How Do Developers Blog? An Exploratory Study

Dennis Pagano
Technische Universität München
Munich, Germany
pagano@cs.tum.edu

Walid Maalej
Technische Universität München
Munich, Germany
maalejw@cs.tum.edu

ABSTRACT

We report on an exploratory study, which aims at understanding how software developers use social media compared to conventional development infrastructures. We analyzed the blogging and the committing behavior of 1,100 developers in four large open source communities. We observed that these communities intensively use blogs with one new entry about every 8 hours. A blog entry includes 14 times more words than a commit message. When analyzing the content of the blogs, we found that most popular topics represent high-level concepts such as functional requirements and domain concepts. Source code related topics are covered in less than 15% of the posts. Our results also show that developers are more likely to blog after corrective engineering and management activities than after forward engineering and re-engineering activities. Our findings call for a hypothesis-driven research to further understand the role of social media in software engineering and integrate it into development processes and tools.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—Process Metrics; K.4.3 [**Organizational Impacts**]: Computer-supported collaborative work

General Terms

Human Factors, Documentation, Measurement

Keywords

Social Software, Open Source, Data Mining, Blogs

1. INTRODUCTION

Social media enable the creation and exchange of user-generated content [9]. Individuals can use them to interact with, share information 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 [14]. For example, facebook recently announced that it has more active users than the population of the USA[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 [2]. For example, several studies have shown the role of Wikis for managing software documentation and collaboration [12]. Other authors suggested the integration of social media into development environments [7, 15]. However, there exists no empirical framework on the role of social media in software engineering. This paper takes a step towards such a framework by exploring the role of blogs – a popular social medium in open source communities.

We report on an exploratory study, which examines the content and the metadata of blogs in four large open source software projects. 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 development activities. 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 our research setting, including the research questions, the used data, and followed methodology. 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 and committing behavior of developers (Section 5). Section 6 surveys related work focusing on studies, which analyze social media and similar artifacts. Section 7 discusses our findings while Section 8 presents their limitations. Finally, Section 9 concludes the paper and sketches our future plans.

2. RESEARCH SETTING

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

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

2.1 Research Questions

We focus our analysis on three aspects: blog *usage*, blog *content*, and *integration* of blogging into 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. In particular we answer the following questions:

- **Publishing frequency:** How often do developers and other stakeholders create blog entries?
- **Post structure:** What are typical elements included or referenced in a blog post?

When analyzing the blog usage we distinguish between active developers (*committers*) and other bloggers.

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

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

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

- **Publishing patterns:** Are there particular patterns, which describe when developers use blogs within their development workflows?
- **Content dependencies:** Are there relationships between the work performed and the information blogged?

2.2 Research Method

Our research method is summarized in Figure 1. It consists of two phases: a data preparation and a data analysis phase. In the preparation phase, we first select the development communities, from which we will collect the research data. To qualify for our study a community must have a public source code repository and at least 50 bloggers. So-called *blog aggregators* like Planet[2] collect blog posts of community members in a single place. Each entry in the aggregator includes meta information about the name of the blogger and the link to the original blog post. Additionally, the aggregator lists all contributing persons as the sources. Contributors are usually selected by a particular community board. The selection underlies a strict quality policy, since the primary goal of the common blog is to provide project related knowledge.

After querying the blog aggregator and the source code repository, we associate the blog posts to the commit history based on the authors' accounts. This mapping enables to distinguish between bloggers who commit source code and other bloggers in the community. We build a list of blogger names and a list of committer names from the collected posts and commit history. Then we try to assign a committer name to a blogger name. Most blog posts include the real name of their authors. However, most commits only

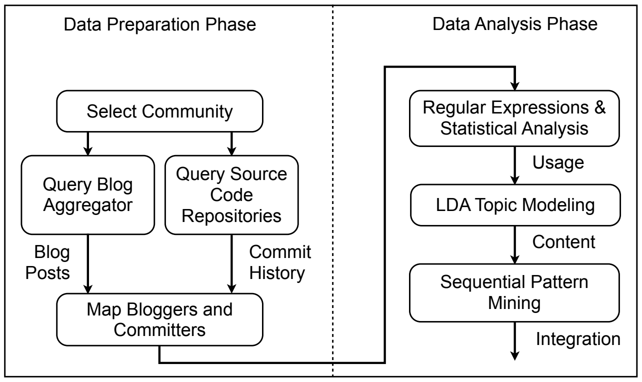

Figure 1: Research Method

contain login names. The real names in the lists were not identical (e.g. "David Wheeler" and "David E. Wheeler"). To compare two real names, we use a text similarity algorithm[3], that finds exact matches as well as slightly diverging pairs. In addition we create login name candidates from the blogger's real name (e.g. "glefur" from "Goulwen Le Fur") and rerun the text similarity algorithm on the login names. Finally, we observed that several commit messages contain meta information such as "Author: 10:44:52 Tim Janik", "Author: PST 2006 Michael Emmel". We extract this information using regular expressions and manually correct the author names when required.

The data analysis phase consists of three steps, which respectively answer the usage, content, and integration questions. To analyze the *usage* of blogs we apply descriptive statistics (for frequency calculation) and regular expressions (for analyzing the blog structure). To analyze the blog *content* and included information we apply the Latent Dirichlet Allocation (LDA) topic modeling technique [4]. When applied on the blogs, this technique extracts keywords which belong together and groups them as topics. Finally, to study the *integration* aspect we order commit messages and blog posts by time and investigate the resulting stream of events. Thereby we look for patterns and regularities using Sequential Pattern Mining [1].

2.3 Research Data

We selected four large open source software communities for our analysis: Eclipse, GNOME, PostgreSQL, and Python. Table 1 shows an overview of the collected data, which we briefly introduce in the following.

Table 1: Overview of Collected Data

	Eclipse	GNOME	PostgreSQL	Python
# posts	10,333	18,323	2,691	18,660
# bloggers	328	342	95	405
# commits	239,659	252,831	30,745	45,116
# committers	467	2,294	34	178
# blogging committers	93	250	12	34

Eclipse is an open development platform, which comprises extensible frameworks, tools, and runtime environments to build, deploy, and manage software systems. The Eclipse

[2]www.planetplanet.org

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

community is one of the largest open source communities including 11 million users and more than thousand active developers. From Planet Eclipse[4] we collected 10,333 blog posts of 328 community members over the last 7 years. Eclipse comprises over hundred subprojects, which use separate source code repositories. We asked 3 Eclipse developers 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 10 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 from 342 community members from the last 10 years. GNOME includes about hundred subprojects, which use different source code repositories. As in Eclipse we asked for the most active subprojects and selected the following 34: 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, 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 subprojects 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 2,691 blog posts of 95 active members from the last 7 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 8 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.

[4]planeteclipse.org
[5]planet.gnome.org
[6]planet.postgresql.org and www.planetpostgresql.org
[7]planet.python.org

3.1 Publishing Frequency

We studied the publishing frequency for the whole communities as well as for the single bloggers, while distinguishing between active developers (committers) and other bloggers.

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 hours. On average, Python is the most active community with one blog post each 3.9 hours. GNOME bloggers publish one blog post each 4.9 hours on average. The mean time between two posts in the Eclipse community is 5.6 hours. PostgreSQL is the least active community with a post each 18 hours. This means the bigger the community is, the more frequent blog posts it has.

We observed a significant difference between the blog publishing frequency and the commit frequency in the communities. On average, Eclipse and GNOME developers committed a source code change twice per hour, while PostgreSQL and Python developers committed once in 4 hours.

Next, we study the total number of blog posts for the single 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 2 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 10 posts, while 75% of the other bloggers accounted for less than 50 posts.

Table 2: Blog Posts per Person (Medians)

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

Analyzing the individual contributions, we found that committers 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 7 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). In all communities except Eclipse, committers posted more frequently than other bloggers.

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 several active members of the Eclipse community. We also randomly selected 5 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.

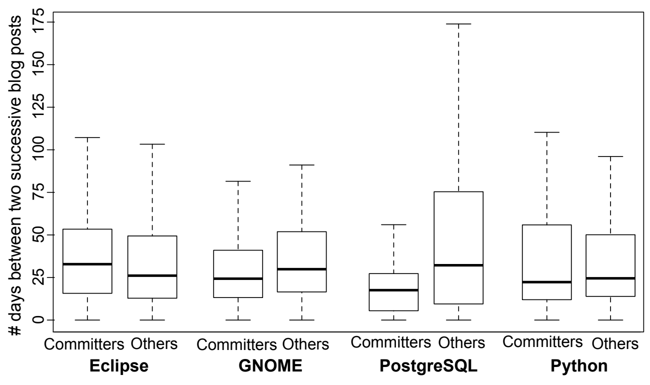

Figure 2: Publishing Frequency of Bloggers

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.

3.2 Post Structure

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

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 Figure 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 of the author with a new Linux init system. 1,102 posts comprise less than 10 words, examples from posts with 3 words are "GNOME lacks stetic." and "Amazing, absolutely amazing.". Over 95% of all posts are shorter than 1,000 words, which is equivalent to about 4 printed pages.

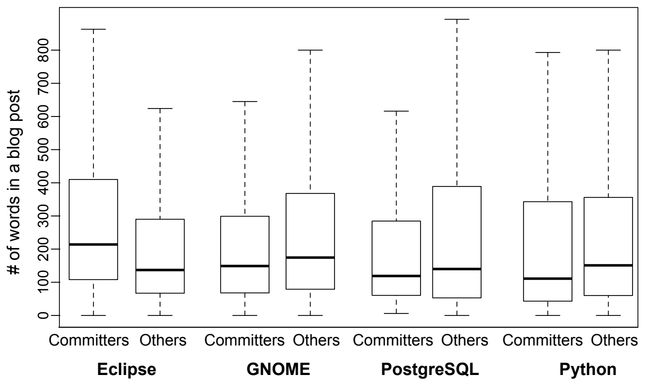

Figure 3: Average Lengths of Blog Posts

All committers except in Eclipse write shorter posts than other bloggers on average. The Eclipse committers account even for the longest blog posts on average in 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.

To our surprise, in only 934 of all 50,701 blog posts (1.8%) we found source code paragraphs. On average each of these posts contained 2.5 code paragraphs. As shown in Table 3, only the PostgreSQL committers published less source code in their blogs than other members of the community. On the other hand, PostgreSQL committers posted 4.3 sections of source code on average as opposed to 1.7 sections by other community members. We conclude that developers do not tend to include source code in their blog posts.

Table 3: Blog Posts Containing Source Code

	Eclipse	GNOME	PostgreSQL	Python
Committers	3.0%	1.2%	1.9%	3.6%
Other bloggers	1.4%	1.0%	3.0%	2.2%

We observed that links are frequently included in blog posts. A 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 83.8% of committers' posts and 77.2% of other bloggers' posts contain links.

To further investigate the usage of links in individual posts, we semi-manually investigated 5,313 links in 400 randomly selected blog post samples (200 posts from developers and 200 posts from other bloggers) from each community using regular expressions. In a first step we removed about 19% of all links because they were off topic (e.g. links to private sites). 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 the ones of other community members (25%). We think that committers more than other community members tend to re-use knowledge rather than to re-write or copy it.

The remaining links were links to newsgroups (1%), microblogs (1%), project download pages (1%), videos (1%), other downloads (3%), issue trackers (3%), code documentation (7%), source code (7%), and other project-related websites (39%) like the official project website.

Last, we investigated the usage of images in our data sets. We found that 14,605 blog posts (28.8% of all posts) contain images. The Eclipse and GNOME communities use images more frequently (> 37% of all posts) than the PostgreSQL and Python communities (< 18%). We think that this partly results from the fact that several subprojects of Eclipse and GNOME are user interface projects, while in PostgreSQL and Python user interfaces are secondary concepts.

Again we manually investigated 1,231 images in randomly selected samples of 400 blog posts from each community. We observed 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 online anymore and 24% were off topic (e.g. vacation pictures). About 20% of the images are screenshots, 7% community pictures (e.g. conferences), 6% graphics such as function plots and charts, and 3% diagrams (e.g. UML diagrams).

Table 4 shows that committers tend to use more screenshots (22.3%) in their posts than other bloggers (18.0%).

Table 4: Images in Blog Posts (Semi-automated Analysis)

	Screenshots	Community	Graphics	Diagrams
Committers	22.32%	6.84%	8.92%	3.39%
Other bloggers	17.96%	6.18%	3.33%	2.65%

We think that the high amount of screenshots in blog posts indicates that software communities and in particular active developers use this medium to communicate on a relative high level. Further, based on the publication of community pictures we conclude 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* developers post. We used the Latent Dirichlet Allocation (LDA) topic modeling technique [4] 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 a document corpus for each community comprising all blog posts of the community's *active* (committing) developers, using 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, we manually inspected the results and added topic descriptions based on the top 20 most influential words (obtained by LDA) and several associated blog posts (randomly selected). Fourth, we grouped similar topics across the data sets to project-independent topic categories.

Table 5 shows the list of extracted topics with their popularity and examples of influential words. To quantify the *popularity* of these 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. First, single words from most topics are present in a large number of posts (e.g. "use") and do not determine the topic sufficiently without other words from the topic. Second, over 90% of the posts contain at least one predominant topic. Table 6 shows the most popular topics with the according frequencies among the communities.

We found that the most popular topic is *"functional requirements & 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. 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. The topic *"source code"* is covered in less than 15% of all blog posts.

Table 5: List of Identified Topics

Topic description	Pop.	Examples of influential words
functional requirements & domain concepts	42.2%	radio, listen, player, sync, song, music, play, ipod, album, artist, band
community & contributions	37.7%	people, community, contribute, group, help, news, post, comment
API usage & project documentation	30.0%	wiki, write, project, api, document, use, review, text, output
release management & announcements	28.6%	release, try, helios, download, board, committee, foundation
solution concepts & technology	26.8%	rest, uri, response, rule, parser, syntax, widget, javascript, client
architecture & packages	24.7%	start, component, register, import, service, osgi, bundle, framework
target platform	23.4%	linux, android, vm, platform, device, system, run, environment
deployment & dependencies	20.9%	ant, zip, publish, target, jar, install, depend, distribute, plugin
conferences	17.4%	session, democamp, present, eclipsecon, conference, event, talk
development activities	16.3%	work, implement, develop, test, code, improve, task, maintain
non-functional requirements	15.7%	cache, memory, perform, high, quality, limit, secure, cost
communication, discussion	15.7%	send, address, mail, call, discuss, answer, question, decision, phone
debugging & troubleshooting	15.0%	debug, address, process, warning, problem, exception, raise, error
licensing	14.8%	free, company, open, source, community, business, foundation
source code	14.7%	void, new, import, public, final, string, class, return, private, true
competitors & related work	14.6%	more, think, performance, product, oracle, sun, mysql, experience
version control	13.3%	trunk, commit, repository, merge, clone, svn, git, master, csv, push
tips, tricks & tutorials	11.2%	tutorial, tool, practical, summary, support, article, website
user interface & user interaction	11.0%	tab, view, menu, dialog, button, text, mockup, select, click, interact
corrective maintenance	10.2%	support, report, fix, improve, bug, bugzilla, issue
database access & external data	10.1%	jpa, import, store, table, sqlite, database, schema
testing	7.5%	write, test, case, manual, unit, check, build, system, patch, junit
continuous integration	2.5%	resource, test, source, build, configure, hudson, project, generate

From these results we conclude that 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. We think that in particular open source communities depend on the active discussion of social aspects, dissemination of community news, and requests for contribution. 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.

Table 6: Most Popular Topics in the Communities

Eclipse	GNOME	PostgreSQL	Python
functional requirements & domain concepts (47%)	community & contributions (36%)	functional requirements & domain concepts (47%)	API usage & project documentation (50%)
community & contributions (33%)	functional requirements & domain concepts (33%)	non-functional requirements (40%)	functional requirements & 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%)

This is reasonable, since such requirements are crucial for a database system (e.g. performance, security, scalability etc.).

To further interpret the different topics developers write about in blogs, we grouped our results into the following categories.

1. Requirements: This category contains topics that are related to requirements engineering, particularly *application domain concepts*, *requirements*, and *user interface design*. Further, topics describing or comparing the project to *competitors* or describing *related work* belong to this category, since these are also covered in requirements engineering.

2. Community: These topics represent community and social aspects like *contributions* of specific members, *communication*, and *project news*. Another topic is *"conferences"* describing events organized by the community and provide a possibility to meet, learn, and exchange.

3. Project knowledge: This category 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 category. But also *tips and tricks* as well as *tutorials* are a form of project knowledge.

4. Deployment: Topics in this category deal with information regarding the deployment of a system. This includes topics that directly describe artifacts and processes related to the system's *deployment*, but also *dependencies* and *plugins*. Further, information about the *target platform* as well as the *runtime environment* is included in this category.

5. Management: This category 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*.

Table 7: Topic Categories and their Popularity

Category	Pop.	Topics
Requirements	51%	functional requirements & domain concepts, non-functional requirements, competitors & related work, user interface & user interaction
Community	45%	community & contributions, communication, conferences
Project Knowledge	34%	tips, tricks & tutorials, API, project documentation
Deployment	33%	deployment & dependencies, target platform
Management	33%	release management & announcements, continuous integration, version control
Implementation	29%	source code, solution concepts & technology
System Design	26%	architecture & packages, database access & external data
Maintenance	25%	corrective maintenance, debugging & troubleshooting, testing
Activities	16%	development activities
Business	15%	licensing

6. Implementation: These topics describe implementation details like *source code*, *solution domain concepts*, and other *technology* related artifacts.

7. System Design: Topics that describe *architecture concepts* and discussions about dependencies and access to *external data* belong to the system design category.

8. Maintenance: This category contains topics related to *corrective maintenance* and *testing*. *Debugging* and *troubleshooting* also belong to the maintenance category although they are part of the entire development process. The reason for this is that debugging and troubleshooting are corrective tasks.

9. Activities: Topics in this category describe *development activities* like modeling, learning and implementing.

10. Business: This category comprises topics that deal with legal issues like *license* questions as well as business strategies and discussions about the *open source* and closed source character of a project.

Table 7 shows the identified categories and how the topics are distributed among them. Further, the table shows the popularity of each category, which is the percentage of all blog posts that contained at least one of the according topics.

The most popular category is *Requirements*, which is predominant in more than half of all blog posts. The second most popular category is *Community,* which is predominant in 45% of all blog posts. We think that this result is reasonable, since community building is a major goal of social media. *Implementation* and *System Design* topics are present in about 30% of all posts. Based on these results, it seems that developers use blogs primarily to describe system requirements and new features. Source code and low-level concepts seem to be less frequent.

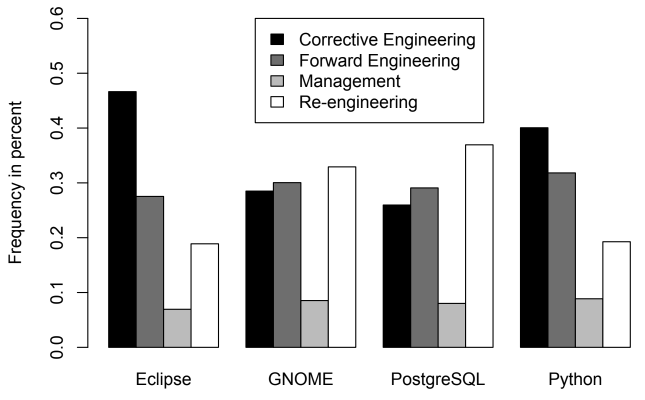

Figure 4: Frequencies of Commit Categories

5. BLOG INTEGRATION

To study *how* blogging activities are *integrated* into developers' workflows, we explore publishing patterns of blog posts and commit messages and investigate relations in the content of these artifacts.

5.1 Publishing Patterns

We examined *when* developers post new blog entries. We wanted to find out if developers make a particular use of social media after having accomplished certain types of development activities. To answer this question, we investigated developers' work descriptions in the commit messages. That is, our goal is to identify from the commit message the development activity performed before writing a post.

To this end we classified the commit messages using the classification algorithm proposed by Hattori and Lanza [8]. 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 82% of all commit messages, while 1% of all 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 4 depicts the classification results.

In the next step, we used a sequential pattern mining algorithm by Zaki [18] to analyze the sequences of developers' commit messages and blog posts. Sequential pattern mining [1] 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 minutes. This represents the mean session duration empirically found by Maalej and Happel [10].

From the data sets we created a sequence database for each project 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 post belong to this sequence. Items that occur within 120 minutes belong to same element within the sequence. For example the sequence $\langle\{f\},\{cf,r\},\{m\},\{c\},\{B\}\rangle$ denotes five elements ending with a blog post. The second element contains the three items c, f, r which took place within 120 minutes. 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 surveyed community.

We analyzed these sequence databases in order to answer the following questions:

- Given developers' blog posts, what is the probability that the last commit message belongs to a certain category?
- Given a commit message in a certain category, what is the probability for a blog post after that commit?

For each project, we generated all sequential patterns of length 2 that ended with a blog post with a maximum gap value of 1. 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 most 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 commit messages of 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 5 shows the results.

We compare these results to the commit classification results (depicted in Figure 4). In the GNOME and the PostgreSQL projects there are less commit messages describing corrective engineering activities than commit messages describing forward engineering and re-engineering activities.

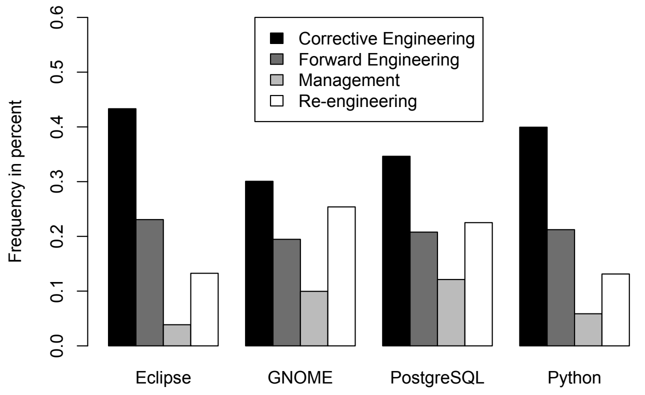

Figure 5: Categories of Commits Before Blog Posts

Table 8: Blogging Probability after Commits

Commit type	Eclipse	GNOME	PostgreSQL	Python
Corrective Engineering	1.49%	2.67%	2.17%	4.20%
Forward Engineering	1.35%	1.64%	1.16%	2.81%
Management	0.90%	2.95%	2.46%	2.79%
Re-engineering	1.13%	1.95%	0.99%	2.87%

Nevertheless, more blog posts are preceded by commit messages describing corrective engineering activities.

We also calculated the average probability for a blog post, given a commit message of a certain category. Table 8 shows the results. In the Eclipse and Python communities most blog posts are published after corrective engineering activities. GNOME and PostgreSQL developers on the other hand publish most blog posts after management activities. Although the situation is different across the studied communities, blog posts are more likely to happen after commit messages describing corrective engineering or management activities, than after forward engineering or re-engineering commits.

5.2 Content Dependency

In the last question we tried to find out if there is a dependency between the *content* of commit messages and the *content* of blog posts. To achieve this, we randomly selected a set of 200 sequences containing at least three commits and a 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)

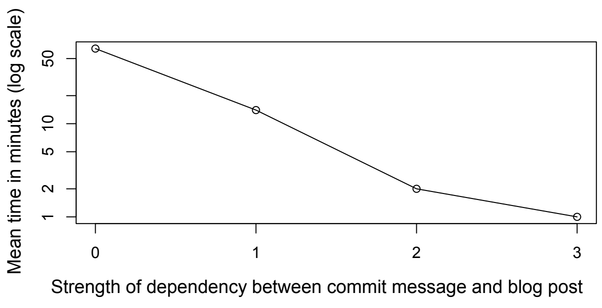

Figure 6: Dependencies between Blogs and Commits in Term of Time

- **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 (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. We think this percentage shows that developers use blogs to illustrate their changes to the code.

To understand the influence of the time between commit messages and blog post on this result, we calculated the average time period for each grade. Figure 6 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. This means that developers are more likely to publish information about their recent activities than about old activities.

6. 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.

6.1 Mining Blogs

To our knowledge there are no published empirical studies on how developers use blogs. However, there exist several studies on mining blogs in general. We distinguish between 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. [6] propose a generative blog topic model to identify external influences on bloggers and their topics. Tseng et al. [16] 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. [5] 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. [13] 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.

6.2 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 [8] 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 [11] used NLP techniques to analyze personal notes and commit messages. They found several regularities in how developers describe their work. This work extends these findings by showing that developers also describe their activities in their 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 [17]. 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. [3] 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. In our work we found similar relations between blogging as social activity and development activities on an individual level.

6.3 Integrating Social Media

Recent papers [2, 7, 15] suggest the integration of social media into the development environment and development processes. Guzzi et al. [7] claim that integrating blog user interfaces into the IDE would foster the reuse and sharing of program knowledge. Treude and Storey [15] 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 can be used to further tighten this integration.

7. DISCUSSION

We observed regular social activities in open source communities. Individual developers only blog occasionally, but the community as a whole constantly shares information and produces several blog posts per day. The more members a community comprises, the more frequently new posts emerge. In particular committers actively contribute to the community blogs. They have the largest number of blog posts. They also publish more frequently and over a longer time period than other members.

Developers continuously describe their work in short and precise commit messages. Blog posts are less frequent than commit messages, but comprise significantly more content. They rarely include source code but frequently high level information and images. Blog posts seem to have rather the character of short documentations, tutorials, and howtos.

Studied developers blog in a high level of abstraction (i.e. requirements and domain concepts). At first glance this is surprising, as we expect developers to use models, technical abstractions, and source code related concepts. However, the public availability of blog information and its general audience explain this granularity.

Further we observed a high amount of posts dealing with community related topics like upcoming conferences. This seems to be typical for social media. Developers include this information orthogonally to other topics in their posts. We think that in particular open source communities depend on the active discussion of social and collaboration activities, as dissemination news and requests for contribution.

We found that developers post more often after corrective engineering or management tasks than after forward engineering or re-engineering tasks. One silent implication of this finding is that bug fixes and management activities represent important information for all stakeholders in a software community. On the one hand, communicating the 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, a high number of solved issues indicates a healthy project. On the other hand, posts about management activities like release announces and release plans make the community aware of the overall project status. Moreover blog posts frequently contain information about recent activities described shortly in previous commit messages.

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. Tools can help developers to reuse available knowledge in their posts, e.g. by linking to blogs and wikis, or capturing particular screenshots. Moreover, tools may annotate blog posts with frequent topics to facilitate information structuring and access. 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.

8. LIMITATIONS

We made four simplifying assumptions during our analysis, which might imply limitations for validity of the results. First, to connect blogs and commit messages we mapped the names of committers and other bloggers using a text similarity algorithm. We chose a pessimistic mapping that creates rather false negatives than false positives. That is, the size of the data sets was rather affected than the analysis results. As a side effect, few bloggers, which we classified as other community members might be committers. Second, 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 different time settings. On the other hand, considering activities within two hours as the same session reduces the effects of such synchronization errors. All the 200 manually investigated sample sequences, were correctly ordered. Third, we used the heuristic of Hattori and Lanza [8] 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. Finally, we selected a number of Eclipse and GNOME subprojects, and could not analyze the behavior of *all* committers in these communities. We think our large selection is representative, which was confirmed by active members of both communities.

9. CONCLUSION

How do developers blog? In this exploratory study we found that developers post a new project-related blog entry every 26 days, on average. While 22% of the posts include screenshots, only 1.8% contain source code. Developers frequently link to existing information like Wiki pages and other blog posts. Topics representing high-level concepts such as *domain concepts and functional requirements* are predominant in more than one third of developers' blog posts, while less than 15% deal with source code concepts. Over 37% of developers' posts include community related information. Further, developers are more likely to blog after corrective engineering and management activities 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. We think that a hypothesis-driven research should further explore the role of social media and integrate it into development processes and tools. Currently we are also studying how developers use other social media like micro-blogs or content communities. In addition we plan to replicate our results using different techniques such as semantic methods and author-topic modeling techniques.

10. ACKNOWLEDGEMENT

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.

11. REFERENCES

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14. IEEE Comput. Soc. Press, 1995.

[2] A. Begel, R. DeLine, and T. Zimmermann. Social media for software engineering. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, pages 33–38. ACM, 2010.

[3] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining Email Social Networks. In *Proceedings of the 2006 international workshop on Mining software repositories*, pages 137–143. ACM, 2006.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3(4-5):993–1022, May 2003.

[5] N. Glance, M. Hurst, and T. Tomokiyo. BlogPulse: Automated trend discovery for weblogs. In *WWW 2004 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics*, volume 2004. Citeseer, 2004.

[6] D. Gruhl, D. Liben-Nowell, R. Guha, and a. Tomkins. Information diffusion through blogspace. *ACM SIGKDD Explorations Newsletter*, 6(2):43–52, Dec. 2004.

[7] A. Guzzi, M. Pinzger, and A. van Deursen. Combining micro-blogging and IDE interactions to support developers in their quests. In *Software Maintenance (ICSM), 2010 IEEE International Conference on*, pages 1–5. IEEE, 2010.

[8] L. P. Hattori and M. Lanza. On the nature of commits. *2008 23rd IEEE/ACM International Conference on Automated Software Engineering - Workshops*, pages 63–71, Sept. 2008.

[9] A. M. Kaplan and M. Haenlein. Users of the world, unite! The challenges and opportunities of Social Media. *Business Horizons*, 53(1):59–68, Jan. 2010.

[10] W. Maalej and H. Happel. From work to word: How do software developers describe their work? *Working Conference on Mining Software Repositories*, pages 121–130, 2009.

[11] W. Maalej and H.-J. Happel. Can development work describe itself? *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pages 191–200, May 2010.

[12] W. Maalej, D. Panagiotou, and H.-J. Happel. Towards Effective Management of Software Knowledge Exploiting the Semantic Wiki Paradigm. In K. Herrmann and B. Brügge, editors, *Software Engineering*, pages 183–197, Bonn, Germany, 2008. GI.

[13] X. Song, Y. Chi, K. Hino, and B. Tseng. Identifying opinion leaders in the blogosphere. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 971–974, New York, New York, USA, 2007. ACM.

[14] The Nielsen Company. Led by Facebook, Twitter, Global Time Spent on Social Media Sites up 82% Year over Year. 2010.

[15] C. Treude and M.-A. Storey. 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*, pages 12–22, Washington, DC, USA, 2009. IEEE Computer Society.

[16] B. Tseng, J. Tatemura, and Y. Wu. Tomographic clustering to visualize blog communities as mountain views. In *WWW 2005 Workshop on the Weblogging Ecosystem*. Citeseer, 2005.

[17] S. Wasserman and K. Faust. *Social network analysis: methods and applications*. Cambridge University Press, 1994.

[18] M. Zaki. SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning*, 42(1):31–60, 2001.