

The Promises and Perils of Mining Ohloh.net

Henk Poley
Vrije Universiteit Amsterdam
hpy300@cs.vu.nl

ABSTRACT

Ohloh.net is a website that indexes open source software projects, and generates some statistics on their source and committers. In this paper I pose a few scientific use cases for the data exposed through the Ohloh website & API, and a review of existing papers on this subject. At the end of the paper I propose some additions to Ohloh, like mining mailinglists; showing related developers for a project; finding unit-tests inside repositories; integrating external software attention databases; and scraping bugtrackers.

1. INTRODUCTION

Ohloh was founded by two former Microsoft managers Jason Allen and Scott Collison in 2004. The website launched February 2006. Ohloh now contains data on approx. 300.000 projects, and 330.000 source committers.

Ohloh displays several comparative code metrics for the projects it indexes. For example it will display warnings when the code to comment ratio is subpar, or when there are only very few developers. Similarly there are analysis badges for positive attributes like long running projects with a stable development track.

1.1 Related websites

Sourceforge.net is a large website that hosts open source projects. Sourceforge has recently (May 2009) acquired Ohloh. Changes to either Ohloh or Sourceforge are not worked out yet. Though further orientation on finding projects -on Ohloh's side- and hosting projects -on Sourceforge's side- seems obvious. Ohloh is readying a site relaunch end june, early august 2009 (Asay [2009]).

Freshmeat.net is a website that indexes open source projects since 1998. In comparison to Ohloh there is less emphasis on developers, and more on the projects. Ohloh.net has surpassed Freshmeat.net since November 2008, according to compete.com site analysis. Alexa.com does not show this change; But did show stark changes in their statistics aggregation overall during this time, and should probably

not be trusted.

DistroWatch.com moves in more or less the same open source space, but collects only high level projects that integrate software together into desktop OSes. This site is less focussed on individual developers, but more on the companies and groups the aggregate their work in distributions. It has about half the visitors as Freshmeat.net, and was surpassed by Ohloh in September 2008 according to compete.com.

On *Advogato.org* developers can register and write a little blurb about themselves. There they can link to projects they worked on. Other registered users can give their opinion about their level of knowledge as a developer (Master, Journeyer, or Apprentice). There seems to be no automated fact aggregation going on, other than displaying contents of RSS feeds from a developer's blog.

The major difference to the other sites in this space, is that Ohloh tries to extract facts from the source code itself.

2. PROMISES

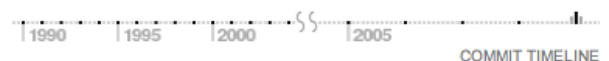
Note: Ohloh limits the number of API calls to 1000 per day. Statistical analysis during the project run time was limited by that fact.

2.1 Promise: Learn what people are using and contributing to

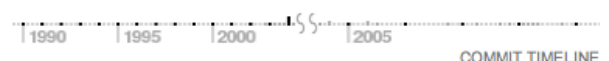
For registered users Ohloh shows a listing of their projects, as long as a user adds it themselves. For projects with known publicly accessible repositories their personal commit history ('Experience') is then shown in abbreviated form on their personal page.

Experience

Bazaar Version Control — Dec 2008 to Present
at CANONICAL, LTD.
24 COMMITS



svnbook — Oct 2002 to Present
contributor
55 COMMITS



Furthermore people can keep lists of applications and li-

braries they used, so called 'stacks'. Adding a project to a stack increments the 'users' counter of the project. This data is then used to sort the result set when you search for projects.

There are also more formal ways to access the data about Ohloh users. In the paper "RDFohloh, a RDF wrapper of Ohloh" Fernandez [2008] worked on exposing the accounts on Ohloh as Friend-Of-A-Friend (FOAF) data in RDF format through the Ohloh API.

RFD is an XML based markup language that stores fact triplets, data like {"Linus Torvalds", "develops", "Linux"}. Research in artificial intelligence promises that this is a good basis for higher level inferences. Query languages like SPARQL and OWL are developed for this. By picking common words -formalized in "ontologies"- different websites should be able to share facts. Friend-Of-A-Friend (or FOAF) is currently a common use case of RDF. With FOAF you encode the 'knows' relationship, or who knows who.

A problem with this data is that different users in the Ohloh database could well be the same person, see sec. 3.2.

2.2 Promise: Journaling lets you keep people up-to-date on what you're coding

This feature requires the project maintainers to keep a weblog hosted on Ohloh itself, what they call a 'journal'. Other people can break into the journal timeline by mentioning (linking) the project in their own posts. There are currently 934 projects with one or more journal entries, either by maintainers or Ohloh users mentioning the project. This means that about 3 promille of the projects have something journaled. A tiny fraction of the total.

Ohloh should consider letting their users add external RSS feeds about their project, automatically aggregating these in the journal posts. Currently RSS feeds are aggregated, but not combined into the journal view.

2.3 Promise: Increase awareness of your open source projects

Ohloh promises that having your project on their site increases awareness of the project. People can find projects on the site by using Ohloh's search engine, or by stumbling on them via an interesting user. When using the project search engine, it sorts the results by the number of accounts that have the project in their stack. This means it won't give an opportunity for your project to be shown, if your project is relatively new with few users. To see if finding a project via contributing people is possible, we look at exposure via high ranked users on Ohloh.

Users on Ohloh are rated by Kudorank. Kudorank starts at 7 after the first pass of their ranking-algorithm, and declines over time. Only receiving a 'kudo' from someone else's Ohloh account increases this score. This happens either by direct vote, or someone using your project in one of their software 'stacks' Luckey [2007b]. This means ranks 8 and above are by influential contributors within the Ohloh kudo system.

kudorank	#contributors
10	63
9	3504
8	10583
7	26465

With this table we can estimate the amount of users with increased exposure at around 14.000 (kudorank 8, 9 & 10).

Set against the total user and committers base of about 330.000 gives us a 4.2% of the total. This matches with similar metrics like the active blogs survey by Technorati. 5.6% on a total of 133 million total blogs had at least one post the last 120 days (blo [2008]). Open source is often described as 'scratching an itch' for the individual developer. This data at least supports this notion, in that the abandoning rate of open source projects is similar to life describing projects like a blog.

New projects should consider having a 'celebrity contributor' on their project, if they want increased awareness.

2.4 Promise: Estimating change in coding behavior

Ohloh indexes commit size on projects with public repositories. Looking at changes in the commit size over time could give an indication of a coding philosophy change.

This was researched in "Continuous Integration in Open Source Software Development" Deshpande and Riehle [2008]. Their research hypotheses was: The average commit size in any year is greater than the average commit size during the next year. This was found to be non existant at 95% confidence level.







Further research could be done by actively looking for interesting strings and code connected to modern coding methods within the Ohloh repository mining backend, see sec. 4.4.

3. PERILS

3.1 Peril: Using the API get you less info than web scraping

During my project it quickly became apparent that the API is only of limited use. Data available to browsers is not seen on their matching .xml file in the API. Most of it can be found elsewhere. But that needs extra trips to the API, quickly dividing the amount of useful data you can access from 1000 API calls (sec. 2).

For example the following page element appears on the main page of my own profile.

	Experience	Measured
 Assembly	10y 6m	0m
 JavaScript	7m	0m
 PHP	5y 7m	0m
 Python	1m	1m
 TeX/LaTeX	1m	1m
 shell script	5y 7m	0m

The XML returned when requesting the profile .xml file (Account call) does not contain this data:

```
<response>
  <status>success</status>
  <result>
    <account>
      <id>39601</id>
      <name>henkpooley</name>
```

```

<created_at>2009-04-08T12:18:29Z</created_at>
<updated_at>2009-06-23T12:04:44Z</updated_at>
<homepage_url></homepage_url>
<avatar_url>http://www.gravatar.com/avatar.php?gravat
ar_id=acacdd37eaaaa0370465edc3374a38d8</avatar_url>
<email_sha1>ca6e4cfef89492bcbe78fb15182963e6f4f47ce3</email_sha1>
<posts_count>0</posts_count>
<location>Schagen, Nederland</location>
<country_code>NL</country_code>
<latitude>52.7882686</latitude>
<longitude>4.802157</longitude>
<kudo_score>
  <created_at>2009-06-22T10:50:29Z</created_at>
  <kudo_rank>7</kudo_rank>
  <position>19321</position>
  <max_position>218377</max_position>
  <position_delta>30</position_delta>
</kudo_score>
</account>
</result>
</response>

```

The programming language facts can now be found by iterating over all the projects of this user (ContributorFact collections call), and then requesting the language data of his contributions (ContributorLanguageFact call). So that is $2 + \#projects$ queries per person.

But when you go there in the API, the user reported experience (first column) is nowhere to be found. I have not done any commits myself on this particular project, but submitted them to a maintainer. So no personal data is available:

```

<response>
  <status>success</status>
  <result>
    <contributor_fact>
      <contributor_id>73392401193649</contributor_id>
      <account_id>39601</account_id>
      <account_name>henkpooley</account_name>
    </contributor_fact>
  </result>
</response>

```

This is what Ohloh has to say about it the API limitations themselves: "The design concept is that for each web page on Ohloh, there may be an equivalent XML-formatted version of the page. Currently, only a small subset of the Ohloh site is available as XML, but more data will become available over time." Luckey [2007a]. We can conclude that since 2007 their API still does not have full coverage of their database.

3.2 Peril: One person could be represented as multiple contributors

In the data from Ohloh different committers can be the same person. This is a problem when trying to find all data

about an individual. For example a search for *torvalds* will return 15 results. One is Linus Torvalds' own Ohloh account. Another is a fan. And 13 others are Linus Torvalds commits to projects that are not linked to his personal account.

Linus.Torvalds KUDO RANK 10
VIEW | 25 FOLLOWERS | 2 POSITIONS
Member
Contributes to [Git](#) and [Linux Kernel 2.6](#).

Linus Torvalds KUDO RANK 9
VIEW
Contributor
Contributes to [Sparse](#).

rick_777 KUDO RANK 8
VIEW | 2 POSITIONS
Member
Rick has been a hobbyist programmer since 1990. Around 1992, he began programming in Turbo Pascal - then he moved to Delphi, HTML/PHP/MySQL, and C++.

Favorite topics: Algorithms, code optimization.

Favorite real life heroes: Richard Stallman, Linus Torvalds, Donald Knuth.

Favorite Anime series: Death Note.

Favorite motto: Don't beg for it, get it yourself. Don't wait for it, make it happen. Contributes to [Saya Video Editor](#) and [Code::Blocks](#).

Linus Torvalds KUDO RANK 8
VIEW
Contributor
Contributes to [Debian GNU/Linux](#).

Linus Torvalds KUDO RANK 7
VIEW
Contributor
Contributes to [Cogito](#).

The paper "Smushing RDF instances" Shi et al. [2008] worked on combining ('smushing') accounts and committers that represent the same person.

The data available on Ohloh is quite sparse. Shi only managed to combine 13 committers on Ohloh, over the 12.613 accounts they had fetched through the API. On mailing lists though they could smush at least a few percent of the people. Ohloh could consider mining mailing lists and trying to combine committers based on that, see 4.1.

A possible solution is to use voluntary combinations, where the users can vote on combining contributors from different code bases into one persona.

3.3 Peril: Can't find people by programming language

Currently you cannot find people who use one -or several- language(s), without spidering the whole site and scraping it yourself. With the 1000 queries per day limit, one would need about a year (330 days) to scrape the entire user base. I encountered this problem when implementing the algorithm in section 4.3.

3.4 Peril: Commit stats do not show changed lines

Encountered in "Continuous integration in open source software development" by Deshpande and Riehle [2008] is the problem that Ohloh only counts the lines that are removed from the previous state, and lines added in the commit. The amount of overlap between added and removed lines is unknown, there is no count of which lines are only slightly changed. The overlap measure could be used to

asses code churn. For example a 'currently refactoring' label could be given to projects that change lots of lines but do not close a lot of bugs, relative to their peers.

4. ADVICE TO OHLOH

Some interesting additions the people at Ohloh could add to their scraper or analyses backend.

4.1 Mine mailinglists

Ohloh should consider mining mailing lists and using this to combine ('smush') their user accounts. Though there is of course no protocol for showing which committer name a mailing list member has, registered users at Ohloh need to supply at least one email address. This could be used to find the registered accounts back in the mailinglists. Ohloh could then add their contribution to a project, even without knowing their commit history.

Apart from the smushing aspect, Ohloh could use metrics like the time it takes for the developer to respond to the mailinglist to improve their ranking metric.

4.2 Find copied libraries & frameworks

Some development frameworks require you to copy over some or all of their code in your project before you can use them. Ohloh could hash files and try to find where the common hashes originated from. This would need a second pass over all the source files in the Ohloh database though.

Excluding duplicated code would lead to fairer project source code statistics. And could help with the smushing problem in sec. 3.2, by listing projects that used someone's code on their own profile.

4.3 Show related developers for your project

A nice idea for Ohloh would be to show other people who might be able to help with your project. This could already be done using their existing data. The proposed algorithm:

1. Collect programming language(s) used by your project
2. Find programmers who can program (all) these languages
3. Remove inactive programmers
4. Remove over-active programmers

I have written some -incomplete- Python code to support this. It is available at Google Code (Poley [2009]). A problem I had in mining the data was that you can't find all programmers by the languages they use, with the search engine. Though if I had started right away with caching data from Ohloh regardless if I knew what to do with it, I might have had a big enough cache of user-data to show some results with this algorithm.

To put the 1000 query limit in perspective. Even the most used languages like C or Java end up being used in about 15% of the commits. Scripting languages like PHP or Python are at about 8%. So real time collection of data (up to a 1000 language-person pairs) would give you around 100 people per language. From which then a subsection needs to be found. And this subsection can't be found with the data from a single API call per contributor. Since this only returns the single top language of a programmer, further reducing the amount of data you can get out of Ohloh directly. See also sec. 3.1.

4.4 Find unit-tests

In 'Continuous Integration in Open Source Software Development' Deshpande and Riehle [2008] they tried to infer if continuous integration occurred more in open source projects since the widespread introduction of testing paradigms. With the limited data from Ohloh they could not find a change in behavior.

Instead Ohloh could look for testing inside the code bases it mines. A good first approximation would be to just look for files with the string 'test' in their name. Most projects use one of the xUnit derived test frameworks, which follow common terminology. Looking for files with commonly used strings in unittests like 'setup', 'teardown' and 'fixture' would be a good second order approximation.

Ohloh currently does not show the filenames used inside the commits, so this kind of analyses needs a trip to the individual repositories.

4.5 Integrate software attention measurements

There are websites like IUseThis.com, Wakoopa.com and Debian Popularity Contest that measure software use. They work either by votes or direct measurement. Importing this data could show a better picture of actual open source software use.

4.6 Scrape bugtrackers

Like mailinglists, bugtrackers contain information about the development of a project. Unlike mail, this is more formalized. Often it is protocol to mention the bug number that is fixed by the commit. By tracking these as links into the bugtracker, research like "When Do Changes Induce Fixes?" Liwerski et al. [2005] could be run on a regular basis. Developers can then be compared on the basis of amount of fixed bugs versus bug introducing fixes.

Also, some bugtrackers expose mail addresses. The comments on the tickets about source code commits could then be traced to their username in the repository.

5. CONCLUSIONS

Ohloh is a promising website for finding open source projects. Though there is still a lot of low hanging fruit. I will send all my suggestions to the Ohloh forum, and look forward to the relaunch of their website around August.

6. ACKNOWLEDGMENTS

The project supervisor is Rahul Premraj. The second reader of this paper is Hans van Vliet. The structure of the paper was inspired by "The Promises and Perils of Mining Git" Bird et al. [2009].

References

- Technorati - state of the blogosphere. 2008. URL <http://technorati.com/blogging/state-of-the-blogosphere/>.
- Matt Asay. Behind the scenes on sourceforge's acquisition of ohloh. 2009. URL http://news.cnet.com/8301-13505_3-10252312-16.html.
- Christian Bird, Peter C Rigby, Earl T Barr, David J Hamilton, Daniel M German, and Prem Devanbu. The promises and perils of mining git. pages 1-10, Apr 2009.

- A Deshpande and D Riehle. Continuous integration in open source software development. *IFIP International Federation for Information Processing*, 275:273–280, 2008.
- S Fernandez. Rdfohlh, a rdf wrapper of ohloh. *1st workshop on Social Data on the Web (SDoW2008)*, Jan 2008. URL <http://ceur-ws.org/Vol-405/paper11.pdf>.
- J Liwerski, T Zimmermann, and A Zeller. When do changes induce fixes? *International Conference on Software Engineering: Proceedings of the 2005 international workshop on Mining software repositories*, 2005.
- Robin Luckey. Ohloh api. 2007a. URL <https://www.ohloh.net/api>.
- Robin Luckey. Ohloh kudosrank. 2007b. URL <https://www.ohloh.net/about/kudos>.
- Henk Poley. pyanohlh - python interface to ohloh api and further webscraping. 2009. URL <http://code.google.com/p/pyanohlh/>.
- L Shi, D Berrueta, S Fernandez, L Polo, and S Fernandez. Smushing rdf instances: are alice and bob the same open source developer? *ISWC2008 workshop on Personal Identification and Collaborations: Knowledge Mediation and Extraction (PICKME 2008)*, 2008.