Expertise Identification and Visualization from CVS

Omar Alonso Premkumar T. Devanbu Michael Gertz
Dept. of Computer Science
University of California at Davis
{oralonso, gertz, ptdevanbu}@ucdavis.edu

ABSTRACT
As software evolves over time, the identification of expertise becomes an important problem. Component ownership and team awareness of such ownership are signals of solid project. Ownership and ownership awareness are also issues in open-source software (OSS) projects. Indeed, the membership in OSS projects is dynamic with team members arriving and leaving. In large open source projects, specialists who know the system very well are considered experts. How can one identify the experts in a project by mining a particular repository like the source code? Have they gotten help from other people?

We provide an approach using classification of the source code tree as a path to derive the expertise of the committers. Because committers may get help from other people, we also retrieve their contributors. We also provide a visualization that helps to further explore the repository via committers and categories. We present a prototype implementation that describes our research using the Apache HTTP Web server project as a case study.

Categories and Subject Descriptors
D.2.8 [Software Engineering]: Metrics - Empirical, Open Source

General Terms: Design, Experimentation

Keywords
Classification, expertise identification, information visualization.

1. INTRODUCTION
A well-known question in software development is "who owns X?" or "who are the experts for Y?" In the context of open source, we define an expert as somebody who has contributed a significant number of transactions over time. While only a relatively small number of people are *developers*, who have commit privileges, many others actually *contribute* to the source code. We thus distinguish between *developers* and *contributors.*

In this paper, we present a mechanism for detecting expertise using a rule-based classification approach. We also examine the relationship of contributors to developers, and how much help the contributors actually provide. To discover this information we mine a CVS data source to find submissions and contributors per transaction. We argue that with a CVS logfile and a good high level description of the source code tree, it is possible to automatically identify expertise. There are several ways of identifying expertise in a particular domain given different information sources. In this work, we would like to identify as experts those developers who have a high number of transactions over a period of time. The overall quality of those transactions and their relationship with other sources like a bug database are out of the scope of this paper.

Recently, exploratory search systems have emerged as a specialization of information exploration to support serendipity, learning, and investigation of large data sets [16]. In this paper, we provide an exploratory tool that allows the examination of the data about expertise and contribution to open-source projects in more detail. By now, it has been well established (*e.g.,* in the Apache project) that there is a relatively small group of developers who actually make those changes. This would indicate that the structure is very similar to a traditional industrial development team. But in fact, there are others who contribute source code, bug fixes, patches etc. Their contributions play a significant role in the success of these projects. Naturally, several questions arise. Who are the experts? Are they in the hundreds?

Research in mining software repositories has been very active lately with many projects working beyond just source code. Email and CVS sources contain rich data for a wide range of analysis [12], [13], [14]. The social aspect is also an important component of the mining process like the identification of active participants, owners in a project, and overall structure of a team [4], [6], and [10]. There are a number of projects about CVS visualization and developers evolution [5], [8], [9], [15]. A project that mixes expertise identification via clustering and visualization in an enterprise setting is presented in [3].

2. DATA PREPARATION
We use a database-driven approach framework for the analysis and exploration of software repositories [1]. This framework provides database and mining techniques for the integration, processing, analysis, and management of different types of open source repository data.

2.1 CVS characteristics
A CVS log file usually has a field where a developer is expected to enter detailed information about the transaction, such as which bug was fixed (if there is one open), who submitted the patch (if there is a submission) and who has reviewed it. From the data characteristic perspective, it is *semi-structured data*: good structure for some items (like author, file name) and unstructured for the messages and comments part, as shown in this example:

MSR'08, May 10–11, 2008, Leipzig, Germany.

```
<entry><date>2004-08-11</date>
<weekday>Wednesday</weekday>
<time>15:44</time><author>wrowe</author>
<file><name>modules/proxy/proxy_ajp.c</name>
<revision>1.6</revision></file>
<msg>Close only when needed.
Submitted by: jfclere</msg></entry>
```

In practice, developers do not always enter all the elements that are expected; the mining tools must be robust in cases where data is missing. However, when available, this is potentially very useful information: the identity of the person who is credited for a contribution to a particular commit. This data gives us useful information about the relationship between contributors and developers, and the effects of the contributor/committer relationship on the productivity of the committer. We can ask questions such as the following:

1. Given a CVS repository, can we automatically detect expertise?
2. Is there a temporal evolution with committers? Are they still active?

We provide answers to the above questions and present a preliminary analysis of the data in this respect.

2.2 Information Extraction and Mining

Given the structure of a textual message in CVS, using regular expressions we can identify some patterns and write simple extraction tasks. As long as there is some sort of formatting regularity, automated extraction is feasible [11]. Ideally, we expect to find the following although in practice not all the data is available.

```
<msg>message
PR:
Obtained from:
Submitted by:
Reviewed by:
</msg>
```

We can easily pull out names of contributors that provided fixes for bugs (identified as "PR") and populate parts of the database schema for later mining. The schema contains information about authors (developers), entries, files, bugs (PR), and contributors.

We obtained a dump of the CVS repository in XML that contains 15,589 entries between 1996 and 2004 for the Apache 2.0 release. A second step was to manipulate XML using the parsed SAX representation, to process the entire file and populate the database schema. The advantage is that one can use SQL, XPath, regular expressions or any combination of existing query languages to perform information extraction. The following script is an example of some rules for detecting a contributor in a message text. The query illustrates the convenience of being able to refer to relational and semi-structured data in a single query.

```
select extractValue(entry,'/entry/author') a,
   count( extractValue(entry,'/entry/msg') ) sub
from cvs_table where regexp_like(
extractValue(entry,'/entry/msg'),'Submitted by:
[a-zA-Z]+')
group by extractValue(entry,'/entry/author')
```

The following table shows, as an illustration, a few records with authors (developers who have a committer id), the number of entries in the CVS log file (transactions) and of those transactions how many have a "submitted by" comment.

AUTHOR	ENTRY	SUBMITTED
nd	1814	89
wrowe	1792	87
trawick	1634	30

With this information we can now retrieve all the names of contributors for a given developer. The CVS log file shows that there are 75 unique developers that account for 15,589 entries. In terms of assistance from other people, we were able to extract data regarding contributions for 75 developers who have entries in the CVS. Out of these, 39 gave no credit in their commit logs to contributors. The most credits were given by developer `nd` with 89 separate credits in total.

2.3 Source Code Directory Structure

In the case of Apache, the CVS repository has a clear directory structure and there is a README file that describes the source code layout in more detail. We can think of the layout as a classification scheme, and it can be very useful for categorizing the repository using a rule-based classifier. From the original layout document, we can derive the following list of categories and directory paths:

CATEGORY	DIRECTORY
Developer documentation	docs/manual/developer/
FAQ	docs/manual/faq/
How to documentation	docs/manual/howto/
Images	docs/manual/images/
Misc. documentation	docs/manual/misc/
Modules documentation	docs/manual/mod/
Platform documentation	docs/manual/platform/
Programs documentation	docs/manual/programs/
SSL Documentation	docs/manual/ssl/
Vhosts documentation	docs/manual/vhosts/
Authorization and auth	modules/aaa/
File and data caching	modules/cache/
WebDAV functionality	modules/dav/
Code in the early stages	modules/experimental/
General inline data fil	modules/filters/
Data generation functions	modules/generators/
Basic HTTP protocol impl	modules/http/
Logging functions	modules/loggers/
URL mapping and rewriting	modules/mappers/
Header metadata	modules/metadata/
Proxy module	modules/proxy/
OpenSSL functionality	modules/ssl/
Modules which test vari	modules/test/
OS Unix	os/unix/
OS Windows	os/win32/
Server MPM	server/mpm/
Perl Library	srclib/pcre/
Rudimentary command line	support/ab
Apache run-time Control	support/apachectl
APache eXtenSion tool	support/apxs

3. CLASSIFICATION

Classification is the task of assigning objects to one or more classes or categories. A well-known approach is to manually define topics or categories and then see if the documents' content matches them. In principle, this approach works well when the number of categories is small. For large collections, manual classification does not scale.

We are interested in deriving the categories that a developer has committed based on the code layout. Since we know the category for a directory path and each transaction has a directory path, it is possible to classify each entry according to the human *labeling*, like "Authorization and authentication" instead of "aaa". For such a classifier, what is needed is a category name (Authorization and authentication) and a query or rule that satisfies that category (location of `aaa` files in the source code structure).

The next step is to create a rule-based classifier on the category directories that we can use to classify transactions at the file name level according to the categories defined above. For example, the file `modules/aaa/mod_authnz_ldap.c` belongs to the category "Authorization and authentication". By classifying the file, we can derive the committer as well. One can argue that a commit does not necessary mean that a person is an expert. However, we classify all transactions so a higher *count* on certain categories, is a clear indication that the person is likely to be an expert.

The classifier was implemented using Oracle's `ctxrule` index that allows the creation of a rule-based classifier. Using the category and directory information, a table is created with an index on the rules (the directory structure). The classifier then queries the table using the `matches` operator, passing the file path as query and returns a label if there is a match. The following code fragment illustrates the classifier in more detail.

```
select all authors from CVS
for each author get all the files names
  if there is a match(category, file name)
     store (author, category) in classify table
  end if;
```

This rule-based classifier has a limitation like the lack of learning new categories and it behaves binary (yes/no classification). The final outcome is then a table that contains the counts (number of transactions) per year in each category. With this at hand, we can now visually explore the experts over time. Note that we are interested in detecting expertise (generalist, specialist) in areas, not necessary code ownership as indicated in [9].

4. VISUALIZATION AND EXPLORATION

In the context of large data sets, information visualization can help users navigate as well as provide new insights and summaries of the data collection. We introduce the *expertise cloud visualization* for exploring CVS.

4.1 Expertise cloud

We borrow the idea of a "tag cloud" from social websites to present an interface that summarizes the activity of the committer in a log file. A tag cloud is a visualization of user generated *tags* that describe the content of a website. In this context a tag is a relevant keyword or topic label that the user has entered. The importance of a tag is proportional to its font size.

Instead of users labeling the content, we take advantage of the data generated in the classification phase and define a tag as a committer's id/name or category label. We also can retrieve the number of transactions per committer, which makes the generation of the size font proportional to the overall transactions in the logfile. Figure 1 shows the expertise cloud with names in alphabetical order from left to right. A nice feature of the cloud is that one can scan it visually very quickly.

Our visualization furthermore allows exploring a single committer. The user can click on a name (say `aaron`) and the cloud now presents all categories the user has made contributions to. These are the expertise areas for a particular person (Figure 2). Also, next to the user name we see a temporal history of contributions over the years. Because involvement in a project varies over time, we encode the time variable in the color, where earlier activity is lighter blue and darker more recent. Our expertise cloud is a bivariate map. In other words, dark blue represents active committers, and color saturation decreases for those who haven't been active in a long time. This range is computed automatically based on number of transactions in the entire history of the logfile.

Figure 1. Cloud of committers

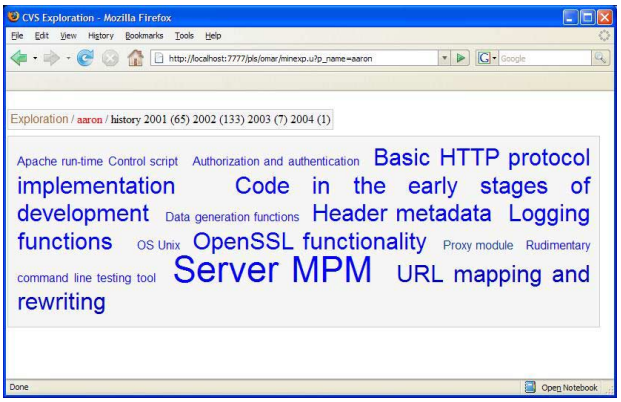

Figure 2. Areas of expertise by a committer

The visualization was implemented using a low-cost approach that performs extremely well: HTML and CSS. There is no plug-in required and it runs in every browser. Also, our visualization is not source code oriented as most of the work on CVS visualization.

5. DEMONSTRATION

Using the visualization tool for the analysis, we can see that a small group of developers perform all commits to the project. These people have been around for a number of years and are the owners of significant portions of the code. For example, just to name a few developers, `fielding`, `trawick`, `nd`, `clar`, `rbb`, `stoddard`, `slive`, `jerenkratz`, and `bnicholes` account for the bulk of transactions, and all of them have been working on the project for several years. This is consistent with the project meritocracy philosophy [7]. The visualization tool helps to get a

good idea of specialists and generalists. Figure 3 shows the activity of a committer who has been working mainly on documentation. On the other hand, as presented in Figure 4, generalists that have a lot of transactions tend to work on several pieces of the system. Still, they are main owners of certain pieces (server MPM as seen at the bottom of the image).

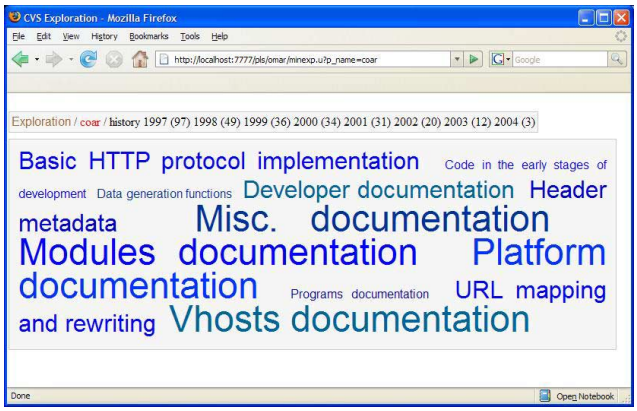

Figure 3. An example of a specialist

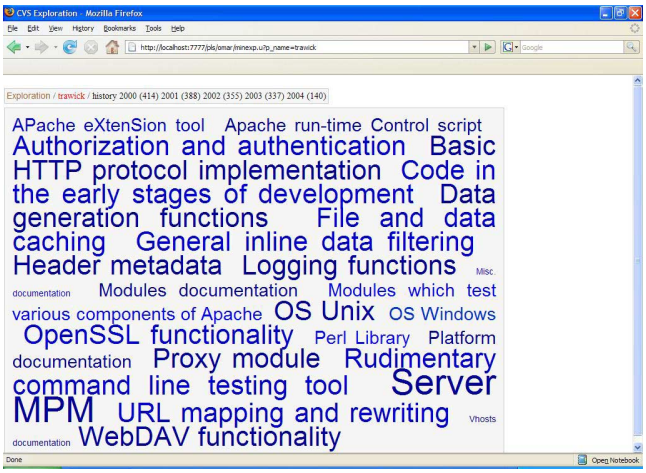

Figure 4. An example of a generalist

As we can see by exploring and navigating the cloud, a small group did over hundreds of transactions, while the rest has a very low activity. This is consistent with similar findings about individual author contributions on the Linux project reported in [12]. The structure indicates a similarity with the onion-like structure as proposed in [4]. Also, the time dimension (as expressed in the color scale of the text) shows that people are fairly active in the project. Finally, the tool provides a similar description of contributors and expert as outlined in the official Apache page [2]. This step was done by manually comparing the experts against the expertise description on the Apache project Web page.

6. CONCLUSIONS AND FUTURE WORK

Clearly, the CVS data source has rich information for discovering contributors and activities in the Apache project. Using information extraction and classification we were able to identify expertise and contributors through developers, which give important insight information about how the work is performed.

We presented an automated way for detecting expertise in a development team using open source as an example. The areas of expertise and the contributors who help are presented in a novel visualization that allows exploration and discovery of patterns. Future work includes using other classification and clustering techniques, a complete evaluation in terms of accuracy, and alternative visualization techniques for the same data set.

7. REFERENCES

[1] O. Alonso, P. Devanbu, and M. Gertz. "Database Techniques for the Analysis and Exploration of Software Repositories". *First MSR Workshop,* ICSE (2004).

[2] Apache contributors, httpd.apache.org/contributors

[3] J. Brunnert, O. Alonso, and D. Riehle. "Expertise People and Skill Discovery Using Tolerant Retrieval and Visualization". *In Proc. Of 30th ECIR*, (2007).

[4] K. Crowston and J. Howison. "The Social Structure of Free and Open Source Software Development". *First Monday*, Vol. 10, No. 2 (February 2005).

[5] M. D'Ambros, M. Lanza, H. Gall. "Fractal Figures: Visualizing Development Effort for CVS Entities". *In Proc. of VISSOFT* (2005).

[6] B. Dempsey *et al*. "Who Is An Open Source Software Developer?" *CACM*, Vol. 45, No. 2, (February 2002).

[7] R. Fielding "Shared Leadership in the Apache Project". *CACM*, Vol 42, No. 4, (April 1999).

[8] E. Gilbert and K. Karahalios. "LifeSource: Two CVS Visualizations". *CHI* (2006).

[9] T. Gîrba *et al*. "How Developers Drive Software Evolution". *IWPSE,* (2005).

[10] S. Huang and K. Liu. "Mining Version Histories to Verify the Learning Process of Legitimate Peripheral Participants". *Second MSR Workshop*, ICSE (2005).

[11] P. Jackson and I. Moulinier *Natural Language Processing for Online Applications: Text Retrieval, Extraction, and Categorization.* John Benjamins Publishing (2002).

[12] Y. Kidane and P. Gloor "Correlating Temporal Communication Patterns of the Eclipse Open Source Community with Performance and Creativity". *NAACSOS* (2005).

[13] L. Lopez J. Gonzalez-Barahona, and G. Robles. "Applying Social Network Analysis to the Information in CVS Repositories". *First MSR Workshop,* ICSE (2004).

[14] A. Mockus, R. Fielding, and J. Herbsleb. "Two Case Studies Of Open Source Software Development: Apache and Mozilla". *ACM TOSEM*, Vol. 11, No. 3, (2002).

[15] L. Voinea and A. Telea. "CVSgrab: Mining the History of Large Software Projects", *Proc. Of EUROVIS* (2006).

[16] R. White *et al*. "Supporting Exploratory Search", *CACM* Vol. 49, No. 4, (April 2006).