Exploring the boundaries of collaborative software development

M. Cameron Jones
Google Inc.
mcj@google.com

Michael B. Twidale
University of Illinois
twidale@illinois.edu

ABSTRACT
In this paper we present some observations of programmers citing and attributing copied code via comments in their source code. We see this as an example of a broader collaborative fabric which is woven into software development today, which is perhaps changing the notion of collaborative software development, broadening it from the narrow scope of collaboration within teams or within and between organizations, to a model more akin to scholarly and academic publication. Conceptualizing collaboration in this way draws in many more concepts from Library and Information Science, which we argue pose novel and exciting avenues for future research.

INTRODUCTION
Collaborative software development is typically framed in the context of teams or organizations. However, the growing number of open-source projects and amount of source code and discussions around code available on the web are expanding the definition of collaborative software development, allowing developers to collaborate outside the contexts of traditional software development teams [5], and even without directly interacting with one another. This kind of software development is collaborative in the same sense that much of science can be considered to be collaborative -- publications respond to or build on the work of others. As the scholarly publication process became more open and faster, more academic papers were published, increasing the interactivity among scientists. Likewise, as more software is 'published' we see similar patterns emerging, including the early signs of 'software citations'. We consider this analogy of software as scholarly publication to be not only powerful and productive for research in its own right, but also to be indicative of a broader framing of research on software and software development which we call Software Informatics.

Software informatics is the science of information, practice, and communication around software; studying the individual, collaborative, and social aspects of software production and use [4]. However, a full discussion of software informatics is beyond the scope of this position paper. In this paper we argue that the "scholarly publication model" of collaborative software development is inherently collaborative, and discuss our observations of an emergent embedded practice of citation and reference among software developers.

CSCW'12, February 11–15, 2012, Seattle, Washington, USA.

SOFTWARE CITATIONS: QUESTIONS OF ATTRIBUTION
Having found some useful code, a developer may use it in various ways; copying it wholesale, copying fragments or copying and then modifying it so that it more closely achieves her intended goals. Is this copying acknowledged, and if so how? Although a lot of copying occurs without attribution, we believe there is evidence of some explicit acknowledgement occurring. Crediting original sources in code has been seen in the Scratch community, where automatic attribution mechanisms exist in the coding environment, but a separate practice of crediting original sources in comments and descriptions has been observed [6]. We speculate that norms will continue to evolve, just as they have within scientific publishing over the centuries, so that we now have formal conventions of citations of prior work in references sections and a requirement that researchers acknowledge all major sources of insight or inspiration when they build on the work of others or reuse and remix their ideas.

In a sample of 6,190 open source PHP-language projects downloaded from Source Forge, we searched for the following phrases using grep: "copied from", "adapted from", "adopted from", "taken from," and "based on" [3]. While these phrases could be present in the code for reasons other than explicitly documenting copied code, many instances we found appear to be evidence of code copying. At least once instance of one phrase was found in 3,025 projects (48.9%).

For each phrase, the line of text on which it occurred was extracted, and among the extracted lines of code, 50 were randomly selected for each phrase (250 total instances). Each of the 250 sample lines was coded for all features which described or documented the source of the copied code, including things like names of projects, files, and people from whom the code was copied, etc. A single line could contain multiple features. The set of features coded in the sample was constructed through open coding, where new features were added as they were encountered in the data. For example, the line of code in the example in Table

File	XPWeb/Lib/php_lib_login_includes/adodb/drivers/adodb-oracle.inc.php
Line	97
Code	if ($argHostname) { // code copied from version submitted for oci8 by Jorma Tuomainen <jorma.tuomainen@ppoy.fi>

Table 1. Example of a comment attributing where code was copied from.

1 contains a project name (oci8), a person name (Jorma Tuomainen), and an email address (jorma.tuomainen@ppoy.fi).

A total of fifteen source-identifying features were identified through the open coding: project name, file name, person name, url, email, function name, book title, pattern, php.net documentation page, module name, blog entry, date, unix 'man' page, php.net notes page, unknown source. Examples of comments containing each feature are provided in Table 2. Overall, 30% of the samples did not appear to reflect an intentional citation-like behavior, nor did they contain any features.

This simple, low cost easily replicable and extensible pilot study gives us some initial senses of software citations. Despite the limitations of the pilot study, we believe it is sufficient to claim that:

- Software citations already exist as a minority practice.
- There appear to be enough to merit further study.
- Even though the phrases searched for may not have any features to do with software citation, given that in our sample 70% did seem to be citation-like, we can get a good-enough precision for further analysis.
- This simple approach is necessarily an under-count of all extant citation-like practices - it is worth looking for other commonly used phrases in order to improve the recall score of citations.
- We must also acknowledge that in these early stages of the practice, there could well be quite a number of completely idiosyncratic citation practices.

We hypothesize that by analogy with scientific citations, norms and conventions for software citations will evolve. Typically this process happens by people copying and

Feature	Example Snippet
project name	* The following code is adapted from the PEAR DB error handling code.
file name	// FIXME - copied from search.php. Does this work for a second blog?
person name	* based on db_mysql.inc by Boris Erdmann and Kristian Koehntopp
url	* This filter was taken from http://vikjavev.no/computing/ump.php
email	* Adapted from db_mysql.inc by Sascha Schumann <sascha@schumann.cx>
function name	// Query is taken from the runThroughTemplates($theRootLine) function in the parent class.
book title	Code adopted from example code by Hugh E. Williams and David Lane, authors of the book "Web Database Application with PHP and MySQL," published by O'Reilly & Associates.
design pattern	Most of the colors were taken from mail.yahoo.com's theme on their
php.net documentation	// taken from http://www.php.net/manual/en/function.array-merge-recursive.php
module name	# All code under here copied from filescenter->bo->fileUpload #
blog entry	Pretty func adapted from ALA http://www.alistapart.com/articles/gettingstartedwithajax
date	// this block is taken from mysql.php 2005070202
Unix man page	* copied from get-html-translation-table man page
php.net notes entry	* Adapted from the comments to htmlspecialchars: http://us.php.net/htmlspecialchars
unknown source	//Most of this code was adapted from another authors code but I do not remember where I found it

Table 2. Example comments containing each of the 15 attribution features identified in this study.

modifying the behaviors of others that they see in reading code as part of reuse practices.

SOFTWARE DEVELOPMENT AS DIFFUSE COLLABORATION AROUND DOCUMENTS OR HOW A LIBRARY AND INFORMATION SCIENCE PERSPECTIVE MAY HELP

We believe there is potential of studying software development from a Library and Information Science (LIS) perspective. There are several reasons why this might be productive even if at first it seems a little odd. Programs are documents (or collections of documents). Like other scientific and literary texts they can refer to other documents, incorporate textual fragments from other documents and modify and combine such fragments. With increasing numbers and availability of documents, additional mechanisms are required to keep track of them, to search and re-find them. LIS studies the creation, management, transformation, searching for and use of documents. Consequently it may help us to understand these activities as applied to a particular genre of documents - namely program files. The availability of these documents as 'publications' on the web means that collaborations by using documents created by others (in different places at different times, for different purposes) becomes much easier.

CODING AS SEARCH

If we recognize that part of the activity of software development is searching for code written by others [c.f., 1, 2], then we can treat this as an optimization challenge. Do some people search better than others? Do some find resources faster, or find a richer set of candidate resources, or are able to rank the quality and relevance of their results more quickly or accurately? Is this a teachable skill? Should it be taught, and if so how?

If code search is to be taught to programmers then it makes sense to think about whether software search skills should be taught in a computer science curriculum, or whether students might better learn through practice. In either case, there is expertise in LIS in analyzing common errors, misconceptions and inefficiencies by end users in searching with various databases, and in analyzing, interpreting and prioritizing the results, as well as expertise in teaching how to search for and interpret results more efficiently and effectively. This is often covered in courses on "information literacy". The skills and techniques taught in such courses that focus on finding books and articles may not map directly into ways to teach programmers to search for and assess discovered code more effectively, but they are likely to be highly informative - and lead to a quite different perspective than viewing the learning of programming as principally about build-from-scratch code.

CONCLUSION

This kind of very lightweight collaborative software development activity merits further study. We know that the process of publication in science led to an explosion of creativity and progress, as it became easier for scientists to build on the work of others. We suspect something similar is happening that is distinct from but complementary to the team-focused aspects of collaborative software development. The emergence of software citations is just one indication of the role of this larger, more diffuse kind of collaboration. If the trend of software citation continues, it opens up the possibility of 'software bibliometrics' to trace the 'impact' of code fragments, snippets, even algorithms. Just as in science, tracking that impact can both help us understand this kind of collaborative software development better, but also help software developers exploit and build upon the work of their published peers.

REFERENCES

1. Brandt, J., Dontcheva, M., Weskamp, M. & Klemmer, S.R. (2010). Example-centric programming: integrating web search into the development environment. In *Proceedings of* (CHI '10). ACM, New York, NY, USA, 513-522. DOI=10.1145/1753326.1753402
2. Hartmann, B. MacDougall, D.l Brandt, J. & Klemmer, S.R. 2010. What would other programmers do: suggesting solutions to error messages. In *Proceedings* CHI '10. ACM, New York, NY, USA, 1019-1028. DOI=10.1145/1753326.1753478
3. Jones, M.C. (2010) Remix and Reuse of Source Code in Software Production. Doctoral Dissertation, University of Illinois. http://hdl.handle.net/2142/18255
4. Jones, M.C. and Twidale, M.B. (2009) Software Informatics? Proceedings of the iConference 2009. http://hdl.handle.net/2142/9600
5. Mamykina, L., Manoim, B., Mittal, M., Hripcsak, G. & Hartmann, B. (2011). Design lessons from the fastest q&a site in the west. In *Proceedings of* CHI '11. ACM, New York, NY, USA, 2857-2866. DOI=10.1145/1978942.1979366
6. Monroy-Hernandez, A., Hill, B. M., Gonzalez-Rivero, J., boyd, d. (2011) Computers can't Give Credit: How Automatic Attribution Falls Short in an Online Remixing Community. In *Proceedings CHI'11*. ACM, New York, NY, USA, 3421-3430. DOI=10.1145/1978942.1979452