Peer Programming: shared programming resources and how they might be used to understand the activity of programming

Christopher Douce

Abstract
Software developers are expected to possess a multitude of professional skills. They need to have an awareness of interaction design, an understanding of networking systems and an appreciation of the subtleties of database architectures. It is impossible for professional software developers to know everything about each of these areas. Due to the demands on their expertise there has been a rapid growth of the number of programming problem-oriented websites and resources. This paper presents the innovation of the 'programmers blog' and 'peer knowledge exchange site' as one way to learn more about the activity of programming and the difficulties that professional software developers face on a day to day basis. A number of social websites are presented, ranging from article based forums to real programming blogs. A number of suggestions as to how these websites could be used in ethnographic study is given.

1. Introduction

Software development is a fascinating area of study. Developing software requires a programmer to possess knowledge of not only a particular problem area or discipline but also knowledge of a range of hardware and software technologies. Developers not only have to know the language of the users, but also the language of programming environments, design notations and requirement specifications.

An efficient software developer must by a polymath who is fluent in many aspects of abstract sounding technology. An important question is, how does a developer learn? What strategies might he or she adopt to cope with the vast amount of skills that have to be mastered?

One of the most important tools in the software engineers armory is the search engine. Search engines such as Google allow a software engineer to discover useful tips and techniques, to gain insight into how others solve problems and to obtain a richer and deeper understanding of programming problems.

Search engines are presenting results from programmer blog sites, discussion forums where software developer share tips and techniques, and websites that allow programmers to share and disseminate knowledge in the form of short, concise and topical articles. In day to day activities, these sites are becoming increasingly valuable. It is all well and good to find the documentation for a new API but there is nothing more useful than finding a set of sample files that does something similar to what you are trying to achieve. It is even more useful to find sample programs annotated with useful comments and accompanied by a tutorial.

The following section presents a number of practical examples of how these sites have been useful in the day to day work of a professional programmer. To an extent this paper could be described as 'self ethnography' or introspection (Dukes, 1965) , but it is important to bear in mind that the programming resources described here were created by a larger set of professionals.

The next section entitled 'a taxonomy of resources' attempts to loosely categorize the sites that have been discovered. This is followed by a section entitled 'an ethnographic perspective' which begins to explore how these resources could be used to study the wider activity of programming and programmer collaboration. This section also outlines an interesting multi-disciplinary research approach that combines ethnographic study, software engineering and emerging developments in computational linguistics. The final sections brings together the various strands of discussion and presents a number of simplistic conclusions.

2. Programming Knowledge Exchanges

This section describes how programming related websites have been used in real life scenarios to solve problems. The problem is described in outline. This is then followed by a brief analysis of how a particular programming resource (or resources) were used. A brief outline of a solution is then given.

2.1. Article and Question Centric Sites

Programming Problem
An element of a web based system had an interactive control which appeared to slow the performance of a site. The original code was based upon an early 'server side' component. The software supplier had since recommended that this component should not be used for any further development work. The first task, in essence, was to construct a fast, efficient navigation control that can be usable within a web browser. The second task was to determine how it could be possible to construct a pop-up menu control.

Resources
Not having a complete understanding of how to solve these two related problems, the approach adopted was to examine available literature (Microsoft Developers Journal) and perform a number of targeted web searches. Two websites stood out, primarily because other individuals working on a related system had discovered them independently. One was called ‘the code project’[1], and another called ‘experts exchange’[2].

The Code Project provides technology specific (usually language specific) tutorials or solutions to what may be described as common problems. Originating from Canada, it provides over ten thousand software development related articles. Each article usually contains clearly presented (and formatted) code and a discussion forum, a feature which is common to the simplest of blog. These discussion forums allow the consumers of these articles to interact with the author (and other readers) allowing successes, failures or suggestions to be communicated.

The Code Project, like many e-commerce sites where communities of users can exchange views, uses a simple rating facility which allows readers to assess the perceived quality of an article. This is particularly useful since if solutions are incorrect or unhelpful, they are marked down, thus reducing their visibility to solution seekers.

Experts Exchange adopts a different (perhaps less useful) model. Rather than adopting an 'article-centric' model they adopt an older ‘problem-response’ model where subscribers can submit problems to a space where other software professionals can respond (in a similar way to the Google Answers service[3]). Numerous responses to a question can be seen. The expert who posed the question can select an answer that is particularly helpful or solves a problem directly.

Solution
Both sites contained valuable information, tips and tricks about how to begin to work with the chosen programming language. As well as providing articles in a 'magazine article' style, The Code Project also provides sample programs which can be downloaded, extracted, examined

[1]http://www.thecodeproject.com
[2]http://www.experts-exchange.com
[3]http://answers.google.com

and changed. Through examples it was possible to learn about how Javascript and DHTML code could work together to create panels that could be shown and hidden.

2.2. Compound Programmer Sites

Programming Problem

A mechanism to securely send data between a web-client and a server was required. It was necessary to ensure that data values sent between a client and server could not be tampered with.

Analysis

A solution to this problem did not come from the article based sites that were mentioned in the earlier section. Instead, an initial solution was found from a site that adopted a 'more traditional' blog approach. This approach can be seen clearly within the dotnetjunkies and sqljunkies websites[4] which combine discussion forums, blogs and articles. These sites can be described as a compound function sites.

The website dotnetjunkies provides sample code and a corresponding 'talk back' discussion forum. This facility provided a range of tips and pointers about how a number of security APIs could be utilised.

Another example of the compound function site can be see in c-sharpcorner.com[5]. This site appears to be particularly active. The front page is peppered with new (often rather small articles) which aim to inform and illuminate (and perhaps infuriate).

Solution

Using other peoples code as a basis of your own solution has its risks. Using code without a complete understanding the author's original intent and an appreciation of the context in which the code was originally applied may later yield hidden deficiencies. In this particular case the original code contained a problem that only became apparent if a particular locale setting was used. This situation was solvable since it was possible to visit the same site several months later (after the problem had become apparent) and find an updated version of the software and new discussion forum comments from both old and new consumers. This code-article-forum approach to sharing problems represents a form of open source software. Problems are discovered, solved and published for peer review which can then inspire traceable criticism and comment.

2.3. Professional Blogs

Problem

There was a need to research the best approach to two generic problems. The first problem was to understand how it might be possible to minimise sever requests from web clients. The activity of investigating this problem would ultimately ensure that the solution would be as scalable as possible. The second problem was to find how it might be possible to deploy solutions through the construction of installation routines.

Analysis

There is another class of social programming site that is becoming more influential: the professional programmers blog. Nowhere is this more apparent than within the world of Microsoft where it seems to be actively encouraged[6]. Using the blogs from Microsoft it is

[4]http://www.dotnetjunkies.com, http://www.sqljunkies.com. Another website, http://www.sqlservercentral.com offers similar ideas and support was discovered later.
[5]http://www.c-sharpcorner.com
[6]http://blogs.msdn.com/ and http://weblogs.asp.net

possible to read the thoughts and ideas of team leaders and developers who are working hard at the code face. Rather than reading mere opinion you can read when they have zero bug bounce, hold freezes of requirements and learn about what conferences these guys are speaking at. This is useful information for the software manager who has to keep one eye on the current code base and one eye on what is on the technology horizon. The AJAX.NET blog was particularly helpful[7]. It allowed terminology to be understood and awareness of how to solve the first problem gained.

Research into the second problem yielded blogs dedicated to database systems and the development of installation routines[8] and how Microsoft are beginning to evaluate the usability of the APIs (see Clarke, 2004)[9]. Within half an hour of starting to look it was possible to find detailed histories of products which also includes a description of design decisions and pointers towards future development directions.

Solution

Finding interesting text in a blog is only the beginning of a solution. As soon as one has found a useful peer the process of understanding the other developers perspective must begin. A programmer not only has to assimilate the names of functions and command line operations, he or she also has to grapple with the idiolect of the writer and the culture of the organisation in which they work for.

Formulating and distilling general purpose programming strategies from a number of disparate sources takes time and energy. The activity of reading the words of the developers whose products you are using may result in greater levels of understanding and a greater awareness of potential 'bear traps' that may be waiting.

3. A Taxonomy of Programming Resources

A number of web-based social programming resources have been identified. These include:

1. ‘Tutorial talk-back’ of the variety illustrated within ‘the code project’. I have called these *article centric sites*.
2. More traditional ‘post and wait’ discussion forums, where users experiencing problems can request assistance from experts. These I consider to be *question centric sites*.
3. True ‘programmer blogs’ written by developers who construct the tools and systems that others utilise. These are particularly welcome since they allow other programmers to understand new developments within a field and allow the blogger to understand the problems that others are encountering.
4. *Compound sites* – those that combine an article centric approach for information dissemination with blogs or question posting.

A fifth category, but one that has not been discussed here, is of course, the more traditional e-mail discussion lists which are used widely within both the open source and commercial software communities (see O'Shea and Exton, 2004). In the sense that some resources can be general purpose 'one stop shops' for technical advice, another trend lies with the construction of sites that are dedicated to a particular technology or set of technologies[10].

[7]http://weblogs.asp.net/mschwarz/
[8]http://blogs.msdn.com/robmen/
[9]http://blogs.msdn.com/stevencl/
[10]http://www.installsite.org

4. An Ethnographic perspective?

What can programming related web sites tell us apart from (a) there are lots of programmers out there, and (b) there are lots of problems to be solved? They can tell us about how a particular technology is being adopted by a community and how programming fashions and ideas are communicated. Any study that examines how programmers work in the wild should consider where they find their information from. This may be from peers working within the same organisation, or even peers within different countries who might be working with competitors.

The sites that have been described allow common problems and misunderstandings to be uncovered. Such misunderstandings may provide invaluable information to designers of applications, programming libraries and languages. A simple example of this can be found within a blog that discusses the sometimes contentious topic of test first development. There is evidence of contention and expressions of concern: 'I don't think this way…', wrote one blog poster. Other programmers were encouraging, 'if you begin, things will fall into place, you will then understand what is being written here'. It was possible to see persuasion, as well as discussion. Other blogs contained tips on how to communicate technical concepts to managers who were less technically skilled (such as tips about how to introduce Ruby on Rails to a suspicious audience).

When considering programmer blogs and ethnography, one naturally returns to the engineering problems that are being discussed and explored. When browsing through discussion and article forums one can begin to see how software engineering (or specifically, the application of search technology) might be used to study the activity of software development. One approach is to attempt to adopt the idea of sentiment analysis. Sentiment analysis is an area of study that crosses the area of corpora and computational linguistics (see Riloff, Wiebe & Phillips, 2005; Wilson, Wiebe & Hwa, 2004) . It is the application of natural language processing techniques to extract subjective information about a given topic. It is easy to see why marketeers might be interested in mining the blogosphere for opinions about products. Programmer blogs could be mined for sentiments and expressions of frustration about programming related topics. The topics might be related to language design, application architecture or process construction. Similarly, one may be able to view how new ideas may be taking hold and identify how programming fashions and trends (such as AJAX) begin to emerge.

Taking a very naïve view, understanding what programmers continually complain about and what topics are continually perceived as difficult may inform tool design, language design and even programmer education. Mining the depths of programmer opinion may allow us to gain further insight into how to step towards improvements in software reliability and quality.

5. Conclusions

The term *blog* has received so much media coverage one can almost be tempted to deride the phenomena as a fad. It is worth pausing for a moment. A focused technical blog can be a rich source of information. Programmer blogs and their close relation, the article focused knowledge exchange boards seem to be replacing the earlier dominant form of programming related messaging: usenet newsgroups. They have one key advantage. In many cases the signal to noise ratio appears to be relatively low and discussions are usually contextualized closely to a central article or focal idea.

It is also important to remember that it may not be a blog post about a particular idea or technology that may yield an interesting insight into the activity of programming. A series of responses from readers may be even more insightful. In some respects, a corporate blog is also an exercise in PR. It can demonstrate the virtue of openness combined with a demonstration of their ability to listen (and learn directly about customer requirements). This has echos in 'community sites' operated by commercial software companies. Whilst primarily driven as a means to provide technical support, being open and providing a space for programmers to discuss problems can be a cost effective support solution[11].

Code is much more than simply text found within an IDE or a source control tree. Code is something that will be shared. Code represents a concise expression of data transformations and manipulations that can represent a thousand words. Code also lives within websites in the forms of snippets, partial solutions which are waiting to be discovered, adopted, utilized and discussed. The form of these solutions will change and adapt depending upon how well they are received and used by others.

By examining how software developers use social websites to solve real problems we may discover how to develop new tools and approaches to share programming related information (see Preece et. al., 2002). Again, returning to the topic of the IDE, almost every IDE worth its salt contains a simple search engine of some form to navigate increasingly large API's. By examining what information sources programmers use in the wild use, perhaps we could begin to ask the question of whether an IDE needs to become a SIDE, a Socially Integrated Development Environment.

There are gigabytes of data that describe the problems programmers face, the discussions they have and the solutions they form. The problem is trying to figure out how harness this information to help us with our quest in understanding how to build strong, efficient and cost effective software systems.

References

Clarke, S. (2004) Measuring API Usability. Dr Dobbs Journal. May, 2004.

Dukes, W. F. (1965). N=1. Psychological Review 64: 74-79.

Hunt, A., and Thomas, D. (1999) The Pragmatic Programmer. Addison-Wesley.

O'Shea, P. and Exton, C. (2005) The Role of Source Code within Program Summaries describing Maintenance Activities. 17th Annual Workshop of the Psychology of Programming Interest Group, Brighton, UK.

O'Shea, P. and Exton, C. (2004) Investigating patterns and task type correlations in open source mailing lists for programmer comprehension. 16th Annual Workshop of the Psychology of Programming Interest Group, Institute of Technology, Carlow, Ireland.

Preece, J., Rogers, Y. and Sharp, H. (2002) Interaction Design. John Wiley & Sons.

Randall, D., Hughes, J., Sommerville, I., Rodden, T. and Bentley, R. (1993) Designing with Ethnography: Making Work Visible. Interacting with Computers , Vol 5, No. 2.

Riloff, E., Wiebe, J. and Phillips, W. (2005). Exploiting subjectivity classification to improve information extraction. Proc. 20th National Conference on Artificial Intelligence (AAAI-2005).

Wilson, T., Wiebe, J. and Hwa, R. (2004). Just how mad are you? Finding strong and weak opinion clauses. Proc. 19th National Conference on Artificial Intelligence (AAAI-2004).

[11]http://community.installshield.com

Acknowledgments
Many thanks are given the the anonymous reviewers whose comments have greatly contributed to the development of this paper. Thanks are also warmly extended to my anonymous proof reader who suggested a number of useful corrections and enhancements.