Categorizing Developer Information Needs in Software Ecosystems

Nicole Haenni Mircea Lungu Niko Schwarz

Oscar Nierstrasz

University of Bern
Switzerland
http://scg.unibe.ch

ABSTRACT

We present the results of an investigation into the nature of the information needs of software developers who work in projects that are part of larger ecosystems. In an open-question survey we asked framework and library developers about their information needs with respect to both their *upstream* and *downstream* projects. We investigated what kind of information is required, why is it necessary, and how the developers obtain this information.

The results show that the downstream needs are grouped into three categories roughly corresponding to the different stages in their relation with an upstream: selection, adoption, and co-evolution. The less numerous upstream needs are grouped into two categories: project statistics and code usage.

The current practices part of the study shows that to satisfy many of these needs developers use non-specific tools and ad hoc methods. We believe that this is a largely unexplored area of research.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement; D.2.9 [**Software Engineering**]: Management; H.3.4 [**Information Storage and Retrieval**]: Systems and Software

General Terms

Human Factors, Management, Measurement

Keywords

Software ecosystems, programmer needs, open source software, grounded theory, program comprehension, frameworks and libraries

WEA '13, August 19, 2013, Saint Petersburg, Russia

1. INTRODUCTION

Software systems do not exist in a void, but rather in larger contexts called software ecosystems. Within an ecosystem a project is surrounded by competing and complementary projects, and developers interact with fellow developers inside and outside of their own projects, borrowing ideas, idioms, code snippets and sometimes even entire code bases. In this paper, we identify and categorize the information needs of developers by analyzing the results of several interviews we conducted with developers that work in an ecosystem context.

Authors define the term "software ecosystem" in different ways [1, 2]. In our work we rely on the definition of Lungu [3]: *"A software ecosystem is a collection of software projects which are developed and which co-evolve together in the same environment."* Our definition focuses on the co-evolution and source-code level inter-dependency aspect rather than the business aspect.

A common relationship between the systems in a software ecosystem is that of *reuse based dependency*: a library or framework (the *upstream*) provides the required source code to another project (the *downstream*). This relationship comes with challenges (*e.g.*, keeping up with the evolution of an upstream library) and information needs (*e.g.*, how is the downstream using a library's API).

In this article we categorize the needs of developers who work in an ecosystem context. The needs are identified by means of an open-question survey. The underlying research question is: *"What are the information needs of a software developer working in an ecosystem context?"* We augment this with information about the developer motivation and the current state of the practice for fulfilling these needs.

Structure of the paper. The remainder of the paper is organized as follows: In section 2, we take a closer look at related previous studies. In section 3, we describe the methodology for this study. In section 4 our field study is evaluated and we present our findings. In section 5, we discuss our research results. Finally, we conclude and plan further work of our study in section 6.

2. RELATED WORK

There have been numerous studies regarding the information needs of developers. Such studies are important since they pave the way for future research and allow the software engineering research community to focus on real-world

problems and thus maximize the impact of the research on the practice.

One of the first such studies is Sillito and Murphy's field study [4] which reported on the information a developer needs to perform a source code change. Their study focus on challenges that developers have when working in a single project context.

Ko and DeLine observe 21 different types of information needs of developers [5]. They collected their data by using a speak aloud protocol while developers are solving real development problems. The study does not touch on the problems inherent in a multi-project context.

Seichter *et al.* examine a management system for software artifacts to support decisions of involved actors. Inspired by social networks, they study different types of interactions within software ecosystems to get information easily [6]. But they do not examine what information the actors need.

Phillips *et al.* identify information needs that support integration decision-making in parallel development [7]. Their findings show needs only in large software companies and do not include the aspect of distributed development in software ecosystems.

One study which is close to ours is that of Begel *et al.* who propose Codebook as a code-based social networking web service that helps developers get information about other activities of their colleagues [8]. They identify the needs inside Microsoft by asking programmers. In contrast to them, we focus on open source needs, rather than only the needs of Microsoft employees.

3. RESEARCH METHOD

To identify the needs of developers working in open source ecosystems, we interviewed several of them via email and in person. We asked the respondents what their information needs were corresponding to their upstream and downstream roles in the software ecosystem in which they craft software.

Figure 1 lists the survey questions. Our research question (section 1) is split into questions 2 and 3 addressing respectively framework and library developers, and developers depending on code from other projects. Questions 2 and 3 are further divided into three subquestions each, asking *what* kind of information they need, *why* this information is important and *how* they obtain that information at the moment.

To analyze the answers which we receive as free-form text, we applied a grounded theory methodology as introduced by Strauss and Corbin [9]. In this methodology, contrary to beginning with a pre-conceived theory, one is evolved from the data, and continuously refined in an iterative process. The data analysis includes coding strategies by breaking down the data collection from surveys or other observations into similar units. The questions are formulated as openly as possible and do not attempt to influence the participant in a certain direction.

We label each mentioned topic with an assigned code[1] This process of qualitative data analysis is known as open coding [10]. By grouping similar codes together we create axial coding categories and themes [9]. The results of the open coding and of the axial coding are presented in section 4 and in subsection 4.3 respectively.

[1]The term "code" refers to a recurring topic in the interviews, and should not be confused with "source code."

1. In what ecosystem are you most active?
2. Are you the developer of a framework or library? If so, what is its name?
   - 2.1. What do you most want to know about the use of your library/framework in your ecosystem?
   - 2.2. Why would that be interesting to know?
   - 2.3. What do you currently do to obtain that information, if anything?
3. Are you using a framework or a library in your ecosystem? If so, name one.
   - 3.1. What do you most want to know about the libraries/frameworks that you are using?
   - 3.2. Why would that be interesting to know?
   - 3.3. What do you currently do to obtain that information, if anything?

Figure 1: The survey as shipped to the participants.

4. RESULTS

In this section, we present a list of codes that resulted from the open-coding process. They represent the information needs, motivations, and current practices of software developers working in an ecosystem context.

We shipped the email survey to a convenience sampled group of 20 framework and library developers. The participants were neither offered nor given compensation for their participation. Participants were assured of their anonymity.

Of the developers asked, 65.0% responded. An additional participant gave us the answers in person. All participants have at least seven years of academic or professional experience. From the 14 answers, we collected initial codes to answer the questions in Figure 1. We assigned each participant a reference letter from A–N. We explain our findings with quotations of the participants with the corresponding reference letter.

The goal of the first question was to put the respondent into the right frame of mind in which he would think about the broader context of his work and the inter-dependencies of the systems he is working on. We do not analyze these answers here, but we mention that we had a variety of ecosystems centered around different languages (Smalltalk, Python), technologies (Moose, SciPy), and online source code repositories (SqueakSource, Github). Two respondents mentioned here two social websites (stackoverflow.com and reddit.com).

4.1 Upstream Findings

The first section of the survey aims at capturing information needs that the developers of libraries and frameworks have as well as their motivations, and current practices. We will list and discuss the codes we extracted for each of the questions individually.

4.1.1 Information Needs

Analyzing the answers to Q.2.1. we found six distinct information needs for the developers that work upstream.

Downstream projects. (A,C,D,F,I) Developers want to know how their code fits in the ecosystem. They want to know how many and which are the downstream projects, and for what purposes is the downstream using their project: *"I'd like to know what people build with my frameworks" (A).* Respondent (D) wanted to know number of passive downstream users that track a project's state.

API usage details. (B,F,J,K,L) Developers monitor the way the downstream is using the API and collect details about invoked methods and their arguments. This provides insight into the effectiveness of an API and its usage: *"L: which parts of the code are actively used?" (L).*

Forked projects. (D,J,L) Developers want to know about the clones of their work. With infrastructures like Github this is particularly easy to do.

Runtime statistics. (B) Some developers want statistics about the usage of their library at runtime to help localize and fix failures: *"which API methods are called how often and which data is passed to them? How often do they fail with an error?" (B).*

Code convention compliance. (E) This includes naming conventions, indentation, comments and so on. A guideline would provide help for maintenance issues, consistency and readability. *"Variation of lint rules in my projects along the project history" (E)* to ensure that downstream users follow the conventions the developer set.

4.1.2 Motivations

From the answers to Q.2.2. we found three main motivations behind the previously listed information needs.

Strengthening self-esteem. (A,I,J,K) Pride in one's work and project motivates information needs. *"It is a good motivation if a lot of people like my code and build cool stuff on top of it" (A)* and *"it helps the self-esteem" (A).* Positive feedback and rising popularity keeps a developer motivated and *"gives inspiration and hints where to orient the project's evolution" (K).*

Maintaining downstream compatibility. (F,I,K) When developers know how their clients use their framework or library, then they are able to estimate the impact of code changes. If needed, they can notify downstream developers on how to stay compatible. A participant explains: *"I want to know [. . .] the impact [. . .] when I modify my source code" (K).* And another developer states: *"I want my clients to know how the library is being used and to assess the impact of possible changes" (F).*

Managing resources. (B,L) Discover unused functionalities to deprecate them out and to better distribute effort. *"To conserve my resources. If people don't use a method or, a whole feature of the API, why maintain it?" (B).*

4.1.3 Current Practices

In the last question Q.2.3. for the upstream developers we asked about the current practices for satisfying their needs.

Mailing lists. (A,F,I,J) People with common interests subscribe to a mailing list to keep up-to-date with a given issue. Problems and solutions are asked and discussed through email communication with all subscriptions.

Repository analytics. (A,C,D) Some source code repositories provide analytics for projects. GitHub provides information about forks, downloads, watches, *etc.* Even monitoring web traffic is of interest: *"I observe the web analytics of my project's home page" (A).*

Monitoring ecosystem commits. (F) In some cases developers track code changes to many projects of interest at once by monitoring news services: *"I am monitoring the RSS of SqueakSource [NB: which includes updates on the changes to several hundreds of active projects]" (F).*

Social media. (A) Developers use social media tools (*e.g.*, Twitter) to publish the latest news about their project.

4.2 Downstream Findings

In the second section we present our findings into what downstream developers need when they are in the process of deciding what libraries or frameworks they want to use for their projects. Again we show their motivations and current practices.

4.2.1 Information Needs

Based on the answers to Q.3.1 we synthesized the information needs for the developers downstream. The needs of the downstream outnumber the needs of the upstream. We list them here in decreasing order of their support.

Upstream changes. (E,F,K,N) Developers want notifications of deprecations and substitutions that affect the API they use: *"What has changed since the last time I loaded [the library]" (K)* and *"if they deprecated some methods" (N)*

They might also care about the developers that make the changes: *"who changed what" (F).*

Finally, when developers have a portfolio of projects, they care about how a third-party upstream impacts it: *"Which of my projects may be impacted by some update of Pharo" (E).*

Available public support. (A,B,E,J) Developers want to know the popularity of a framework *"Are they popular enough to find support on the web in blogs and on StackOverflow?" (B).*

A related factor is the responsiveness of the developer team and associated community to provide support: *"How likely are they to fix bugs and to respond to feature requests" (B). "Whether there are bugs that were left unresolved for a long time" (E).*

Documentation. (B,G,H,N) The potential users of an API require its documentation: *"I am basically happy with a good API documentation" (B).*

Some developers want to understand the internals of an upstream project and thus require architectural documentation: *"[...] expose connections between high-level elements [...] what methods [...] of the packages invoke each other" (N).*

License type. (A,I,L) A common request is: *"Is the license compatible with ours?" (A).*

Implementation quality. (B,E) A potential client of a library wants to know how robust its implementation is, how often it is updated, how responsive the developers are, and how fast the library is evolving. *"Whether [the project's code] works or not" (E).*

People want to know the level of activity around a library: *"Whether they [the libraries] are intensively maintained" (B).*

Compatibility with other systems. (L) A downstream client often depends on multiple upstreams. They want to know whether an individual upstream works with the rest of the configuration. *"Does the current version [of the upstream] run on the version of the system I use [downstream]" (L).*

Real contextual sample code. (C) Developers want example code snippets which are extracted from other projects with similar functionality. *"I'd like to see example code extracted from other projects using the same libs that corresponds to functions I'm trying to figure out how to use" (C).*

Comparison with similar upstreams. (A) Find related libraries and frameworks that provide similar functionalities but are independently developed. *"Comparison with similar frameworks" (A)* gives the opportunity to consider an alternative upstream.

4.2.2 Motivations

With the help of Q.3.2. we list what the motivations behind the downstream information needs are.

API understanding. (C,F,G,I,L,M) Developers want to use functionalities provided by the API right away. This is eased when API names are intuitive and well documented. *"To see whether I can construct on the libraries or not" (F).* A participant's answer is that he would like *"to spend less time figuring out how to use new libraries" (C).*

Keeping up with upstream evolution. (E,I,J,L) Developers of downstream projects want to keep up to date with upstream changes. The only way to improve something is to know the existing problems and to know how it is expected to work (I). *"To know whether I have to update my projects or not" (E)*, *e.g.*, if there are any new releases. The same respondent correlates to the credibility of the upstream: *"I am interested to see if the change was performed by someone I trust" (E).*

Choosing the right upstream. (B,I) Choosing the right upstream will impact the future of a project: *"For example, [our testing framework] uses JUnit 4, but later I learned that less than 5% of all users of JUnit use version 4 and all others still use version 3. So we are stuck with a bad choice" (B).*

Another developer argues: *"...if I don't know how to use [NB: the library] after an hour, I throw it away. I won't look one single day into its code just to see how to use it" (I).*

Influencing upstream. (B,J) Sometimes developers would like to modify the upstream to conform to their needs, but this is not always possible: *"Sometimes I need to collaborate and influence design of frameworks I use and to ensure I can progress even if the maintainers I depend on are not responsive" (J).*

Estimating the impact of changes. (F,H) Before updating to a new version of the upstream, developers want to estimate the impact of changes. They are *"interested in what the change affected" (F).*

4.2.3 Current Practices

Based on the answers to Q.3.3. we list the current practices downstream developers use.

Monitoring news. (C,E,F,G,I,J) Developers read mailing lists and monitor repositories for commits and activities to be up to date. Developers monitor the RSS feeds of the upstream projects: *"I am monitoring the RSS of SqueakSource" (F).*

Searching the Internet. (A,B,C,G,H,I) Downstream developers search the internet for the the upstream developer's website or third parties blogs and tutorials. Before using a specific framework, downstream developers like to play around and modify example code to see how it works.

Developers often estimate the relevance of a library by its popularity online, and in programming related forums. *"I look at the most popular tags on Stackoverflow and pick that library" (B).*

Continuous integration. (F,K,L) Some developers commit code changes to the project repository several times a day. As one respondent states, *"I am building regularly to ensure that at least things still work" (F).* This supports fast deployment and uncovers compatibility problems in early stages.

Unit tests. (E) I load the latest upstream version and run my unit tests.

4.3 Summarizing the Results

Axial coding is the process in which the categories discovered and described in the previous sections are grouped together in larger themes. Looking at our data we can see different themes for the two types of information needs: the upstream needs and the downstream needs.

Downstream Needs.

The downstream needs are classified into three main categories which correspond to the lifecycle of a relationship with an upstream:

1. **Choosing an upstream**: Available public support, Implementation quality, License type, Comparison with similar upstreams.

2. **Learning about an upstream**: Documentation, Real contextual sample code

3. **Co-Evolving with an upstream**: Upstream changes, Compatibility with other upstream systems.

Looking at the current practices, we see that opportunities for research abound. The current practices are often manual and lack dedicated tool support.

Upstream Needs.

The upstream needs are classified into two main categories corresponding to the type of information the developers require:

1. **Downstream project statistics**: Downstream projects, Forked projects

2. **Downstream code usage**: API usage details, Runtime statistics, Code convention compliance

The developer motivation in this role is bi-modal: self esteem and the desire to ensure that the downstream users of their code benefit from it.

The current practices are well-supported with respect the first need but fail to support the second need. We envision this as an area of great potential for future research.

5. DISCUSSION

Our qualitative research method based on grounded theory does not guarantee completeness of our results [11]. Most of the results depend on the selected participants and their opinion and experience.

Begel *et al.* carried out a somewhat similar study at Microsoft [8]. They discovered that programmers are often interested in finding the people responsible for certain parts of the code base. In our case, the respondents were less interested in who wrote the code, but more interested in its quality and functionality.

Sillito *et al.* identified several categories of information developers need when trying to change a system [4]. All these are specific to their single-system evolution context and refer to implementation details (*e.g.*, method calls, data structures, type hierarchies). In contrast, we examined the needs that occur across different code bases.

To triangulate the results reported here we plan to run a follow-up study in which we will verify our results in a closed-question survey, where the participants do not answer with free text, but instead agree or disagree to the hypotheses we present.

6. CONCLUSIONS AND FUTURE WORK

The emergence of software ecosystems and the growing interconnectedness of software mark the beginning of software ecosystem research. The purpose of this study is to discover the upstream and downstream developer information needs in software ecosystems.

We have conducted email interviews with several software builders with both academic and industrial background and applied a grounded theory approach to analyzing the data. We have shown that upstream and downstream developers have different needs and that many of these needs are not adequately addressed in the current practices.

In the future we plan to cross-examine the results described in this paper by running a closed-question survey.

7. ACKNOWLEDGMENTS

We gratefully acknowledge the financial support of the Swiss National Science Foundation for the project "Agile Software Assessment" (SNSF project No. 200020-144126/1, Jan 1, 2013 - Dec. 30, 2015).

8. REFERENCES

[1] David G. Messerschmitt and Clemens Szyperski. *Software Ecosystem: Understanding an Indispensable Technology and Industry*. MIT Press, Cambridge, MA, USA, 2003.

[2] Jan Bosch. From software product lines to software ecosystems. In *Proceedings of the 13th International Software Product Line Conference*, SPLC '09, pages 111–119, Pittsburgh, PA, USA, 2009. Carnegie Mellon University.

[3] Mircea F. Lungu. *Reverse Engineering Software Ecosystems*. PhD thesis, University of Lugano, 2009.

[4] Jonathan Sillito, Gail C. Murphy, and Kris De Volder. Questions programmers ask during software evolution tasks. In *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*, SIGSOFT '06/FSE-14, pages 23–34, New York, NY, USA, 2006. ACM.

[5] Andrew J. Ko, Robert DeLine, and Gina Venolia. Information needs in collocated software development teams. In *Proceedings of the 29th international conference on Software Engineering*, ICSE '07, pages 344–353, Washington, DC, USA, 2007. IEEE Computer Society.

[6] Dominik Seichter, Deepak Dhungana, Andreas Pleuss, and Benedikt Hauptmann. Knowledge management in software ecosystems: software artefacts as first-class citizens. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, pages 119–126. ACM, 2010.

[7] Shaun Phillips, Guenther Ruhe, and Jonathan Sillito. Information needs for integration decisions in the release process of large-scale parallel development. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1371–1380. ACM, 2012.

[8] Andrew Begel, Yit Phang Khoo, and Thomas Zimmermann. Codebook: discovering and exploiting relationships in software repositories. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 125–134, New York, NY, USA, 2010. ACM.

[9] Anselm Strauss and Juliet Corbin. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications Inc., 1998.

[10] Paul Cairns and Anna L. Cox. *Research Methods for Human-Computer Interactions*. Cambridge University Press, 2008. Chapter 2, 7, 9.

[11] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. *Research methods in Human-Computer Interaction*. Wiley, 2010.