussions on the topic of issue management based on Semantic Web technologies.

Mandriva R&D head is available for making a new demo of SWIM v2.0 at the convenience of the NEPOMUK reviewing team.

Mandriva is confident that SWIM will keep evolving both through other R&D projects that will bring additional features especially in the natural language processing area, and by the involvement of new contributors, who already expressed a strong interest for the approach, in particular within the Baetle community [5] aiming at modelling and instrumenting the software issue tracking process, and the HELIOS project's team [6], working on application lifecycle management tooling.

[4] Sebastian Trüg blog entry about Swimpomuk: `http://nepomuk.forge.mandriva.com/wiki/Swimpomuk`

[5] Baetle: Bug And Enhancement Tracking LanguagE, `http://code.google.com/p/baetle/`

[6] `http://www.helios-platform.org/`

3 Scenario and demo script

This section reminds of the general context of the case study, then depicts a typical scenario as a walkthrough of SWIM capabilities, which is illustrated by a demo script going through the steps needed for trying out the main features.

3.1 Context reminder

Figure 1 is a reminder of the case study context: Mandriva Linux users members, an open online community which interacts without any clearly defined organizational structures, have a NEPOMUK enabled desktop, which includes rich user interfaces for manipulating knowledge in the form of semantic graphs, and for communicating over a P2P network.

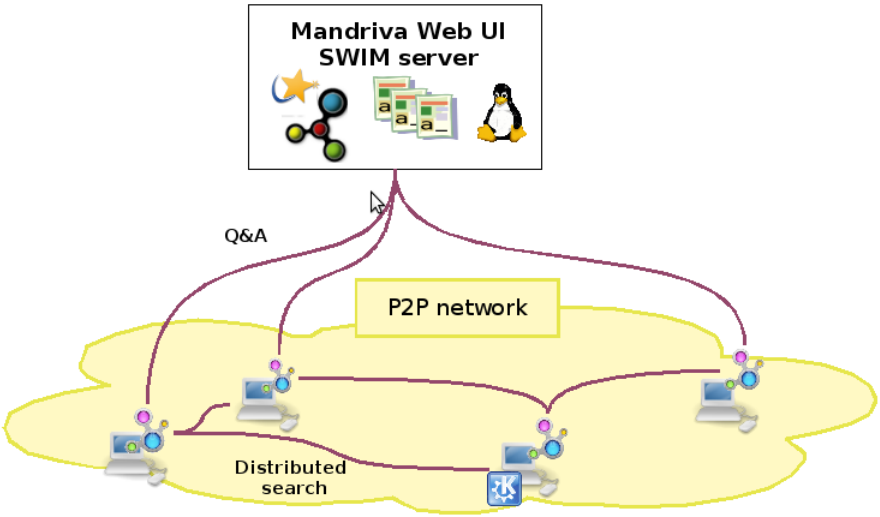


Figure 1: Big picture of the Mandriva Linux community case study

The goal the prototype is to improve the efficiency of the Mandriva Linux community when answering collectively to technical questions, and to support the users in their usage of a Mandriva Linux system. In the first version of the prototype, the emphasis was brought on the server aspects and on their integration into the XWiki engine, the Mandriva community platform infrastructure. In the second version of the prototype, the services running server side for dealing with metadata have been improved and they were made generic enough for not relying on a specific underlying platform. The developments that led to SWIM v2.0 mainly focused on a tighter integration of the system with the NEPOMUK-KDE semantic desktop, and on social features implementation.

3.2 Scenario

Kim is a user of Mandriva Linux 2009. He's running the system on an IBM ThinkPad T41 notebook. Kim just acquired a new EPSON scanner named "EPSON Perfection 1670". He plugs it into his laptop. A panel pops up at the bottom right corner of his screen, displaying a set of resources that relate

to the peripheral: an electronic manual for the scanner, a list of available drivers, some discussions related to the device, and existing bugs known on specific computer systems. Kim browses each referenced driver page. On one of them, corresponding to a package provided by the company Avasys, Kim notices that the Avasys driver is marked as "works out of the box" on Mandriva Linux 2009 by a person named Jim Weston, annnotated by other members of the community as a master in the area of drivers. Kim downloads the package provided by Avasys and gets his scanner installed successfully indeed. Since he remembers it's not the first time he followed a recommendation by Jim on driver related issues, he decides to tag Jim with the keyword "drivers" and to state that he trusts him for driver related matters.

Kim then scans a few printed images of his wife, Anna. He's getting inspired by the pictures and feels like transforming a portrait of Anna into a painting-like picture. The default image design software available in Mandriva 2009 is called "The GIMP". He then looks for existing documentation about it using Google, from which it gets tons of links. As he feels perplex about the results, he gives a try to the SWIM semantic search engine. While browsing the list of results, he notices that his friend Thomas Parker recommends to beginners the following GIMP documentation: `http://www.tutorialsphere.com/tutorials/gimp` (which is listed only on the fourth page of Google when submitting the query 'gimp tutorial'). After a few days, Kim has read all the documents he's interested in on that site. Eager to progress further, he buys an electronic version of a book about Gimp, stores it, and uses Dolphin, the NEPOMUK file manager, for drawing a link between the software and the book. Kim also keep browsing additional resources – How-tos, guides, FAQ – which he annotates with Swimpomuk, the SWIM client for KDE, which works as a Konqueror sidebar.

While keeping working on customizing his pictures, Kim notices his laptop is producing an annoying whistle. Having no clue on how to stop it, he files an issue in the Mandriva forum. A few minutes later, Venkatesh Paraman, an IBM specialist, posts an answer pointing at a page titled "Problem with high pitch noises" in the ThinkWiki [7], a wiki for IBM/Lenovo ThinkPad users running a Linux based operating system: `http://www.thinkwiki.org/wiki/Problem_with_high_pitch_noises`. By skimming over the page, Kim quickly identifies a possible fix, consisting in limiting the ACPI CPU power states. The recommended fix solves the problem indeed. Kim also notices there's an in-depth discussion in the associated "Talk" page [8]. Since Kim is always enthusiast about givning feedback and sharing his knowledge, he annotates with Swimpomuk the discussion thread that he started on the Mandriva forum, by entering the following semantic relations:

- the issue is in "Resolved" status,
- it occurs with the computer system "IBM ThinkPad T43",
- it has the following tags: "noise", "acpi",
- it relates to the manufacturer "IBM",
- it occurs when running the Mandriva 2009 operating system (and probably others as well),
- the "Talk" page relates to the issue.

Since these annotations will be useful to the whole community, Kim uploads them to the SWIM server from Swimpomuk. Swimpomuk however filters out all the annotations relating to Kim's desktop entries.

[7] `http://www.thinkwiki.org`
[8] `http://www.thinkwiki.org/wiki?title=Talk:Problem_with_high_pitch_noises`

Kim then continues browsing other Web resources about Gimp with Firefox. Kim is using Firefox v3.0, which he installed from an unstable Mandriva repository since he really wanted to try it out for its numerous new features. The Firefox window is suddenly moved to another desktop area without him doing any action for that. Kim searches for any existing bug entry pertaining to the problem, by using the SWIM Web UI. He is headed to the Firefox page listing all the open bugs. He identifies one that corresponds to the problem, which references an entry "upstream", i.e. in the project's issue tracker itself – Mozilla Bugzilla in this case. The corresponding content outlines a possible workaround. Kim follows the instruction and gets Firefox 3 work fine again, until a new more stable package gets released by Mandriva developers.

After a few days, Kim has acquired expertise in using Gimp. He remembers a photo of his son taken by Anna during their stay in Barcelona the Summer before. In order to find it, he searches for all photos of his son tagged with the keyword "Barcelona" in the local network, comprising Anna's desktop computer and his. He quickly finds the one he had in mind, downloads it from Anna's computer and loads it into Gimp. Later on, Kim receives an email from his friend Thomas. Kim replies to him, and lets him know that he made much progress on Gimp. Using Swimpomuk, he compiles all the relevant annotations he collected over the Web, and adds them to his message in case Thomas also wants to improve his Gimp expertise.

3.3 Demonstration

This section shows how to run the Kim scenario depicted above using the SWIM tools. It first explains how to get and install the needed software, then it presents the steps to be followed for each action mentionned in the above scenario.

3.3.1 Requirements

Swimpomuk – the SWIM client for KDE – runs on a KDE4 Linux desktop. Running the prototype requires for instance Mandriva 2009 either natively or within a virtual machine such as VirtualBox. Mandriva 2009 ISO can be downloaded from the following page: `http://www.mandriva.com/en/download/free`. The ISO file has then to be burnt onto a CD, which can be booted either for installing Mandriva on the hard disk or for loading it in memory for a "live" session. The installed system needs then to be updated to the latest package versions. This can be done by launching the following commands in a terminal:

```
urpmi.addmedia --distrib ftp://distrib-coffee.ipsl.jussieu.fr/pub/linux/
MandrivaLinux/official/2009.0/i586/
urpmi.update -a
urpmi --auto-select
```

Swimpomuk is available as a dedicated package in the development repository of Mandriva, known as "Cooker". It can be downloaded from the Mandriva Cooker public repositories such as the following one: `ftp://distrib-coffee.ipsl.jussieu.fr/pub/linux/MandrivaLinux/devel/cooker/i586/media/contrib/release/`, and installed by running the canonical Mandriva installation command:

```
urpmi ./swimpomuk
```

Since Mandriva Linux, KDE4 and Swimpomuk evolve continuously, it is recommended to refer to the SWIM project's home page at `http://nepomuk.forge.mandriva.com/wiki/SWIM` for getting up to date instructions on how to install and run the software and its dependencies.

3.3.2 Demo script

When Kim plugs his scanner into his IBM ThinkPad T41 laptop, a panel pops up, as illustrated in figure 2. The panel resource set contains in particular links to drivers suitable for Kim's scanner model.

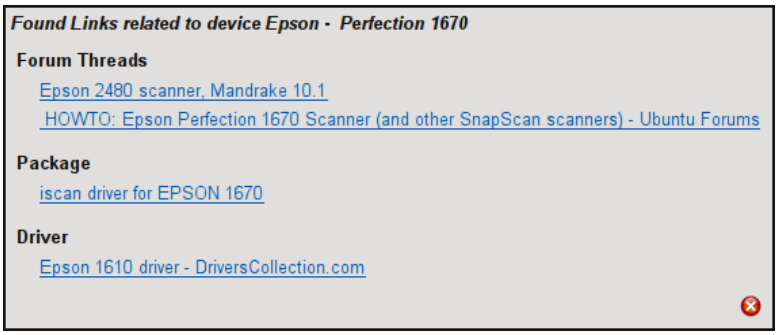


Figure 2: Contextual information prompt when plugging a USB peripherical

Kim browses the driver links available from the help panel. They link to pages hosted on the SWIM server – SWIM Web home is represented on figure 3 – containing annotations from other members of the community, as illustrated by figure 4 relating to a Mandriva 2009.0 package of a driver by Avasys, where we can see that Jim Weston states that the package works "out of the box", that is to say that no manual configuration is needed after the installation.

Figure 3: SWIM at `http://club.mandriva.com/swim`

The SWIM Web page presenting Jim Weston [9] can be obtained either through the SWIM search engine, or by browsing the list of experts (figure 5), available from the SWIM menu . The Jim Weston annotation page contains all statements submitted by the community of users, as illustrated by figure 6, where

[9] In SWIM, the users are identified using the following dedicated scheme: `http://doc4.mandriva.org/users/flastname`. In upcoming versions, the identification scheme will be open.

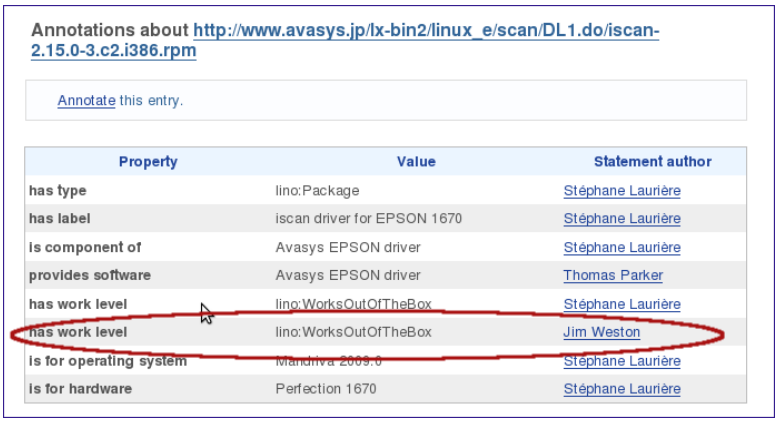


Figure 4: Annotations of a driver package by various experts

Jim is marked as a "Master" by Kim indeed, and is tagged with the keyword "drivers".

Figure 5: Expert list

Kim continues his exploration of the digital imaging realm under Linux and looks for a beginner tutorial for The GIMP. To do so, he first uses the SWIM search engine: Kim selects the resource type he's looking for, i.e. "Documentation", enters "GIMP" in the resource name field, and gets a list of results, with links to their annotations. Figure 7 presents the annotations related to a tutorial hosted on the tutorialsphere.com Web site. Among other entries, there's one by Kim's friend Thomas stating that the tutorial is very pedagogical and is suitable for beginners.

After having bought an electronic book about The GIMP, Kim also draws a link between the book file and the software, using Dolphin, as illustrated by figure 8.

Then, annoyed by the whistle produced by his laptop, Kim files a question in the Mandriva forum, and gets quickly an answer by Venkatesh Paraman, as shown by figure 9. The answer leads to a fix indeed.

Kim shares the acquired knowledge with the Mandriva community by annotating the discussion using Swimpomuk: he first adds the forum entry to the

Annotations about http://doc4.mandriva.org/users/jweston

Annotate this entry.

Property	Value	Statement author
has type	nao:Party	Jim Weston
has label	Jim Weston	Jim Weston
has creator	Jim Weston	Jim Weston
is trusted as	lino:Master	Anna Herzog
is trusted as	lino:Master	Kim
is trusted as	lino:Master	Timothée Béranger
has tag	drivers	Anna Herzog
has tag	drivers	Kim
has tag	drivers	Timothée Béranger
has tag	thinkpad	Timothée Béranger

Figure 6: Annotations of the expert Jim Weston

Annotations about http://www.tutorialsphere.com/tutorials/gimp

Annotate this entry.

Property	Value	Statement author
has type	lino:Documentation	Thomas Parker
has type	lino:HowTo	Thomas Parker
has label	Tutorial sphere GIMP tutorial	Thomas Parker
has comment	This tutorial is extremely well done, especially for beginners.	Thomas Parker
relates to software	The GIMP	Thomas Parker
relates to software	http://www.gimp.org	Thomas Parker

Figure 7: Annotations related to TutorialSphere.com GIMP resources

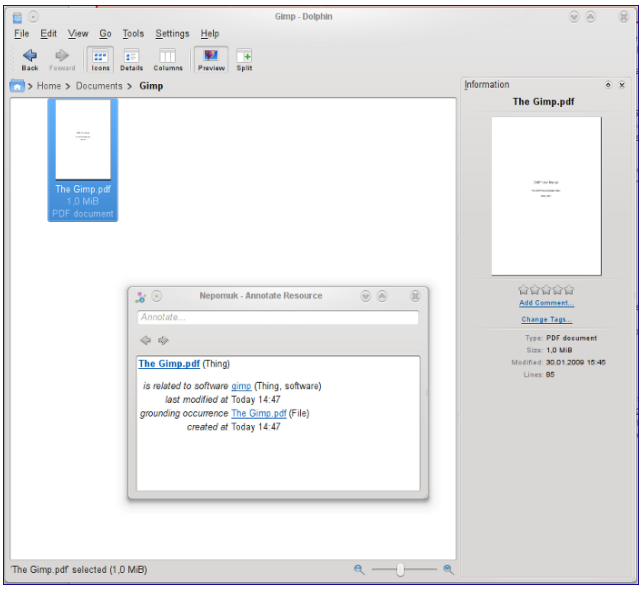


Figure 8: Linking The GIMP with an electronic book stored locally

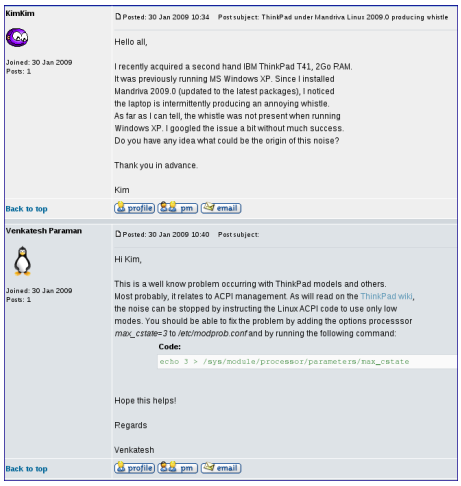


Figure 9: Forum discussion about the whistle produced by Kim's laptop

Swimpomuk's clipboard panel, and drag'n drop it to the metadata panel; then, since Swimpomuk automatically identifies the resource as a discussion, it suggests several annotation possibilites, and Kim can pick the approriate one for marking the issue as "Resolved", as illustrated by figure 10; this results in the forum entry being annotated as "Resolved" indeed (figure 11).

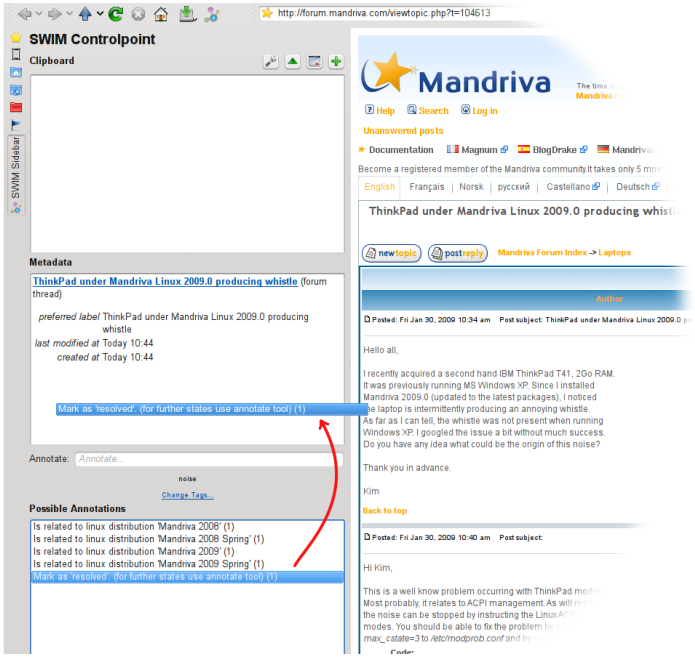


Figure 10: Setting a resolution state using the Swimpomuk suggestions

Kim then uses the annotation tool to link the forum thread to his computer system label, i.e. "ThinkPad T43": when typing "thinkpad t43" in the annotation field, Swimpomuk automatically identifies the text as a system name, and suggests potential relations, as illustrated by figure 12.

After also tagging the discussion with the tags "noise" and "acpi" using the same mechanisms, Kim uses the Swimpomuk clipboard to also link the discussion to the talk in the ThinkPad wiki which helped him solve and understand the problem: he simply drags the link he previously added to the clipboard onto the metadata area, just like he did with the resolved state before (fig-

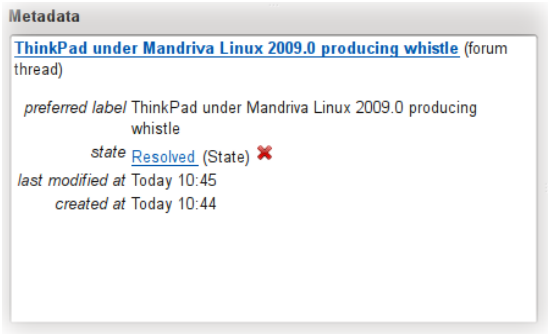


Figure 11: Swimpomuk metadata panel displaying the issue as solved

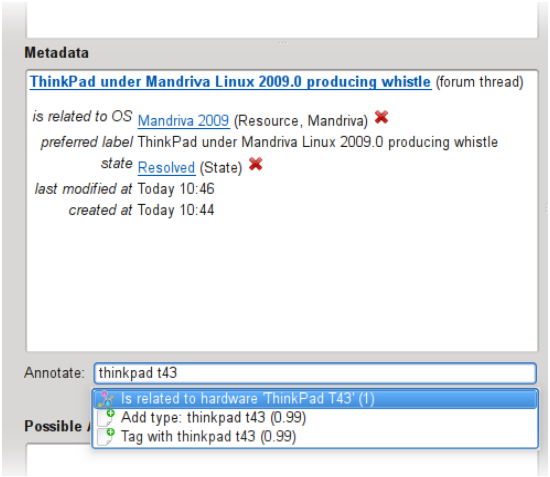


Figure 12: Annotating the discussion by specifying the related system

ure 13).

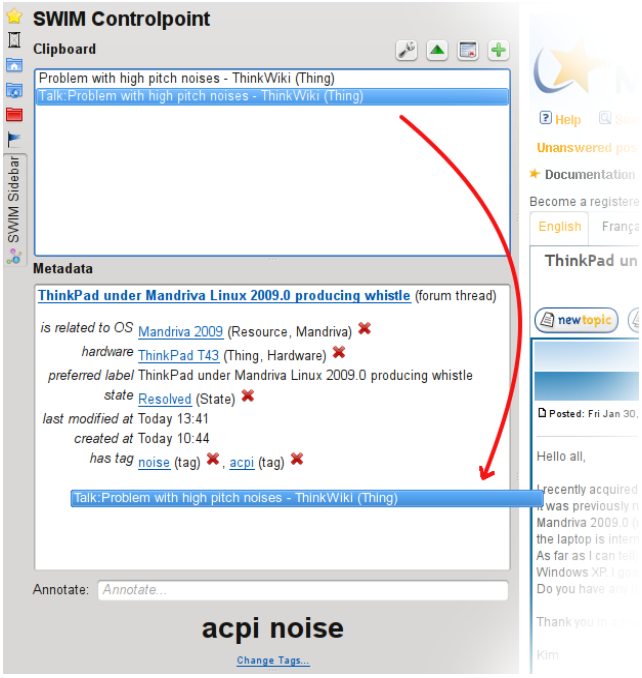


Figure 13: Linking the discussion with another wiki discussion

Eventually, the metadata panel contains all the annotations, as shown on figure 14.

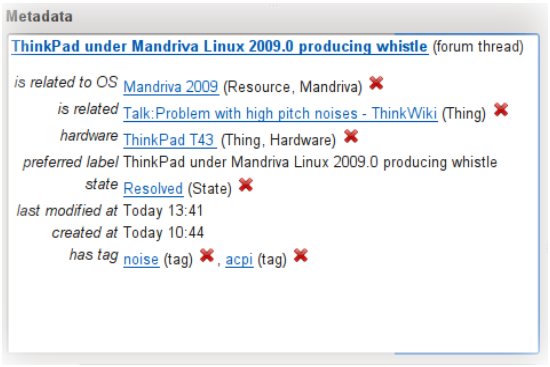


Figure 14: Annotations panel containing all semantic relations

Finally, Kim shares these annotations with the Mandriva community by uploading all annotations to the SWIM server (figure 15).

When Kim's Firefox browser start wandering from one desktop to another, Kim goes through existing Firefox bugs, by hitting the "Firefox" link available from the SWIM search results when looking for the Firefox application. The Firefox SWIM page includes the application bugs indeed, sorted by state, as well as a general description of the application, a link to manual annotations, and the list of available application packages, as illustrated by figure 16.

The annoying bug is listed first in the list, and its annotation page contains links to upstream bugs, as illustrated by figure 17. The link with the upstream database was extracted automatically by SWIM data extractors which continuously crawl external bug databases over the Web.

Once the Firefox problem is solved, Kim gets back to his image design work.

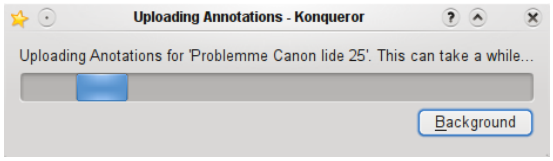


Figure 15: Upload of Kim's annotations to the SWIM server

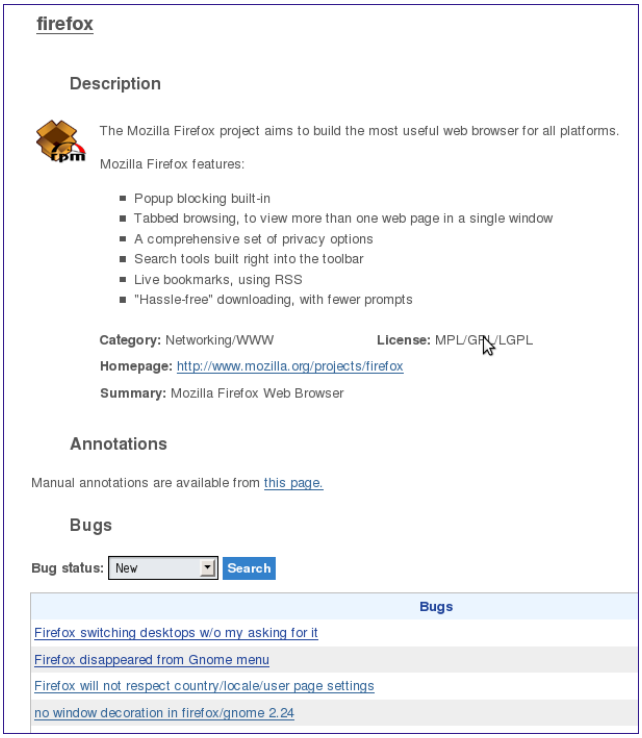
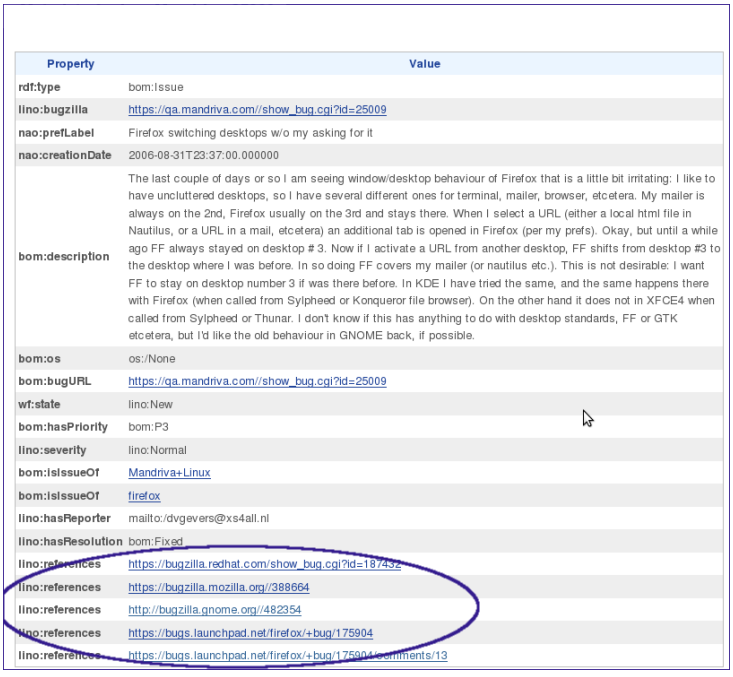


Figure 16: Firefox page on the SWIM server: bugs, packages and link to manual annotations



Property	Value
rdf:type	bom:Issue
lino:bugzilla	https://qa.mandriva.com//show_bug.cgi?id=25009
nao:prefLabel	Firefox switching desktops w/o my asking for it
nao:creationDate	2006-08-31T23:37:00.000000
bom:description	The last couple of days or so I am seeing window/desktop behaviour of Firefox that is a little bit irritating: I like to have uncluttered desktops, so I have several different ones for terminal, mailer, browser, etcetera. My mailer is always on the 2nd, Firefox usually on the 3rd and stays there. When I select a URL (either a local html file in Nautilus, or a URL in a mail, etcetera) an additional tab is opened in Firefox (per my prefs). Okay, but until a while ago FF always stayed on desktop # 3. Now if I activate a URL from another desktop, FF shifts from desktop #3 to the desktop where I was before. In so doing FF covers my mailer (or nautilus etc.). This is not desirable: I want FF to stay on desktop number 3 if was there before. In KDE I have tried the same, and the same happens there with Firefox (when called from Sylpheed or Konqueror file browser). On the other hand it does not in XFCE4 when called from Sylpheed or Thunar. I don't know if this has anything to do with desktop standards, FF or GTK etcetera, but I'd like the old behaviour in GNOME back, if possible.
bom:os	os:/None
bom:bugURL	https://qa.mandriva.com//show_bug.cgi?id=25009
wf:state	lino:New
bom:hasPriority	bom:P3
lino:severity	lino:Normal
bom:isIssueOf	Mandriva+Linux
bom:isIssueOf	firefox
lino:hasReporter	mailto:/dvgevers@xs4all.nl
lino:hasResolution	bom:Fixed
lino:references	https://bugzilla.redhat.com/show_bug.cgi?id=187432
lino:references	https://bugzilla.mozilla.org//388664
lino:references	http://bugzilla.gnome.org//482354
lino:references	https://bugs.launchpad.net/firefox/+bug/175904
lino:references	https://bugs.launchpad.net/firefox/+bug/175904/comments/13

Figure 17: Semantic annotations related to a Firefox bug

He searches for all available pictures tagged "Barcelona" in the home network, using the NEPOMUK social query client. Figure 18 illustrates this action, which terminates the demo script.

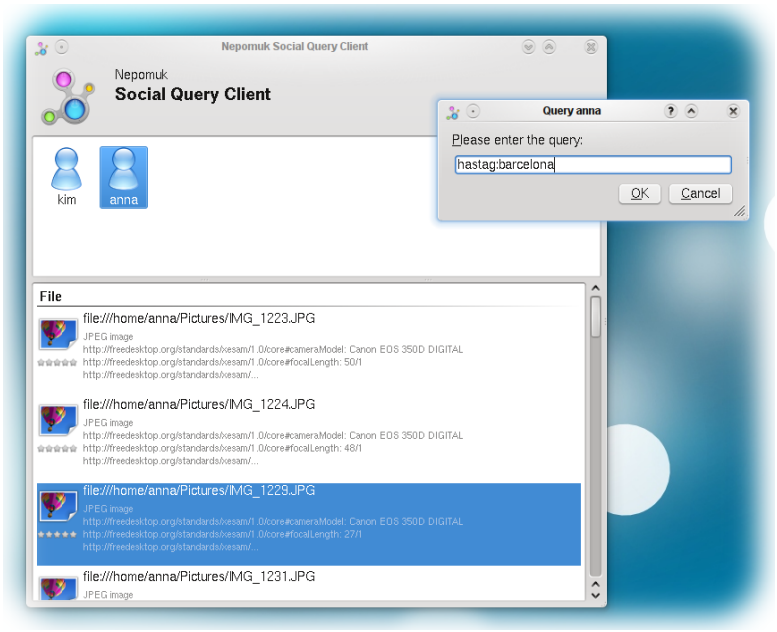


Figure 18: Cross desktop semantic search

4 Prototype architecture

This section presents the SWIM architecture enabling the scenario depicted in the previous section 3. The cornerstones of the SWIM system are comprised of the following items:

- the LINO domain specific ontology (Linux Issue Ontology),
- the SWIM server (previously named "Linuxpedia") relying on the NEPOMUK core components, including a Web UI implemented on top of the XWiki engine,
- the SWIM client for KDE,
- the NEPOMUK social query daemon.

4.1 Domain ontologies

The SWIM help desk data model was designed on the one hand by using the requirements expressed by Mandriva and on the other hand by using as inspirational sources the data models of existing domain ontologies such as BOM (Bug Ontology Model), Baetle (Bug And Enhancement Tracking LanguagE), the Bugzilla data model and other issue trackers models (Launchpad, JIRA etc.). Since no existing ontology covers all the needs expressed during the specification phase, a dedicated ontology was created: LINO, standing for "Linux Issue Ontology".

4.1.1 BOM and Baetle ontologies

BOM is part of the EvoOntis ontology set [10] [3], a set of software ontologies and data exchange format based on OWL and designed by the distributed information systems research group at the University of Zurich's Department of Information Technology. As stated on the EvoOntis Web site,

> *"EvoOntis provides the means to store all elements necessary for software analyses including the software design itself as well as its release and bug-tracking information".*

BOM's main classes are the following: Comment, Defect, Issue, Project, Version, Enhancement, Feature. All BOM classes and properties are documented on the BOM document site [11]. Some of the BOM classes and properties were extended within LINO.

Baetle is a project by Henry Story, a Semantic Web specialist at Sun Microsystem. It aims at enabling the following items, as stated on the project's Web site:

- *SPARQL end points on a bug database that can be queried,*
- *Bugs in one open source project that refer to bugs in other open source projects, and that specify their dependency on these,*
- *A format for bug databases to exchange their bug data,*
- *Relating bugs to software artifacts so that these issues can be tracked.*

As SWIM goals include those objectives, contacts have been established with the Baetle community, and some Baetle concepts were reused in LINO.

[10] http://www.ifi.uzh.ch/ddis/evo/
[11] http://www.ifi.uzh.ch/ddis/evoont/2008/11/bom/doc/ontologies/bom.html

4.1.2 Linux Issue Ontology

The Linux Issue Ontology is meant to enhance BOM and Baetle by additional classes and properties covering more specifically the Linux software and hardware domains. The LINO ontology is used for describing the metadata associated with software or hardware entries. LINO lets the user describe the relations between an issue and the related problematic hardware or software, other similar issues, the context of the issue, the software packages involved in the problem etc. The complete version of the LINO ontology is available in the Appendix section 8 of the document, with comments for each class and property.

4.2 SWIM components

The SWIM component set comprises a SWIM server, a set of libraries on top of the NEPOMUK core infrastructure for interacting with databases storing issues related to Linux applications, a Web user interface implemented on top of XWiki, a KDE application – Swimpomuk – letting users annotate issues using the domain ontologies, and a daemon service enabling social search – NEPOMUK social query service daemon. The components are represented on figure 19.

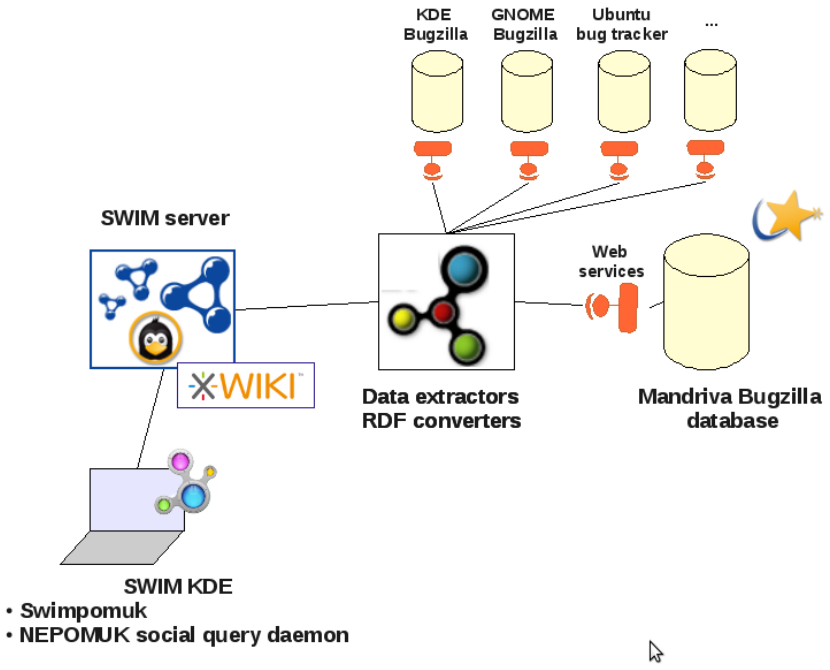


Figure 19: SWIM components overview

4.2.1 Usage of the NEPOMUK components

The following paragraphs describe more specifically how each NEPOMUK component is used within SWIM.

NEPOMUK middleware The NEPOMUK middleware is used as a registry of services [4].

NEPOMUK RDF repository The RDF repository is used for storing all the annotations added to help desk resources, using the NAO [5], NRL [6] ontologies, as well as the domain ontology LINO. The NEPOMUK RDF API is implemented on top of the open-source Virtuoso database engine.

NEPOMUK DataWrapper The DataWrapper [2] is used both server side and client side. Several aperture crawlers have been implemented: one for converting the bugs referenced in Bugzilla databases into RDF statements; another one for converting the hardware information available from the Web site Hardware4Linux.info [12]. On the client side, the DataWrapper component is used for crawling the user's personal resources and storing their metadata in the RDF repository.

NEPOMUK-KDE NEPOMUK-KDE consists of a set of libraries and user interfaces for manipulating desktop metadata, as described at `http://nepomuk.kde.org`. This infrastructure lets the user browse their personal semantic Web, annotate existing resources and create new ones. By putting into practice the NEPOMUK-KDE libraries in the context of the SWIM help desk, users can bridge their personal desktop model with the semantic graph of the SWIM help desk knowledge base. By doing so, the user extends the public knowledge with personal views and links, benefits from contextual assistance and can customize his learning process. In addition, users can share their personal semantic Webs with others and get recommended resources through their network of friends.

4.2.2 SWIM server

The SWIM server is a shared metadata repository used for storing statements relating to bugs, help desk resources, software packages and other public material. The SWIM server repository serves two purposes: first, it helps the Mandriva Linux users identify bugs that relate to some issues they may be running into when using a given application; second, it is meant to ease the linking of Linux distribution bugs with "upstream" bugs, i.e. bugs referenced in the related application bug repository. This feature is likely to solve a common problem related to Linux distributions: there are actually as many Linux information systems as distributions for documentation and for development. The SWIM server prototype is an effort to interlink those information systems with the objective to improve the Linux user experience and the Linux process. By being more and more adopted by the community, this Semantic Web platform will let developers and users draw links between bug entries showing up in various places (for instance in Debian bug database, Firefox bugzilla, Mandriva bugzilla etc.). Once these resources get linked indeed, users can be updated about the bug status simultanously in different places, even if the underlying bug management systems are not the same. Metadata can then apply to a specific bug associated with different URIs, which eases the maintenance of the bug status and the query by users or developers on related comments.

The SWIM server is implemented on top of the NEPOMUK RDF API and uses the open-source Virtuoso RDF database as its backend. Virtuoso was chosen among other alternatives for its good performance. SWIM is initially feeded by the results of the Mandriva bug extraction using a Bugzilla DataWrapper. This DataWrapper is coded in Python and is available as open-source software publicly available from the Nepomuk Mandriva subversion repository [13] under the GPL license. The SWIM Bugzilla DataWrapper indexes the Buzilla database

[12] `http://www.hardware4linux.info`
[13] `http://nepomuk.forge.mandriva.com/`

over the Web and enhances the extracted data by linking the bugs to issues referenced in other bug repositories. The results are then displayed on the SWIM server.

For example, the URL `https://qa.mandriva.com//20002` will create the following entries in the SWIM server:

```
<https://qa.mandriva.com//20002>
  bom:isIssueOf <software:/mDNSResponder> .

<software:/mDNSResponder>
  a lino:Software ;
  nao:prefLabel "mDNSResponder" ;
  lino:version "98" .

<software:/mDNSResponder>
  lino:package <package:/mDNSResponder-98-4mdk> .

<package:/mDNSResponder-98-4mdk>
  a lino:Package ;
  nao:prefLabel "mDNSResponder-98-4mdk" .
```

The list of all bugs related to a given application is then created by issuing a SPARQL query such as:

```
select * where {?bug a bom:Issue . ?bug bom:isIssueOf ?software}
```

The SWIM Web user interface running on the SWIM server was implemented as a set of Groovy scripts on top of the XWiki engine, interacting with the database through the NEPOMUK RDF API. It mainly gives access to the following services: search service, annotation service, annotation display, query service, and semantic graph browsing. Figure 20 shows how an annotation stating that Kim is trusted as a journeyer would be entered by the current user of the system.

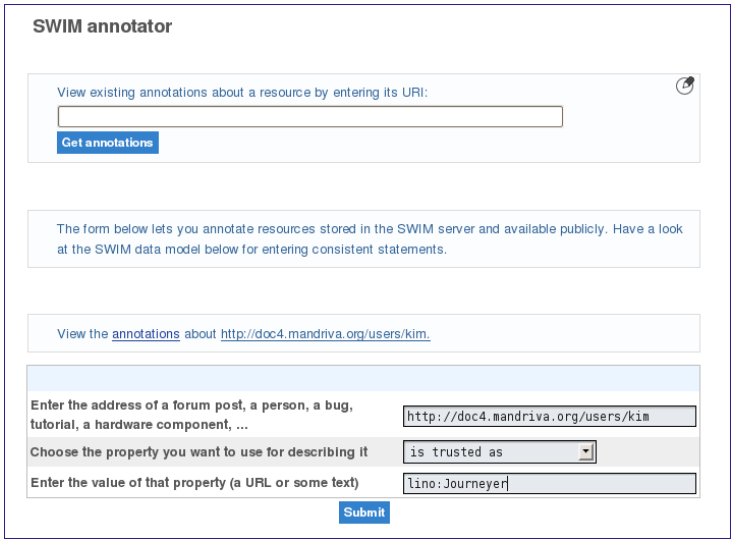


Figure 20: SWIM Web annotator

4.2.3 SWIM client for KDE

The SWIM client for KDE, Swimpomuk, is a KDE 4 client for the Mandriva SWIM server as well as the local NEPOMUK system. It allows to annotate Web pages – especially forum posts and bug reports – with different types of data. As outlined in the scenario section 3, one can draw links between pages, link a page to a specific software or hardware, pages can be tagged, or linked to local files. The created information is stored in the local NEPOMUK system and can optionally be uploaded to the SWIM server. Swimpomuk consists of two main components: a Konqueror sidebar and a small system tray application called "Bugger" which proposes information on newly plugged in devices such as scanners, digital cameras etc. The Swimpomuk sidebar comprises a set of panels:

- A clipboard panel, which allows users to keep Web resources that they want to annotate or to link with one another.
- A metadata panel, which displays the annotations of the current browser page that are available locally or remotely and which lets the user annotate further the loaded Web page.
- A suggestion panel, which searches into the configured metadata repositories (including the local one), extracts from them the resources that may be linked to the current one and suggest the matching entries as relation candidates.
- A semantic query panel for searching into the local and the remote metadata repositories.

The sidebar also provides four buttons in s small toolbar at the top:

- Configure connection options for the SWIM server (username/password),
- Upload annotations to the SWIM server: uploads all annotations (except those to local files) of the current web page to the SWIM server,
- Clear clipboard,
- Add current page to the clipboard.

Using the clipboard is straight-forward: using the "add to" clipboard button, one adds the currently browsed page to the clipboard. Later on, the links can be dragged from the clipboard to the metadata area to link pages.

The metadata area displays all annotations of the current Web page. Each annotation has a little red cross behind it which can be clicked to delete the specific annotation. As mentioned before, links can be dropped from the clipboard (and also the annotation recommendation area) to the metadata area. This will immediately create a new annotation. The current Web page will be linked to the dropped page. The type of link is determined automatically based on the type of page – e.g. a bug report, a forum thread or a hardware4linux.info page. Below the metadata display, the annotation tool allows to perform arbitrary annotations by simply entering a piece of text. The framework then performs auto-completion on this text and propose a set of possible annotations. Typical examples include the annotation with a hardware component (when entering the manufacturer or the component name) or a piece of software, as illustrated by figure 21.

The system searches for matching software components and proposes them in a drop-down list. Once chosen, the new annotation is created and the metadata display is updated, as shown on figure 22.

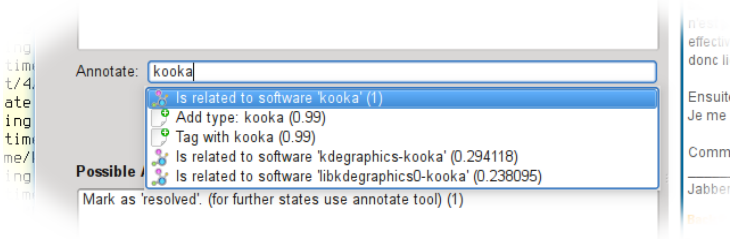


Figure 21: Text completion example

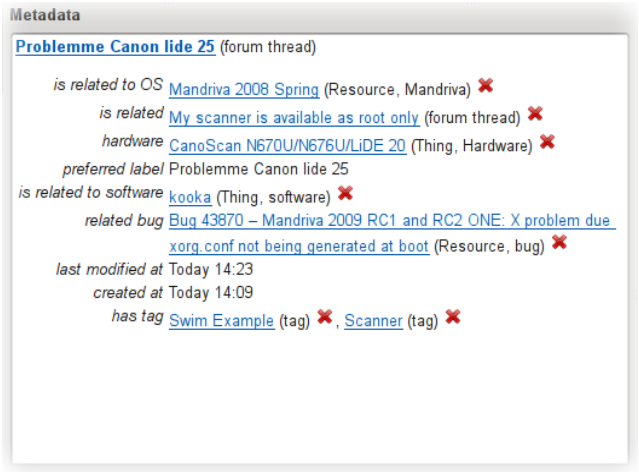


Figure 22: Swimpomuk metadata panel

The area at the bottom of the side bar displays a list of recommendations and common annotations. Common annotations include setting the state of a bug or forum post to closed/solved or linking a post to a specific version of Mandriva. These annotations can also be performed using the annotation tool described above. Other recommendations are based on annotations that other users uploaded to the SWIM server.

As mentioned above, the toolbar also contains a button to start the uploading of local annotations. Once this button is clicked and local annotations exist, these are uploaded to the SWIM server. Since uploading annotations to the server can take some time, the process can be sent to the background which allows to continue daily work.

4.2.4 NEPOMUK-KDE social query component

In addition to its annotation and search capabilities, SWIM makes it possible to issue distributed search queries across a newtork of trusted peers. This feature relies on the NEPOMUK Social Query Daemon server that is embeded into Soprano. The component serves the standard NEPOMUK interfaces; however, since the implementation extends the KDE version of NEPOMUK, no integration with PSEW have been developed in 2008. The SWIM development team will consider such an integration in the future.

Running the NEPOMUK-KDE social query component allows other users in the local network to query the NEPOMUK store. The system relies on a local NEPOMUK service providing read-only access to the annotations' repository over an HTTP interface which is published via the zeroconf protocol. Other NEPOMUK users in the local network can then use the NEPOMUK social query client to run queries on the remote repositories. The query engine is the same also used for local desktop search. Due to the possible security risk that the daemon poses, it is not started automatically. It has to be started manually which can be done with the following command:

```
qdbus org.kde.NepomukServer /servicemanager org.kde.nepomuk.ServiceManager.
startService nsqd
```

Each user in the local network running the the daemon can be queried using the service client. When simply double clicking on the user to be queried, a dialog will request the query string which is then executed on the remote repository.

5 Evaluation

This section describes the assessment that was performed on the second version of the SWIM prototype. The evaluation session took place in January 2009. It consisted first in measuring the completeness of SWIM v2.0 with regard to the use cases planned initially, second in collecting the input from users invited to use SWIM by following a set of scenarios.

5.1 Completeness level

The objective of the second SWIM implementation milestone was to carry out a sound integration and extension of the NEPOMUK components into the domain specific area of Mandriva Linux technical issues. Most of the use cases introduced in the deliverable D11.1 [1] have been implemented, and sound foundations have been laid down for implementing additional advanced features in the future. Table 1 reminds of the features initially planned and outlines their implementation state.

Feature	Description	Status
Data source metadata extraction	SWIM contains a DataWrapper which extracts metadata from a Bugzilla instance over the Web.	Available
Wiki support for ontology design	SWIM users can design ontologies using the Mandriva wiki based on XWiki. The wiki doesn't manage ontology consistency maintenance though when parts of the ontology evolve.	Partially available
Ontology import	Users can import additional ontologies in their local metadata database.	Available natively in Soprano
Ontology export	SWIM users can export the semantic graph stored locally, or any sub-graph of it using Soprano.	Available natively in Soprano.
Support for semi-structured forms	Using Swimpomuk, the user can enter both semantic links and plain comments, i.e. both structured and unstructured data. A more advanced support of semantic forms could consist of integrating the iPad editor by Cognium Systems as a SWIM client. This option will be considered in the future.	Available
Ontology refactoring	The LINO ontology can be refactored only by editing its Trig definition file. The update of the statements using the ontology requires a manual operation at this stage, though.	Partially available
Embedded semantic documents and support for combined documents	The SWIM client lets user combine resources resulting from a semantic query for creating material for their own use.	Available

Conversion of natural language to statements	The NEPOMUK API for plugging an automatic text processing component has been made available in NEPOMUK-KDE. Basic conversion of text entries into statements is showcased by the Swimpomuk recommendation panel. However, a real integration of NLP WP1 components could not be achieved within the timeframe. EDG will focus on NLP technologies integration into SWIM in the future, in particular in the course of the SCRIBO research project (2009-2010).	Partially available
SPARQL query support	Both the SWIM client and the SWIM server let users submit SPARQL queries to the system.	Available
Text completion based on the domain ontologies, text comparison and template management	As illustrated in the scenario section 3, Swimpomuk completes text entries by suggesting possible relations or keywords matching the entered text.	Available
Real-time collaborative editing	A WYSIWYG real time editor has been developed and integrated into XWiki.	Available
Support for semantic search	As described in the scenario section 3, the SWIM client features basic semantic search harnessing the LINO domain ontology.	Available
Support for P2P social search	As described in the scenario section above 3, the SWIM client features social search across desktops in the LAN.	Available
Search result ranking	The user should be able to give a personal score to the results brought by the system.	This feature is postponed to future implementations.
Ontological representation of desktop events and contextual recommendation	This feature is illustrated in the scenario 3 showcasing contextual information availability when plugging a USB peripheral .	Available
Personal workflow and task pattern support	A SWIM user can create a task instance and link relevant material to this task for getting assisted later on by their personal knowledge workbench for carrying out a given task.	Available
Content synchronization	Users can annotate documents offline and upload their annotations to the SWIM server by hitting the "Upload annotations" button in the Swimpomuk application.	Available
Contextual recommendation	When browsing a question or a bug page, the user gets contextual data as a set in the "Potential relations" panel of Swimpomuk.	Available

Trust and reputation management	Users can annotate expert profiles by using the `lino:hasTrustLevel` and `nao:hasTag` properties. `lino:hasTrustLevel` has values instantiating a subclass of the `lino:TrustLevel` type, i.e. one of the following ones, directly derived from the Advogato.org [14] Web site establishing a secure trust network in the community of open-source developers: `lino:Master`, `lino:Journeyer`, `lino:Apprentice`, `lino:Observer`.	Available
Group management	Groups are managed through the XWiki application on the SWIM server. On the client side, it is planned to enhance Swimpomuk by letting the user enter the public address of the desktop owned by people they trust.	Partially available
Social network visualisation	The social network of users can only be displayed in the form of Web links.	Partially available
Email integration	A mail gateway for the Mandriva forum was plugged in to the system, so that users can receive mail updates when answers are brought to questions they subscribed to, or when new questions are submitted in a category they subscribed to.	Available
Desktop notification	The notification system focuses on automatic prompting of relevant resources when plugging a USB hardware component.	Partially available
Access rights management	SWIM v2.0 users have three distinct information spheres: their personal semantic desktop, the local area network of peers' desktop if their NEPOMUK social query daemon is activated, and the public annotations' sphere on the SWIM server, where resources can be restricted in view or edition mode to specific groups using the XWiki rights management system. In the next version of SWIM, only users declared as "trusted peers" on a user's desktop will be able to access that user's metadata repository.	Available
Content statistics	Statistics can be drawn from each desktop metadata database by running SPARQL queries manually. By storing these statistics in a graph, they can then be uploaded to the SWIM server via Swimpomuk, so that administrators can monitor the usage of the system on a set of desktops.	Partially available

Table 1: Status of the semantic and social features of the SWIM application

In terms of non-functional requirements, SWIM is integrated with the KDE version of the NEPOMUK semantic desktop platform. The Web user interface is compatible with all major operating systems and navigators.

[14] `http://www.advogato.org`

5.2 SWIM usage evaluation

SWIM usage evaluation consisted in interviewing several Mandriva Linux users through a questionnaire inviting them to describe their impressions or suggestions while their going through a set of scenarios.

5.2.1 Evaluation scenarios and questionnaire

The scenarios listed in table 2 were proposed to a panel of 10 users for the purpose of evaluation, with associated questions. The gathering of users' feedback was carried out through a Web form available from the SWIM home page on the Mandriva Club. The form itself describes the steps to be followed for each scenario, before entering the answers, as illustrated by figure 24.

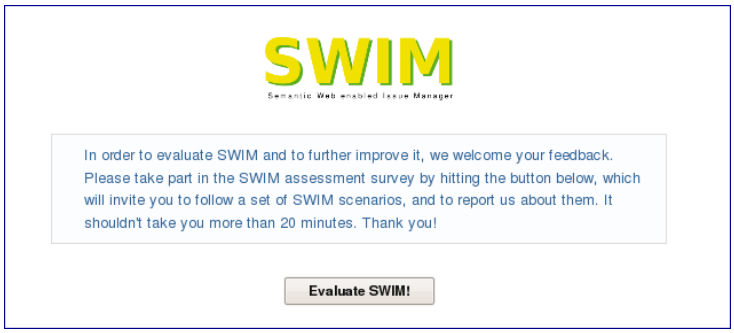


Figure 23: SWIM evaluation home page

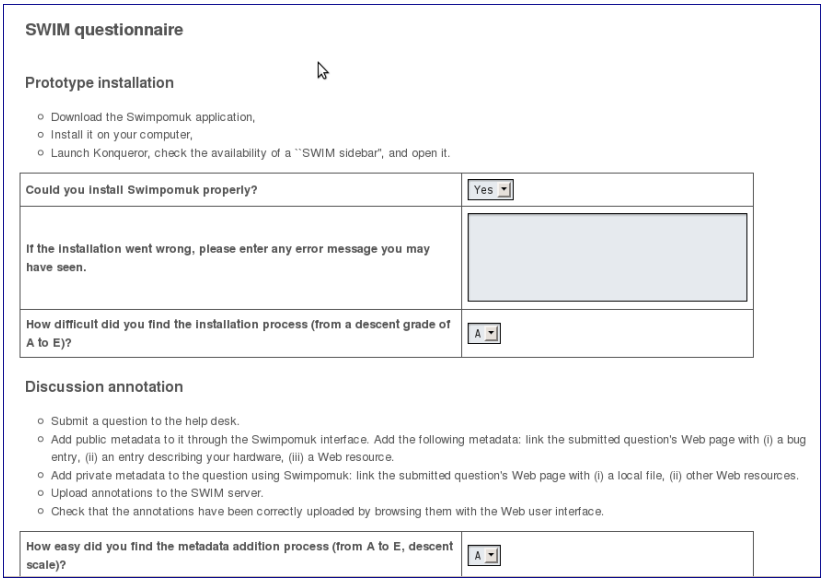


Figure 24: SWIM online evaluation questionnaire

Scenario title	Scenario steps	Questions
Prototype installation	• Download the Swimpomuk application, • Install it on your computer, • Launch Konqueror, check the availability of a "SWIM sidebar", and open it.	• Could you install Swimpomuk properly? • If the installation went wrong, please enter any error message you may have seen. • How difficult did you find the installation process (from a descent grade of A to E)?
Discussion annotation	• Submit a question to the help desk. • Add public metadata to it through the Swimpomuk interface. Add the following metadata: link the submitted question's Web page with (i) a bug entry, (ii) an entry describing your hardware, (iii) a Web resource. • Add private metadata to the question using Swimpomuk: link the submitted question's Web page with (i) a local file, (ii) other Web resources. • Upload annotations to the SWIM server.	• How easy did you find the metadata addition process (from A to E, descent scale)? • Did the upload process go well? • What is your understanding of the "Upload annotations to the SWIM server" action? • Do you have any other remarks or suggestion for improving the annotation feature?
Getting contextual help when plugging a USB peripheral	• Plug a USB peripheral into your computer, such as a digital camera, a printer or a scanner. • If a panel shows up at the bottom right corner of the screen, open several of the proposed links.	• Which USB component did you plug in? • Does a panel pop up at the bottom right corner of your screen? • If so, what does it contain? Do the suggested links relate to your peripheral indeed? • Were the suggestions helpful indeed (A to E)? • Do you have any other remarks or suggestion for improving the contextual help feature?

Annotating an expert using the Web UI	• Browse the list of experts on the SWIM "Experts" Web page, • Choose the expert "Kim" and go to his annotations page, • Annotate Kim by tagging him with the keyword "printers", • Inform the system that you trust Kim as a journeyer (enter the following property value: lino:Journeyer in the appropriate field).	• Did you manage to find Kim's annotation page? • Did you manage to tag him? • Did you manage to add that you trust him as a journeyer? • What score do you give to the SWIM Web UI (A to E)? • Do you have any remarks or suggestion for improving SWIM Web UI?
Sharing private resources with others	• Start the NEPOMUK social query daemon by issuing the following command: `qdbus org.kde.NepomukServer /servicemanager org.kde.nepomuk.ServiceManager.startService nsqd` • Run a SPARQL query using the social query dialog.	• Did the social query daemon start successfully? • If not, which message did you get? • Which query did you run? • What results did you get? • How would mark the service (from A to E)? • Do you have any other suggestion for improving the social query feature?

Table 2: Evaluation scenarios

5.3 Results and lessons learned

The results have been gathered and categorized either as being related to SWIM's added value or to its usability. Table 3 presents a summary of the answers received. Globally, even if some of the prototype features still require an advanced understanding of the Semantic Web underlying technology, users were satisfied with the capabilities offered by SWIM. The most technology-savvy users were even enthusiast about it and about its potential.

Scenario	Answers' summary
Prototype installation	100% of the users could install the prototype, and all of them found the process straightforward (score: A).
Discussion annotation	60% of the users considered the annotation process was easy (score: A or B), while 40% found it not so easy (two answered "C" and two answered "D"). The upload process went well for all, and the meaning of the underlying action was well understood. Most of the users found that the Web display of the annotations could be much more friendly, and that the Swimpomuk panels are a bit complex.
Getting contextual help when plugging a USB peripheral	50% of the users plugged in their printer, 40% a digital camera, and 10% a scanner. The panel popped up for all them. However, it contained no data for 40% of them (due to no reference in the database). It contained relevant information for 60% of them. Suggestions related to that service mostly revolved around the facts that the USB peripheral related pop-up panel should be configurable, so that it shows up only after the first plug, and that more accurate data should be provided.

Annotating an expert using the Web UI	All users managed to find the page correspoding to Kim and to annotate it as expected. All of them considered the user interface could be much improved, though, rating it "C" in average. Suggestions include: an improvement the semantic search engine user interface, a better performance of the search service,
Sharing private resources with others	The social service started successfully for all users. Only 30% of them could enter a relevant query though, since the others were not familiar enough with the SPARQL language, hence rating quite poorly the service. The service got an average mark of "C". Suggestions include: the need to get a visual user interface for creating queries, and the need for a more advanced rights management service, so that desktop's items can be shared only with specific people or groups.

Table 3: Evaluation results

From informal discussions with the users, it appears that the most appreciated features of the prototype were in general the following:

- the Swimpomuk metadata panel for browsing and adding annotations to resources,
- the trust level that can be attributed to experts,
- the panel that pops up when plugging in a USB peripheral,
- the possibility to distinguish between public annotations and private ones that remain on one's computer.

In terms of usability, the Swimpomuk user interface was globally well appreciated, while the Web user interface was considered as quite complex to use at this stage. Further efforts will be devoted to making it more convenient in the upcoming SWIM versions.

6 Exploitation

The annotation capabilities of SWIM, enabling a semantic interlinking between bugs, issues, discussions, documentation and people, tackles a problem that is more and more pointed out as crucial in open-source software engineering, as illustrated by the quotes below.

- Henry Story, Semantic Web evangelist and creator of the Baetle ontology:

  > *Open source software is creating a global software space, with dependencies between projects, is meshing software from many different sources. But we are not meshing the data about the software!*

- An interview of Mark Shuttleworth, the creator of the Ubuntu Linux project, who [15]

  > *We can't expect free software developers to agree on a single bug or patch tracking program, but we can create secure open APIs (application programming interfaces) between Bugzillas. These, Shuttleworth called "Porous federated containers."*

- A freelance Linux consultant wrote a blog titled "When bugtracking systems are fences, not bridges" [16] emphasizing the need for connecting the bug databases:

  > *So there is a lot we, as a community, can improve by connecting users to upstream, or rather, by connecting bug tracking systems to upstream for the benefit of all.*

- A discussion started January 29, 2009 on Slashdot [17]:

  > *Submitting bug reports – and waiting for responses etc. – seems to be SOP for developers and users alike, these days. Every project has some sort of bug-tracker – bugzilla, trac, mailing list, etc. E.g., we currently track 200+ external bugs across 40 OSS projects. Half the bugs depend on something else getting fixed, first. Every bug has its own email thread, etc. Management asks "How we doin' overall?," and suddenly everyone involved gets to work removing dried gum from the bottom of their shoe. What do Slashdotters use/recommend for centrally keeping track of all the bugs you track across all those different bugtrackers? In particular, managing communications and dependencies across bugs?*

The effort consisting in harnessing the Semantic Web principles for improving community based software engineering is still at its infancy. The SWIM prototype counts among the first steps in the software industry toward Semantic Web based software engineering. Exploitation of the prototype will first consist in globally improving the connection between Linux bug trackers, discussions and people. This is likely to be useful to all distributions communities, and in particular to the Mandriva Linux ecosystem.

Second, the market of help desks with social capabilities able to harness an in depth integration with a semantic knowledge base is emerging. The commercial exploitation of the prototype will consist in packaging SWIM as an enterprise collaborative annotation system, around which Mandriva considers offering a range of professional services.

[15] http://www.linuxformat.co.uk/modules.php?op=modload&name=Sections&file=index&req=viewarticle&artid=17

[16] http://dag.wieers.com/blog/when-bugtracking-systems-are-fences-not-bridges

[17] http://ask.slashdot.org/article.pl?sid=09/01/28/218259

7 Conclusion

SWIM is a community help desk featuring tight integration with the NEPOMUK-KDE semantic desktop. It provides a wide range of functionalities and a strong added value to the community of Mandriva Linux users. By letting users annotate collectively various data sources related to the Linux use and development processes, SWIM represents a novel approach for collective management of knowledge.

By using SWIM, users confirmed they are benefiting from NEPOMUK possibilities in the sense that they can create a personal semantic Web on their desktop indeed, and interlink it with the public semantic knowledge maintained by the Mandriva Linux community on the SWIM server. Users consider that this infrastructure improves their issue solving and learning processes, which was from the outset the most outstanding goal of the NEPOMUK Mandriva case study.

Mandriva expects that the combination of Web 2.0 community driven approach with Semantic Web principles tightly integrated into the Linux desktop will lead to a rich eco-system of content providers and consumers around the Mandriva Linux platform, thus easing the use and the dissemination of the Mandriva products. As of January 2009, SWIM is an additional service of the Mandriva Club platform, which is one of the business line of Mandriva. Mandriva will consider deriving from SWIM commercial offers targeting the enterprises by delivering products aiming at enhancing the efficiency of technical support teams. The SWIM approach could also be extended to other specific areas by replacing the LINO ontology with other ones dedicated to the target domains.

8 Appendix

8.1 Linux Issue Ontology

```
@prefix bom:     <http://www.ifi.unizh.ch/ddis/evoont/2008/11/bom#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix nao:     <http://www.semanticdesktop.org/ontologies/2007/08/15/nao#> .
@prefix nrl:     <http://www.semanticdesktop.org/ontologies/2007/08/15/nrl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix pimo:    <http://www.semanticdesktop.org/ontologies/2007/11/01/pimo#>
.
@prefix nco:     <http://www.semanticdesktop.org/ontologies/2007/03/22/nco#> .
@prefix lino:    <http://nepomuk.kde.org/ontologies/2008/11/25/lino#> .
@prefix wf:      <http://www.w3.org/2005/01/wf/flow#> .

<http://nepomuk.kde.org/ontologies/2008/11/25/lino> {
    # some improvements for WF and BOM
    bom:description
        rdfs:subPropertyOf nao:description .

    wf:state
        nrl:maxCardinality "1"^^xsd:nonNegativeInteger .

    # this is missing from bom
    lino:hasResolution
        a rdf:Property ;
        rdfs:domain bom:Issue ;
        rdfs:range bom:Resolution ;
        rdfs:label "resolution" ;
        rdfs:comment "The resolution of an issue. Only such issues that have
state Resolved, Verified, or Closed can have a resolution." .

    lino:Package
        a rdfs:Class ;
        rdfs:subClassOf bom:Product ;
        rdfs:comment "A software package, such as an rpm or a deb." ;
        rdfs:label "package" .

    lino:Software
        a rdfs:Class ;
        rdfs:subClassOf bom:Component ;
        rdfs:label "software" ;
        rdfs:comment "A software such as KMail or ssh" .

    lino:relatedPackage
        a rdf:Property ;
        rdfs:domain rdfs:Resource ;
        rdfs:range lino:Package ;
        rdfs:label "package" ;
        rdfs:comment "An optional software package (rpm or deb or other)
providing the software that shows the bug." .

    lino:providesSoftware
        a rdf:Property ;
        rdfs:domain lino:Package ;
        rdfs:range lino:Software ;
        rdfs:label "provides software" ;
        rdfs:comment "The software a package provides. A package can provide
multiple softwares" ;
        nrl:minCardinality "1"^^xsd:nonNegativeInteger .

    lino:Bug
        a rdfs:Class ;
        rdfs:subClassOf bom:Issue ;
        rdfs:label "bug" ;
        rdfs:comment "Represents the actual bug which as an extension to bom:
Issue." .

    lino:ForumPost
```

```
        a rdfs:Class ;
        rdfs:subClassOf pimo:Thing ;
        rdfs:label "forum post" ;
        rdfs:comment "Represents a post in a web forum." .

    lino:ForumThread
        a rdfs:Class ;
        rdfs:subClassOf pimo:Thing ;
        rdfs:subClassOf lino:Discussion;
        rdfs:label "forum thread" ;
        rdfs:comment "Represents a thread in a web forum." .

    lino:belongsToThread
        a rdf:Property ;
        rdfs:label "belongs to thread" ;
        rdfs:domain lino:ForumPost ;
        rdfs:range lino:ForumThread .

    lino:report
        a rdf:Property ;
        rdfs:subPropertyOf pimo:occurence ;
        rdfs:label "report" ;
        rdfs:comment "A lino:Bug can have multiple bug reports. A simple
indicator for matching bug reports is bom:fixes" ;
        rdfs:domain lino:Bug ;
        rdfs:range bom:Issue .

    lino:relatedBug
        a rdf:Property ;
        rdfs:subPropertyOf pimo:isRelated ;
        rdfs:label "related bug" ;
        rdfs:comment "relates a forum thread or a bug to another bug." ;
        rdfs:domain lino:Bug ;
        rdfs:domain lino:ForumThread ;
        rdfs:range lino:Bug .

    lino:relatedThread
        a rdf:Property ;
        rdfs:subPropertyOf pimo:isRelated ;
        rdfs:label "related thread" ;
        rdfs:comment "relates something to a forum thread." ;
        rdfs:domain rdfs:Resource ;
        rdfs:range lino:ForumThread .


    lino:operatingSystem
        a rdf:Property ;
        rdfs:comment "The operating system the software exhibiting the bug was
 running under." ;
        rdfs:domain bom:ComputerSystem ;
        rdfs:label "operatingSystem" ;
        rdfs:range lino:OperatingSystem .

    lino:isRelatedToOS
        a rdf:Property ;
        rdfs:label "is related to OS" ;
        rdfs:comment "relates to an operating system" ;
        rdfs:domain lino:Bug ;
        rdfs:domain lino:ForumThread ;
        rdfs:range lino:OperatingSystem .

    lino:software
        a rdf:Property ;
        rdfs:label "is related to software" ;
        rdfs:comment "relates to a software package" ;
        rdfs:domain lino:Bug ;
        rdfs:domain lino:ForumThread ;
        rdfs:domain lino:Documentation ;
        rdfs:range lino:Software .

    lino:architecture
```

```
        a rdf:Property ;
        rdfs:comment "The underlying architecture of the bug reporter's system
." ;
        rdfs:domain bom:ComputerSystem ;
        rdfs:label "architecture" ;
        rdfs:range lino:Architecture .

    lino:Architecture
        a rdfs:Class ;
        rdfs:label "Architecture" ;
        rdfs:subClassOf rdfs:Resource .

    lino:OperatingSystem
        a rdfs:Class ;
        rdfs:label "OperatingSystem" ;
        rdfs:subClassOf rdfs:Thing .

    lino:BSD
        a rdfs:Class ;
        rdfs:label "BSD" ;
        rdfs:subClassOf lino:OperatingSystem .

    lino:MAC
        a rdfs:Class ;
        rdfs:label "MAC" ;
        rdfs:subClassOf lino:OperatingSystem .

    lino:Linux
        a rdfs:Class ;
        rdfs:label "Linux" ;
        rdfs:subClassOf lino:OperatingSystem .

    lino:Fedora
        a rdfs:Class ;
        rdfs:label "Fedora" ;
        rdfs:subClassOf lino:Linux .

    lino:NetBSD
        a rdfs:Class ;
        rdfs:label "NetBSD" ;
        rdfs:subClassOf lino:BSD .

    lino:Redhat
        a rdfs:Class ;
        rdfs:label "Redhat" ;
        rdfs:subClassOf lino:Linux .

    lino:FreeBSD
        a rdfs:Class ;
        rdfs:label "FreeBSD" ;
        rdfs:subClassOf lino:BSD .

    lino:Windows
        a rdfs:Class ;
        rdfs:label "Windows" ;
        rdfs:subClassOf lino:OperatingSystem .

    lino:Mandriva
        a rdfs:Class ;
        rdfs:label "Mandriva" ;
        rdfs:subClassOf lino:Linux .

    lino:OpenSuSE
        a rdfs:Class ;
        rdfs:label "OpenSuSE" ;
        rdfs:subClassOf lino:Linux .

    lino:Slackware
        a rdfs:Class ;
        rdfs:label "Slackware" ;
        rdfs:subClassOf lino:Linux .
```

```
    lino:Hardware
        a rdfs:Class ;
        rdfs:label "Hardware" .

    lino:Manufacturer
        a rdfs:Class ;
        rdfs:label "manufacturer" ;
        rdfs:comment "The manufacturer of a piece of hardware" .

    lino:hardwareManufacturer
        a rdf:Property ;
        rdfs:comment "The manufacturer of a piece of hardware" ;
        rdfs:domain lino:Hardware ;
        rdfs:label "manufacturer" ;
        rdfs:range lino:Manufacturer .

    lino:hardware
        a rdf:Property ;
        rdfs:subPropertyOf pimo:isRelated ;
        rdfs:comment "An optional hardware component the bug report refers to
." ;
        rdfs:domain bom:Issue ;
        rdfs:label "hardware" ;
        rdfs:range lino:Hardware .

    lino:version
        a rdf:Property ;
        rdfs:comment "The version string of a software or an operating system.
 Example: 2.01 or 1.2.9" ;
        rdfs:domain lino:Hardware, lino:OperatingSystem, lino:Package, lino:
Software ;
        rdfs:label "version" ;
        rdfs:range rdfs:Literal .

    lino:bugzilla
        a rdf:Property ;
        rdfs:comment "Link to the original bug report on a bug reporting
system." ;
        rdfs:domain bom:Issue ;
        rdfs:label "bugzilla" ;
        rdfs:range rdfs:Resource .

    lino:bugSeverity
        a rdf:Property ;
        rdfs:comment "The severity of the bug. This includes blocking,
critical, grave, major, crash, normal, enhancement, wishlist, minor and
trivial" ;
        rdfs:domain bom:Issue ;
        rdfs:label "bug severity" ;
        rdfs:range lino:Severity .

    lino:Severity
        a rdfs:Class ;
        rdfs:label "severity" ;
        rdfs:comment "The severity of a bug. Use predefined instances only." .

    lino:Crash
        a lino:Severity ;
        rdfs:label "crash" .

    lino:Normal
        a lino:Severity ;
        rdfs:label "normal" .

    lino:Wishlist
        a lino:Severity ;
        rdfs:label "wishlist" .

    lino:Grave
        a lino:Severity ;
        rdfs:label "grave" .
```

```
    lino:Blocking
        a lino:Severity ;
        rdfs:label "blocking" .

    lino:Critical
        a lino:Severity ;
        rdfs:label "critical" .

    lino:Enhancement
        a lino:Severity ;
        rdfs:label "enhancement" .

    lino:High
        a lino:Severity ;
        rdfs:label "high" .

    lino:Low
        a lino:Severity ;
        rdfs:label "low" .

    lino:Major
        a lino:Severity ;
        rdfs:label "Major" .

    lino:Minor
        a lino:Severity ;
        rdfs:label "Minor" .

    lino:Wishlist
        a lino:Severity ;
        rdfs:label "wishlist" .

    lino:Trivial
        a lino:Severity ;
        rdfs:label "trivial" .

    lino:Unconfirmed
        a wf:State ;
        rdfs:label "Unconfirmed" ;
        rdfs:comment "This bug has recently been added to the database. Nobody
 has validated that this bug is true. Users who have the 'canconfirm'
permission set may confirm this bug, changing its state to NEW. Or, it may be
directly resolved and marked RESOLVED." .

    lino:New
        a wf:State ;
        rdfs:label "New" ;
        rdfs:comment "This bug has recently been added to the assignee's list
of bugs and must be processed. Bugs in this state may be accepted, and become
ASSIGNED, passed on to someone else, and remain NEW, or resolved and marked
RESOLVED." .

    lino:Assigned
        a wf:State ;
        rdfs:label "Assigned" ;
        rdfs:comment "This bug is not yet resolved, but is assigned to the
proper person. From here bugs can be given to another person and become NEW,
or resolved and become RESOLVED." .

    lino:Reopened
        a wf:State ;
        rdfs:label "Reopened" ;
        rdfs:comment "This bug was once resolved, but the resolution was
deemed incorrect. For example, a WORKSFORME bug is REOPENED when more
information shows up and the bug is now reproducible. From here bugs are
either marked ASSIGNED or RESOLVED." .

    lino:Resolved
        a wf:State ;
        rdfs:label "Resolved" ;
        rdfs:comment "A resolution has been taken, and it is awaiting
verification by QA. From here bugs are either re-opened and become REOPENED,
are marked VERIFIED, or are closed for good and marked CLOSED." .
```

```
    lino:Verified
        a wf:State ;
        rdfs:label "Verified" ;
        rdfs:comment "QA has looked at the bug and the resolution and agrees
that the appropriate resolution has been taken. Bugs remain in this state
until the product they were reported against actually ships, at which point
they become CLOSED." .

    lino:Closed
        a wf:State ;
        rdfs:label "Closed" ;
        rdfs:comment "The bug is considered dead, the resolution is correct.
Any zombie bugs who choose to walk the earth again must do so by becoming
REOPENED." .

    lino:x86
        a lino:Architecture ;
        rdfs:label "X86" .

    lino:x86_64
        a lino:Architecture ;
        rdfs:label "X86_64" .

    lino:DebianTesting
        a lino:Debian ;
        rdfs:label "Debian Testing" ;
        lino:version "Testing" .

    lino:Mandriva2008
        a lino:Mandriva ;
        rdfs:label "Mandriva 2008" ;
        lino:version "2008.0" .

    lino:Mandriva2008_1
        a lino:Mandriva ;
        rdfs:label "Mandriva 2008 Spring" ;
        lino:version "2008.1" .

    lino:Mandriva2009
        a lino:Mandriva ;
        rdfs:label "Mandriva 2009" ;
        lino:version "2009.0" .

    lino:Mandriva2009_1
        a lino:Mandriva ;
        rdfs:label "Mandriva 2009 Spring" ;
        lino:version "2009.1" .

    lino:hasTrustLevel
        a rdf:Property ;
        rdfs:label "has trust level" ;
        rdfs:domain nao:party ;
        rdfs:range lino:TrustLevel .

    lino:TrustLevel
        a rdfs:Class;
        rdfs:comment "A trust level, like in Advogato.org" ;
        rdfs:label "trust level" .

    lino:Master
        a lino:TrustLevel;
        rdfs:label "master" .

    lino:Journeyer
        a  lino:TrustLevel;
        rdfs:label "journeyer" .

    lino:Apprentice
        a lino:TrustLevel;
        rdfs:label "apprentice" .

    lino:Observer
```

```
      a lino:TrustLevel;
      rdfs:label "observer" .

   lino:Documentation
      a rdfs:Class;
      rdfs:label "documentation" .

   lino:FAQ
      a rdfs:Class;
      rdfs:subClassOf lino:Documentation .

   lino:Guide
      a rdfs:Class;
      rdfs:subClassOf lino:Documentation .

   lino:HowTo
      a rdfs:Class;
      rdfs:subClassOf lino:Documentation .

   lino:Discussion
      a rdfs:Class .

   lino:Driver
      a lino:Software ;
      rdfs:label "driver" .

   lino:WorkLevel
      a rdfs:Class ;
      rdfs:label "work level" .

   lino:WorksOutOfTheBox
      a lino:WorkLevel ;
      rdfs:label "works out of the box" .

   lino:WorksWithModification
      a lino:WorkLevel ;
      rdfs:label "works with modification" .

   lino:NeutralWorkLevel
      a lino:WorkLevel ;
      rdfs:label "don't know or not supported by hardware" .

   lino:WorksPartially
      a lino:WorkLevel ;
      rdfs:label "works partially" .

   lino:WorksNot
      a lino:WorkLevel ;
      rdfs:label "does not work" .

   lino:hasWorkLevel
      a rdf:Property ;
      rdfs:comment "The compatibility level of the package for the
corresponding hardware/operating system." ;
      rdfs:domain lino:Package ;
      rdfs:range lino:WorkLevel .

   lino:downloadURL
      a rdf:Property ;
      rdfs:comment "The address the package can be downloaded from." ;
      rdfs:label "has download address" ;
      rdfs:domain lino:Software;
      rdfs:range  rdfs:Resource .

   lino:forOperatingSystem
      a rdf:Property ;
      rdfs:comment "The operating system the package is for." ;
      rdfs:label "for operating system" ;
      rdfs:domain lino:Package;
      rdfs:range  lino:OperatingSystem .
```

```
     lino:forHardware
        a rdf:Property;
        rdfs:comment "The hardware this driver is for." ;
        rdfs:label "for hardware" ;
        rdfs:domain lino:Driver ;
        rdfs:range  lino:Hardware .

}

<http://nepomuk.kde.org/ontologies/2008/11/25/lino/metadata> {
    <http://nepomuk.kde.org/ontologies/2008/11/25/lino/metadata>
          a       nrl:GraphMetadata ;
          nrl:coreGraphMetadataFor <http://nepomuk.kde.org/ontologies
/2008/11/25/lino> .


    <http://nepomuk.kde.org/ontologies/2008/11/25/lino>
          a       nrl:Ontology , nrl:KnowledgeBase, nrl:DocumentGraph ;
          nao:hasDefaultNamespace "http://nepomuk.kde.org/ontologies
/2008/11/25/lino#" ;
          nao:hasDefaultNamespaceAbbreviation "lino" ;
          nao:lastModified "2008-11-25T18:34:34.881Z" ;
          nao:serializationLanguage "TriG" ;
          nao:status "Unstable" ;
          nrl:updatable "0" ;
          nao:version "1" .
}
```

References

[1] Stéphane Laurière, Alexandre Solleiro, Sebastian Trüg, Cristian Bogdan, Kristina Groth, and Pär Lannero. Nepomuk Deliverable D11.1 – Mandriva Community Nepomuk Scenario Report. http://nepomuk.semanticdesktop.org/xwiki/bin/Main1/D11-1, 2006.

[2] Leo Sauermann and Enrico Minack. Nepomuk Deliverable D2.1 - Adapters, Extractors, and Knowledge Structure Services. http://nepomuk.semanticdesktop.org, 2006.

[3] University of Zurich. EvoOnt – A Software Evolution Ontology. http://www.ifi.uzh.ch/ddis/msr/.

[4] Gerald Reif, Tudor Groza, Siegfried Handschuh, Medhi Jazayeri, and Cédric Mesnage. Nepomuk Deliverable D6.2A – Intermediate Nepomuk Architecture. http://nepomuk.semanticdesktop.org/xwiki/bin/Main1/D6-2A, 2007.

[5] Ludger van Elst Simon Scerri, Michael Sintek and Siegfried Handschuh. Nepomuk Annotation Ontology Specification. http://www.semanticdesktop.org/ontologies/nao/, 2007.

[6] Ludger van Elst Simon Scerri, Michael Sintek and Siegfried Handschuh. Nepomuk Rrepresentation Language. http://www.semanticdesktop.org/ontologies/nrl/, 2007.