The FLOSSWALD Information System on Free and Open Source Software

Alexandre Hanft
Intelligent Information Systems
Lab, University of Hildesheim
alexandre.hanft@uni-hildesheim.de

Meike Reichle
Intelligent Information Systems
Lab, University of Hildesheim
meike.reichle@uni-hildesheim.de

Abstract. *We propose the implementation of an intelligent information system on free and open source software. For a first prototype we model the knowledge base and the case format used for case-based retrieval. Based on information from the Debian Project package repository and open source software directories we intend to investigate means of mapping the provided technical information on vague attributes that are more intuitive to users who are unexperienced in using open source software. Thereby, we intend to build a system that will be able to give intelligent software recommendations depending on more intuitive specifications provided by the user.*

1 Introduction

Free and open source software has produced a large and diverse range of software which often offers numerous alternatives for the same task, such as text editors, e-mail clients or web browsers. Also because of this existing project descriptions are mostly technically phrased and focus on the project's technological features.

However, in order to choose from a range of available software especially less experienced computer users mainly ask for qualitative attributes such as usability, stability and an agreeable look. Already existing software directories also offer mainly technically oriented search possibilities. What's missing here is the link between the user's qualitative expectations and the technical attributes of a project.

We intend to learn translating these qualitative attributes into a set of technical features. Our plan is to design and implement an information system on free and open source software, FLOSSWALD, that offers searches by technical as well as qualitative attributes. The system's knowledge base will consist of software descriptions, improved with tags and user feedback on the results.

First, we illustrate our motivation to launch FLOSSWALD. Section 2 introduces the idea behind FLOSSWALD, including a first analysis of the available data sources. Section 3 presents related work that we have examined in the course of the project's creation. This paper closes with a conclusion and an outlook in section 4.

1.1 Motivation

The Free and Open Source Software Community is a complex social and technical network that consists of thousands of individual groups and projects that produce software

in all degrees of quality, from low class to very high grade. But most of it is only known to its users, other insiders or those who know where and how to look for it. While the popularity of free and open source software is rising, a growing number of less-experienced computer users who have just recently begun using some of the most popular free and open source software such as the Firefox browser or the Thunderbird e-mail client are now considering to exchange also other programs for a free and open source alternative. These computer users usually only know very few isolated projects and don't know about the actual free and open software scene.

This rising interest in free and open source software also creates a new need for information on free and open source software projects and their nature and quality. However, knowledge about this topic is still very much restricted to insiders who are themselves active in the Free and Open Source Software Community. While simple technical questions such as "What database does application XY use?” are sufficiently easy to answer using e.g. web search machines, more vague questions such as “What is the right e-mail client for me” or “Which GNU/Linux distribution should I put on my small company’s webserver?” are a much harder task. Such questions don’t only need technical information but also meta information such as how old or established a project is, how many users or developers it has, how mature its code base is, or how reliable it is to still be around in a few years.

Existing information services such as Freshmeat or Sourceforge rely heavily on technical criteria and language and are thus of only small use to inexperienced users. As a consequence, we plan to implement an intelligent information system that meets this need by offering more intuitive search criteria and intelligent search tools based on user preferences and learned similarities between software or user groups.

2 The FLOSSWALD Project

First we introduce the project and describe the investigated data sources for our knowledge base: Debian packages, DebTags, Debian changelogs, the Debian bug tracking system and the FLOSSmole project The last part of this section presents the planned implementation as an instantiation of a more general framework for knowledge-based systems.

2.1 Concept

FLOSSWALD, the Free/Libre Open Source[1] SoftWare and AppLication Directory, is a project proposal that aims to use a case-based reasoning system that includes information about the individual softwares in its knowledge base. We decided to us a case-based system, because we are dealing with vague criteria and use a large collection of individual information entities. The system is further equipped with several machine learning components that are meant to improve system performance by creating additional knowledge in the form of concepts, e.g. user groups or similarities (such as "of a similar kind" or "do the same task"), from the provided data.

2.2 The Knowledge Base

To build up the knowledge base we include data from the Debian GNU/Linux package repository and the FLOSSMOLE project [Howison et al. 2006] that provides raw data, mined at Freshmeat, Sourceforge, ObjectWeb and Rubyforge. The Debian Project

Debian GNU/Linux is a free operating system developed by more than a thousand volunteer developers and many more contributors such as package maintainers, translators and documentation writers all over the world, who collaborate mainly via the Internet. It has, due to its age and popularity, probably the largest selection of prepackaged free and open source software of all GNU/Linux distributions. Its upcoming release will include17,583 binary packages[2] and 10,228 source packages. And, what's most important, all of these packages come with a textual description, which

[1] In this paper we use the term "free/libre and open source software" in order to include both, the Open Source and the Free Software community.

[2] http://packages.debian.org

```
Package: 3dchess
Priority: optional
Section: games
Installed-Size: 136
Maintainer: Debian QA Group <packages@qa.debian.org>
Architecture: i386
Version: 0.8.1-12
Depends: libc6 (>= 2.3.6-6), libx11-6, libxext6, libxmu6,
libxpm4, libxt6, xaw3dg (>= 1.5+E-1)
Filename: pool/main/3/3dchess/3dchess_0.8.1-12_i386.deb
Size: 33564
MD5sum: fecee217870b621286f75e528496d3b1
SHA1: 88343e19f566cf5cd11ef099bad97fbabf4e316d
SHA256: 3601709708044f7e489a0a74dbe4aca0e04b2fe1bc
533655b268af36fb6abd2c
Description: 3D chess for X11
3 dimensional Chess game for X11R6. There are three boards, stacked
vertically; 96 pieces of which most are the traditional chess pieces with
just a couple of additions; 26 possible directions in which to move. The AI
isn't wonderful, but provides a challenging enough game to all but the most
highly skilled players
Tag: game::board::chess, interface::3d, use::gameplaying, x11::application
```

already offers a great wealth of information about a certain software. The textual description can be analyzed with different information retrieval tools, extracting important terms or finding similar descriptions in other packages. The package's size, dependencies, section and priority can also provide conclusions on its suitability for an existing system, its nature or purpose. Additionally to this, the Debian package repository offers several other sources of information.

DebTags [Zini 2005] is a project started by Enrico Zini. Those tags are shown alongside package descriptions where available and give meta information on the software. The tags include numerous different ontologies representing different "perspectives" such as what the software is used for, what interfaces are used, what role the software has (server, client), its programming language, used protocols and many others. DebTags are in a machine readable format and thus allow for smarter search and navigation interfaces than the original full text Debian package search.

Debian further collects detailed anonymised usage data on its packages. Debian GNU/Linux users can voluntarily install a program called *popularity contest*[2] (popcon) that sens anonymised reports to the Debian

[2] http://popcon.debian.org/

project indicating what packages are installed on a user’s system and when they have been used the last time. These data allow a first take on e.g. the popularity of a particular software.

Debian Changelogs and the bug tracking system[3] can give information on the up-to-dateness and stability of a package.

The FLOSSmole Project

The FLOSSmole (formerly OSSmole) project is a collaborative project.

> *[...] designed to gather, share and store comparable data on and analyses of free and open source software development for academic research.* [Howison et al. 2006, S.1]

Additional analysis of the FLOSSMOLE data and the adequacy of these data sources has been done in [Reichle and Hanft 2006].

The Case-Format

In order to allow a case-based retrieval we model a first prototype with myCBR [myCBR], an open source plugin for the OWL Editor Protégé. In order to include all available data, we combine our data sources, namely the Debian package descriptions and the data mined by the flossmole project. We remove unnecessary slots and map correspondent ones. This is necessary, since some slots, such as a textual description of the project or the software's main task are given by several sources and many projects are

[3]http://bugs.debian.org/

covered by different data sources. Additionally we introduce a project ID for all software descriptions and do some normalisation, such as outsuorcing frequent terms e.g. licenses.

Also, we decided to separate our case base in two different parts, one holding general project information such as a project's description or license, the other holding qualitative data such as a software's priority, vitality or popularity. We do this because we consider these data to be entirely different. The information provided in the first part (fig. 1) can be considered to be universally valid while the data in the second part (fig. 2) are subject to constant change and cannot be assumed correct under all circumstances. Because of this we intend to only use the data from the first part for retrieval while the data from the second part are at the moment only used as additional information when displaying the according results and possibly for ordering them. However, in the next development stage, when we introduce vague attributes, these subjective data may prove useful. It needs to be investigated, whether e.g. a high popularity may be taken as an indication for any vague attributes e.g. "easy to use". An example for a case format can be seen in fig. 1.

```
<?xml version="1.0" encoding="UTF-8"?> <Document>
<Instances_for_Class class="Project">
<Prototype>
<slotvalue slot="project_id" value="Integer" minval="1" />
<slotvalue slot="projectname_short" value="String" />
<slotvalue slot="license_id" value="integer" />
<slotvalue slot="url_project_page" value="String" />
<slotvalue slot="real_url_homepage" value="String" />
<slotvalue slot="subscriptions" value="Integer" minval="0" />
<slotvalue slot="Section" value="Symbol" />
<slotvalue slot="Architecture" value="Symbol" />
<slotvalue slot="Source" value="String" />
<slotvalue slot="Depends" value="String" />
<slotvalue slot="Installed-Size" value="Integer" minval="0" />
<slotvalue slot="desc_full" value="String" />
```

Figure 1: The package description of a Debian package

```
<Instances_for_Class class="Project_qualitative_data">
<Prototype>
  <slotvalue slot="Priority" value="Symbol" />
  <slotvalue slot="project_id" value="Integer" minval="1" />
  <slotvalue slot="rating" value="Float" minval="0.0" />
  <slotvalue slot="rating_count" value="Integer" minval="0" />
  <slotvalue slot="rating_rank" value="Integer" minval="1" />
  <slotvalue slot="vitality_score" value="Float" minval="0.0"
/>
  <slotvalue slot="vitality_percent" value="Float" minval="0.0"
/>
  <slotvalue slot="vitality_rank" value="Integer" minval="1" />
  <slotvalue slot="popularity_percent" value="Float"
minval="0.0" />
  <slotvalue slot="popularity_score" value="Float" />
```

Figure 2: The package description of a Debian package

2.3 Integration into a Knowledge-Based System

Due to the heterogeneity and the large amount of free and open source software, it seems to be appropriate to use case-based reasoning to give the user advice according to their vague descriptions and (soft) constraints. In such a case-based reasoning system, each software project should be represented by an individual case. Additionally, users should have the option to tag already known software with freely chosen tags to enrich the software descriptions with attributes that conform better with most users' level of abstraction than the mostly technical information we get from the afore mentioned sources. To finally create the "link" between more formal project descriptions and the vague descriptions from the users' perspectives we plan to use machine learning technology.

[Althoff et al. 2006] presented a framework for knowledge-based systems (KBS) that appears to be promising for realising FLOSSWALD. The integrationhas been sketched up more detailed in [Reichle and Hanft 2006].

3 Related work

Existing information systems on free and open source software include Freshmeat, GNU Savannah, Berlios and Sourceforge. Many of these directories (e.g. Savannah, Berlios and Sourceforge) also offer development infrastructure such as web space, mailing lists, version control systems, bug tracking systems, or wikis. Those directories thus only include information on projects they also host. This has led to a certain redundancy since many projects registered with e.g. Sourceforge, so they are listed there but do not use the provided infrastructure but their own. Other software directories have been discussed in [Reichle and Hanft 2006].

Althoff et al. 1999 have created an information system that is designed as an experience factory and holds information on CBR technology and tools. This system is also based on a CBR system that can be queried and updated over a web interface. This system gave the original idea for the FLOSSWALD and we hope to be able to reuse some of the experiences made in the development of this system.

4 Conclusion & Outlook

We are confident that FLOSSWALD will be of great use to computer users new to free and open source software, who will most likely do vague searches, based on similarities or ratings as well as to experienced users who are searching with highly defined criteria such as required libraries or avoiding particular technologies or protocols. After evaluating the data provided by the Debian project and the FLOSSmole project and designing a knowledge base and case structure to flexibly work with them, our next step will be to implement a CBR system based on the knowledge-based systems framework that is able to deal with the provided information and define the particular components (such as the maintenance of the knowledge base) where machine learning modules can be used to improve the system's performance.

5 References

[Althoff et al. 1999] Althoff, K.-D., Nick, M., Tautz, C. (1999). CBR-PEB: An Application Implementing Reuse Concepts of the Experience Factory for the Transfer of CBR System Know-How. In Proceedings of the Seventh Workshop on Case-Based Reasoning during Expert Systems '99 (XPS-99), Wuerzburg, Germany.

[Althoff et al. 2006] Althoff, K.-D., Hanft, A. & Schaaf, M. (2006). Case Factory – Maintaining Experience to Learn. M. Gцker & T. Roth-Berghofer (eds.), Proc. 8th European Conference on Case-Based Reasoning (ECCBR'06), LNCS 4106. Springer Verlag. pp 429-442.

[Howison et al 2006] Howison, J., Conklin, M., Crowston, K. (2006). FLOSSmole: A Collaborative Repository for FLOSS Research Data and Analyses. International Journal of Information Technology and Web Engineering. 1(3). July-September, 2006. pp 17-26.

[myCBR] https://mycbr.opendfki.de, last visited on 22th of october 2006

[Newman et al 1998] Newman, D.J. & Hettich, S. & Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learning databases at http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA: University of California, Department of Information and Computer Science. last visited: 07/25/2006

[Reichle and Hanft 2006] Reichle, M. & Hanft, A.. The FLOSSWALD Information System on Free and Open Source Software. Workshop FGWM 2006. In Schaaf, M. & Althoff, K.-D.(eds) Proceedings of the LWA 2006. Hildesheimer Informatik-Berichte volume 1/2006, University of Hildesheim

[TREC 2005] The Fourteenth Text REtrieval Conference (TREC 2005) in Gaithersburg, Maryland, National Institute of Standards and

Technology (NIST), at http://trec.nist.gov/pubs/trec14/t14_proceedings.html. last visited: 07/25/2006

[Zini 2005] Zini, E. (2005). A cute introduction to Debtags at http://debtags.alioth.debian.org/paper-debtags.html. Last visited: 07/25/2006