Empir Software Eng
DOI 10.1007/s10664-012-9211-2

How do open source communities blog?

Dennis Pagano · Walid Maalej

Editors: Arie van Deursen, Tao Xie and Thomas Zimmermann

Abstract We report on an exploratory study, which aims at understanding how software communities use blogs compared to conventional development infrastructures. We analyzed the behavior of 1,100 bloggers in four large open source communities, distinguishing between committing bloggers and other community members. We observed that these communities intensively use blogs with one new entry every 8 h. A blog entry includes 14 times more words than a commit message. When analyzing the content of the blogs, we found that committers and others bloggers write about similar topics. Most popular topics in committers' blogs represent high-level concepts such as features and domain concepts, while source code related topics are discussed in 15% of their posts. Other community members frequently write about community events and conferences as well as configuration and deployment topics. We found that the blogging peak period is usually after the software is released. Moreover, committers are more likely to blog after corrective engineering than after forward engineering and re-engineering activities. Our findings call for a hypothesis-driven research to (a) further understand the role of social media in dissolving the collaboration boundaries between developers and other stakeholders and (b) integrate social media into development processes and tools.

Keywords Social software · Open source · Data mining · Blogs · Social software engineering

1 Introduction

Social media enable the creation and exchange of user-generated content (Kaplan and Haenlein 2010). Individuals can use them to interact with, share information

D. Pagano (✉) · W. Maalej (✉)
Fakultät für Informatik, Technische Universität München, Munich, Germany
e-mail: pagano@cs.tum.edu, maalejw@cs.tum.edu

Published online: 25 May 2012

with, and meet other individuals presumably with similar interests, forming large data, knowledge, and user bases. In recent years the number of users and use-cases of social media has grown rapidly (The Nielsen Company 2010). For example, Facebook recently announced that it has more active users than the population of Europe.[1] The usage of Facebook, Blogger & Co. is no longer limited to private scenarios such as finding school friends, sharing photos, or keeping a vacation diary. Professionals use more and more social media e.g. to organize a conference, market a new product, or coordinate an open source project.

The software engineering community has also recognized the potentials of social media to improve communication and collaboration in software projects (Begel et al. 2010). For example, several studies have shown the role of Wikis for managing software documentation and collaboration (Maalej et al. 2008). Other authors suggested the integration of social media into development environments (Treude and Storey 2009; Guzzi et al. 2010; van Deursen et al. 2010). However, there exists no framework on the use of social media in software engineering. This paper takes a step towards such a framework by exploring the role of blogs as a popular social medium in open source communities.

The overall goal of our research is to increase the *socialness* of software, making the involvement of software users and their communities a first order concern of software systems and processes (Maalej and Pagano 2011). In particular, we aim at systematically utilizing valuable user experiences and volunteered resources, e.g. published in communities. We hypothesize that social media represent an important enabler for dissolving the communication and collaboration boundaries between developers and other stakeholders. We therefore envision a more intensive and scientific use of social media in software engineering projects.

As a first step, we investigate in this paper how open source communities currently use social media. We divide the blogging ecosystem in active software developers and other stakeholders (including the users). We are particularly interested in understanding how and why these groups use blogs, and how their blogging activities are related to other project activities. From the results we try to draw conclusions on how current communication means between developers and other stakeholders can be improved (e.g. by revising software tools and processes), and how otherwise stakeholders (in particular users) can be stronger involved in the development lifecycle.

In the following we report on an exploratory study, which examines the content and metadata of blogs in four large open source communities. The contribution of the study is threefold. First, it gives empirical evidence on what a developer's blog post typically looks like. Second, it explores usage patterns of blogs during development and identifies dependencies to other activities: in particular committing code and releasing software. Third, it gives first insights to tool vendors and practitioners into how to better integrate social media into development tools and processes.

The remainder of the paper is structured as follows. Section 2 introduces the research questions, research data, and methodology used. The following three sections summarize our research findings on the usage of blogs (Section 3), the information included (Section 4), and the dependencies between blogging, releasing,

[1] http://www.facebook.com/press/info.php?statistics

and committing activities (Section 5). Section 6 discusses our findings while Section 7 presents their limitations. Section 8 surveys related work focusing on studies, which analyze social media and similar artifacts. Finally, Section 9 concludes the paper and sketches our future plans.

2 Research Setting

We first summarize the questions that drive our research. Then, we describe the overall method we used to collect and analyze the data. Finally, we present the actual data sets collected to perform our analysis.

2.1 Research Questions

Our research goal is to understand how and why blogs are currently used by different groups in a software community. We focus on three aspects: the actual *usage* of blogs in software projects, the *content* of these blogs, and the *integration* of blogging activities into the development workflows.

Blog usage describes *how* software development communities blog (i.e. share information in blogs). For that, we analyze the publishing frequency as well as the structure of blog posts, answering the following questions:

- *Publishing frequency:* How often do community members blog?
- *Post structure:* What are typical elements of a blog entry and how often are they included or referenced?

Blog content describes the *information* published in the blogs. This includes identifying topics (i.e. semantic entities) and their frequencies. In particular we answer the following questions:

- *Topics:* Which topics are discussed in blogs of development communities?
- *Topic popularity:* How popular are these topics (i.e. frequency distribution) across different communities?

Blog integration describes *how* blogging activities are *integrated* in the development workflows. We examine usage patterns and content dependencies between blogs, source code repositories, and release repositories answering the following questions:

- *Publishing patterns:* Are there particular patterns, which describe when blogs are used in the communities? In particular:
  - Release dependency: When are blogs posted in relationship to the software releases?
  - Activity dependency: When are blogs posted in relationship to particular development activities?
- *Published Information:* Are there relationships between the work performed and the information blogged? In particular:
  - Content dependency: Are blog post topics and particular development activities related?

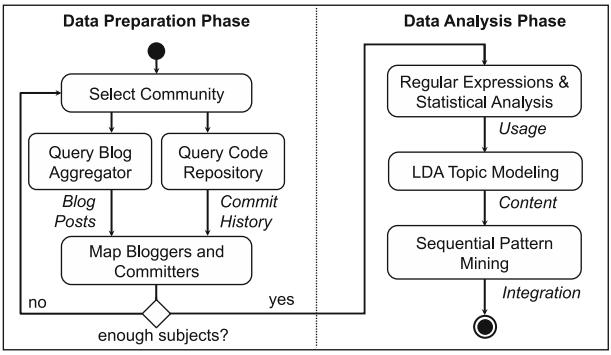

Fig. 1 Research method

– Time dependency: To which degree are work performed and information blogged related in terms of time?

When answering these questions we distinguish between active developers (*committers*) and other bloggers (*others*). This allow us to compare the behavior of developers and other stakeholders in the studied communities.

2.2 Research Method

Our research method consisted of two phases: a data preparation and a data analysis phase, as depicted in Fig. 1.

2.2.1 Data Preparation Phase

In the data preparation phase, we selected the development communities, from which research data is to be collected. To qualify for our study a community must have:

– A public source code repository and a public release history.
– More than 100 bloggers. In a community with 100 different bloggers blogs are popular enough to be studied and give a representative impression on the community.
– More than 1,000 project-related (e.g. not private) blog posts. This threshold provides enough data to discover statistical significant dependencies and means about the blogs.
– More than ten blogging committers. Studying at least ten different blogging committers gives enough variation to get common and different behavior of the developers.

Our study is of exploratory nature and is not designed to be generalizable to an arbitrary software development project. For the studied cases these criteria are meant to provide enough data and subjects to represent the community.

To select the communities based on the above criteria, we analyzed Planet,[2] a popular *blog aggregator.* Planet is used by 43 open source communities to collect

[2] www.planetplanet.org

blog posts of their members in a single place. Each entry in a particular "planet" (i.e. community aggregator such as planet Mozilla) includes meta information about the name of the blogger and the link to the original blog post. Additionally, the aggregator lists all contributing bloggers, who are usually selected by a community board. The selection underlies a strict quality policy, since the primary goal of the common blog is to provide project related knowledge. It is important to notice that in this study we did not investigate comments or responses to blog posts, but only the original blog posts.

After querying the blog aggregator and the source code repository, we mapped the blog posts to the commit history based on the authors' names. This mapping enabled us to distinguish between bloggers who commit source code and other bloggers in the community.

We first built a list of blogger names and a list of committer names. The real names in the lists were not identical (e.g. "David Wheeler" and "David E. Wheeler"). In addition most blog posts include the real name of their authors, while most commits contain only login names. Therefore, we created login name candidates from the blogger's real name (e.g. "glefur" from "Goulwen Le Fur"). We also observed that several commit messages contain meta information such as "Author: 10:44:52 Tim Janik", "Author: PST 2006 Michael Emmel". We extracted this information using regular expressions and manually corrected the author names when required.

We then compared the entries in the blogger and committer lists using a text similarity algorithm,[3] which is based on the Dice similarity coefficient on bigrams (Dice 1945):

$$d(X, Y) = \frac{2\,|X \cap Y|}{|X| + |Y|} \quad \text{, for two strings } X, Y.$$

The intersection of two strings is defined as the set of common *bigrams*. The Dice coefficient is frequently used in information retrieval to find exact matches as well as slightly diverging pairs. It evaluates to 100% if and only if the words are equal. For instance, the strings "Dennis" and "Tennis" have a Dice coefficient of 80% while "Walid" and "Kalid" lead to a coefficient of only 75%. Although both pairs only differ in one letter, the former includes more common information, since the strings are longer.

Consequently, login names should match more precisely than real names, as they are shorter. We manually investigated random probes of the matchings to find suitable thresholds and ensure the correctness of this approach. We finally used a threshold of 80% to match two real names (e.g. "David Wheeler" and "David E. Wheeler"). To match two login names (e.g. "wheeler" and "dwheeler") we required their Dice coefficient to be above 90%.

2.2.2 Data Analysis Phase

The data analysis phase consisted of three steps, which respectively answer the usage, content, and integration questions. To analyze the *usage* of blogs we applied descriptive statistics (for frequency calculation) and regular expressions (for analyzing the blog structure). We also conducted statistical tests to exclude the hazard factors

[3] http://www.catalysoft.com/articles/StrikeAMatch.html

Table 1 Overview of collected data

	Eclipse	GNOME	PostgreSQL	Python
# posts	10,333	18,323	3,385	18,660
# bloggers	328	342	112	405
# commits	239,659	252,831	30,745	45,116
# committers	467	2,294	34	178
# blogging committers	93	250	12	34

and report on the error rates. To analyze the blog *content* and included information we used the Latent Dirichlet Allocation (LDA) topic modeling technique (Blei et al. 2003). When applied on the blogs, this technique extracts keywords which belong together and groups them as topics. Finally, to study the *integration* aspect we ordered commit messages and blog posts as well as commit messages and releases by time and investigated the resulting stream of events. Thereby we looked for patterns and regularities using Sequential Pattern Mining (Agrawal and Srikant 1995). We detail on each of these analysis steps in the corresponding result section.

2.3 Research Data

Our data preparation phase led to the selection of four large open source software communities: Eclipse, GNOME, PostgreSQL, and Python. Table 1 shows an overview of the collected data, which we introduce in the following.

Eclipse is an open development platform, which comprises extensible frameworks, tools, and runtime environments to build, deploy, and manage software systems. The Eclipse community is one of the largest open source communities including 11 million users and more than a thousand active developers. From Planet Eclipse[4] we collected 10,333 blog posts of 328 community members over the last seven years. Eclipse comprises over one hundred sub-projects which use separate source code repositories. We asked two Eclipse committers and one Eclipse blogger (all were involved for around three years) to name the most active projects in the community. Based on their independent feedback, we selected the following 22 projects: atl, cdo, cdt, compare, dsdp, dtp, e4, ecf, eclipse platform, emf, equinox, gef, gmf, jdt, m2m, m2t, mylyn, pde, rap, riena, swt, and xtext. From the repositories of these projects we collected 239,659 commit messages of 467 developers, written over the last ten years.

GNOME is a large community, which develops a free and open source desktop environment. The GNOME community describes itself as "a worldwide community of volunteers who hack, translate, design, QA, and generally have fun together." From Planet GNOME[5] we collected 18,323 blog posts of 342 community members from the last ten years. GNOME includes about hundred sub-projects, which use different source code repositories. As in Eclipse we selected the following list after asking two active committers and one blogger (all were involved for three to four years) for the most active sub-projects: banshee, empathy, eog, epiphany, evolution, f-spot, gdk-pixbuf, gdm, gedit, gimp, glib, gnome-applets, gnome-bluetooth, gnome-control-center, gnome-disk-utility, gnome-keyring, gnome-packagekit, gnome-panel,

[4] planeteclipse.org

[5] planet.gnome.org

gnome-power-manager, gnome-session, gnome-shell, gnome-terminal, gnome-utils, gnome-vfs, gparted, gtk+, gvfs, libgnome, metacity, nautilus, pitivi, policykit-gnome, seahorse, and totem. For these 34 sub-projects we collected 252,831 commit messages of 2,294 developers, written over the last 14 years.

PostgreSQL is an open source database management system developed and maintained by a global community of developers and companies. The PostgreSQL community uses two planet websites[6] with slightly different contributors. We merged both contributor lists and obtained 3,385 blog posts of 112 active members from the last seven years. PostgreSQL is a single project hosted in a single source code repository. We collected 30,745 commit messages from 34 developers, written over the last 15 years. PostgreSQL accounts for the smallest data set.

Python is an interpreted, general-purpose programming language, which emphasizes code readability and maintainability. From Planet Python[7] we collected 18,660 blog posts from 405 community members, published over the last eight years. Thus, Python comprises the largest number of bloggers and blog posts in our data. Like PostgreSQL, Python uses a single source code repository. We collected 45,116 commit messages from 178 developers, written over the last 20 years. Python is the oldest project in the collected data.

After the data preparation phase we found 93 matches of committer and blogger names in Eclipse, 250 in GNOME, 12 in PostgreSQL, and 34 in Python. These people are actively committing source code and blogging in their community.

3 Blog Usage

We explore *how* the studied communities use blogs, by analyzing the publishing frequency and the post structure.

3.1 Publishing Frequency

We studied the publishing frequency for the whole communities as well as for the individual bloggers, while distinguishing between active developers (committers) and other bloggers. Table 2 shows the number of posts by committers and others in our data sets. In all communities we found less entries from committers (13 to 26% of all entries) than from other bloggers, except in GNOME. Committers in this community contributed 81% of all blog posts. This result originates not least from the distribution of committers and other bloggers in the corresponding data sets (see Table 1).

In all studied communities we observed several blog posts each day. The mean time between two successive blog posts within a community is 8.1 h. On average, Python is the most active community with one blog post each 3.9 h. GNOME bloggers publish one blog post each 4.9 h on average. The mean time between two posts in the Eclipse community is 5.6 h. PostgreSQL is the least active community with a post each 18 h. This means the bigger the community is, the more frequent blog posts it has.

[6] planet.postgresql.org and www.planetpostgresql.org

[7] planet.python.org

Table 2 Blog post ratios of committers vs. others in the studied communities

	Eclipse	GNOME	PostgreSQL	Python	Total
Committers	2,651 (26%)	14,893 (81%)	432 (13%)	3,025 (16%)	21,001 (41%)
Others	7,682 (74%)	3,430 (19%)	2,953 (83%)	15,635 (84%)	29,700 (59%)

We observed that the *committing frequency* in all four communities is significantly higher than the respective *blogging frequency*. On average, Eclipse and GNOME developers committed a source code change twice per hour, while PostgreSQL and Python developers once in 4 h. This reflects also that the Eclipse and GNOME communities are larger than PostgreSQL and Python in terms of their development efforts.

Next, we study the total number of blog posts for the individual bloggers. The distribution of the individual number of blog posts is positively skewed in all data sets. We therefore use medians rather than means to describe the distribution. In all communities, committers had written more blog posts than other bloggers. As shown in Table 3 we found the biggest difference in the Python community, where committers wrote more than twice as many posts as other bloggers. PostgreSQL committers blogged nearly twice as much as other community members. In all four data sets, 75% of the committers had published more than ten posts, while 75% of the other bloggers accounted for less than 50 posts. For all communities but PostgreSQL the difference between the median number of blog posts by committers and others is statistically significant. A two-sample Wilcoxon rank sum test with continuity correction rejected the null hypothesis that the medians are equal ($p < 0.05$ for all communities but PostgreSQL: $p = 0.073$, CI = 95%).

Analyzing the individual contributions, we found that committers in general blog more frequently than other community members. Figure 2 shows the publishing frequency of bloggers in the four communities. The distributions of the frequencies in the data sets are positively skewed. There are bloggers who post only once in seven months. We therefore use again medians rather than means to describe the distribution. The average time between two successive blog posts based on median lies between 17 and 33 days in all data sets. For committers the median blog post rate is 26 days (39 days based on mean). 75% of all committers publish a blog post latest every 44 days. Other bloggers in the communities post once in 28 days (43 days mean). We conclude that in all communities except Eclipse, committers posted more frequently than other bloggers. However, neither a two-sample Wilcoxon rank sum test for medians, nor a two-sample t-test for means could show that the observed difference is statistically significant (medians: $p = 0.15$, means: $p = 0.079$).

Eclipse committers publish every 34 days based on median and 44 days based on mean. Other bloggers in the Eclipse community post every 29 days based on median and every 40 days based on mean. 25% of the Eclipse committers post less often than once in 54 days, while 75% of other bloggers in the community post more often than once every 50 days. We discussed this observation with active members of the

Table 3 Total number of blog posts per person (medians)

	Eclipse	GNOME	PostgreSQL	Python
Committers	18	28	21	35
Other bloggers	14	24	11	17

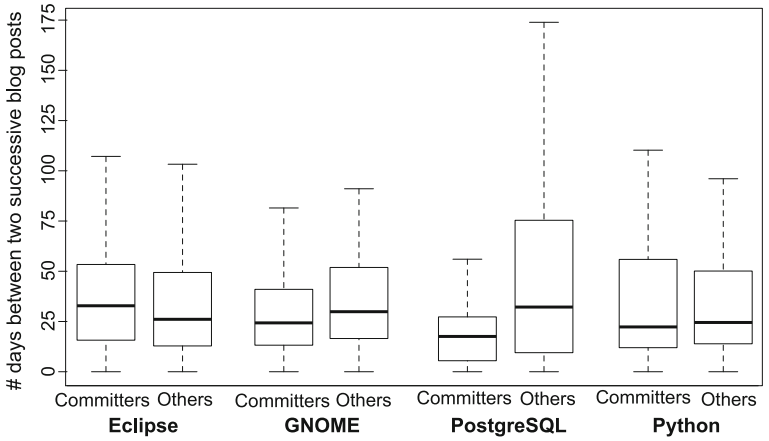

Fig. 2 Publishing frequency of bloggers

Eclipse community (committers and other bloggers). We also randomly selected five frequent Eclipse bloggers and investigated their web profiles and activities. We found that Eclipse *evangelists* regularly provide information to the community without being actively involved in the development of the software itself.

Finally, we analyzed *how long* the community members have used blogs. Committers had a medium blog usage time of 2.2 years. 75% of them have used blogs longer than 1.2 years. The medium usage time of other bloggers is 1.6 years, 75% of them have blogged for less than 2.6 years. This observation is statistically significant for both medians and means (two-sample Wilcoxon rank sum test with continuity correction for medians, two-sample t-test for means, $p < 0.001$).

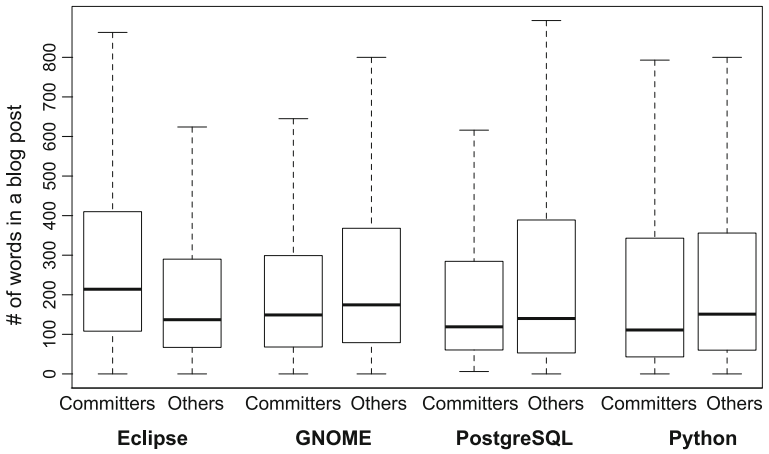

Fig. 3 Average lengths of blog posts

Table 4 Blog posts containing source code

	Eclipse	GNOME	PostgreSQL	Python	Total
Committers	332 (13%)	884 (6%)	46 (11%)	438 (14%)	1,700 (8%)
Other bloggers	583 (8%)	209 (6%)	579 (4%)	2,674 (17%)	4,045 (14%)

To summarize, studied open source communities frequently and continuously publish information in blogs. Committers blog slightly more often and for longer time periods than other community members.

3.2 Post Structure

To study the structure of blogs, we examined the *length* of the posts and analyzed the included *source code*, *links*, and *images*. We used regular expressions to extract these elements. In the remainder of this section we describe the analysis results.

3.2.1 Blog Post Length

The median blog post length is 150 words (273 words average), which is about 14 times the median length of a commit message (11 words) in our data sets. As shown in Fig. 3, the distribution of post lengths is positively skewed in all data sets, since single posts comprise several thousand words. The longest blog post of our data sets was entered in the GNOME community. It contains 9,265 words and describes experiments with a new Linux init system. 1,102 posts (2%) comprise less than ten words. Examples of posts with only three words are "GNOME lacks stetic" and "Amazing, absolutely amazing". Over 95% of all posts are shorter than 1,000 words, which corresponds to four printed pages.

On average, committers of all studied communities except Eclipse write shorter posts than other bloggers (for Eclipse, GNOME, and Python two-sample Wilcoxon rank sum test leads to $p < 0.001$, PostgreSQL: $p = 0.14$). The Eclipse committers account for the longest blog posts across all communities (214 words median). Over 75% of their blog posts are longer than 100 words. The Python committers on the other hand account for the shortest blog posts, 25% of which comprise less than 45 words.

3.2.2 Source Code

We scanned the HTML source of the blog posts for the tag "`<pre>`", which is commonly used to markup source code paragraphs. From all 50,701 studied blog posts we were able to find source code paragraphs in 5,745 posts (11.3%).[8] As shown in Table 4, GNOME has overall the least amount of blog posts with source code while Python has the highest. This reflects the nature of these systems. The use of Python as an API library is rather source code based, while GNOME present other aspects such as user interfaces. A senior GNOME committer, who regularly attended

[8]In an earlier version (Pagano and Maalej 2011) we reported a much smaller rate. After a manual analysis of 100 randomly selected blog posts, we discovered that the original scanning algorithm missed certain blocks. We are more confident with the results presented in this paper.

Table 5 Blog posts containing links

	Eclipse	GNOME	PostgreSQL	Python	Total
Committers	2,384 (90%)	12,418 (83%)	366 (85%)	2,332 (77%)	17,500 (83%)
Others	6,171 (80%)	2,893 (84%)	1,934 (65%)	12,328 (79%)	23,326 (78%)

community events, confirmed this finding. Since April 2007 he contributed over 600 commits, and also blogged occasionally with 14 blog posts since December 2009. He explained that developers only include source code in their posts to show "new cool features" and explain "how things are done right".

In Eclipse and PostgreSQL committers published more source code in their blogs than other bloggers as opposed to Python. In the GNOME community, both groups show approximately the same behavior. However two-sample t-tests could not prove that these differences are statistically significant ($p > 0.05$). *Committers and other bloggers tend to include source code only in about every tenth post.*

3.2.3 Links

We observed that links are frequently included in blog posts. In total of 40,826 posts (80.5%) contain links. We compared the behavior of active developers against other community members in including links in their blog posts. We found that on average 83.3% of committers' posts and 78.5% of other bloggers' posts contain links, as shown in Table 5.

We randomly selected 400 blog posts from each community (200 posts from developers and 200 posts from other bloggers) and investigated their links using regular expressions and manual peer reviews. This sample included in total 5,313 links. In a first step we removed about 19% of all links, which we rated as off topic (e.g. links to private sites). In the remaining links, we found that committers on average included more links to Wikis (11%) than other bloggers (8%). Similarly, the posts of committers included more links to other blog posts (28%) than posts of other community members (25%). The remaining links were links to project-related websites (39%) like the official project website, reference documentation of source code (7%), source code (7%), other downloads (3%), issue trackers (3%), newsgroups (1%), micro-blogs (1%), project download pages (1%), and videos (1%).

We think that committers tend to reference knowledge rather than to re-write or copy it more frequently than other community members. Committers also seems to know public knowledge sources better than other bloggers.

3.2.4 Images

Last, we investigated the usage of images in our data sets. We found that 14,605 blog posts contain images (28.8% of all posts). As shown in Table 6, developers in all

Table 6 Blog posts containing images

	Eclipse	GNOME	PostgreSQL	Python	Total
Committers	1,463 (55%)	5,523 (37%)	41 (10%)	274 (9%)	7,301 (35%)
Others	2,672 (35%)	1,387 (40%)	343 (12%)	2,902 (19%)	7,304 (25%)

Table 7 Images in blog posts (semi-automated analysis)

	Screenshots	Community	Graphics	Diagrams
Committers	22.3%	6.8%	8.9%	3.4%
Other bloggers	18.0%	6.2%	3.3%	2.6%
Total	20.1%	6.5%	6.1%	3.0%

communities but Eclipse publish about 2 to 10% less images in their blogs than other community members. In Eclipse, every second developer post includes images while every third post of other community members does.

Overall the Eclipse and GNOME communities upload more images (> 37% of all posts) than the PostgreSQL and Python communities (< 18%). We think that this partly results from the fact that several sub-projects of Eclipse and GNOME are user interface projects, while in PostgreSQL and Python user interfaces are secondary concepts.

Again we created a random sample of 400 blog posts from each community and manually assessed 1,231 included images. We found that 25% of all included images are thumbnails of social bookmarking sites, comment counters, or other automatically added images. 15% of all images were not accessible (broken links) and 24% were off topic (e.g. vacation pictures). About 20% of the images are screenshots, 7% community pictures (e.g. conferences or meetings), 6% graphics such as function plots and charts, and 3% diagrams (e.g. UML diagrams). Table 7 shows that committers tend to use slightly more screenshots (22.3%) in their posts than other bloggers (18.0%).

To summarize, studied software communities and in particular active developers use blogs to communicate on a relative high level of abstraction. Further, based on the publication of community pictures, it seems that the community itself is an important topic in blogs.

4 Blog Content

We analyzed the content of the blogs to find out *which information* is included in community posts and *how frequently*. We used the Latent Dirichlet Allocation (LDA) topic modeling technique (Blei et al. 2003) to derive topics from the blog posts in each project. With LDA a topic emerges as a set of words that are correlated with a certain probability because of their co-occurrence in the same document. Our topic extraction process involved four steps. *First*, we created two document corpora for each community: the first comprising all blog posts of the *active* (committing) developers, and the second comprising all other blog posts in the community. To create these corpora we used a list of English stop words and a stemmer to remove word inflexions beforehand. *Second*, we performed multiple runs of the LDA algorithm on the four data sets and experimented with different numbers of topics. We found that using 50 topics leads to the most meaningful results (i.e. a total list of 200 topics from the four data sets). *Third*, both authors independently inspected the results and manually added topic descriptions (labels) based on the top 20 most influential words (obtained by LDA). To support the labeling, we randomly selected few associated blog posts and analyzed them manually until we identified a topic label which covers the different aspects of the associated words. We repeated

this step until the agreements on the labels by the authors were over 90%. In this step, a label could be assigned to multiple topics. This led to fewer descriptions than the original topics. We observed the resulting merged topics with the same labels (i.e. topics with the same semantics but potentially different words or word sets) are more consistent than the results of running LDA with an up-front reduced number of topics (i.e. pure syntactic clustering). *Fourth*, we grouped similar topics across the data sets to project-independent *themes* (such as grouping the topics "*domain concepts*", "*legal requirements*", and "*features*" to the theme "*requirement*").

To quantify the *popularity* of the topics we calculated the occurrences of the topics within the blog posts using the document-topic matrix from LDA. Since a single blog post may contain multiple topics, we selected the most predominant topics per post by evaluating for each post and topic the number of words in the post belonging to that topic. We defined a topic to be predominant if at least 10% of the words in the blog post belong to the topic. This threshold results from two observations:

1. Single words from most topics are present in a large number of posts (e.g. "use"). These words do not determine the topic sufficiently without other words from that topic.
2. Over 90% of the posts contain at least one predominant topic.

In the following we report on the topics included in committers' blogs as well as in the posts of other bloggers. Then we discuss themes and compare their popularities among committers and other bloggers.

4.1 Topics of Committers

Table 8 shows the list of topics extracted from committers' blogs across the four communities with their popularity and examples of influential words. From 200 topics we were able to identify semantic descriptions for 194 topics with 23 labels. The remaining six topics exhibited heterogeneous words with inconsistent or no meanings in the associated blogs (e.g. "word", "net", "50", "77", "att", "en", "45", "resolv", "ironpython", "de", "på", "och", "det", "et", "silverlight", "som", "int", "85", "ou", "spreadsheet"). These "unknown" topics together have a popularity of 12.3%.

We found that the most popular topic is *"features & domain concepts*". This topic is predominant in around 42% of all blog posts. Further the topics *"community & contributions"* as well as *"API usage & project documentation"* are predominant in about a third of the blog posts over all studied communities. The topic "*source code*" is covered in less than 15% of all blog posts.

Table 9 shows the most popular topics with their corresponding frequencies among the communities. With the exception of PostgreSQL, the topic "*architecture & packages*" is predominant in about one third of all posts. However, in the PostgreSQL data set we were unable to extract this topic. Python blogs frequently include information about "*API usage & project documentation*". As this is an old, infrastructure project (> 19 years) we think that bloggers particularly stress the reuse of its API. In PostgreSQL blogs *non-functional requirements* represent a popular topic. This is reasonable, since such requirements are crucial for a database system (e.g. performance, security, or scalability).

Table 8 List of identified topics in the blogs of committers

#	Topic description	Pop.	Examples of influential words
C1	Features & domain concepts	42.2%	radio, listen, player, sync, song, music, play, ipod, album, artist, band
C2	Community & contributions	37.7%	people, community, contribute, group, help, news, post, comment
C3	API usage & project documentation	30.0%	wiki, write, project, api, document, use, review, text, output
C4	Release management & announcements	28.6%	release, try, helios, download, board, committee, foundation
C5	Solution concepts & technology	26.8%	rest, uri, response, rule, parser, syntax, widget, javascript, client
C6	Architecture & packages	24.7%	start, component, register, import, service, osgi, bundle, framework
C7	Target platform	23.4%	linux, android, vm, platform, device, system, run, environment
C8	Deployment & dependencies	20.9%	ant, zip, publish, target, jar, install, depend, distribute, plugin
C9	Conferences	17.4%	session, democamp, present, eclipsecon, conference, event, talk
C10	Development activities	16.3%	work, implement, develop, test, code, improve, task, maintain
C11	Non-functional requirements	15.7%	cache, memory, perform, high, quality, limit, secure, cost
C12	Communication, discussion	15.7%	send, address, mail, call, discuss, answer, question, decision, phone
C13	Debugging & troubleshooting	15.0%	debug, address, process, warning, problem, exception, raise, error
C14	Licensing	14.8%	free, company, open, source, community, business, foundation
C15	Source code	14.7%	void, new, import, public, final, string, class, return, private, true
C16	Competitors & related work	14.6%	more, think, performance, product, oracle, sun, mysql, experience
C17	Version control	13.3%	trunk, commit, repository, merge, clone, svn, git, master, csv, push
C18	Tips, tricks & tutorials	11.2%	tutorial, tool, practical, summary, support, article, website
C19	User interface & user interaction	11.0%	tab, view, menu, dialog, button, text, mockup, select, click, interact
C20	Corrective maintenance	10.2%	support, report, fix, improve, bug, bugzilla, issue
C21	Database access & external data	10.1%	jpa, import, store, table, sqlite, database, schema
C22	Testing	7.5%	write, test, case, manual, unit, check, build, system, patch, junit
C23	Continuous integration	2.5%	resource, test, source, build, configure, hudson, project, generate

Table 9 Most popular topics in committers' blogs

Eclipse	GNOME	PostgreSQL	Python
Features & domain concepts (47%)	Community & contributions (36%)	Features & domain concepts (47%)	API usage & project documentation (50%)
Community & contributions (33%)	Features & domain concepts (33%)	Non-functional requirements (40%)	Features & domain concepts (42%)
Architecture & packages (31%)	User interface & user interaction (32%)	Community & contributions (39%)	Community & contributions (42%)
Target platform (26%)	Architecture & packages (32%)	Release management & announcements (38%)	Deployment & dependencies (36%)
Solution concepts & technology (26%)	Development activities (31%)	Conferences (25%)	Architecture & packages (36%)

To summarize, developers include more high-level than low-level concepts in their blogs. The large amount of posts dealing with community aspects fits to the "social nature" of blogs. In particular open source communities depend on the active discussion of social aspects, dissemination of community news, and requests for contribution.

4.2 Topics of Other Bloggers

Table 10 shows the list of topics extracted from the blogs of other community members. From the studied 200 topics we were able to label 177 topics with 22 labels. The remaining 23 topics included heterogeneous words or noise. We were unable to agree on their semantics even after consulting associated blog posts. The popularity of these "unknown" topics is 35.8%. Most of the identified topics were also found in the committers' blogs, except "*education, learning & training*", "*development tools & technology*", "*applications, related tools & technologies*", and "*business & industrial use*".

The results are slightly different from the topic popularities in the committers' blogs (shown in Table 8). Other bloggers blog most frequently about "*community & contributions*". This topic is predominant in about 38.6% of their blogs. The second topic is "*features & domain concepts*" (38%), which we had found to be the most frequent committer topic. Unlike in committers' blogs, the topic "*deployment, configuration & dependencies*" is among the top five topics. Other bloggers seem to share more knowledge about using and configuring the software systems than committers in the studied communities. Surprisingly, "*solution concepts & technology*" is also ranked among the most popular topics. When looking at several posts with these topics, we found that the four studied software systems have rather a framework nature. Other bloggers, who are not contributing to these frameworks, often use them to build their own tools. Thereby they make considerable experience on how to engineer specific tasks. Therefore blogs of other community members often share knowledge about solution concepts and how-to's.

Table 11 shows the most popular topics in the different communities. The topic "*deployment, configuration & dependencies*" is predominant in more than 40% of all posts in the Eclipse and PostgreSQL communities. We think that this is an indicator for the variety of different installations and configurations of these systems. Moreover, we found the topic "*source code*" among the top five topics in the Python community, which is reasonable as Python is a programming language.

Table 10 List of identified topics in the blogs of other community members

#	Topic description	Pop.	Examples of influential words	#C
O1	Community & contributions	38.6%	community, member, meet, discuss, people, blog, mentor, team, comment, contribute, newsgroup	C2
O2	Features & domain concepts	38.0%	use, feature, video, audio, effect, player, flash, filter, user, apply	C1
O3	Solution concepts & technology	37.1%	thread, send, close, websocket, deadlock, request asynchronous, handle, response	C5
O4	Deployment, configuration & dependencies	35.9%	install, deploy, system, include, configure, jar, file, web.xml, plugin	C8
O5	Conferences	26.4%	present, eclipsecon, talk, interesting, conference, attend, session, people, guadec, thank, organize	C9
O6	Release management & announcements	23.0%	release, galileo, europa, plan, update, version, now, announce, available, download	C4
O7	User interface & usability	21.4%	swing, grid, style, button, look, theme, design, usable, switch, widget, window, gui, form	C19
O8	Source code	20.6%	int, string, void, return, class, implements, extends, public, import, new, final	C15
O9	Version control	19.5%	repository, change, commit, file, store, svn, local, version, git, branch, merge, push, trunk	C17
O10	Target platform & operating system	18.3%	platform, phone, device, android, linux, system, mobile, nokia, iphone, hardware, ubuntu	C7
O11	Licensing, legal & commercial use	17.9%	source, open, foundation, license, brand, commercial, gpl, company, trademark, free	C14
O12	Education, learning & training	16.6%	webinar, video, demo, show, present, learn, school, student, university, education, research	
O13	Development tools & technology	16.5%	script, tool, source, project, pydev, dlr, debug, editor, emacs, ide, highlight, vim, eclipse	
O14	Software evolution	15.0%	version, extension, now, patch, bug, fix, feature, improve, new, issue, report, change, update, major	C20
O15	Non-functional requirements & quality	14.9%	buffer, performance, time, statistic, more, checkpoint, tune, optimize, benchmark, high	C11
O16	Development activities & work descriptions	14.7%	team, design, develop, process, concept, time, implement, day, start, work, week, month, today	C10
O17	User support	13.3%	user, problem, wiki, patch, need, help, start, show, work, support, please, soon, thank, beta, conflict	C18
O18	Testing	11.4%	test, automate, junit, unit, hudson, write, case, doctest, suite, assert, fail, run, code, unittest, pass	C22
O19	Applications, related tools & technologies	9.3%	software, adobe, mac, windows, pro, upgrade, microsoft, edition, cable, power, modem, oracle	
O20	Persistency management	9.3%	database, create, store, server, sync, service, row, query, key, data, table, model, insert	C21
O21	Frameworks & architecture	9.3%	rcp, rap, sdk, android, layer, osgi, service, provider, framework, spring, distribute, bundle, ajax	C6
O22	Business & industrial use	6.1%	business, market, company, job, public, interest, cost, person, industry, manage, offer, vendor	

Table 11 Most popular topics in other bloggers' posts

Eclipse	GNOME	PostgreSQL	Python
Community & contributions (49%)	Community & contributions (44%)	Solution concepts & technology (46%)	Features & domain concepts (37%)
Deployment, configuration & dependencies (44%)	Features & domain concepts (32%)	Deployment, configuration & dependencies (46%)	Solution concepts & technology (35%)
Features & domain concepts (42%)	Conferences (29%)	Features & domain concepts (41%)	Development tools & technology (34%)
Solution concepts & technology (39%)	User interface & usability (28%)	Target platform & operating system (33%)	User interface & usability (31%)
Release management & announcements (38%)	Solution concepts & technology (28%)	Community & contributions (33%)	Source code (30%)

To summarize, other bloggers exhibit a similar blogging behavior as committers, writing about many high-level and few low-level topics. We observed frequent posts about community aspects and about using and configuring the studied software systems.

4.3 Comparison of Themes

To further interpret and compare the information included the different blogs, we grouped the resulting topics into the following themes:

1. **Requirements**. This theme contains topics that are related to requirements engineering, particularly *application domain concepts*, *features*, *user interface design*, *legal requirements and licenses*, as well as *non-functional* requirements. Topics describing *competitors* or describing *related work*, as well as discussions about *business* strategies and the *open source* and *closed source* character of a project belong to this theme as well.
2. **Community**. These topics represent community and social aspects like *contributions* of specific members, *communication*, and *project news*. *Conferences* organized by the community that provide possibilities to meet, learn, and exchange belong to this theme as well.
3. **Project knowledge**. This theme captures topics related to knowledge and public information in a project. The system's *API*, *documentation*, and information regarding the *usage of the system* belong to this theme. But also *tips and tricks,* documented *user support* as well as *tutorials* are a form of project knowledge.
4. **Deployment**. Topics in this theme deal with information regarding the deployment and configuration of a system. This includes topics describing artifacts and processes of system *deployment*, but also *dependencies* and *plugins*. Information about the *target platform* as well as the *runtime environment* is also included in this theme. Further, this theme contains topics about deployment context, i.e. other technological relationships such as other software *applications*, *related tools and technologies*.

5. **Management**. This theme comprises all management related topics. Apart from *release management*, we found several other management topics like project and technology management. As representatives of configuration management we found *continuous integration* and *version control*.
6. **Implementation and Design**. This theme describes implementation details like *source code*, *solution domain concepts*, and other *technology* related artifacts. Topics that describe *architecture concepts* and discussions about dependencies and access to *external data* belong to this theme as well. Further, this theme contains topics describing development tools.
7. **Activities and Tasks**. This theme contains topics describing *development activities and task* such as *corrective maintenance* and *testing*, *debugging*, *troubleshooting*, *modeling, and implementing*.

Table 12 shows the identified themes and their popularities by committers and other bloggers across the communities. The theme popularity denotes the percentage of all blog posts that contained at least one of the associated topics in the according community.

The three most popular themes are common among committers and other bloggers. However, the popularity ranks are slightly different. The most popular theme in both cases is "*Requirements*", which is predominant in more than half of all committers' blog posts, and in nearly half of all other bloggers' posts. The second most popular theme in committers' blogs is "*Community,*" which is predominant in around 45% of all their posts. In the posts of other bloggers this theme ranks third, with 43% popularity. Since community building is a major goal of social media this result seems reasonable. "*Implementation and Design*" topics are present in about 38% of committers' and 43% of others' posts. It seems that developers use blogs primarily to document system requirements and features. Source code and low-level concepts are rather less frequently discussed. This difference is particularly high in committers' posts (54% "*Requirements*" vs. 38% "*Implementation and Design*").

Table 12 Themes and their popularity among committers and other bloggers

Theme	Mean	Eclipse	GNOME	PostgreSQL	Python
Blogging committers					
Requirements	54%	58%	40%	66%	49%
Community	45%	39%	40%	50%	52%
Implementation and design	38%	46%	37%	23%	48%
Project knowledge	34%	30%	28%	30%	51%
Deployment	33%	32%	32%	31%	38%
Management	33%	30%	31%	43%	29%
Activities and tasks	31%	19%	35%	36%	35%
Other bloggers					
Requirements	48%	49%	40%	56%	45%
Implementation and design	43%	51%	31%	46%	45%
Community	43%	50%	47%	40%	34%
Deployment	37%	44%	32%	46%	27%
Activities and tasks	33%	40%	29%	31%	34%
Project knowledge	32%	35%	23%	41%	30%
Management	24%	41%	16%	18%	21%

The remaining four themes differ slightly regarding their popularities among committers and other bloggers. While committers tend to write more about *project knowledge*, other bloggers include more information about the *deployment* and *configuration* of the system in their posts. Other bloggers share more information about using and configuring the software in particular environments, while committers tend to share more generic project knowledge. Among other bloggers, the "*management*" theme has the least popularity with only about 24%. We think that this is reasonable, since in an open source community these stakeholders are less considered with management issues than the core members (i.e. committers).

To summarize, high-level concepts such as requirements and features as well as community aspects are important topics in the blogs of both active committers and other bloggers. Committers share more generic project knowledge, while other bloggers often discuss usage and configuration of the systems.

5 Blog Integration

To study *how* blogging activities are *integrated* into developers' workflows, we first explore publishing patterns of *blog posts*, *releases*, and *commits*. We then investigate relations between the *content* of blogs and commit messages.

5.1 Publishing Patterns

In this section we examine *when* developers post new blog entries with respect to other activities in the software projects. To investigate the influence of the project status on passive community members (non-committers), we study relationships between releases and their blog posts. For committers on the other hand, we examine relations between specific development activities expressed in their commit messages and their corresponding blogging behavior.

5.1.1 Release Dependency

We studied whether the project status has any influence on the blogging behavior of non-committers. We first created a histogram for each community, depicting the number of blog posts that were entered per day over the whole project lifetime, and marked the release dates of the according products on these figures. The results in Fig. 4 show an increasing activity in all four communities over time. From the mid of 2005 the blogging activities in all communities are continuous. Interestingly, we can observe *peaks* in the number of blogs per day around the release dates, especially in the last three years. Consequently, we hypothesize that releases influence the number of non-committing bloggers' posts. To further study this hypothesis, we tested the relative distribution of blog posts between two releases in the four communities. We first estimated the relative position of each blog post between the previous and the following release. Then, we calculated the distribution of the blog posts on this relative interval (see Fig. 5). Afterwards, we showed that this distribution is not uniform using the Kolmogorov-Smirnov test. Therefore we could reject the null hypothesis with $p < 0.002$ for all four communities. The corresponding histograms illustrate that most blog posts are made during the first 5% of the time span between two subsequent releases in all but the PostgreSQL communities.

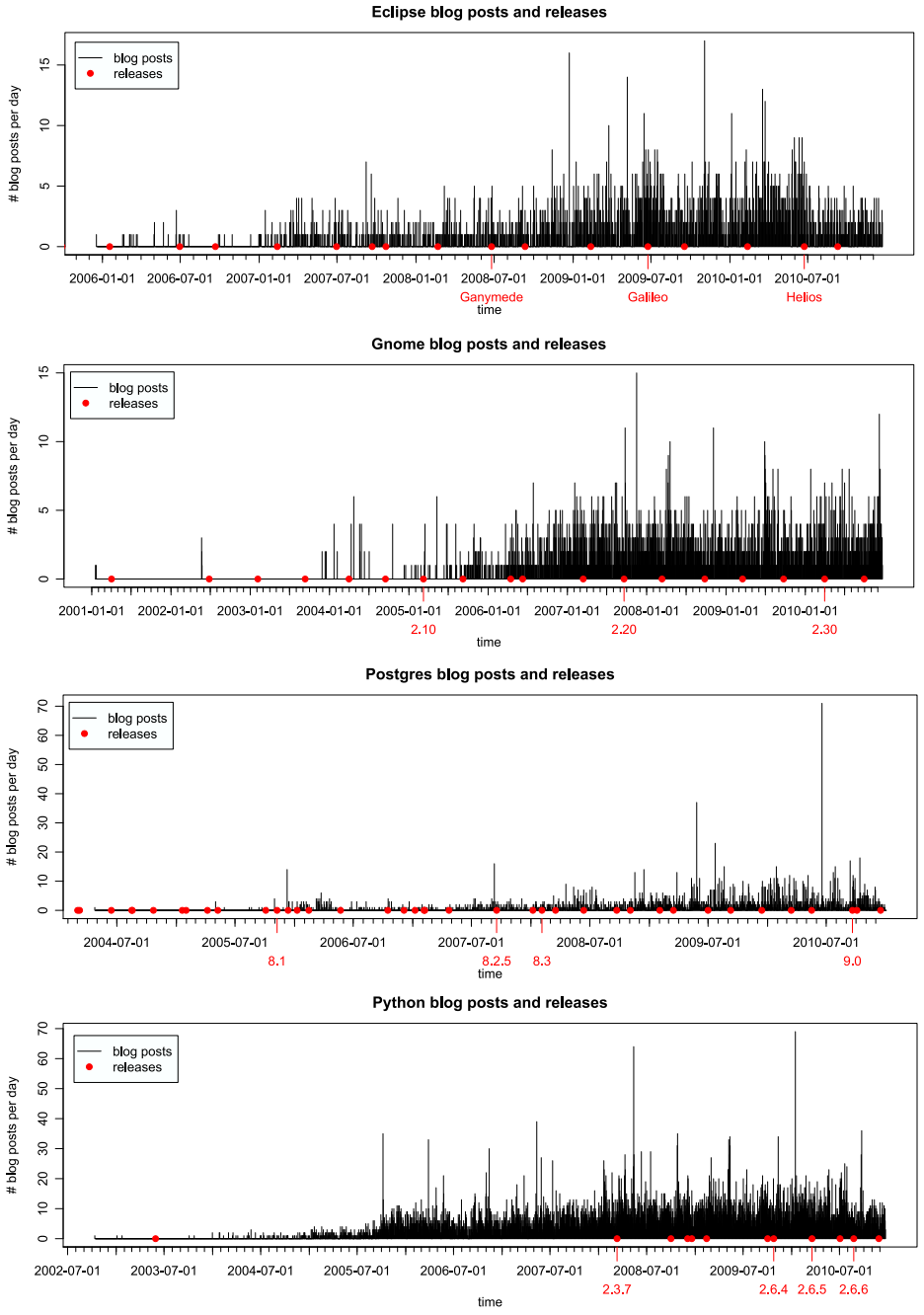

Fig. 4 Histogram of non-committing bloggers' posts and releases

We also studied the corresponding histograms for committers' blogs. For the GNOME and PostgreSQL communities we were not able to identify any significant difference for the blogging frequency compared with the release times, although

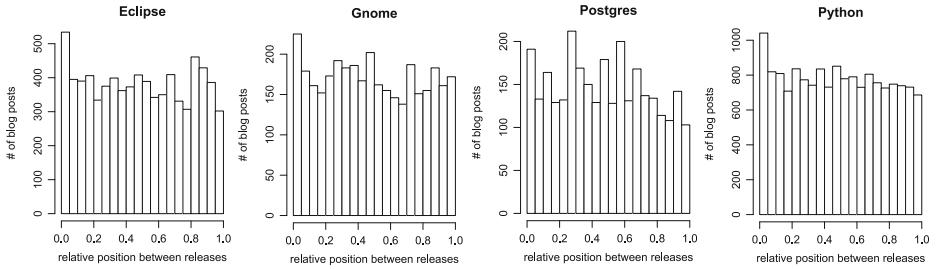

Fig. 5 Relative distributions of non-committing bloggers' posts between releases

it seems that in these communities blogs are slightly more frequent *after* product releases. The Eclipse and Python committers instead seem to post more frequently *before* product releases (cf. Fig. 6).

To summarize, in particular non-committers often blog more frequently shortly after product releases. Reporting about new product features and usage experiences could be triggers for these posts.

5.1.2 Activity Dependency

Next, we wanted to find out if committers make a particular use of social media after having accomplished certain types of development activities. To answer this question, we investigated their work descriptions in the commit messages. That is, our goal was to identify from the commit message the development activity performed before writing a post.

To this end we classified commit messages made by bloggers using the classification algorithm proposed by Hattori and Lanza (2008). The algorithm classifies commit messages according to disjoint sets of keywords by assigning a commit message the category of the first matching keyword in the text. Hattori and Lanza proposed the four categories *forward engineering*, *re-engineering*, *corrective engineering*, and *management* and provided an according keyword list for each category. Before the classification we applied a word stemmer on the commit messages to obtain more generic matches. The algorithm was able to classify about 83% of all bloggers' commit messages, while 1% of their commit messages were empty. We checked the validity of the classification using a random sample of 500 messages. We observed an accuracy of over 75%. Figure 7 depicts the classification results.

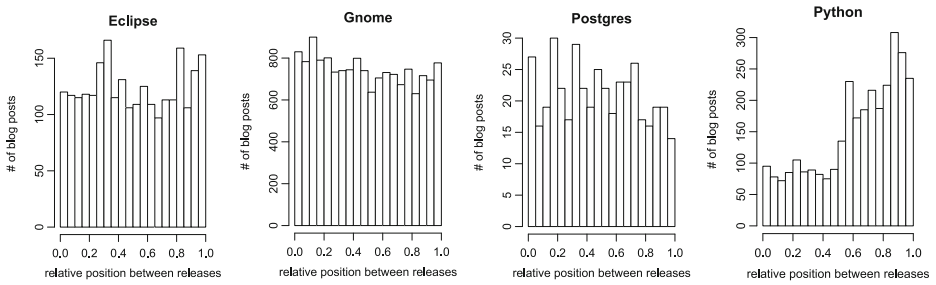

Fig. 6 Relative distributions of committers' posts between two subsequent releases

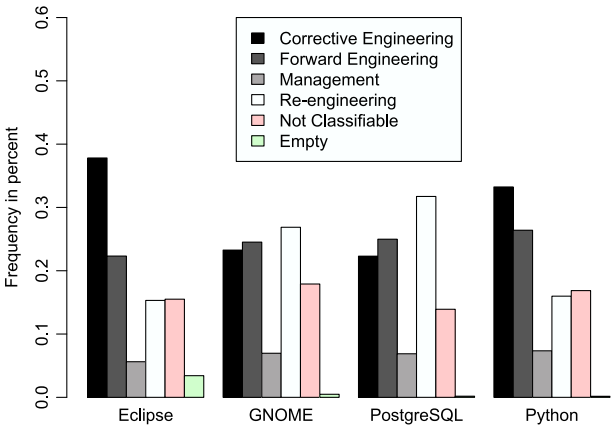

Fig. 7 Frequencies of commit categories

In the next step, we used a sequential pattern mining algorithm by Zaki (2001) to analyze the sequences of developers' commit messages and blog posts. Since we assume that the analyzed values are discrete (blogging and committing are non-continuous activities) and that the temporal order of the activities is important (e.g. blogging before a commit activity would be a different finding from blogging after the commit activity) sequential pattern mining is more appropriate than standard association rule mining (Agrawal et al. 1993).

Sequential pattern mining (Agrawal and Srikant 1995) allows finding frequent patterns in *sequence databases*. A sequence database contains a number of *data sequences*, which are ordered lists of *elements*. Elements are also called *itemsets* since they may contain multiple items. In our case, an item is either a commit message of a certain category or a blog post. We denote a commit message by a small letter according to its category (e.g. "c" for corrective engineering) and a blog post by "B". An example sequence would be $\langle\{f, m\}, \{c\}, \{B\}\rangle$, which comprises three elements. The first element contains two commit messages describing a forward engineering and a management activity. The second element represents a commit message describing a corrective engineering activity. The last element denotes a blog post. Given a sequence database S and a minimum support σ, sequential pattern mining yields all subsequences s of the sequences in S that are contained in a fraction of at least σ percent of all sequences. Each subsequence found is called *sequential pattern*.

Sequential pattern mining allows to find regularities in elements of a linear order. In our case the linear order is established by the time when a commit message or a blog post is published. We assume that items within an element happen at the same time or as part of the same *session*. We consider a *session* as a time interval, in which a developer performed a particular activity. Restricting a session to simultaneously published commit messages or blog posts makes less sense. Instead we use an upper bound of 120 min. This represents the mean session duration reported by Maalej and Happel (2009).

From the data sets we created a sequence database for each community as follows. The list of items I_d contains all commit messages and blog posts of developer d in chronological order. The items are then inspected one by one, oldest item first. A sequence ends when a blog post is made. All items before and including this blog

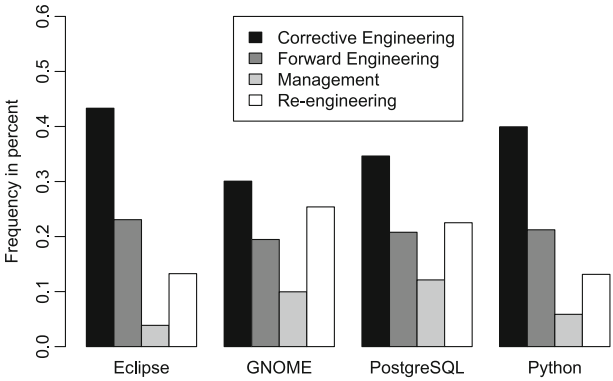

Fig. 8 Categories of commits before blog posts

post belong to this sequence. Items that occur within 120 min belong to same element within the sequence. For example the sequence $\langle\{f\}, \{c, f, r\}, \{m\}, \{c\}, \{B\}\rangle$ denotes five elements ending with a blog post. The second element contains the three items c, f, r which took place within 120 min. As an additional step, we removed sequences $\langle\{B\}\rangle$ that consist only of a single blog post. We repeated the sequence generation process for each developer d, resulting in a sequence database per community.

We analyzed these sequence databases in order to calculate the probability that the last commit message given a developers' blog post belongs to a certain category. To this end, we generated all sequential patterns of length 2 that ended with a blog post with a maximum gap value of 1 for each community. That is, two adjacent elements in a resulting sequential pattern are at most consecutive. For example the result $\langle\{c\}, \{B\}\rangle$ means that a developer published a post after describing a corrective activity in a commit message. We compared the according support values for all patterns across the different communities.

Our results show that a plurality of blog posts (30 to 43%) follow a commit message describing a corrective engineering activity. Least blog posts (13 to 25%) follow a commit message describing a management activity. Regarding the forward engineering and re-engineering categories we found two different situations. In Eclipse and Python, there are less re-engineering than forward engineering commits which precede blog posts. In the other two projects we observed the opposite situation. Figure 8 shows the results.

We compare these results to the commit classification results (depicted in Fig. 7). In the GNOME and PostgreSQL communities there are less commit messages describing corrective engineering activities than commit messages describing forward engineering and re-engineering activities. Nevertheless, more blog posts are preceded by commit messages describing corrective engineering activities.

To summarize, committers blog most frequently after corrective engineering activities, and least frequently after management activities.

5.2 Published Information

In the last question we analyzed if and how information in commit messages and blog posts is related in the studied open source communities.

5.2.1 Content Dependency

First, we investigated relations between committers' blog post topics and the categories of their preceding commits. Initially, we calculated the most predominant topic of each blog post in the obtained sequential patterns using the LDA document-topic matrix. Then, we associated the category of the preceding commit message with this topic and thus successively created a distribution of commit categories per blog post topic. The results are shown in Table 13. For each topic in committers' blog posts the distribution of the associated commit categories is stacked in a barplot, including the fraction of commit messages that were not classifiable by the heuristic of Hattori and Lanza.

Table 13 Distribution of commit categories per committer blog post topic – ■ corrective engineering, ■ forward engineering, ■ management, □ re-engineering, □ not classifiable

#	Topic description	Pop.	Distribution of preceding commit categories
C1	Features & domain concepts	42.2%	
C2	Community & contributions	37.7%	
C3	API usage & project documentation	30.0%	
C4	Release management & announcements	28.6%	
C5	Solution concepts & technology	26.8%	
C6	Architecture & packages	24.7%	
C7	Target platform	23.4%	
C8	Deployment & dependencies	20.9%	
C9	Conferences	17.4%	
C10	Development activities	16.3%	
C11	Non-functional requirements	15.7%	
C12	Communication, discussion	15.7%	
C13	Debugging & troubleshooting	15.0%	
C14	Licensing	14.8%	
C15	Source code	14.7%	
C16	Competitors & related work	14.6%	
C17	Version control	13.3%	
C18	Tips, tricks & tutorials	11.2%	
C19	User interface & user interaction	11.0%	
C20	Corrective maintenance	10.2%	
C21	Database access & external data	10.1%	
C22	Testing	7.5%	
C23	Continuous integration	2.5%	

We found "*API usage & project documentation*" among the top three topics associated with *corrective engineering* commits. We think that this is reasonable, since corrective actions can influence a system API (e.g. deprecated methods) and project documentation (e.g. issue tracking). The three most frequent topics associated with *forward engineering* commits include "*features & domain concepts*", as well as "*non-functional requirements*", which reflect the incorporation of new features and implementation of new (functional and non-functional) requirements. *Management* commits were mostly associated with "*licensing*", "*version control*", and "*development activities*", which fits the character of these topics as they are mostly unrelated to direct coding. Further, among the three most frequent topics associated with *re-engineering* commits we found "*database access & external data*", as well as *"non-functional requirements*". The former topic rises the assumption that external data sources are a frequent trigger for refactoring activities, the latter is concerned with code quality, which is often targeted in re-engineering activities.

To summarize, even if commit messages and blog posts are completely different project media, our results show that their contents are related in many cases, depending on factors such as presumably the time between their creations. This will be analyzed in detail in the next section.

5.2.2 Time Dependency

Last, we studied the *dependency degree* between the content of commit messages and the content of blog posts, with respect to the time between them. To achieve this, we randomly selected a set of 200 sequences, each containing at least three commits and one blog post. The three commit messages represented activities from the same category (e.g. three consecutive commit messages describing corrective engineering activities). For each of these sequences we created a document containing the last three commit messages and the following blog post. Then two independent persons manually rated the degree of dependency between the commit messages and the blog post, by giving each sequence one of the following grades:

- **3**: Information strongly related to the commits (e.g. summary of what has been done in the commits)
- **2**: Information partly related to the commits (e.g. advice on coding conventions after code refactoring)
- **1**: General project information (e.g. plans or infrastructure)
- **0**: Unrelated information (e.g. private notes)

For example the commit message "Enable multiple selection in download dialog, now you can cancel more than one download at a time. Note that this has no effect over the Pause button, only over Stop. Bug #327734." and the following blog post "Also, gnome bug #327734 has just been half fixed, meaning that you can now cancel more than one download at a time" are strongly related and should be rated with grade 3. We found that in 94 of 200 cases (46.5%) the content of commit messages and blog posts are unrelated (grade 0), whereas in 13 cases (6.5%) they are strongly related (grade 3). We found information partly describing the commit messages in 18 cases (9.0%). In total 15.5% of the evaluated blog post samples include information which refer to one or more of the developer's previously entered commit messages. This percentage shows that developers also use blogs to describe their changes to the code summarizing the work they have done. To understand the influence of the time

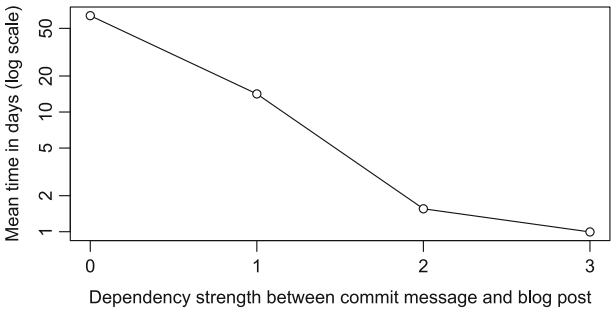

Fig. 9 Dependencies between blogs and commits in terms of time

between commit messages and blog post on this result, we calculated the average time period for each grade. Figure 9 shows the results. The strength of dependency between a commit message and a blog post decreases with an increasing time period between the commit and the post.

To summarize, developers also use blogs to summarize their work. They are more likely to publish information about recent activities they have performed than about old activities.

6 Discussion

In this section we highlight three main findings. First, we discuss the importance of blogs as a project medium and blogging as a project function. Second, we discuss the purpose of blogging in open source software projects based on our results, differentiating between blogging committers and other stakeholders. Finally, we derive insights for future research, in particular how to integrate blogs into development environments and blogging into developers' workflows as well as how to dissolve boundaries between developers and other stakeholders.

6.1 Blogging is a Project Function

In all studied open source communities we observed regular and frequent blogging activities since several years and across many releases. This is not surprising, as blogs became one of the most popular media for sharing and accessing software engineering knowledge in the last years (Parnin and Treude 2011). While individual developers only blog occasionally, the community as a whole constantly shares information and produces an average of up to six blog posts per day. These posts are written equally by committers as well as other community members.

Unlike committers in large open source projects, which have been studied quite thoroughly (e.g. Mockus et al. 2002), *other community members* are less researched. This non-committing group includes not only actual *users* of the software, but also other stakeholders such as *evangelists*, *community coordinators, companies' proxies*, and *managers*. Evangelists might have created the project long time ago. They have large experience and special interests in the success of an open source project, and therefore advertise it and demonstrate its usefulness. Managers and coordinators might be hired by the community to plan releases or organize conferences. Crowston et al. (2005) studied the social structure of open source projects and suggested an

"onion model" for describing it. Accordingly, a small group of core developers is surrounded by several layers of *peripheral* helpers, ranging from occasional problem solvers, to mainstream users whose contribution is limited to the occasional submission of bug reports. We think that our non-committing group of other members mainly overlaps with the periphery of an open source community according to the onion model.

Committers publish more frequently and over a longer time period than other members. This can be explained by their deep knowledge about the software as well as their high level of involvement in the community. But other community members blog regularly and about a broad spectrum of topics: from component dependencies to education and training. We observed that blogs are an important medium, and blogging (*sharing* knowledge) is an important activity for the studied open software communities. Future studies should also investigate, how community members *discover* and *use* (e.g. *comment*, *rate*, or *share*) knowledge captured in blogs, as this was not part of this research.

6.2 Why Do Developers and Other Stakeholders Blog?

Our results show that the studied communities use blogs as documentation tool to share knowledge and experiences, and to socialize and maintain community structures. A major finding of this study is that the studied developers blog in a high level of abstraction, e.g. frequently about features and domain concepts. At first glance this is surprising, as we expected developers to blog about models, technical abstractions, and source code related concepts. However, the public, "social", and rather informal nature of blogs can be one of the reasons behind this information granularity. Blogs enable developers to document features, dependencies, known issues, and qualities of new releases in an informal and time-ordered way and to a broad audience. Studies showed that developers describe their work in short but regular commit messages (Maalej and Happel 2010). Blog posts on the other hand are less frequent than commit messages, but comprise significantly more content. They rarely include source code but frequently high-level information and images. Therefore, blog posts seem to have rather the character of short documentations and tutorials.

Unexpectedly, we found that non-committers blog about more technical topics than committers in all studied communities. We think that this due to the framework nature of the studied projects. Non-committers use Eclipse, GNOME, PostgreSQL and Python as infrastructure for their own projects. In their blogs, they frequently reflect their technical experiences, share code examples, patterns to solve particular engineering tasks, and howtos. This shows the importance of blogs for making end users an integral part of software projects, enabling them to share their experience and helping to create and maintain project knowledge (Maalej and Pagano 2011).

Further, we observed a high fraction of posts dealing with community building aspects, such as advertising or summarizing events and conferences. This seems to be typical for social media. Developers and other stakeholders include this information orthogonally to other topics in their posts. In particular, open source communities depend on an active discussion of social and collaboration activities—where collaborators barely know each other. This serves as dissemination of results, requests for contribution, as well as creation and diffusion of a "social and community feeling", by using terms as "great", "community", "fun", etc.

6.3 Impacts

6.3.1 Integration of Blogging

Social activities like blogging are currently barely integrated into development processes and tools. However, we think the way developers blog calls for revisiting current development practices with more emphasis on integrating social activities and media.

Blog posts frequently contain information about recent activities described shortly before in previous commit messages. We found that developers post more often after corrective engineering than after forward engineering or re-engineering tasks. One silent implication of this finding is that bug fixes represent important information which should be shared with all stakeholders in a software community. Communicating these corrective actions to the community might have two implications. First, developers publicly show their personal contributions and merits—an important social and motivational factor. Second, many solved issues indicate a healthy project. On the other hand, posts about release announces and release plans make the community aware of the overall project status. We found that currently in particular non-committers publish more blog posts shortly after a new release.

Tools can help developers to reuse available knowledge in their posts, e.g. by linking to blogs and Wikis, or capturing particular screenshots. In particular, facilitating blogging after a particular development activity, or after using a new software release would be beneficial to developers. Moreover, tools may annotate blog posts with frequent topics to facilitate information structuring and access. To analyze the relationship between blog content and time of blogging further studies and experiments need to be conducted. An exploration of further social activities and their roles—in particular in requirements engineering—will also lead to a better integration of blogging into development processes.

6.3.2 Dissolving Boundaries Between Developers and Other Stakeholders

In today's software projects, users are neither an integral part of the software engineering *processes*, nor of the software *systems* themselves (Maalej and Pagano 2011). On the one hand, software engineering processes are rather transactional and focus on a small number of representative users to give feedback in requirements engineering activities. Contributing to other activities such as testing, documentation, integration, or design is exceptional. On the other hand, user feedback mechanisms in software systems are not standardized and rather ad hoc—if they exist at all. Typically the software "core features" are more important than user feedback or community features (typically found in the "Help" menu). Communication channels that allow for collaboration among users or between users and developers are usually decoupled from a software system and its development infrastructure.

Blogs seem to support dissolving the boundaries between developers and other stakeholders and encourage the involvement of users and their communities as a first order citizen of software projects. In all studied open source communities, we did not discover any *major* and *significant* difference between developers' and other stakeholders' blogging habits. Both groups seem to frequently rely on social media to publish project-related information in a similar level of abstraction and formality. In particular the content of blogs presents strong overlaps in the discussed

topics and their popularities. Developers as well as other stakeholders discuss about requirements, implementation, and community aspects. On the one hand, developers report about their recent development activities to communicate their project work to a broad audience, including users and other stakeholders. On the other hand users and other stakeholders seem to have their blogging peak time shortly after new versions are released—reporting on their experiences with the new changes. Utilizing these experiences and the volunteered resources provides a huge benefit for software projects. We claim that communities should be created *systematically* and integrated in software systems utilizing social media such as blogs. In (Maalej and Pagano 2011) we envision a software framework that enables the development and maintenance of such *social* software.

7 Results Validity

7.1 External Validity

Although our study was neither designed to be generalizable nor representative for all developers and communities, we think that most of the results have a high degree of generalizability, in particular for large open source communities. At the design time of the study, we knew neither the entire population of software development blogs, nor of blogging developers. Therefore we were unable to study a random representative sample of blogs and bloggers. Instead, we aimed at a rather exploratory, hypothesis-generating study to better understand blogs, their usage, and role in the development project. The four studied communities should rather be interpreted as four cases than as one homogeneous dataset. However, the careful selection of these communities, their leading role in open source software, and the large number of their blogs and bloggers give confidence that many of the results apply for other comparable communities as well.

We think that our results are representative for each of the studied communities due to the following reasons.

- Our datasets include all community blogs from the last seven years.
- We conducted statistical tests to check the statistical significance of our results and exclude hazard factors.
- We got similar results using different analysis methods (e.g. descriptive statistics and topic analysis).
- In two of the studied communities (Eclipse and GNOME), we were able to contact three senior active members. Among them were both committers, who had contributed for around three to four years (92 to over 600 commits each), and evangelists, who had been involved at least three years in the community. While discussing the results in detail they confirmed the findings based on their experiences.

Nevertheless, there are three limitations which should be considered when interpreting the results. First, for Eclipse and GNOME we were unable to analyze the blogs of *all* sub-projects. Both communities are very broad and use different infrastructures. We think though that the large sub-project selection is representative, which was confirmed by active members of both communities. Second, some of the findings

for PostgreSQL—in particular those comparing the behavior of committers and other bloggers—were not statistically significant (i.e. $p > 0.05$). This results from the relatively low number of blogging committers in this community. Since comparable results from the other communities passed the statistical tests, we think that the results of PostgreSQL are not due to chance. Finally, we only studied blogs collected from the community aggregators. We did not collect other blog posts (e.g. private blogs of users via a web search engine). Our study is community centric rather than developer or user centric. The aggregators collect the official blogs of all "known" members. They are controlled by a community board and obey strict quality measures. We found that our data mostly excludes "noise" such as private or project unrelated topics. However, studying other blogs from outside the aggregators might lead to additional results such as the behavior and the topics of unsatisfied stakeholders.

7.2 Construct Validity

We made the following simplifying assumptions during our analysis, which might partly limit the construct and internal validity of the results:

- To connect blogs and commit messages we mapped the names of committers and bloggers using a text similarity algorithm (Dice 1945). We chose a pessimistic mapping that creates false negatives rather than false positives. That is, the size of the data might be affected but not the analysis results. As a side effect, few bloggers, which we classified as other community members, might be committers. Since the number of non-committers is much larger than the number of blogging committers a pessimistic approach is more appropriate. We were able to map between 19,1% (Python) and 73,1% (GNOME) of all blog authors in the data sets. For each community, we randomly picked about half of the remaining committers, and we manually searched their names among the other community members—without any success.
- To study the integration of blog posts into the development workflow, we use the commit time and blog publishing time to order the corresponding artifacts chronologically. However, blogs and repositories may reside on different servers with slightly different time settings. To reduce the effects of such synchronization errors we considered activities within 2 h as the same session. All the 200 manually investigated sample sequences were correctly ordered.
- We used the heuristic of Hattori and Lanza (2008) for the categorization of commit messages. Testing the categorization of 500 randomly selected commits shows that this algorithm has an accuracy of about 75% on our data. Therefore, the calculated blogging probability after a commit category might be erroneous. This marginal error does not bias the resulting trends, though.
- We used regular expressions to extract source code blocks, images, and links. Unlike for links and images, there are no standard HTML tags for code blocks. Therefore, more advanced mining mechanisms such as Bettenburg et al. (2008, 2011) might have extracted more code blocks leading to higher ratios. However, the weak frequency of the according source code topic gives confidence that this ratio remains below 15%.
- Part of our results relies on manual analysis such as the categorization of images, links and the labeling of blog topics. These results are subject to experimenter

bias. To reduce this risk, we conducted pair analysis independently from each other. We iterated this activity by refining the rating criteria to improve the inter-raters agreement. We only reported on results where the rates of inter-rater agreement were over 90%.

8 Related Work

We focus the related work discussion on three fields: studies on mining blogs, studies on mining artifacts similar to blogs, and research on social media in software engineering.

8.1 Mining Blogs

To our knowledge there are no published empirical studies on how developers and development communities use blogs—except the first published version of this paper (Pagano and Maalej 2011). However, there exist several studies on mining blogs in general. We distinguish studies, which target community-related aspects and others, which target content-related aspects of blogs.

Several authors discussed how to discover, visualize, and study the dynamics of communities by analyzing the content of blogs. Gruhl et al. (2004) propose a generative blog topic model to identify external influences on bloggers and their topics. Tseng et al. (2005) explore different communities of interest in a set of blogs. They propose visualization techniques that help to explore further dependencies between blog topics. The goal of our study is to explore basic questions on how developers blog. Therefore we focus on single, open source, development communities. We model blog posts as community documents with multiple latent topics, assuming that topics included in these blogs are either related to software development or to the project. This enables us to quantify the popularity of particular topics of interest to the software engineering research, such as "*API usage*", or "*community & collaboration*".

Other blog studies aimed at extracting meaningful knowledge, automating trend discovery, and identifying opinion leaders. Glance et al. (2004) visualize the popularity of blog topics over time and show a correlation with real world trends. Similarly, we observed that developers are more likely to blog about recent activities. Song et al. (2007) identify opinion leaders in a set of blogs based on information novelty and influence on other blogs. We also observed that committers, technology experts, and evangelists share their knowledge in their development communities by using blogs. In addition, we were able to quantify the semantic entities in developers' blogs, i.e. which types of information are included.

8.2 Mining Related Artifacts

There is a large research community, which applies data mining techniques to analyze development artifacts. Related studies analyze commits, work descriptions, and other social media. Several authors explored commit histories to identify reasons for software changes and to understand the software evolution. Our work is based

on these results. In particular we use the algorithm of Hattori and Lanza (2008) to classify commit messages according to the development activity accomplished by the commit.

Other authors analyzed informal artifacts, in which developers summarize what they have done in a particular work session. Maalej and Happel (2010) used NLP techniques to analyze personal notes and commit messages and found regularities in how developers describe their work. This work extends these findings by showing that developers also describe their activities in blogs.

Researchers spend considerable effort on the analysis of other social media such as social networks and mailing lists. Social network analysis itself is an established research field (Wasserman and Faust 1994). Several publications study individual and group behavior as well as the explicit or latent structure of social networks. We focus our research on the medium blog as well as the blogging behavior of developers. Bird et al. (2006) create social networks from developer email communication and study similarities to development teams. They show that sub-community movements in these social media reflect development activities. Bacchelli et al. (2010) showed how e-mail archives enclose significant information on the software system they discuss. The authors presented a benchmark for recovering traceability links between e-mails and source code. In our work we found similar relations between blogging as social activity and development activities on an individual level.

8.3 Integrating Social Media

Recent papers (Begel et al. 2010; Guzzi et al. 2010; Treude and Storey 2009; van Deursen et al. 2010) suggest the integration of social media into the development environment and development processes. Guzzi et al. (2010) claim that integrating blog user interfaces into the IDE would foster the reuse and sharing of program knowledge. Treude and Storey (2009) discussed how the informal and lightweight use of social media can be integrated into development processes. The authors concluded that informal processes are usually carried out via communication mechanisms. Our study is not based on blogs that are already integrated into the development environments and processes. We analyzed the current practices of using social media by a large number of developers. Our findings on the blogging frequency, blogging time, and information included give empirical evidence to the claims of these studies as well as new insights into integration "features". For example the type of images, links, and information included by developers and the probability of blogging after certain activities or releases can help to further tighten this integration. van Deursen et al. (2010) envision an IDE that uses both tagging and blogging strategies to facilitate collaboration, program comprehension, and traceability in development teams. In our vision, even users and other stakeholders contribute to project activities using such social media.

9 Conclusion

How do developers and other stakeholders in open source communities blog? In this paper we reported on a first study which systematically explores this question. We found that open source communities blog frequently and continuously with 2–6

blogs per day. An average developer posts a new project-related blog entry every 26 days, slightly more often than other community members. While 29% of the posts include images, only few contain source code blocks. Developers frequently link to existing information like Wiki pages and other blog posts. Topics representing high-level concepts such as *features and domain concepts* are predominant in more than one third of the blog posts of studied communities, while only 15 to 20% deal with *source code* concepts. This reveals the documentation function of blogs in software projects. Moreover, more than 43% of the studied posts include community related information, which reveals the social and motivational function of blogs.

We also identified interesting patterns in *when* community members blog. The peak time of blogging seems to correlate with the releases of the software, and occurs usually *after* the software is released. In addition, committers are more likely to blog after corrective engineering than after forward engineering and re-engineering activities. Their blog posts frequently contain information about activities described shortly before in commit messages.

Our results represent a starting point towards the empirical framework of the use of social media in software engineering. We think that there are two lines of future research towards this framework: a *hypothesis-driven* and a *content analysis* line. Hypothesis-driven research enables us to explore the role of social media and allows for a need-driven integration of these media into development processes and tools. A content analysis research enables a more in-depth analysis of the knowledge shared in blogs, giving more reliable results on the roles, efficiency, and the quality of blogs and blogging. Questions such as "does the post report on requirements or on user experience?", "how are posts structured?", or "which decision rationale is discussed and explained in the blog posts?" are better answered with a manual content analysis. Currently, we are studying how developers and end users use other social media like micro-blogs or content communities. In addition we plan to replicate our results and conduct a more in-depth analysis using different content analysis techniques on a representative sample.

Acknowledgements This work has been supported by the FastFix project, which is funded by the 7th Framework Programme of the European Commission, grant agreement no. FP7-258109. We would like to thank Enrique Garcia Perez, Damir Ismailović, Amel Mahmuzić, Helmut Naughton, Tobias Roehm, Alex Waldmann, and the anonymous MSR'11 and EMSE reviewers for their valuable feedback. We are further thankful to Jonas Helming, Felix Kaser, and Daniel G. Siegel for helpful insights into the Eclipse and GNOME communities.

References

Agrawal R, Srikant R (1995) Mining sequential patterns. In: Proceedings of the eleventh international conference on data engineering. IEEE, pp 3–14

Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of the 1993 SIGMOD conference on management of data, ACM, Washington, DC, USA, pp 207–216

Bacchelli A, Lanza M, Robbes R (2010) Linking e-mails and source code artifacts. In: Proceedings of the 32nd ACM/IEEE international conference on software engineering—ICSE '10, p 375

Begel A, DeLine R, Zimmermann T (2010) Social media for software engineering. In: Proceedings of the FSE/SDP workshop on future of software engineering research. ACM, pp 33–38

Bettenburg N, Adams B, Hassan AE, Smidt M (2011) A lightweight approach to uncover technical artifacts in unstructured data. In: 2011 IEEE 19th international conference on program comprehension, pp 185–188

Bettenburg N, Premraj R, Zimmermann T, Kim S (2008) Extracting structural information from bug reports. In: Proceedings of the 2008 international workshop on mining software repositories—MSR '08, p 27

Bird C, Gourley A, Devanbu P, Gertz M, Swaminathan A (2006) Mining email social networks. In: Proceedings of the 2006 international workshop on mining software repositories. ACM, pp 137–143

Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3(4–5):993–1022

Crowston K, Heckman R, Annabi H, Masango C (2005) A structurational perspective on leadership in Free/libre open source software teams. In: Proceedings of the 1st conference on open source systems (OSS), Genova, Italy

Dice LR (1945) Measures of the amount of ecologic association between species. Ecology 26(3):297–302v

Glance N, Hurst M, Tomokiyo T (2004) BlogPulse: Automated trend discovery for weblogs. In: Proceedings of the WWW 2004 workshop on the weblogging ecosystem: aggregation, analysis and dynamics, ACM, New York, NY, USA

Gruhl D, Liben-Nowell D, Guha R, Tomkins A (2004) Information diffusion through blogspace. ACM SIGKDD Explorations Newsletter 6(2):43–52

Guzzi A, Pinzger M, van Deursen A (2010) Combining micro-blogging and IDE interactions to support developers in their quests. In: Proceedings of the 26th international conference on software maintenance (ICSM), IEEE, 2010, pp 1–5

Hattori L, Lanza M (2008) On the nature of commits. In: ASE workshops. IEEE, pp 63–71

Kaplan AM, Haenlein M (2010) Users of the world, unite! The challenges and opportunities of social media. Bus Horiz 53(1):59–68

Maalej W, Happel H (2009) From work to word: how do software developers describe their work? In: Working conference on mining software repositories, pp 121–130

Maalej W, Happel H-J (2010) Can development work describe itself? In: 2010 7th IEEE working conference on mining software repositories (MSR 2010), pp 191–200

Maalej W, Pagano D (2011) On the socialness of software. In: Proceedings of the international conference on social computing and its applications. Sydney, Australia, IEEE

Maalej W, Panagiotou D, Happel H-J (2008) Towards effective management of software knowledge exploiting the semantic wiki paradigm. In: Herrmann K, Brügge B (eds) Software engineering. Bonn, Germany, GI, pp 183–197

Mockus A, Fielding RT, Herbsleb JD (2002) Two case studies of open source software development: Apache and Mozilla. ACM Trans Softw Eng Methodol 11(3):309–346

Pagano D, Maalej W (2011) How do developers blog? an exploratory study. In: Proceedings of the 8th conference on mining software repositories. ACM

Parnin C, Treude C (2011) Measuring API documentation on the web. In: Proceeding of the 2nd international workshop on web 2.0 for software engineering, Web2SE '11. ACM, New York, NY, USA, pp 25–30

Song X, Chi Y, Hino K, Tseng B (2007) Identifying opinion leaders in the blogosphere. In: Proceedings of the sixteenth ACM conference on conference on information and knowledge management. ACM, New York, New York, USA, pp 971–974

The Nielsen Company (2010) Led by Facebook, Twitter, global time spent on social media sites up 82% year over year

Treude C, Storey M-A (2009) How tagging helps bridge the gap between social and technical aspects in software development. In: ICSE '09: proceedings of the 2009 IEEE 31st international conference on software engineering. IEEE Computer Society, Washington, DC, USA, pp 12–22

Tseng B, Tatemura J, Wu Y (2005) Tomographic clustering to visualize blog communities as mountain views. In: WWW 2005 workshop on the weblogging ecosystem. Citeseer

van Deursen A, Mesbah A, Cornelissen B, Zaidman A, Pinzger M, Guzzi A (2010) Adinda : a knowledgeable , browser-based IDE. In: Proceedings of the 32nd ACM/IEEE international conference on software engineering. ACM, vol 2, pp 203–206

Wasserman S, Faust K (1994) Social network analysis: methods and applications. Cambridge University Press

Zaki M (2001) SPADE: an efficient algorithm for mining frequent sequences. Mach Learn 42(1): 31–60

Dennis Pagano received the German Diploma degree in computer science (equivalent to B.Sc. + M.Sc.) from the Technische Universität München in 2008. He is currently a 4th year Ph.D. student at the TUM and a scientific staff member at the Chair for Applied Software Engineering. His research interests include human factors in software engineering with a focus on user involvement, mining software repositories, and recommender systems. He participated actively in the open source research projects TeamWeaver and FastFix at the TUM.

Walid Maalej leads a research group on human and context aspects in software at the TU München (Germany). In his Ph.D. he developed a context-aware approach to detect developer's intentions and support software engineering tasks. His current research interests include group recommendation systems and collaboration with end users. Walid published more than 30 papers and supervised more than 20 theses on these topics. He co-organized international events such as the Social Software Engineering and Managing Requirements Knowledge workshop series. Walid served on the PCs of numerous conferences, including the ESEC/FSE 2011, RE'11, and OSS'12. He previously served as consultant for Siemens, Deutsche Telecom, and Rohde & Schwarz, and TATA Consulting Services. He has been distinguished by the Werner-von-Siemens-Ring foundation as Germany's Junior Researcher in Informatics for 2010–2013.