Summer 8-2-2013

Online Ecosystems in Software Development

Corey J. Jergensen
University of Nebraska-Lincoln, cjergens@cse.unl.edu

ONLINE ECOSYSTEMS IN SOFTWARE DEVELOPMENT

by

Corey Jergensen

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Professor Anita Sarma

Lincoln, Nebraska

June, 2013

ONLINE ECOSYSTEMS IN SOFTWARE DEVELOPMENT

Corey Jergensen, M.S.

University of Nebraska, 2013

Advisor: Anita Sarma

Software projects are no longer developed as individual, monolithic projects. Instead, they exist as part of an ecosystem where related projects are developed together using a common underlying technical infrastructure and common project cultures.

This work explores characteristics of online software communities by comparing and contrasting two software ecosystems that are both related to programming, but provide different functions. The first is Stack Overflow, a programming question and answer forum. The second is Gnome, an open-source, Linux desktop environment. Both are examples of online communities because: (1) the communities are composed of smaller projects or topics, (2) users contributions are provided almost entirely by volunteers, and (3) they are managed by their members.

In this work, we investigate various aspects of each community. We start by observing how users join each community. We find that the shared infrastructure of the Gnome ecosystem allows users to join additional projects within the community at an accelerated rate. In Stack Overflow, we find there are a higher number of users attempting to contribute than typically observed in other online communities. We observe that the reduction in the overhead in joining communities leads to a higher number of contributions being provided by a larger base of users.

We then study the distribution of contributions amongst members in each community. We find that each community has a skewed distribution of contribution with a small minority providing more content than the remainder of the community. In

Stack Overflow however, we observe that the high and low contribution groups of users provide a similar amount of contribution, a finding different from what has been observed in other communities.

Last, we look at the role experienced users fill in the community. In GNOME, tenured users contribute less code and may spend an increased amount of time on community and project management. In Stack Overflow, tenured users may perform moderator tasks that help the community function and provide feedback on the site's operation. In each community, tenured users show a vested interest in the community and its continued success.

ACKNOWLEDGEMENTS

I would like to thank a number of individuals who have provided help and support. My advisor, Dr. Anita Sarma, for enabling me to complete this research, her much needed direction and especially her patience. Dr. Patrick Wagstrom for his assistance and collaboration in the studies of the Stack Overflow and Gnome communities. The members of my defense committee (Dr. Anita Sarma, Dr. Myra Cohen and Dr. Patrick Wagstrom) for taking time to review and provide feedback on my thesis. The various grants that have supported my work (AFSOR FA9550-10-1-0406, NSF IIS-1110906 and NSF CCF-1016134).

Contents

1 Introduction **1**
1.1 Research Questions 4

2 Background **7**
2.1 Open Source Development Communities 10
2.2 Online Question and Answer Forums 11

3 Stack Overflow **13**
3.1 Community Description 13
3.2 Research Questions 17
3.3 Data Collection 19
3.4 User Distribution 21
3.5 Contribution Distribution 25
3.6 Tenured Users 29
3.7 Summary 31

4 GNOME **33**
4.1 Community Description 33
4.2 Research Questions 35
4.3 Data Collection 36
4.3.1 Subset of Projects 37
4.3.2 Project Metrics 38
4.4 User Joining 42
4.4.1 Progression Paths 42
4.4.2 Broad Factors that Affect Code Centrality 46
4.5 Contribution Distribution 49
4.6 Tenured Users 51
4.6.1 Predicting Source Code Centrality 51
4.6.2 Project Tenure and Source Code Centrality 53
4.6.3 Ecosystem Tenure and Source Code Centrality 54
4.7 Summary 55

5 Threats to Validity **57**

6 Conclusions **61**
6.1 Future Work 64

A Stack Overflow **66**
A.1 Data Schema 66
A.2 High and Low Contribution User Breakdown 71
A.3 Interviews 74

B Gnome **76**
B.1 Data Schema 76

Bibliography **79**

List of Figures

3.1 Example Stack Overflow user profile. 14
3.2 Example list of Stack Overflow questions. 15
3.3 Example Stack Overflow question with answer and comments. 16
3.4 Number of high and low contribution users by month. 26
3.5 Number of answers provided by high and low contribution users by month. 26
3.6 Number of high and low contribution users providing accepted answers by month. 27
3.7 Number of accepted answers provided by high and low contribution users by month. 28

4.1 Example progression paths that can be taken by GNOME developers. 44

B.1 Schema diagram for GNOME community data. 77
B.2 Schema diagram for GNOME community data. 78

List of Tables

3.1 Characterization of the Stack Overflow community based on different contribution roles that users occupy. 24

4.1 Progression across social and technical mediums. 45
4.2 Major Component Loadings from Principle Component Analysis. . . 47
4.3 PCA regression model. 48
4.4 Summary of the amount and relative percentage of high and low contribution users and the contribution they have provided to projects. . 50
4.5 Base regression model. 52
4.6 Enhanced regression model. 53
4.7 Experience regression model. 55

A.1 Summary of the amount and relative percentage of high and low contribution users and the contribution they provide by month. 72
A.2 Summary of the amount and relative percentage of high and low contribution users providing accepted answers and the amount of accepted answers they provide by month. 73

Chapter 1

Introduction

Software development is as much a social activity as a technical activity. Software is typically developed in teams that are often globally distributed. These teams are comprised of developers with different levels of expertise and knowledge about different aspects of the project. They therefore need to coordinate among themselves to correctly complete their tasks. For example, in a web application, a front-end developer may require the expertise of another developer with whom he or she needs to interact with regarding the project API. Developers therefore need to interact with others to seek expertise and coordinate or reallocate tasks between collaborators.

Newcomers to a project will typically not be aware of this knowledge distribution or know from whom to seek help. They first need to not only learn the technical infrastructure of a project, but also the social norms and culture in the project. For example, in order to contribute code to a project, a developer must not only learn about how to use the code tracking (versioning) system used by the project, but also the social norms surrounding the code contribution process in the project. A project, for instance, could require commit messages to follow a certain format and if the changes submitted do not include the appropriate commit message format it might

be rejected without consideration.

These same principles extend beyond the code management process. Projects include a variety of infrastructures and a required set of communication and coordination processes around these systems. In open source development, projects typically employ at least an issue tracking system and mailing lists to facilitate operations.

The use of different kinds of infrastructure (i.e. bug tracker, code tracker, etc.) across projects can increase the overhead and put additional burden on users who are required to use different systems in each project. If a user needs to learn how to operate a different issue tracker for each project they use, they may choose to report issues to fewer projects. To reduce the complexity of working with all of these systems, projects may choose to use the same systems across a range of projects. By using a common infrastructure for multiple projects, the amount of effort required for an individual to become part of a project can be reduced.

Interestingly, projects are colaesing around a common set of tools and infrastructures. This might occur either because these projects socially or technically depend on each other (Gnome community [1]) or because they are hosted through a common site (Source Forge [2], GitHub [3]). In this work, we define such groups of projects as online software communities or software ecosystems. For example, an open source community would share components that manage code or track issues while a question and answer based community would share a common platform for getting questions answered. This commonality helps users by providing a consistent experience.

This work will focus on online software communities with two different types of content: question and answer forums and open source development. We choose these domains because they are key parts of software development. Question and answer

[1] http://www.gnome.org
[2] http://sourceforge.net
[3] http://github.com

forums provide assistance to developers who need help with programming or programming related tasks. A shared infrastructure that transcends multiple programming languages or programming topics allows developers to provide contributions to a wide range of topics. Open source communities on the other hand provide a common location for developers to collaborate on software development. The shared infrastructure lowers the boundaries of joining additional projects within the community and encourages collaboration between projects.

The first community that we investigate, Stack Overflow, is an online, technical question and answer forum dedicated to programmers and programming related tasks. Stack Overflow has seen tremendous growth since it was founded in 2008. With over 1.9 million registered users that have provided 9 million answers to 4.9 million questions, Stack Overflow has become a sizable resource for individuals seeking programming assistance. On the flip side, users who have knowledge and expertise and would like to share their knowledge. The site is organized to allow easy discovery of topics where a user may have answers. Users can garner reputation in the community by providing answers to the questions of others. We consider Stack Overflow as an ecosystem because it provides a common platform where developers with different programming questions can seek assistance without the requirement of learning a new system for each project or language.

The second, GNOME is a linux desktop environment originally announced in August 1997. The community has become home to a myriad of desktop tools from email clients and system utilities to music players and games. The community is home to over 1,200 projects that have been contributed to by over 1,000 developers. GNOME is a good representation of a software ecosystem because of the technical dependencies between projects in the community and its utilization of common systems for users to (1) discuss the projects via mailing lists; (2) report and manage program issues via

the issue tracker; and (3) manage individual project code via a software configuration management (SCM) system. These shared systems lower the boundaries between projects by providing the same experience and access across projects and therefore encourage project participation.

Beyond both being programming related communities, these communities also share other traits. First, each community is voluntary. Participants in the community choose to participate by contributing to projects they find interesting. Each community also has a socialization process. New users must first learn how to contribute and learn about the project culture before they can become part of the community. Finally, these communities are run and operated by the community. Each community is responsible for its own maintenance and success.

1.1 Research Questions

A primary goal of our work is to understand online communities by investigating users, their contributions and their progression in becoming a member in these communities. The insight gained from these questions can then be used to improved existing communities and shape new communities. As mentioned earlier, we studied two communities in particular: Stack Overflow and GNOME. In this section, we present three research questions that drive our research. The knowledge gained from these questions can be used to to improve the effectiveness of other online, software communities.

RQ1: How do users join online, software communities?

In the first research question, we investigate the process that users take when joining an online, software community. Success of any community depends on sustaining its user base and its contributors. This is especially true for online communities that

depend largely on voluntary contributions. Therefore, a key challenge for any online community is to ensure the socialization process of newcomers such that there is a steady growth of contributing members. In this question we seek to understand the joining process of each community and their effectiveness at socializing newcomers. A better understanding of users' socialization process in these communities will provide knowledge to other communities about the involvement of newcomers to their communities.

Newcomers to a community are not the only important part of online, software communities. Effective communities also rely on the contributions from their regular user base. Work or contributions to an online community is distributed across its members.

RQ2: How are the contributions in online, software communities distributed amongst community members?

Typically, the majority of an online community's contribution is provided by a small group of core users [21, 36]. This is known as a power law distribution (Pareto law or law of the vital few) [36] and has been found in other online communities [9, 20, 35]. Here we seek to investigate the distribution of contributions amongst users in online, software communities.

RQ3: What is the role of tenured users in online, software communities?

In the final research question, we seek to understand the behaviour of senior members in these communities. These members are likely to have accrued substantial knowledge of their community and its operation. Here we investigate the behaviour of these users and how it differs from other members in the community. We do so by studying the role tenured members occupy in a community and their impact on the project. We first investigate if their role differs from the communities' newer members. We then investigate the role that these users occupy and how this affects

the communities in which they participate.

The remainder of this thesis is organized as follows: In Chapter 2 we discuss background and related work. Chapter 3 and 4 describes our studies on Stack Overflow and GNOME communities. Chapter 5 discusses threats to validity. Chapter 6 concludes with a summary of our findings by comparing the communities and our proposed future work.

Chapter 2

Background

In this chapter we discuss previous work that has investigated online and open source development communities. We begin by discussing general characteristics of online, software communities before discussing open source development and Q&A forum research in more detail.

Previous work on online communities have studied the process by which users gain access and become involved [6,11,28]. Qualitative work by von Krogh et al. found that users must go through a "joining script" when becoming a contributor to a project [47]. This "joining script" represents a barrier that the user must overcome before becoming part of the project [47]. Other studies have found other barriers that must be overcome before joining a project [18, 24]. In some communities, commit access is only given after a new developer has proved their potential as an active community member [16, 47]. In most cases, it is the responsibility of the user to identify technical tasks appropriate to their skill set to start contributing [16, 47]. In their analysis of the Freenet project, von Krogh et al. found that only one in six newcomers were given specific tasks to work on when starting out on a project. Many projects lack an explicit architecture or system design, making it difficult for newcomers to understand

the system before they can start contributing [47]. A study of new developers in a corporate setting found that similar barriers exist when developers join a project [14]. These new developers must learn about the technical landscape and the social culture of any projects they work on to contribute. All of these barriers limit the number of potential contributors to projects.

Previous work on online communities has found a very skewed distribution of contribution amongst community members with a core group of users providing the majority of the site's content and the majority of users providing minimal contributions [21, 25, 36]. A study of the Yahoo! Answers community, a general question and answer forum, found that the majority of users have posted only one question [21]. Further, a study of Wikipedia, an online encyclopedia created entirely by its users, found that the modal number of edits for a user is one [25]. These communities follow a power law distribution, also known as the law of the vital few or the Pareto principle.

Even when a community can attract new users, retaining members can be difficult, especially in the case of voluntary contributors [40,50]. A study of Wikipedia showed that 60% of editors did not return after their first day of membership [25]. A case study of the various open source communities found that the social network and the strength of the ties in a community was a good indicator for retention of members [40].

The majority of users in an online community only consume content while only a small fraction provide content [21, 29]. These user are known as lurkers or free riders [21]. Previous work has shown that users may "lurk" for a variety of reasons. Some users may "lurk" to learn about a community before contributing [29, 37, 44], waiting until they become comfortable with the community [44]. Others may be learning about a new topic [29].

Work has also been done to show the impact of tenure on users in an online

community. In question and answer forums, research has shown that the quality of a user's answers increases as they are socialized and gain experience with the community [17,23]. In open source communities, the consistent finding is that members who play a more central role in a project exert more influence on the technical direction of the project [7, 16, 30]. Duchenaut claimed that developers are required to have a certain social status before they can implement a high impact change [16]. von Krogh et al. found that developers who had contributed to a project across multiple releases made more far-reaching changes while new developers made more localized changes [47]. Research in the Apache and Postgres open source communities found that developer status was dependent on the tenure of the individual and that the social status of an individual was a strong criterion for success.

There exist many reasons why users contribute to projects including identification with the community, altruism, willingness to help others and feelings of personal gratification when the community succeeds [36,51]. Studies have found that monetary rewards help in creating high quality answers [22, 25]. These rewards have been found to attract more views, but not necessarily increase participation from the users [51]. Another study found that software development environments are increasingly incorporating social interactions allowing developers to make inferences about other based on the visible cues of development activity and social reputation [13]. Users may contribute to obtain reputation within the community. The visibility gained within the community is considered a strong motivational factor [53].

Users may also contribute for more self fulfilling reasons. Higher performance from users was noted when users were presented high-challenge goals. It was found that completing these tasks helped to raise the contributor's self-efficacy [4]. Users may also be motivated by altruistic factors. Constant et al. found altruism to be a strong motivation for users answering questions [10]. Additional work has also shown

that users are driven to contribute towards the common, public good [24, 30, 50].

2.1 Open Source Development Communities

Open source projects have been studied by researchers in many areas of the computer and social sciences including software engineering, computer-supported cooperative-work and management [12, 45]. Many modern open source projects resemble large enterprise projects that are comprised of numerous smaller, related projects [8]. These open source communities though are composed of a mixture of members such as volunteers, students, and industry paid contributors [8]. One example is Eclipse which is operated by the Eclipse Foundation. Eclipse has contributors from a number of large technology companies including IBM, Intel and Oracle, in addition to substantial participation from volunteers and university students.

Open source projects target many different problem spaces. These include desktop environments (KDE, GNOME), web servers (Apache), programming languages (Perl, PHP) and integrated development environments (Eclipse) [31].

Open source projects have an underlying social organization by which the projects operate [5, 7, 33]. Research into the social structure of open source projects has been performed in a wide variety of projects including Apache [34], Freenet [47], Netscape [26], Mozilla [35], Python [16] and others [45]. These studies show that these projects have a layered userbase known as the "Onion Model". In the Onion Model, a project has a central core of members which control the operation and future direction of the project surrounded by layers of members with decreasing involvement in the project. This model is generally considered meritocratic because as members gain experience through contributions to the project, they migrate toward more central roles in the project. For example, the central core project members may contribute

a majority of the project's source code. This core would then be surrounded by a layer of members who report project bugs which in turn would be surrounded by a layer of members who do no more than use the project. Despite the amount of research performed, no consistent naming scheme has arisen for the roles in open source projects [16, 52].

2.2 Online Question and Answer Forums

Ackerman and Malone were the first to investigate Question and Answer (Q&A) forums [1]. They proposed a blueprint for Q&A forums in their design of Answer Garden in 1990. They suggested in their work that providing a common repository of questions and their answers would encourage users to contribute because they would not have to duplicate their answers across forums.

General purpose Q&A forums often suffer from a poor percentage of questions answered, between 5-53% of questions do not receive a response [2, 15, 23]. There are many hurdles to overcome when getting a question answered. The general-purpose nature of Q&A sites typically attract conversational posts geared toward eliciting discussions from the community [25]. These subjective questions receive opinionated answers that may not answer the original question. On top of this, responses to a question do not always guarantee an answer [23] and questions may receive no answer at all if they are poorly written [15].

Investigations of Yahoo! answers, which at one point controlled 74% of the Q&A forum traffic in the United States [22, 25], have found that most users provided only questions or answers while only a fraction of the user population provided both [21]. Previous work also shows that the majority of users (70%) provide only questions while a small portion of the users provide the site's answers [21].

Study of KiN, a Q&A forum in South Korea, found altruism, learning, and competency motivate top users to answers, but that user participation is often sporadic [36].

In previous studies of Stack Overflow, Mamykina et al. found that questions receive an answer very quickly with a median answer time of 11 minutes [32]. They also found that frequent users have a higher answer ratio and account for 66% of the total answers [32]. In a different study of Stack Overflow, Otkay et al. found that when answers are of equal quality, users prefer newer answer and that the presence of existing answers does not discourage additional users from answering [41]. Additionally, Treude et al. qualitatively analyzed question characteristics and found that Stack Overflow is particularly effective for code reviews, conceptual questions or for novices [46]. Work by Parnin et al. [43] found that Stack Overflow can be a rich source of documentation for software APIs. They found that harnessing the crowd to create documentation produced a rich set of code examples and discussions.

Chapter 3

Stack Overflow

The first community we explore is Stack Overflow, which is an online site dedicated to programming and programming related questions and answers. Since its inception, Stack Overflow has become a defacto repository for finding answers to programming related tasks. Programmers can ask a variety of questions and expect a quality answer or showcase their knowledge of certain topics all in a centralized location.

We choose to focus on the Stack Overflow community because of its rapid growth and success at growing its user base. Here we first consider the success of users in the Stack Overflow community. We then investigate the distribution of work between groups that exist in the Stack Overflow community. Last, we examine the role of tenured users within the community.

3.1 Community Description

Stack Overflow was founded in August of 2008 by Jeff Atwood and Joel Spolsky. Discouraged with current options for seeking programming related assistance online, they created a unified platform where individuals with a wide variety of questions could find answers. The site has experienced tremendous growth since its inception

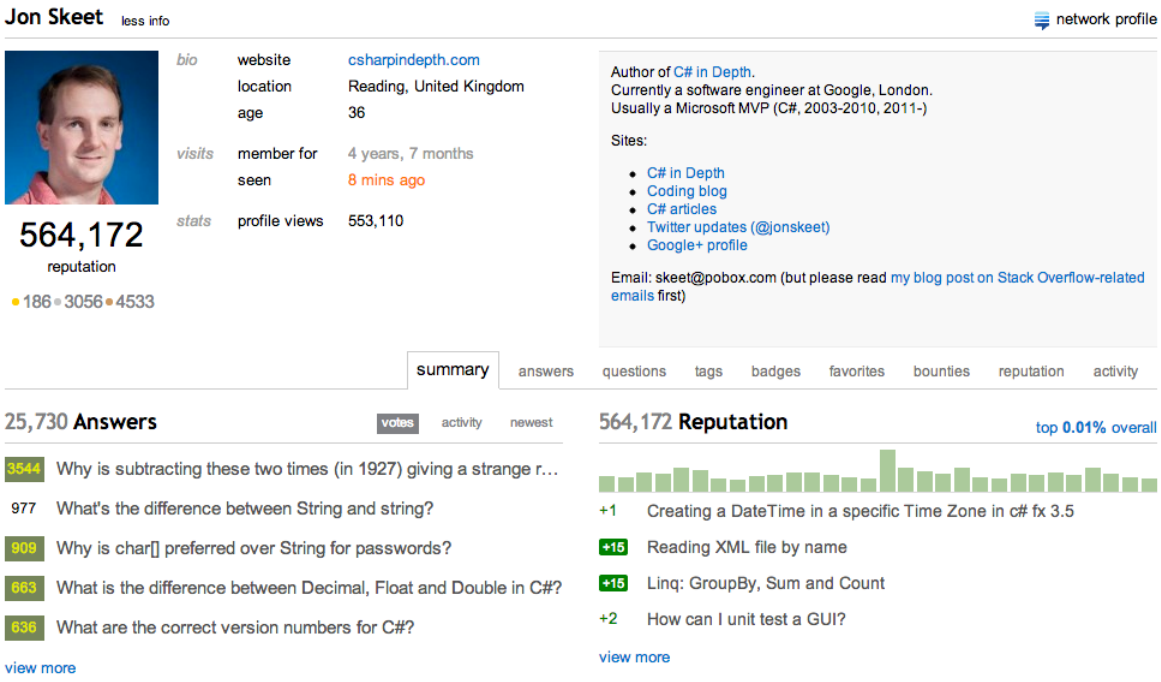

Figure 3.1: Example Stack Overflow user profile.

and has quickly become the de facto source for programming related assistance online. As of April 2013, the site boasts 9 million answers to 4.9 million questions provided by 1.9 million users.

Stack Overflow has the following main components that allow the site to function:

- **Users** are the site's content providers. Anyone who visits Stack Overflow can view the site's content, but users are required to register an account (or name and email address when answering as a guest) before participating. In this work, we refer to users as individuals with registered accounts. For an example of a user profile, see Figure 3.1.

- **Questions** are inquiries by the site's users. Questions can be posted by any user with an account. Stack Overflow requires that questions be objective and seek factual answers. Questions that don't follow this rule are deleted, closed or moved by the community. Each question can also be marked with up to five

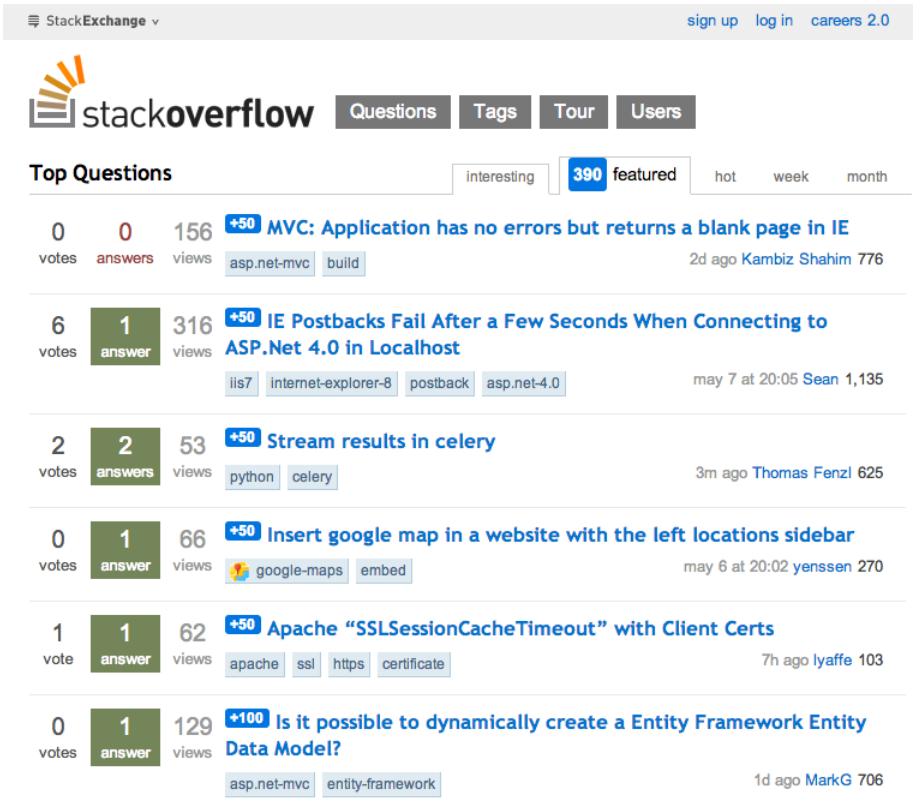

Figure 3.2: Example list of Stack Overflow questions.

tags. For a list of sample questions on Stack Overflow, see Figure 3.2, or a more detailed view of a question, see Figure 3.3.

- **Tags** are simple strings of text. Tags can refer to but are not limited to programming languages, development environments or more general topics such as "templating" and are used to organize questions to make it easier for users to find questions they can answer. A question with sample tags can be seen in Figure 3.3.

- **Answers** are responses to questions. A question can have any number of answers contributed by Stack Overflow users. Once one or more answers have been provided for a question, the original poster is advised to select an answer

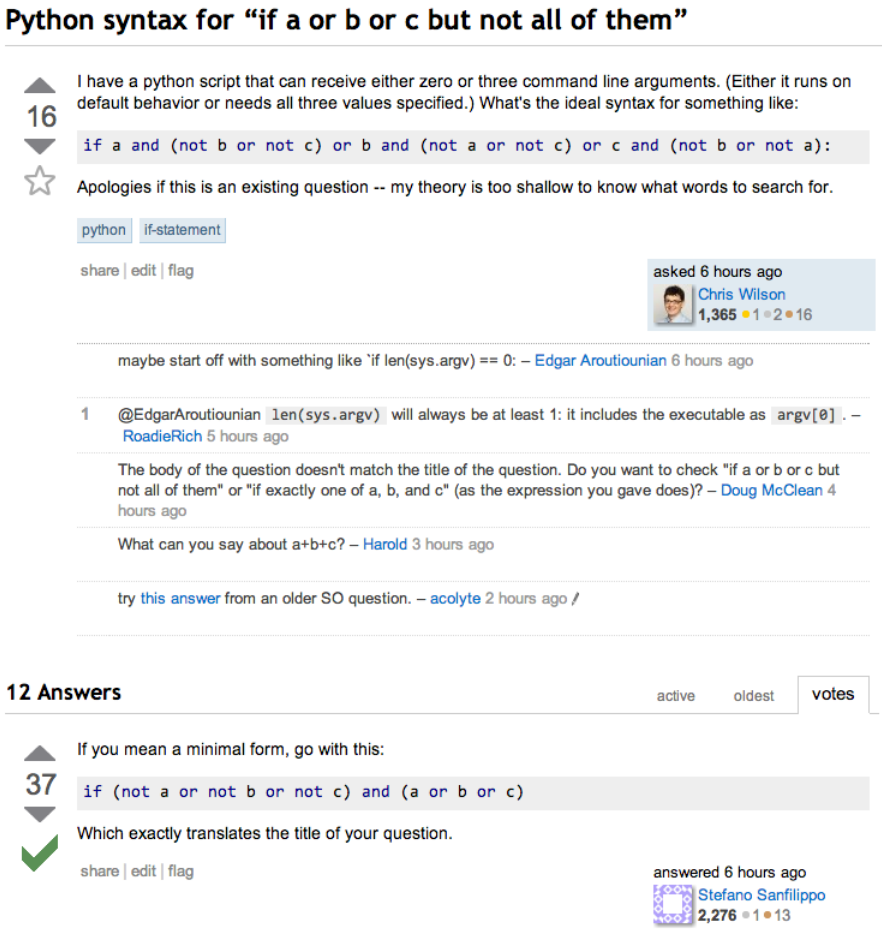

Figure 3.3: Example Stack Overflow question with answer and comments.

which they deem correct or the most helpful by “accepting” it. An accepted answer will always appear first in the list of answers for a question and is easily identified by a large green checkmark. A sample answer can be seen in Figure 3.3.

- **Comments** are responses to questions or answers that can be used to clarify or discuss the question or answer. Stack Overflow discourages users from posting answers to obtain clarifications because both questions and answers should be objective. Sample comments in response to a question can be seen in Figure 3.3.

- **Votes** can be cast for any question or answer to reflect its usefulness or quality.

Note that good quality questions and answers garner positive votes or "upvotes" while low quality questions and answers garner negative votes or "downvotes". Therefore, the total number of votes a post receives is an indicator of the quality of the post. Users can then gain or lose reputation based on the votes their questions and answers receive.

- **Reputation** is the total number of points a user has gained on Stack Overflow. Reputation is obtained from the votes of others on posts provided (10 points for an answer upvote, 5 points for a question upvote), having answers accepted (15 points) and accepting answers to questions a user has asked (2 points). Reputation can be lost when a user's content is downvoted (-2 points) or when downvoting another user (-1 point). Reputation also serves the secondary function of unlocking new abilities on Stack Overflow once a user has obtained enough reputation. These abilities include basic tasks such as voting (15 points required) or commenting on other's posts (50 points required) as well as more important abilities like editing other user's posts (2000 points required) or closing questions that are not suitable (3000 points required).

3.2 Research Questions

We begin by addressing the first research question in the scope of the Stack Overflow ecosystem.

RQ1: How do users join Stack Overflow?

In this question, we investigate how users become involved with Stack Overflow. We divide users into groups based on their success at providing contributions that are accepted by the community (providing answers that are accepted and questions that have accepted answers). We then observe the number of users in each group to

judge the effectiveness of Stack Overflow in socializing new users.

By dividing the users of Stack Overflow into independent groups, we examine the division of the users. This presents a view of the site from the perspective of the users involved. This helps us to understand the distribution of different kinds of users within Stack Overflow, but does little to examine the actual operation or active contributors of the site. To better understand this we present the next research question.

RQ2: How are contributions in Stack Overflow distributed amongst community members?

In this question, we explore the breakdown of contribution for different groups of users in Stack Overflow. We define two classes of contributors, high and low contribution, and analyze the amount of content provided by each group. We define different user groups because the previously defined roles capture the success of a user but do not address their continued participation in the community.

This by itself though does not fully explain the breakdown of contribution. To explore this idea further, we look at the overall success of each group at providing content by analyzing the number of accepted answers provided by each group.

In the final research question, we seek to understand the role of Stack Overflow's senior members. These members of the community have likely accrued a substantial amount of knowledge about the community and its operation. Here we investigate the role of these users and how it differs from other members in the community.

RQ3: What is the role of Stack Overflow's tenured users?

To explore the role of tenured users in Stack Overflow, we conduct interviews with some of the community's tenured users. We chose interviews over quantitative analysis because we wanted to understand the motivation and reasoning used by actual users from the community. Interviews allow us to gain insight into the users thoughts and motivations that would not be possible with quantitative analysis. By exploring the

interviewees' experience with Stack Overflow, we can determine the roles these users serve and their motivations in continuing to contribute to Stack Overflow.

3.3 Data Collection

Stack Overflow places a Creative Common wiki licence on all of its user provided content. In tandem with this, they also release a dump of the site's content every few months. The data is released as a series of XML files, each containing a type of community activity (post, comment, etc.). These files also contain data about the connections between the types of activity (see Appendix A.1 for a full description of these files and their content). For example, each post contains a reference to the identifier of the user that provided the post. Comments contain references to the post that they are in response to as well as to the user who provided the comment.

Scripts were written to process the data from the files and convert it into a format that is easier to process. A MySQL database was constructed to store the data from the files. We did so because a relational database allows the data to be queried in a flexible format and simplifies the process of exploring the relations that exist between data items. For example, a query can easily be constructed to select all posts provided by a single user in a given period of time. This query can then be refined to select only answers provided during that time period.

Data used in this work was extracted from the June 2011 data dump of Stack Overflow and its sister site's (Server Fault[1], Super User[2], etc.[3]) [3] and covers 34 months of site activity from August 2008 to May of 2011. The data extracted from this dump contains 5.5 million posts (questions and answers), 6.2 million comments

[1]http://serverfault.com
[2]http://superuser.com
[3]https://stackexchange.com/sites

and 13.6 million votes posted by 645 thousand users.

After the XML files were processed, additional processing steps were performed for use in analysis. Each post was characterized by the length of the post, source code contained within the post. Posts were pruned that contained missing references (i.e. answers posted by users who later deleted their account).

First, the number of words was calculated and stored for each post after removing embedded code from the post's body. Embedded code was removed because verbose code segments could artificially inflate the number of words in a post.

Next, posts were marked with whether or not they contained inline or embedded code. We differentiate between inline and embedded code because inlined code likely refers to specific code items while embedded code is likely a code example. The difference here will affect the way a user makes use of the code included.

Then for each post, the reputation of the posting user was estimated at the time that they posted. This was performed according to the reputation gain guidelines posted on the Stack Overflow Frequently Asked Questions page. These rules are as follows

- 10 points for receiving an upvote to an answer.
- 15 points for having an answer accepted.
- 2 points for accepting an answer.
- 5 points for receiving an upvote to a question.
- -2 points for having an question or answer downvoted.
- Bounties can be specified to encourage answers to overlooked posts. In the case of a bounty, one user specifies an amount of reputation points that will be given from their own reputation to the user that provides the accepted answer.

In addition to these rules, there are a few more rules in Stack Overflow which were not taken into account: when downvoting, the user issuing the downvote loses one reputation. We could not include this in our calculation because the vote data that is provided by Stack Overflow does not contain references to users who provide votes. Also, Stack Overflow limits the amount of reputation a user can obtain in a given day, excluding reputation gained from bounties. This limitation dramatically increases the complexity of reputation calculation and therefore was not included. Our reputation points for a user may therefore be higher in cases of users who have exceeded the daily reputation cap.

Lastly, posts with no associated user accounts were removed. Users who have deleted their Stack Overflow account are no longer referenced by the posts they created. This makes analysis on these posts incomplete and thus were removed. In total, 86,643 posts, only 1.56% of the total posts, were removed.

To augment the archival data dump, interviews were performed with members of the Stack Overflow community. These members include one founder/designer, two high reputation and two low reputation users. Users were chosen from different levels of community involvement to obtain a broader perspective of the community's participants. Questions asked during the interviews can be found in Appendix A.3.

Note that there may arize some threats to validity based on how the data was collected and its analysis. We discuss these threats to validity associated with this work in Chapter 5.

3.4 User Distribution

The most important part of a community is its members. Without contributions from its members, a community would not function. It is for this reason we look at the

success of users within the Stack Overflow community. Most communities have a very skewed base of users that contribute, with a small minority providing the majority of the content and a majority of the contributors having little to no contribution [21,36].

Further, most online communities suffer from high attrition rates [40,42,50]. Users may leave a community for a variety of reasons including a lack of interest or external factors such as work. A successful community needs to engage new users and retain individuals as members of the community.

To understand how successful Stack Overflow has been at enabling new users, we look at the "joining" path which users follow. We define joining levels based on a user's success in getting their questions and answers accepted. We note that it is possible that a high quality, useful answer does not necessarily need to accepted. However, we posit that a user who provides high quality posts will eventually be accepted.

We identify five stages that a user may find themselves in when they contribute to Stack Overflow:

Registered Lurkers are users who have registered for an account but have not provided any content. This does not include users who visit the site seeking help without having an account. Lurkers may be users who have not yet contributed or users who have left the site altogether. Research in other communities has shown that individuals may create accounts to observe a community before participating, learn about a new topic or become comfortable with the community before participating [29, 29, 37, 44, 44].

Stalled users are individuals who have contributed but none of their contributions have been accepted. This includes users who have posted one or more questions or answers and have not had any answer accepted or question for which they have accepted an answer. We further divide these users into two categories:

- **Single contribution users** are individuals who have tried to contribute only once and have been unsuccessful. We observe these users separately as they may have joined Stack Overflow and posted once out of curiosity before losing interest and moving out of the site. It is well known that online communities have high attritions rates [40, 50]. The popular, online encyclopedia Wikipedia has recorded that 60% of users do not return after their first day of membership [25].

- **Multiple contribution users** are individuals who have attempted posting (question or answer) at least twice and have not been successful. This would indicate that the users are interested in the community and are attempting to participate. These users may lack the knowledge of the community necessary to provide meaningful content though and remain unsuccessful. For example, a user in this category may ask subjective questions, which are discouraged on Stack Overflow, or they may provide answers that are not clear enough to solve the inquirer's request.

Partial success users are individuals who have been successful with either questions or answers but not with both. Partial success users are users that have attempted more than once in both pursuits, but have not been accepted in both. These are users who are actively trying to be a full contributor but remain only partially successful. For example, a developer who has asked several questions that have been accepted but has been unable to get an answer accepted after multiple attempts would be a partial success user.

Limited users are individuals who have posted only questions or only answers and have been accepted. We can further divide this group into "question only" and "answer only" users. As with the partial success users, we include users who have attempted the other pursuit (questions for answer only and answers for question only)

Table 3.1: Characterization of the Stack Overflow community based on different contribution roles that users occupy.

User Role	**Number**	**Percent (%)**
Registered Lurker	206,373	32.01
Stalled	253,827	39.37
- Single contribution	190,784	29.59
- Multiple contribution	63,043	9.78
Partial Success	17,037	2.64
Limited	101,281	15.71
- Question only	72,338	11.22
- Answer only	28,943	4.49
All Rounder	66,294	10.28
Total	**644,812**	**100**

only once and have not been successful.

All rounder users are users who have been successful in getting both questions and answers accepted.

From the results of this analysis (see Table 3.1), we see there are a large number of registered lurkers (32%). This is to be expected and is typical of online communities [21, 29]. Though this number may seem high, it is not as high as has been reported in other communities which can range from 50% to 90% [38, 39]. We note though that this is only lurkers who have created accounts and does not include all lurkers in the Stack Oveflow community. This may be because Stack Overflow does not require individuals to register for an account if they simply wish to read questions and answers. For this reason we propose that these registered lurkers may be curious about the community and want to be a part of the community or are preparing to post.

The largest group of users on Stack Overflow are stalled users (39.4%). Of these users, the majority have contributed only once (29.6%). These users might have made a contribution out of curiosity, but might not have been engaged by the community

and therefore left. The remainder of the users have tried multiple times without success (9.8%). Though this may only be about 10% of the overall community, this group still represents a large number of users, about 63K, who wish to be part of Stack Overflow but have been unable to provide accepted content.

Finally, we found a small number of all rounders (10.3%). This is consistent with other findings but, still larger than other communities such as KiN (5.4%), a social, Q&A site in South Korea [36]. This may be because most users are either "questioners" or "answerers" with a small group of users who provide both.

3.5 Contribution Distribution

The law of the vital few states that 80% of the effects arise from 20% of the causes [21,36]. In online communities, this equates to a small minority of users providing the majority of a site's content. Previous work involving Yahoo! Answers found a power law distribution in the contributions across community members and found that the majority of the community had posted only one question, while a core group of high contribution users provided most of the answers [41].

Previous work on Stack Overflow by Mamykina et al. [32] found a similar power law distribution between high and low contribution users. In their study, the authors separated users into groups by their monthly contribution patterns. In their work, if a user contributed more than 20 answers during a given month they were considered as "high activity" users. Otherwise, the user was considered a "low activity" user.

Since Stack Overflow has nearly doubled in size since this previous work, we reanalyze the community to observe any changes in that pattern that might have taken place. We use the same threshold to divide high (>20 answers per month) and low (<20 answers per month) activity users [32]. We take the contribution amounts

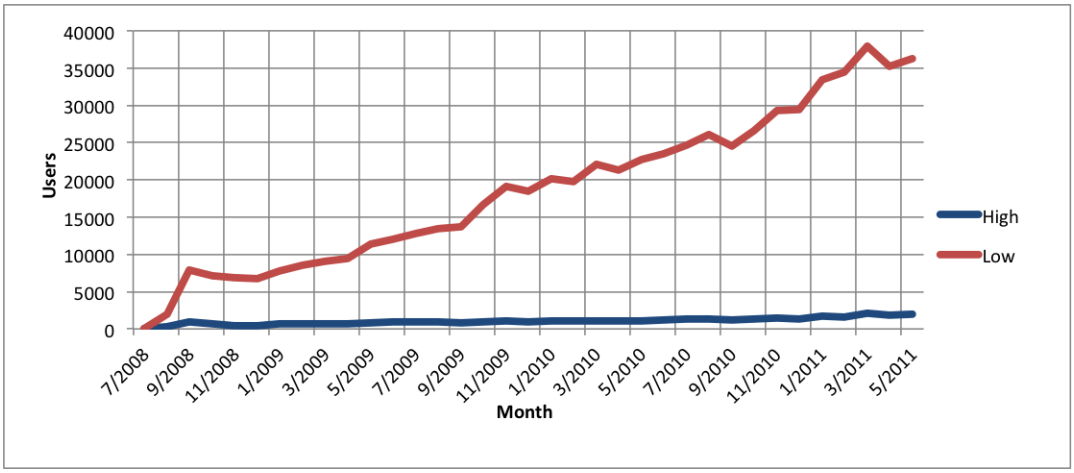

Figure 3.4: Number of high and low contribution users by month.

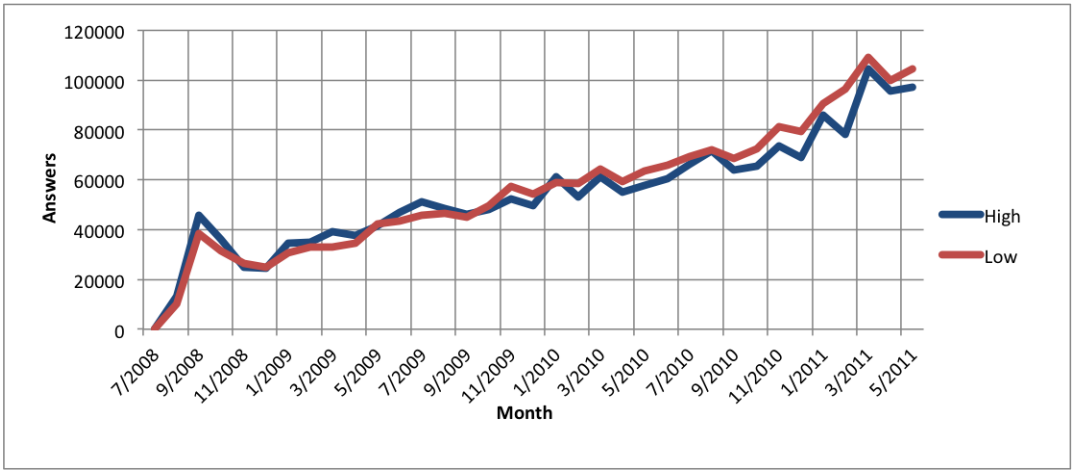

Figure 3.5: Number of answers provided by high and low contribution users by month.

for each user during each month over the history of Stack Overflow. Since we are looking at each month individually, it is possible for a user to be classified as high activity in one month but low activity in another month. Users that did not provide any answers during a given month were excluded from this analysis.

When we plot these users over time, Figure 3.4, shows that the number of low-contribution users has grown at a much higher rate than the number of high contribution users. At the same time, the percentage of the users on the site that are low contribution has stayed relatively constant at about 5.4% (see Table A.1). The

relative amount of answers provided by each group has also remained similar. In total, 50.8% of the answers on Stack Overflow has been provided by low contribution users (see Table A.1) and the distribution of contributions between the high and low contribution groups has remained similar (see Figure 3.5). This is a markedly different trend than what is found in other online communities [21, 36], as well as from the previous work on Stack Overflow [32]. This shows that the low contribution users of Stack Overflow play an important role in providing answers to the community and should not be discounted due to their infrequent contributions.

While overall answer output may be a valid method for measuring community performance, it does not take into account the quality of the contribution being provided. For this reason, we perform similar analyses as before but with only accepted answers. Note, here we assume that acceptance of a post is a proxy of its quality. To do this, we first labeled each user as either high or low based on all of the answers they provided during a given month. Users without any accepted answers were then removed. We then charted the number of users providing accepted answers as well as the amount of accepted answers that they have provided for each month.

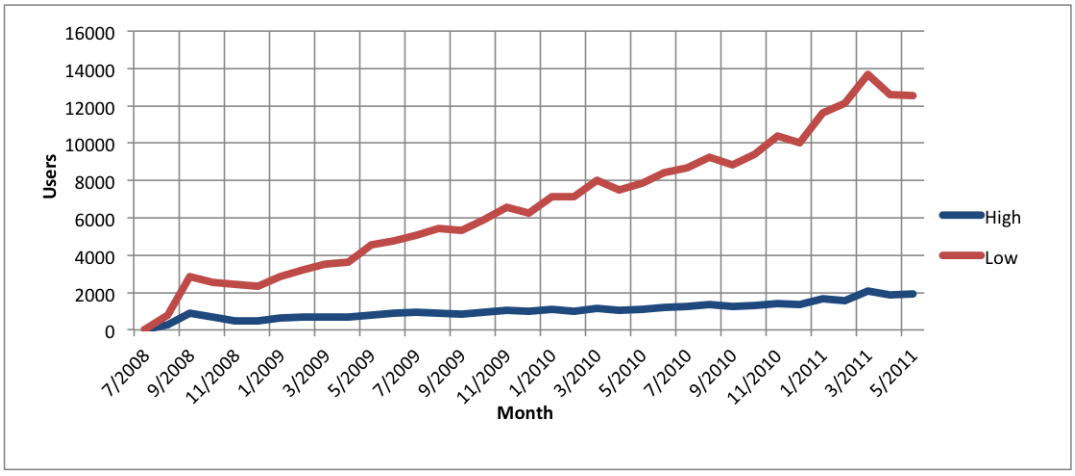

Figure 3.6: Number of high and low contribution users providing accepted answers by month.

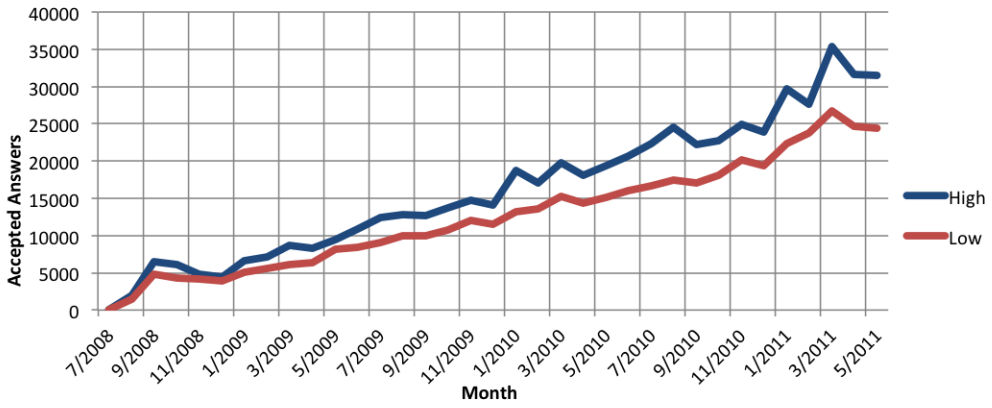

Figure 3.7: Number of accepted answers provided by high and low contribution users by month.

Figure 3.6 shows the number of high and low contribution users over time. Here we can see that there are still many more low contribution users providing accepted answers than high contribution users but the difference is not as divergent as when viewing all answers. From the figure we can see that many of the low contribution users have not provided accepted answers; their numbers in this case are drastically reduced, while the high contribution users numbers remain relatively unaffected (differing by only 14 users in the final month of analysis).

Figure 3.7 shows the number of accepted answers provided by high and low contribution users by month. Similar to what was found previously, the number of accepted answers provided by each group is growing at a similar rate. For accepted answers though, the high contribution users have consistently provided more accepted answers each month.

We see that for both analysis, "all answers" and "accepted answers only", the number of low contribution users has grown at a much larger rate than the number of high contribution users, but the relative percentages of these groups has remained stable over time. This stable distribution also extends to the content provided by

each group as the number of answers and accepted answers provided by each group has stayed proportional.

In summary, our analysis shows that almost 95% of users can be characterized as low-activity users and that they have provided about 50.84% of the answers on Stack Overflow. Even though this shows that half the site's answers come from the site's top 5%, this is still contrary to the law of the vital few. This also contradicts earlier work by Mamykina et al. [32] who found that 94.4% of users are low activity and they provided only 34.4% of the site's answers. Extending this analysis for only accepted answers shows a similar trend where 86.5% of users providing accepted answers have provided 43.7% of the site's accepted answers. We note that, even though the percentages are not as close as when viewing all answers, the contribution of accepted answers is fairly evenly distributed between high and low contribution users. This indicates that Stack Overflow has been able to harness a large group of low activity users who are helping Stack Overflow with its content. This fact could be because answers may require knowledge of very niche topics where a broad user base is required.

3.6 Tenured Users

In a community, as people stay longer they gain knowledge and experience; they become tenured in the community. These users often follow a different role from the communities less experienced users (e.g. moderating, maintenance, etc.). To see if such a phenomenon occurs in Question and Answer forums, we collected a sample of the communities tenured members and conducted interviews.

Instead of using only time since a user has joined Stack Overflow to denote tenure, which would be inaccurate if a user had created an account without ever contributing,

we instead opt for the site's built-in reputation mechanism. Stack Overflow keeps a record of experience in the community by awarding reputation to users who provide useful content. By using this measure, we are using the site's built-in mechanism for experience as a proxy for tenure.

From the interviews performed on high reputation users, we find that some tenured users chose to answer different questions than low tenure users. For example, Interviewee A spoke about a decline in the number of questions they chose to answer as they gained reputation. They said that instead they would seek out difficult or extremely technical questions. This is different than newer users who are "playing" Stack Overflow for the game aspect of the site. For example, Interviewee B commented that after they joined the site, they answered as many easy questions as they could since their main goal was to gain reputation. Answers to these easy questions garnered them the most reputation for the amount of time they could commit to the site.

Interviewee C spoke of the desire to maintain the quality of the site's content so that it could be used by others to produce "better code" which would be easier to maintain. This user was dedicated to editing questions and answers so that their content could be easier to access for more users. He also noted that closing and deleting posts was crucial in keeping up the quality of Stack Overflow content. If a question is subjective in nature or asks for content not applicable to Stack Overflow, users with moderator abilities should vote to delete or transfer the question. Keeping questions factual and objective is an important goal of Stack Overflow which seeks to provide an objective resource to developers in need of assistance.

Interviewee A, mentioned an increase in their involvement over time. To address some of the early issues in Stack Overflow's design, Stack Overflow users began using the site itself to discuss design decisions. These meta discussions gave rise to a separate discussion board, Meta Stack Overflow, the goal of which was to focus on

Stack Overflow related questions. The user interviewed mentioned their extensive contribution to Meta Stack Overflow and their involvement in discussions revolving around Stack Overflow operations. This higher-level involvement shows an investment in the community that goes beyond simply contributing. Community discussions help construct a better Stack Overflow for individuals to use.

The theme that emerges between these high reputation users is an increased involvement in tasks that will further the Stack Overflow community as a whole. By working to answer the site's more difficult questions, moderating site content and involvement in the future design of Stack Overflow, these users are invested in the success of the community as a whole.

3.7 Summary

In this chapter we have analysed the Stack Overflow community by observing the progression that users make when joining the community, identifying the main knowledge contributors and determining the role of the community's tenured users.

First, we find that the largest group of users are stalled (253K). Of these stalled users, a large portion have attempted contributing multiple times (190K). This shows that a large number of users are attempting to contribute but are lacking the knowledge to do so effectively. On the other hand, this does show a large number of users with interest in contributing that can be leveraged by educating these users to provide more effective answers. We believe that mentoring or shepherding can help Stack Overflow.

The next largest group of users are registered lurkers. The percentage of registered lurkers is in line with other Q&A forums that track user accounts that have no activity. This also shows a large potential of users, especially users who have made multiple

contributions but are stalled users. These users may be interested in the community and are either learning about the community before contributing or have not yet been able to contribute.

The smallest user groups are partial success users (17K) and all rounder users (66K). The low number of partial success users when compared to the number of limited users (101K) shows that users who have attempted both are largely successful as many have become all rounders.

Next, we find that the low contribution users contribute almost half of the site's answers and over 40% of the site's accepted answers. This shows that a considerable amount of the site's content is provided by low contribution users, a finding which is atypical of other online Q&A forums.

Last, we find that tenured users help manage the community. We find that high reputation users may choose to answer fewer, more difficult questions instead of many easier questions. We also find that these users help moderate the community by deleting off-topic posts and editing posts to be useful to a majority of the site's visitors. We have also found that some tenured users take part in discussions on Meta Stack Overflow about site operations and potential bugs. These discussions on Meta Stack Overflow have affected design decisions and changes that improve the Stack Overflow experience.

Chapter 4

GNOME

The second community we investigate is the Gnome project which provides desktop environment to its users including applications ranging from system utilities, email clients to media players. Started in 1997, Gnome is now used by millions of users around the world and has had code contributions by over 3500 users. Gnome has become a staple of the open source community.

We choose to focus on the Gnome community because of its success in providing a full desktop experience for users around the globe. We first consider the path users take when getting involved with the Gnome community and its projects. We then investigate the distribution of work between the members of Gnome projects. Last, we examine the impact of tenure on the contribution provided by developers.

4.1 Community Description

Gnome is an open source, Linux desktop environment started by Miguel de Icaza and Federico Mena in 1997. Control of the community was handed to the non-profit organization, the Gnome Foundation upon its establishment in 2000. This foundation is responsible for the management, operations and future directions of the

community.

GNOME is composed of many smaller projects that serve specialized functions within the environment. These functions include very technical libraries that facilitate other applications, everyday applications such as an email client or file browser, and entertainment providers such as games and video players. A central governing body, the GNOME Foundation, dictates the release schedule. This means that each of the projects in the ecosystem may operate independently from the GNOME community, but they will adhere to a community release cycle and branding. When the GNOME project releases an update, individual projects contribute their current release to the larger branded release.

The GNOME community therefore encompasses a number of entities, which not only include the established projects, but also the infrastructure, the people and their contributions. These components can be classified by the following descriptions:

- **Projects** are the organizational units that GNOME uses. Projects are formed around functionality within the ecosystem ranging from low-level system components, such as system libraries to user-focused projects such as email clients.

- **Mailing lists** are the primary public discussion medium for GNOME projects. On a mailing list, individuals can exchange emails with other individuals about a project's design or functionality. These communications may range from help inquiries to in-depth design discussions.

- **Bug trackers** are central repositories for managing problems (bugs, issues) that may arise in GNOME projects as well as for recording user requested features. Using these systems, individuals in a project can track the project's progress. GNOME uses a bug tracker named Bugzilla.

- **SCM repositories** are used to track code in the GNOME ecosystem. Project

members can track who and when others are contributing code to individual projects in the community and at what time. GNOME uses the git version management system for tracking code contributions.

- **Users** are individuals who contribute to the community projects. Users can be identified by a variety of tokens such as an email address (on the mailing list), a user account (on the bug tracker) or commit information (on the SCM repository). The sum of all these tokens constitutes a user's identity in the GNOME ecosystem.

4.2 Research Questions

Previous studies of open source software have largely investigated communities consisting of a single, large, monolithic product (e.g. Apache, Mozilla). These studies found that the new community members became involved by starting with less central activities like community mailing lists and gradually working their way toward more central activities such as code contributions in a model know as the onion model [11, 26, 52].

The overall goal is to investigate whether the same process also holds true for GNOME, which is an amalgamation of a set of disparate projects. With this in mind, we ask three specific research questions.

RQ1: What is the process that users follow to join the GNOME *ecosystem?*

In this question we seek to understand how users get involved in the GNOME community and its projects. Using activity traces (e.g. mail messages, code commits) created by community members who have contributed code to GNOME projects, we trace the path that individuals take when becoming a member of a project. We observe the order in which users first interacted with the systems in individual projects

and the community as a whole. We can then compare these observations with previously studied monolithic communities to study the effect of this organization.

RQ2: Who are the main contributors in a Gnome *project?*

The next question studies the varying contribution amounts between users in Gnome projects. Work within a project is not evenly divided between contributors. Previously studied communities have shown a skewed relationship between the highest and lowest volume contributors. Here we seek to evaluate the distribution of contributions among the developers in Gnome projects.

RQ3: What is the impact of tenure on the contributions provided by developers?

The final question asks what role senior members play in Gnome projects. Experienced users in a project will typically assimilate large amounts of knowledge about the project and its operation. These users may use this knowledge to steer the direction of the project and provide assistance to newer users within the system. This question seeks to understand what role these tenured users play and how it differs from the role of other users in a project.

The analysis conducted for RQ1 and RQ2 appeared in FSE 2011 [27].

4.3 Data Collection

The data that was collected for the Gnome ecosystem contains three main sources of archival data: mailing lists (for project discussion), Bugzilla (for bug management) and CVS Repositories (for source code versioning and storage). Note that when the data was collected, Gnome was using the CVS version control system and has since transitioned to the git version control system. Data from each of these sources was collected for each of the projects in the Gnome community.

Collection of the data used in this study was completed in 2007 by Patrick

Wagstrom and is available online [48]. Data was collected for each of the main sources of archival data spanning 11 releases over 10 years from 1997 to 2007. The collected data includes references for over 2.5 million code changes by over 1,000 developers to more than 1,200 projects and 790,000 comments on 250,000 bugs reported by 26,000 different users.

Once the data had been collected, it was loaded into a single, large database. Post-processing was performed to link entries to projects and users together across data sources. While a large part of this process was automated (matching performed by comparing email addresses and provided user names across data sources), about 10% of the user accounts were matched manually by consulting members of the GNOME community. A full description of data used can be found in Appendix B.1.

Threats to validity associated with this data and our subsequent analysis can be found in Chapter 5.

4.3.1 Subset of Projects

In this work, a sample of six GNOME projects were highlighted for analysis. These projects were selected because of their long history within the ecosystem and their availability of archival artifacts. These projects are also well-known in the GNOME community and share many users that work on multiple projects. The following is a description of the projects selected.

- Eye of GNOME: An end-user application for viewing images and lightweight graphics manipulation.
- Epiphany: A web browser that is customized to integrate into the GNOME desktop environment.
- GConf: A library and several tools for applications to manage settings in a

standard and unified method. Most end user applications in the ecosystem rely on this library as a critical piece of infrastructure. All projects in the subset we examine utilize this library as a key component.

- GnomeUtils: A collection of utilities for developers and end-users alike to make the most out of their desktop experience.

- GnomeVFS: A system level library for the transparent manipulation of files and other file-like resources on the local machine and across network connections. The use of this library is not required by all end-user applications; however, all applications in this subset utilize this library.

- Gnumeric: An extensible end-user spreadsheet application.

4.3.2 Project Metrics

The following is the set of metrics that were collected for developers in the GNOME community. These metrics were collected in each of the subset projects for each GNOME release.

Code Commits: Total number of contributions to the CVS repositories. We further distinguish between source code, documentation and translation commits since each requires a different set of skills.

- Source Code Commits: Total number of commits with files containing project source code. Source code files were identified by matching the file extensions to those known language extensions (i.e. ".c" for C language files and ".py" for python language files.)

- Documentation Commits: Total number of commits with files containing project documentation. Documentation files were identified by matching the file exten-

sions to those of know documentation formats (i.e. ".txt" for text documentation.)

- Translation Commits: Total number of commits with files containing project translations. Translation files are identified by the file extension ".po".

Mail Messages: Contributions provided to project mailing lists. We divide these contributions into message count, message started and message responses because of the function of the type activity these contributions represent. For example, the start of a message thread may represent a user question while message responses may be discussion about a particular part of the project.

- Message Count: Total number of mail messages posted to a mailing list.
- Message Started: Number of mail message threads started.
- Message Responses: Number of mail messages posted in response to other mail messages.

Bug tracker: Total number of contributions to project bug trackers. We further distinguish between bug tracker activity and bug tracker comments because each type of contribution denotes a different skill set used on the bug tracker.

- Bug Tracker Activity: Total number of bugs filed, comments posted and actions posted on the project bug tracker.
- Bug Tracker Comments: Total number of comments to a project bug tracker.

Experience: Total amount of experience in the GNOME community. We divide experience into project specific experience (project experience, project active experience, prior experience) and community experience (total experience and total active experience).

- Project Experience: Number of releases since a developer's first contribution to a project.

- Project Active Experience: Total number of releases during which a developer has provided contribution.

- Prior Experience: Number of releases a developer was active in other projects in the GNOME ecosystem before joining a project, counted per project that they were active on. For example, if a user became active during a release on two projects before joining a third project during the next release, their prior experience would be counted once for each project they were active on which in this case is two.

- Total Experience: The number of releases since a developer's first contribution to the GNOME community.

- Total Active Experience: Number of releases a user has been active on any project in the GNOME community. Users who are new to the ecosystem will have equal total experience and total active experience.

Sourcecode Centrality: The measure of how central or important contributions are to a project. This metric is called Source Code centrality and has many steps in its calculation.

To determine the centrality of code, we first construct a network from the projects' source code. There are many ways to do this including call graphs, packages like in Java or the concept of logical commits [19]. In this work, we employ logical commits to construct a network of the projects structure, that is we identified files/program that are related to each other logically because they were committed at the same time. We chose to use logical commits because it does not depend on the structure

and dependencies of the code and thus will work across varying types of files including files of different programming languages, configuration files and build files, while other program analysis methods work with files that are of the same programming language.

Construction of the network with logical commits involves creating a graph node for each file in a software project and adding or increasing the weight of an edge each time two files are committed together. Doing this will not only emphasize how often a file is committed, but will highlight the importance of the other files with which a file was committed.

Once a network is created, we can apply metrics to the network to determine how central or important a file is within the network. There are many approaches to determining the centrality of nodes in a graph. In this work we apply eigenvector centrality because it avoids issues with network structure while maintaining a consistent implementation [9, 28].

Mathematically, eigenvector centrality is the first eigenvector of the adjacency matrix representation of a network. In general, this implies that nodes in the network that are connected with high weight will be scored higher or considered more central. In the context of this work, eigenvector centrality will highlight important files from a project's source code by identifying files that are often committed alongside other "important" files. We can then use this measure to infer the importance of a commit by taking the average centrality of the files it contains. Finally, we can aggregate project commits for a developer to calculate the average centrality of the commits that they provided.

4.4 User Joining

In this section, we investigate how users become involved with the GNOME community and its individual projects. We begin by observing the order in which users who have contributed code to a project arrived at that position. We select only users who have contributed code in our dataset since they can be considered model community members because they have successfully become a code contributing member of a project. We then identify predictors from the community and study their impact on code provided by developers.

4.4.1 Progression Paths

We model the stages of involvement with a project by investigating the three main sources of data for each project: mailing list, bug tracker and code contributions. These stages correspond to stages of involvement previously observed in other open source communities [47].

Each of these sources has a different barrier for entry and a user is not required to participate in any medium before another. For example, consider the mailing list of a project, a user often only needs to send an email to the list address to participate (permission may be required), whereas the bug tracker system requires creation of an account. The code repositories on the other hand require commit access granted to the individual before they can contribute code.

To identify the order in which a user becomes involved, we identify the first contribution a user makes in each source of data, if it exists. We then determine during which GNOME release each of the contributions occurred. The order in which a user becomes involved with a project can then be determined using GNOME releases.

We next take each of the progressions and group them into five categories based

on the order of the types of contributions provided. We identify mailing list and bug tracker contribution to be more socially oriented while code contribution is very technically oriented. Based on this, we can define paths for users that shift from one type of contribution to another. These paths are as follows:

- **Social-technical path:** This path includes users who start in a social medium (mailing list) and later became involved in a technical medium (bug tracking or code contribution) over a series of releases. The key criterion for this path is that only one new type of contribution was provided during any given release. This path is typical of what has been identified in previous work on open source project joining [35].

- **Accelerated path:** This path includes users who start in a social medium (mailing list) and later become involved in a technical medium (bug tracking or code contribution) much like the social-technical, path but at an accelerated rate. These users become involved with multiple mediums (mailing list, bug tracker, SCM system) during the span of a single GNOME release. For example, a user may be active on the mailing list during a single release and become active on both the bug tracker and contribute code during the next release. Note that this pattern might be occurring because the user passes through the stages in a time period shorter than the release period (about six months).

- **Technical-social path:** This path includes users who have joined a project in an order opposite to what is found in previous studies of other communities [35]. Users in this path have started in a technical medium (bug tracking or code) and later become active in a social medium (mailing list).

- **Technical only path:** This path includes users who have participated only in technical mediums (bug tracker or code contribution).

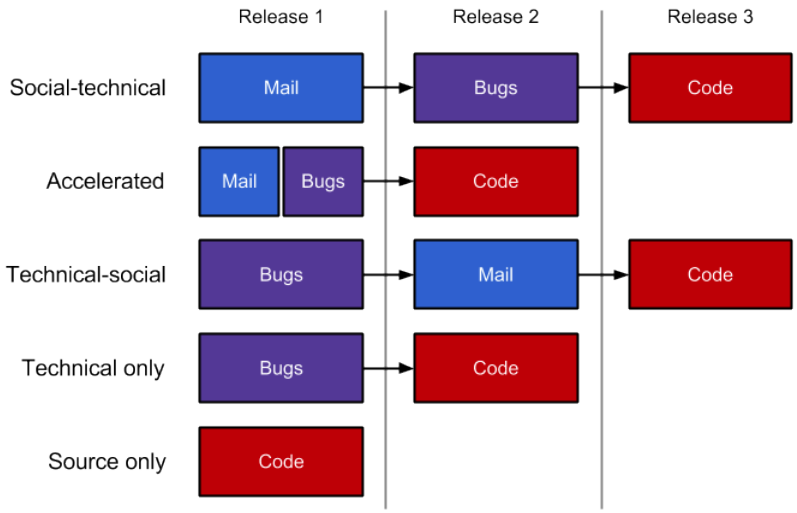

Figure 4.1: Example progression paths that can be taken by GNOME developers.

- **Source only:** We further divide the technical only path to highlight users who have only provided code contributions. This path also directly contradicts the onion model as these are users that are providing technical contributions without prior socialization which is intended to help them contribute.

Table 4.1 provides an overview of the number of users who have followed each type of progression path at the project level.

We see that in Table 4.1, the traditional social-tech path is the smallest progression path (1.82%). This is in direct contrast to previous studies which found users progressed from social to technical mediums as part of a socialization process inside the project [35]. Even when including users who have followed the accelerated path, the number of user progressions following previously identified patterns is still less that 10%. As noted earlier, a reason for the accelerated path could be that our window of analysis (GNOME release) was large enough that we do not see the transition between roles.

We next highlight the presence of users who have gone opposite of the traditional model, users who have started participating on a project by first becoming involved in

Table 4.1: Progression across social and technical mediums.

Category	Individual projects		Ecosystem subset	
	members	percent	members	Percent
Social-tech	24	1.82%	25	5.45%
Accelerated	100	7.58%	82	17.86%
Tech-social	118	8.95%	103	22.44%
Technical	224	16.98%	74	16.12%
Source only	853	64.67%	175	38.13%

a technical medium before contributing in the social medium. These user progressions account for 8.95% of the total progressions made by users. This is in direct contrast to previous findings and it shows that users inside the GNOME community may not require socialization in an individual project.

The analysis also shows that the majority of user progressions do not involve a social component (81.65%), but rather are technical (16.98%) or source only (64.69%). This shows that very few users that have contributed to a technical medium for a project have participated in a social medium. This indicates that socialization is either not required at the project level or that socialization occurs elsewhere.

To determine if socialization was occurring at the ecosystem level instead of at the level of individual projects, we conduct the same analysis at the ecosystem level which in this case is our subset of projects. We do this by expanding our analysis to contributions across these projects in our project subset (see Section 4.3.1).

Similar to our earlier investigations, we identify the developers for each of the projects in the subset. Then instead of determining the progression path of an idividual with an individual project, the first contribution by the individual to any project in the subset is used to construct an ecosystem progression path. Table 4.1 contains the results of this analysis.

Immediately noticeable in Table 4.1 is the increase in the percentage of social-technical paths, an increase of 4.63% increasing the percentage to 5.45%, and accel-

erated paths, an increase of 10.38% to 17.86%. When combined, these paths account for 23.31% of the total paths. Also noticeable is the large drop in source only users, a decrease of 28.54% to 38.13%. This shift may indicate that socialization still proceeds from social to technical mediums, but is occurring as users join the ecosystem instead of when they join an individual project.

4.4.2 Broad Factors that Affect Code Centrality

In this section we use statistical analysis to determine the types of contributions and activities of developers that impact the amount of central code they provide to a project. We take the types of contributions provided by users as well as their experience in the community and predict the average centrality of the code they provide. We use code centrality in this case because it indicates how "central" or important the code provided by a developer is to a project. This will give us insight into the types of activities performed by the communities most central code providers as well as other insights into the roles of other community members.

Because of the high correlation between our predictor variables, we use Principle Component Analysis to study the relationships among the predictor variables. PCA is useful in cases where predictor variables are correlated and standard linear regression is not applicable. We perform PCA using each of the predictors identified. Four components were identified with a standard deviation above one. These components are presented in Table 4.2.

An explanation of the components is as follows:

- Component 1 (general contribution inverse): This component is the inverse of project contribution which equates to the total amount of participation a developer has in a project. Developers who score very high on this component have contributed very little to the project.

Table 4.2: Major Component Loadings from Principle Component Analysis.

Component	1	2	3	4
Std Dev	2.3286	1.8840	1.1644	1.0550
Source Commits	-0.371			0.254
Doc Commits	-0.386			0.275
Trans Commits	-0.145		-0.215	0.492
Message Count	-0.388			-0.159
Message Started	-0.343		0.194	-0.442
Message Responses	-0.341		0.187	-0.445
Bug Tracker Activity	-0.367			0.232
Bug Tracker Comments	-0.388			
Prior Experience		-0.146	0.773	0.311
Total Experience		-0.502	0.229	
Total Active Experience		-0.476	-0.300	-0.139
Project Experience		-0.509	0.147	
Project Active Experience		-0.467	-0.310	-0.126

- Component 2 (project experience inverse): This component is the inverse of the time spent in a project. Developers scoring high on this component are new to the project.

- Component 3 (broad social experience): This component reflects broad social experience. Developers who score highly with this component have broad experience in the GNOME community and may transition in and out of projects often. Developers who score highly with this component are especially active on the project mailing list, starting and responding to others' messages.

- Component 4 (technical/translation): This component reflects technical medium expertise. Developers with a high score for this component have been active with both code commits and bug tracker activity, but have not been active on the project mailing list. This component also highlights prior experience with the community and has a larger loading for translation commits and a negative loading for total active experience. One possible explanation for this

Table 4.3: PCA regression model.

Variable	Estimate	Significance
Component 1 (General contribution inverse)	-1.0341	<0.0001
Component 2 (Project experience inverse)	0.1848	<0.0001
Component 3 (Broad social experience)	-0.1230	0.00001
Component 4 (Technical/translation)	0.6600	<0.0001
Adj R-squared: 0.6268, F: 514.1 on 10 and 3045 DF, p <0.0001		

is a community translator who transitions between projects as translations are needed.

We now create a regression model to evaluate the relationship between the components and Source Code Centrality. The results of this model are presented in Table 4.3.

If we look at the estimate for the first component (general contribution inverse) in Table 4.3 (-1.0341), we can see that developers who have more contributions have a tendency to provide code that is more central. The negative estimate value indicates that the inverse of the first component is indicative of more central code. A user scoring highly according for Component 1 has provided very little contribution to a project while the inverse of this, as implied by the negative estimate, implies that users who provide more contribution provide more central code.

Component 3 (broad social experience) indicates that users with more social contributions have lower central code contributions. This could be because of the distribution of work between developers in a project. Users who spend their time contributing socially may not have time to contribute to technical mediums. Also, this could indicate that different types of work are divided amongst developers in a project with some users focusing on social mediums while other concentrate on technical contributions.

Component 4 (technical/translation) hints that users with prior experience in

the community may have reduced joining times in other projects. The loading for translation commits appears to indicate that this is at least true for community translators.

4.5 Contribution Distribution

In this section, we analyze the distribution of contribution of developers in the GNOME community. Previous work in other communities [21, 25, 36] has shown a very skewed distribution between the communities' highest and lowest contributing members. In this section we investigate the distribution of contribution amongst the developers in the GNOME community.

To analyze the distribution of contribution in the GNOME ecosystem, we divide the contributors from each of the projects in the selected subset of projects. From this selection of users, we divide the users into high and low contribution users and determine the amount of content that each group has provided.

We begin the analysis by selecting users from each of the projects in the subset who have provided source code. The definition of source code in this context is only files that contain code to be executed as the program runs. This includes files in a variety of programming languages (C, C++, python, etc.) and build files (Makefile, etc.), but excludes other supporting files like documentation and translations. We select only source as the contribution medium for this analysis because each project may have different conventions about how documentation and translations are stored (e.g. documentation stored outside of version control).

We first identify the source code developers for each of the subset projects and aggregate the amount of contribution that each developer provides. That is, for each developer, we count the total number of commits to source code.

Table 4.4: Summary of the amount and relative percentage of high and low contribution users and the contribution they have provided to projects.

Project	High Contribution				Low Contribution			
	Users		Contribution		Users		Contribution	
Eye of Gnome	20	19.61%	1804	89.26%	82	80.39%	217	10.74%
Epiphany	17	20.00%	6812	97.37%	68	80.00%	184	2.63%
GConf	29	19.59%	1964	83.68%	119	80.41%	383	16.32%
GnomeUtils	32	19.63%	5608	89.56%	131	80.37%	654	10.44%
GnomeVFS	35	20.00%	4719	86.79%	140	80.00%	718	13.21%
Gnumeric	24	19.35%	18711	96.76%	100	80.65%	627	3.24%
Subset	73	19.89%	39075	92.16%	294	80.11%	3326	7.84%

We next divide each of the projects' users into two contribution categories by selecting the top 20% of contributors as high contribution and consider the remaining 80% of users as low contribution users. The 20% distinction was determined by using the Pareto principle which has been used similarly in other studies of online communities [32].

Then, we determine the total amount of contribution provided by each the high-contribution and low-contribution users by simply totaling the number of commits amongst the developers in each group. Table 4.4 provides a summary of this analysis.

From the analysis we can see that the amount of contribution provided by each group is highly skewed. We see that the top users provide the majority of the content while the large majority of users provide very little. For the projects in the subset, on average 90.57% of the contributions were by high-contribution users and the most skewed project was Epiphany where 97.37% of the contribution was provided by high-contribution users.

By taking the source code commits from each of the project from the subset and taking the top 20% of this grouping, we can see that 92.16% of the commits in total have come from the highest contributing users.

Therefore we find that the difference in the amount of contribution provided by

the high and low contribution users is very skewed. These findings are in line with previous studies of other communities [21, 25, 36]. The distribution of contribution between the differing contribution groups matches the Pareto principle. This indicates that within the core of the onion model, there is a further distinction high contribution and low contribution members.

4.6 Tenured Users

In this section we examine the impact that tenure with the GNOME community and individual GNOME projects has on the centrality of the code that a developer provides. Using various statistical methods, we examine tenure in a variety of ways, including tenure with a single project and previous experience with the community.

4.6.1 Predicting Source Code Centrality

We first define the value, Source Code Centrality, to be predicted by the factors in the regression models. We define source code centrality as the average centrality of code commits provided by a developer. For example, if a developer provides contributions to more central files in a project, their average Source Code Centrality would be high than a developer that contributes to less central files in a project.

We next identify factors to serve as predictors of source code centrality. Presented here are the basic predictors that will be used to predict source code centrality which include Source Code Commits, Mail Count, Tracker Activity, Project Experience and Project Active Experience. For a description of these variable, please see Section 4.3.2.

Before a regression model can be constructed, the predictor variables must be checked for independence. The assumption when building a regression model is that each of the predictors are not correlated and thus are independent. If variables are

Table 4.5: Base regression model.

Variable	Estimate	Significance
Mail Count	0.0675	<0.0001
Tracker Activity	0.0100	<0.0001
Project Experience	-0.0522	0.0253
Adj R-squared: 0.4777, F: 311.4 on 9 and 3046 DF, p <0.0001		

found to be correlated (dependent), they must be removed before regression analysis can be performed. We found three variables that had sufficient independence for use as control variables: Mail Count, Tracker Activity and Project Experience.

Source Code Centrality is heavily dependent on Source Code Commits and has a very high correlation and therefore both cannot be used in the regression model. In addition, Project Active Experience can not be used because of its high correlation, about 0.96, with Project Experience. In this work, we choose Project Experience over Project Active Experience because even if a developer had not been active with a project in a given a period of time, they may still be using or learning about the project in some manner. For example, a developer in a project may be inactive for a release on a project but still use the project and follow its development.

To account for the variance that exist between the projects used in the model, we create a regression model with a separate intercept for each project. The output of this regression model can be seen in Table 4.5.

The model shows that both social and technical activities, Mail Count and Tracker Activity respectively, increase the centrality of the code provided by the developer. These activities indicate that, as a user is more active on a project, their code is more central to the project.

The negative sign on the estimate for Project Experience indicates that the longer a user is with a project the less central their code contributions are going to be. We note that this does not imply that the user is not important to the project, but they

Table 4.6: Enhanced regression model.

Variable	Estimate	Significance
Mail Count	0.0653	<0.0001
Tracker Activity	0.0099	<0.0001
New Developer	0.5917	<0.0001
Normal Developer	0.5489	<0.0001
Experienced Developer	0.2327	0.2315
Adj R-squared: 0.4773, F: 279.9 on 10 and 3045 DF, p <0.0001		

may be occupying a different, non-coding project role. The tenured users in this case may be fulfilling a different role than core code contributer (i.e. project management or bug triage).

4.6.2 Project Tenure and Source Code Centrality

Given that Project Tenure had a negative effect on centrality of contribution, we further investigate tenure to gain additional insight into the development patterns of tenured users in a project. We expand our regression model by dividing Project Experience into three subcategories: New Developer (first release with a project), Normal Developer (second through fifth release with a project) and Experienced Developer (sixth or more release with project). The results of this regression model are contained in Table 4.6.

We see again that the centrality of tenured users' commits declines over time, lower estimate (0.2327) for Experienced Developer than the New Developer and Normal Developer (0.5917 and 0.5489). The Experienced Developer estimate is also not significant (0.2315) and thus has little predictive power in this model. We also see a decrease of the estimate between New Developers and Normal Developers but to a much lesser degree (0.5917 to 0.5489).

This model suggests that tenured developers become less involved with project code that is central, but the function of Experienced Developers is still unclear. To

understand why experienced users might be making less central code commits, we investigated the contribution pattern of a set of users. In a random selection of six experienced developers: three remained central in their contributions, two made fewer contributions because they had assumed a broader leadership role in the community (e.g. release manager and foundation board), and the sixth shifted away from source code contribution and contributed by focusing on bugs and evaluation of new feature requests to the project. This indicates that these experienced developers have used their knowledge of the GNOME community and its projects to help the community as a whole function.

4.6.3 Ecosystem Tenure and Source Code Centrality

We now expand the model to take into account tenure with larger GNOME community (other projects in the GNOME ecosystem). We add the variables Prior Experience, Total Experience and Total Active Experience to the existing model from Section 4.6.1

We note that, in general, developers were active without any major breaks in their participation causing the correlation between Total Experience and Total Active Experience to be very high (0.97). Therefore these factors could not be used together in the same regression model. The correlation between Total Experience and Project Experience was 0.78, therefore they also could not be used together in a regression model. We select Prior Experience as the more relevant additional metric for depicting experience in the ecosystem.

Building on the previously developed models (see Section 4.6.1), we add Prior Experience. The results of this regression model are shown in Table 4.7. From this model we can see that Prior Experience is not significant. Therefore we can conclude that the prior experience of a user with other project does not play a major role in determining the centrality of a developer's code contributions.

Table 4.7: Experience regression model.

Variable	Estimate	Significance
Mail Count	0.0675	<0.0001
Activity Count	0.0100	<0.0001
Project Experience	-0.0533	0.0232
Prior Experience	-0.0138	0.6809
Adj R-squared: 0.4775, F: 280.2 on 10 and 3045 DF, p <0.0001		

4.7 Summary

We have analyzed (1) the progression that developers take when joining a project individually as well as in the context of the ecosystem, (2) the distribution of contributions in a sample of projects and (3) the impact of tenure on the centrality of code provided by developers in the Gnome ecosystem.

First, we find that many users do not follow the traditional socialization path in individual projects. Few developers in Gnome projects have followed a traditional path (1.82%) and most developers have contributed only through technical mediums (81.65%) with 64.67% of users contributing only source code. This indicates that socialization is either occurring in a different manner or at the ecosystem level.

When progression paths were observed at the ecosystem level, more developers were found to be following the more traditional and accelerated progression paths. This indicates that there is socialization occurring at the ecosystem level which may reduce the time required for developers to join additional projects in the same ecosystem.

Exploring the distribution of code contributions to projects in the ecosystem revealed a very skewed pattern of contribution, where a small minority of developers provide the majority of source code commits. We observe that, on average, 92.16% of commits were provided by high contribution developers. This is in line with the Pareto principle and has been observed in other online communities [21, 25, 36].

Last, statistical models indicate that a developer provides less central code as they gain tenure with a project. The general indication is that tenured members provide less central code to a project than newer members. When project tenure was categorized into three groups, the trend again was for newer users to provide more central code. In a sample of tenured users we found that it is not clear what the exact role in project most tenured users perform. We have an indication that experienced users may assume other roles in a project (i.e. bug management) or may provide contributions to the larger ecosystem (i.e. release management or foundation board).

Chapter 5

Threats to Validity

As our findings are in contrast to many previous studies of individual Open Source projects and Q&A forums, we must be conscientious of the limits of our research and the possible threats to the internal, construct, and external validity of our work.

Internal validity: Our analysis is based on data extracted from public project archives mailing lists, bug trackers, and source code repositories. We have largely assumed that social interactions and decision making are conducted via mailing lists and we could have missed communication that occurred through personal email, IRC channels or face-to-face meetings during conferences and developer meetings. However, research literature suggests that Open Source socialization processes largely occur through project mailing lists, so in this respect we are in line with previous literature but we realize that this might not be sufficient to understand the communication in modern open source projects [7, 34]. Further Open Source norms and traditions encourage project discussions and communications to take place in the developer mailing lists [19]. Next, our treatment of comments in the bug tracker as a technical activity may be problematic. While most discussions are about a particular issue or defect, there are situations where users post non-technical information to the

bug tracker, such as feature requests, however these requests rarely come from experienced project developers who have commit access and are a minority of discussions.

Our analysis of Stack Overflow has two main threats to validity. First, our use of "acceptance" as a proxy for quality in the community. We assume that in general the best answer is chosen as the accepted answer and that quality questions are answered. If users do not select the best answer in response to a question then our analysis of user distribution may not be accurate. With the Stack Overflow policy of allowing only objective questions and answers, a user that consistently provides quality, objective responses to the questions of others should have their answers accepted. Second, all analysis run was validated manually. If the analysis was not validated properly this would be a major threat to the results of the study. However, in this study each set of analysis was validated by hand first on a smaller set of data before being performed on the main dataset.

In addition, the data for the communities studied are from different periods of time. The data collected for the analysis of the GNOME ecosystem ends before the Stack Overflow data begins. This discrepancy may account for some of the differences observed between the two communities. Also, the GNOME ecosystem has also transitioned to the git version tracking system since the data for this study were collected. Git encourages a distributed model of contribution from code which may encourage additional developers to contribute to GNOME projects by alleviating some of the barriers observed in this study. However, since the analysis studies the communities and not the same users in each community this should not be a factor.

Construct validity: There are three major construct validity threats to our study of GNOME. First, the use of a release as a unit of analysis for our regressions and analysis of interaction patterns, which was done to provide a consistent way of comparing across projects as all projects followed the same 6 month cycle, may be

problematic if developers move through the onion model phases in shorter time periods than the 6-month release cycles in GNOME. We accommodated this concern where feasible, for example, when calculating the progression paths in section 5.1 we optimistically considered individuals to have followed a social-technical path even when their first contributions to social and technical mediums were in the same release. The second threat to construct validity lies in the creation and use of Source Code Centrality as a proxy for core contributions. Past research has often used the raw amount of contribution (e.g., number of commits or lines of code changed) as a measure of developer activity [40, 49] or mailing list communications as a measure of centrality in a project network [5]. In our study, we needed a finer grained measure to characterize both the volume and importance of contributions, which allowed us to identify whether prior experience enabled developers to start making complicated, central changes earlier. Logical commits were used to create the network as they provide a language agnostic method to associate source code files and eigenvector centrality was used because of its general robustness in the face of various network topologies. Third, our analysis centers on code contributions in a project and, therefore, ignores contributions of other stakeholders in a project (e.g., architects, testers) and their socialization processes. However, note that past literature that has demonstrated the existence of the onion model investigated the socialization process of developers based on their code contributions and is similar to our study.

In the study of Stack Overflow, the measure of tenure was used as a proxy for tenure within the community. It is possible that very experienced users in the community have a low reputation score. In cases such as this, another measure of tenure would have been more appropriate. In our study though, we chose Stack Overflow's measure of reputation because it is the measure that is used by the community to rate users on their experience.

External validity: Finally, there is a chance that GNOME may not be a suitable sample of an Open Source ecosystem and Stack Overflow may not be a suitable sample of an Q&A forum. This limits the degree to which we can generalize our findings to other online, software ecosystems. Further, our study of GNOME considered a subset of six projects with significant history and participation as representative of the greater GNOME ecosystem. The fact that these six projects are well adopted within the community, have long histories, and significant participation may make them outliers in the broader context of the GNOME ecosystem. It is possible that other projects might demonstrate different socialization processes.

Chapter 6

Conclusions

In this work, we have investigated two different types of online software communities: Stack Overflow, a question and answer site, and GNOME, an open source desktop environment. In this section we now compare and contrast each site and draw conclusions from our work.

RQ1: How do users join online, software communities?

When we studied "user joining" in Stack Overflow, we observed that users show different levels of success when joining the community. First, a large group of stalled users were discovered (about 253K) with many having attempted contributing more than once (about 191K). The next largest group of users identified were registered lurkers (about 206K). This is an interesting observation that users who actively try to participate almost equal the number of registered lurkers. This shows that Stack Overflow has a large base of users who may be waiting to or lack the socialization to contribute.

When we observed "user joining" in the GNOME ecosystem, we find a different socialization pattern than what has been identified for users joining a new project in previously studied (monolithic), open source projects. In the subset of projects

that we studied, only 1.82% of developers followed the traditional social-technical progression path. Even when including accelerated progression paths, the percentage is still less than 10%. When we expanded the investigation to the ecosystem level (a subset of six projects from GNOME), this percentage grows to 5.45%. When including accelerated progression paths, this percentage grows to 23.31%. This is an indication that socialization may be occurring at the community level, allowing developers to move between projects at an accelerated pace by transferring their knowledge. Note that the majority of progression paths either start with or only include technical contributions.

Implications: Comparing the two communities, we can see that there are a large number of users with technical knowledge and expertise to share with others. The low barrier of entry for Stack Overflow allows these users to contribute easily by answering questions that others may have. In GNOME, once users have been socialized into the community, they often contribute directly to projects without the requirement of socialization to individual projects. The results for this research question indicate that a common infrastructure in an ecosystem indeed helps garner the contributions of more users.

RQ2: How are the contributions in online, software communities distributed amongst community members?

When we investigate the distribution of contribution between high and low contribution members in Stack Overflow, we find a distribution that is much less skewed than that which has been identified in other online communities [38, 39]. We find that about half (50.8%) of the community's answers have been provided by low contribution users. Even when considering only accepted content, we find that low contribution users have provided about 43.7% of the community's accepted answers. This shows that low contribution users play a key function in Stack Overflow and

provide a substantial portion of the site's content.

When we investigate the distribution of contribution in the GNOME community, we find the distribution to be very skewed. On average, 90.57% of project source code commits are provided by the top 20% of project contributors. When considering commits from each of the projects in our subset, 92.16% of source code commits were provided by the top 20% of contributors. This skewed distribution was found in other online communities [21, 36] and is considered typical.

Implications: Comparing the two communities, we find a different distribution of contribution between the two: Stack Overflow and GNOME are skewed in their distribution of contribution between users, but Stack Overflow is skewed to a much lower degree. This may be because of the type of contributions provided in each community and the barrier that must be overcome before contributing. In GNOME, a user must obtain an account to the SCM system and receive commit access from the project before contributing. On Stack Overflow, a user can contribute answers to any question after creating their own account, any of which could be accepted if they are of sufficient quality. By lowering the barrier of entry and allowing easier contribution across a myriad of topics, Stack Overflow has been able to harness the knowledge of users who may have otherwise not been able to contribute.

RQ3: What is the role of tenured users in online, software communities?

Interviews with tenured community members revealed that these members may invest their time into community management and moderating posts. Such users from Stack Overflow highlighted in the interviews that they spent their time answering fewer, but more difficult questions instead of answering numerous, simpler questions. Other interviewees also noted that they participated in community moderation tasks such as closing duplicate or subjective questions from the forum. One interviewee discussed his participation in early, meta-based discussions about the design and function

of Stack Overflow. Changes emerging from this discussion eventually made their way into the site's design. Each of these activities helps to manage the community and signifies a vested interest of the community members in their community's success.

In the GNOME community, we observed a general trend of tenured users providing less central source code to projects. Regression analysis revealed that the more releases that a developer was involved in a project, the less central their code was on average. When the analysis was extended to block user tenure into categories (new developer, normal developer and experienced developer), the trend again was that the more tenure a user had with a project, the less central their code contributions were on average. Ecosystem tenure on the other hand was found to have little effect. When a sample of tenured users was selected, they were found to occupy a mixture of roles including project and community managers. The indication here is that tenure in the GNOME community decreases the centrality of code provided by developers and that this may be because these users have assumed project and community leadership roles.

Implications: In each of the studied communities, we observe that tenured users may fill more management and moderation roles. Members of Stack Overflow expressed this interest through discussions of the sites design and forum moderation tasks, while GNOME community members filled roles like project and community release managers. A commonality is that in each community, tenured users showed a vested interest in the success of the community.

6.1 Future Work

In this section, we identify two main themes from the implications of the research questions and present directions for future work.

First, we observe that lowering the barrier of entry for contributions allows and encourages a large base of users to contribute. This is apparent in the Stack Overflow community where we observe a more equal distribution of contribution between high and low contribution users. Applying this concept to an open source development community would allow a greater number of users to provide contributions to a wider range of projects. This concept has already been implemented by the GitHub community. Labeled "Social Coding", GitHub allows users to present code directly to a project that can be reviewed, discussed and accepted or rejected. Future work will investigate the impact of this implementation and its impact on the operation of projects in the GitHub community.

Second, we observe the appearance of specialized roles within the communities. One type of specialized role identified was the management and moderator role fulfilled by the tenured users within each community. Similar to observations of the structured communities of commercial development, senior members in the community with knowledge of the environment and its operation are tasked with leading and managing the work of others. Another specialized role identified was the role of the translators in the GNOME community. These users have less involvement with individual projects, but instead provide projects with translation help that they are best suited to provide. Future work will investigate further the function these specialized roles play in their respective communities and investigate the existence of additional specialized roles in these as well as other communities.

Appendix A

Stack Overflow

A.1 Data Schema

Included in this section is an description of the types of data provided in the June 2011, Stack Overflow data dump as described by the "readme.txt" file provided with the data.

- badges.xml
  - Id
  - UserId, e.g.: "420"
  - Name, e.g.: "Teacher"
  - Date, e.g.: "2008-09-15T08:55:03.923"
- comments.xml
  - Id
  - PostId
  - Score

  - Text, e.g.: “@Stu Thompson: Seems possible to me - why not try it?”
  - CreationDate, e.g.:“2008-09-06T08:07:10.730”
  - UserDisplayName: populated if a user has been removed and no longer referenced by user Id
  - UserId

- posts.xml
  - Id
  - PostTypeId
    - 1: Question
    - 2: Answer
  - ParentID (only present if PostTypeId is 2)
  - AcceptedAnswerId (only present if PostTypeId is 1)
  - CreationDate
  - Score
  - ViewCount
  - Body
  - OwnerUserId
  - OwnerDisplayName: populated if a user has been removed and no longer referenced by user Id or if the user was anonymous
  - LastEditorUserId
  - LastEditorDisplayName=“Jeff Atwood”
  - LastEditDate=“2009-03-05T22:28:34.823”

  - LastActivityDate="2009-03-11T12:51:01.480"
  - CommunityOwnedDate="2009-03-11T12:51:01.480"
  - ClosedDate="2009-03-11T12:51:01.480"
  - Title=
  - Tags=
  - AnswerCount
  - CommentCount
  - FavoriteCount

- posthistory.xml
  - Id
  - PostHistoryTypeId
    * 1: Initial Title - The first title a question is asked with.
    * 2: Initial Body - The first raw body text a post is submitted with.
    * 3: Initial Tags - The first tags a question is asked with.
    * 4: Edit Title - A question's title has been changed.
    * 5: Edit Body - A post's body has been changed, the raw text is stored here as markdown.
    * 6: Edit Tags - A question's tags have been changed.
    * 7: Rollback Title - A question's title has reverted to a previous version.
    * 8: Rollback Body - A post's body has reverted to a previous version - the raw text is stored here.
    * 9: Rollback Tags - A question's tags have reverted to a previous version.

    * 10: Post Closed - A post was voted to be closed.
    * 11: Post Reopened - A post was voted to be reopened.
    * 12: Post Deleted - A post was voted to be removed.
    * 13: Post Undeleted - A post was voted to be restored.
    * 14: Post Locked - A post was locked by a moderator.
    * 15: Post Unlocked - A post was unlocked by a moderator.
    * 16: Community Owned - A post has become community owned.
    * 17: Post Migrated - A post was migrated.
    * 18: Question Merged - A question has had another, deleted question merged into itself.
    * 19: Question Protected - A question was protected by a moderator
    * 20: Question Unprotected - A question was unprotected by a moderator
    * 21: Post Disassociated - An admin removes the OwnerUserId from a post.
    * 22: Question Unmerged - A previously merged question has had its answers and votes restored.
- PostId
- RevisionGUID: At times more than one type of history record can be recorded by a single action. All of these will be grouped using the same RevisionGUID
- CreationDate: "2009-03-05T22:28:34.823"
- UserId
- UserDisplayName: populated if a user has been removed and no longer referenced by user Id

  - Comment: This field will contain the comment made by the user who edited a post
  - Text: A raw version of the new value for a given revision
    * If PostHistoryTypeId = 10, 11, 12, 13, 14, or 15 this column will contain a JSON encoded string with all users who have voted for the PostHistoryTypeId
    * If PostHistoryTypeId = 17 this column will contain migration details of either "from <url>" or "to <url>"

- users.xml
  - Id
  - Reputation
  - CreationDate
  - DisplayName
  - EmailHash
  - LastAccessDate
  - WebsiteUrl
  - Location
  - Age
  - AboutMe
  - Views
  - UpVotes
  - DownVotes

- votes.xml
  - Id
  - PostId
  - VoteTypeId
    - 1: AcceptedByOriginator
    - 2: UpMod
    - 3: DownMod
    - 4: Offensive
    - 5: Favorite - if VoteTypeId = 5 UserId will be populated
    - 6: Close
    - 7: Reopen
    - 8: BountyStart
    - 9: BountyClose
    - 10: Deletion
    - 11: Undeletion
    - 12: Spam
    - 13: InformModerator
  - CreationDate
  - UserId (only for VoteTypeId 5)
  - BountyAmount (only for VoteTypeId 9)

A.2 High and Low Contribution User Breakdown

.

Table A.1: Summary of the amount and relative percentage of high and low contribution users and the contribution they provide by month.

Month	Users				Answers			
	High Contrib.		Low Contrib.		High Contrib.		Low Contrib.	
8/1/2008	290	13.0%	1939	87.0%	13428	57.0%	10119	43.0%
9/1/2008	956	10.7%	7968	89.3%	45607	54.2%	38592	45.8%
10/1/2008	737	9.3%	7172	90.7%	36039	53.5%	31322	46.5%
11/1/2008	479	6.5%	6859	93.5%	24862	48.6%	26261	51.4%
12/1/2008	490	6.8%	6753	93.2%	24631	49.8%	24848	50.2%
1/1/2009	684	8.1%	7781	91.9%	34685	53.2%	30506	46.8%
2/1/2009	704	7.6%	8518	92.4%	34880	51.3%	33087	48.7%
3/1/2009	723	7.4%	9023	92.6%	39023	54.0%	33193	46.0%
4/1/2009	698	6.9%	9483	93.1%	37458	52.1%	34384	47.9%
5/1/2009	783	6.4%	11372	93.6%	41349	49.5%	42112	50.5%
6/1/2009	894	6.9%	12010	93.1%	46883	51.8%	43544	48.2%
7/1/2009	968	7.0%	12845	93.0%	51321	52.9%	45740	47.1%
8/1/2009	906	6.3%	13443	93.7%	48586	51.1%	46541	48.9%
9/1/2009	870	6.0%	13648	94.0%	46217	50.7%	44921	49.3%
10/1/2009	972	5.5%	16663	94.5%	47934	49.1%	49628	50.9%
11/1/2009	1050	5.2%	19148	94.8%	52471	47.7%	57484	52.3%
12/1/2009	985	5.1%	18517	94.9%	49697	47.7%	54414	52.3%
1/1/2010	1107	5.2%	20172	94.8%	61300	51.0%	58854	49.0%
2/1/2010	1017	4.9%	19800	95.1%	53174	47.7%	58396	52.3%
3/1/2010	1137	4.9%	22131	95.1%	61367	48.8%	64323	51.2%
4/1/2010	1026	4.6%	21279	95.4%	54903	48.1%	59130	51.9%
5/1/2010	1102	4.6%	22765	95.4%	57946	47.7%	63456	52.3%
6/1/2010	1192	4.8%	23551	95.2%	60579	47.9%	65846	52.1%
7/1/2010	1279	4.9%	24606	95.1%	66337	48.9%	69296	51.1%
8/1/2010	1380	5.0%	26111	95.0%	71646	49.8%	72132	50.2%
9/1/2010	1247	4.8%	24560	95.2%	63993	48.3%	68523	51.7%
10/1/2010	1289	4.6%	26553	95.4%	65485	47.4%	72620	52.6%
11/1/2010	1429	4.7%	29259	95.3%	73619	47.5%	81361	52.5%
12/1/2010	1356	4.4%	29382	95.6%	69140	46.5%	79461	53.5%
1/1/2011	1657	4.7%	33434	95.3%	86106	48.7%	90709	51.3%
2/1/2011	1559	4.3%	34444	95.7%	78161	44.8%	96430	55.2%
3/1/2011	2072	5.2%	37995	94.8%	104431	48.9%	109017	51.1%
4/1/2011	1852	5.0%	35211	95.0%	95567	48.8%	100102	51.2%
5/1/2011	1942	5.1%	36220	94.9%	97373	48.2%	104460	51.8%
Total	36832	5.4%	650658	94.6%	1896198	49.2%	1960914	50.8%

Table A.2: Summary of the amount and relative percentage of high and low contribution users providing accepted answers and the amount of accepted answers they provide by month.

Month	Users				Accepted Answers			
	High Contrib.		Low Contrib.		High Contrib.		Low Contrib.	
8/1/2008	280	26.2%	790	73.8%	2016	58.3%	1440	41.7%
9/1/2008	906	24.2%	2837	75.8%	6469	57.1%	4865	42.9%
10/1/2008	711	21.8%	2549	78.2%	6081	58.3%	4356	41.7%
11/1/2008	469	16.1%	2439	83.9%	4788	53.6%	4153	46.4%
12/1/2008	473	16.7%	2351	83.3%	4490	53.7%	3871	46.3%
1/1/2009	660	18.7%	2864	81.3%	6617	56.9%	5019	43.1%
2/1/2009	692	17.8%	3197	82.2%	7170	56.4%	5533	43.6%
3/1/2009	712	16.9%	3510	83.1%	8726	59.0%	6059	41.0%
4/1/2009	686	15.8%	3649	84.2%	8326	56.7%	6352	43.3%
5/1/2009	775	14.5%	4579	85.5%	9517	53.7%	8208	46.3%
6/1/2009	885	15.6%	4772	84.4%	10845	56.2%	8469	43.8%
7/1/2009	958	15.9%	5086	84.1%	12407	57.7%	9099	42.3%
8/1/2009	902	14.3%	5424	85.7%	12763	56.1%	10003	43.9%
9/1/2009	868	14.1%	5304	85.9%	12650	56.1%	9918	43.9%
10/1/2009	965	14.1%	5881	85.9%	13679	56.1%	10718	43.9%
11/1/2009	1043	13.7%	6544	86.3%	14692	55.1%	11996	44.9%
12/1/2009	973	13.4%	6265	86.6%	14135	55.2%	11478	44.8%
1/1/2010	1097	13.3%	7130	86.7%	18681	58.6%	13204	41.4%
2/1/2010	1013	12.4%	7143	87.6%	17019	55.6%	13573	44.4%
3/1/2010	1134	12.4%	8007	87.6%	19724	56.4%	15247	43.6%
4/1/2010	1024	12.0%	7519	88.0%	18130	55.9%	14326	44.1%
5/1/2010	1093	12.2%	7878	87.8%	19346	56.0%	15172	44.0%
6/1/2010	1189	12.3%	8440	87.7%	20642	56.3%	16026	43.7%
7/1/2010	1273	12.8%	8690	87.2%	22352	57.3%	16661	42.7%
8/1/2010	1370	12.9%	9255	87.1%	24542	58.4%	17465	41.6%
9/1/2010	1245	12.3%	8842	87.7%	22251	56.6%	17050	43.4%
10/1/2010	1287	12.1%	9392	87.9%	22759	55.8%	18037	44.2%
11/1/2010	1421	12.0%	10373	88.0%	24953	55.3%	20166	44.7%
12/1/2010	1351	11.9%	10019	88.1%	23896	55.2%	19391	44.8%
1/1/2011	1653	12.5%	11617	87.5%	29710	57.1%	22318	42.9%
2/1/2011	1553	11.3%	12147	88.7%	27647	53.8%	23779	46.2%
3/1/2011	2069	13.1%	13692	86.9%	35322	56.9%	26706	43.1%
4/1/2011	1850	12.8%	12615	87.2%	31684	56.2%	24675	43.8%
5/1/2011	1928	13.3%	12552	86.7%	31473	56.3%	24469	43.7%
Total	36508	13.5%	233366	86.5%	565502	56.3%	439822	43.7%

A.3 Interviews

Please tell us a bit about yourself:

- What attracted you to Stack Overflow in the first place?
- What motivates you now to contribute to Stack Overflow?
- How did you learn to contribute to the site to get accepted by the community? (either being upvoted or getting your answer accepted)

Contribution activity: We found that high activity (>20 answer per month) and low activity user groups contribute similar volumes of answers, which is highly atypical of online communities. This shows that Stack Overflow attracts a large base of low-activity users who are contributing (about 45%) as well as stalled users, who have not gotten an answer accepted or question answered.

- Have you ever made any distinctions between high-activity vs. low-activity users in your interactions in the site (the reputation points of a user can allude to their high-activity)?
- What underlying design of Stack Overflow do you think attracts people to contribute, even low activity and stalled users?
- Does the community believe in encouraging stalled or low-activity users to contribute? If yes, how are these users encouraged?
- Do you feel that there are any effects on the community/quality of the board in having a high percentage of content provided by low activity users?

Answer Quality: Our Statistical analysis of answers (C# and Ruby tags) shows that technical criteria (embeds code, length of answers) trump social cues (e.g., reputation of posters, favorite count):

- What factors do you consider the most important when determining the quality of an answer?
- Can you recall the characteristics of an answer that you have recently accepted?
- Does the social factors (reputation points of the poster, favorite counts, discussion on the question or answer) affect your perception of quality? If yes, in what way. If no, why not?
- How do you think the game-centered reputation gains have an impact on how users answer questions?
- Do you feel that the score of a parent question elicit "good" answers? Why?
- Are there other design elements of Stack Overflow that affect how users ask or answer questions? (e.g., peer pressure, moderation pressure?)

Stalled Users: Our analysis shows that a large number of users have difficulty getting an answer accepted or question answered in Stack Overflow. Nearly 40

- With so many questions already answered, do you think it more difficult now to provide content to Stack Overflow?
- What is your advice to aspiring new users?

Is there anything else that you would like to tell us that we have not already covered?

Appendix B

Gnome

B.1 Data Schema

(On next page.)

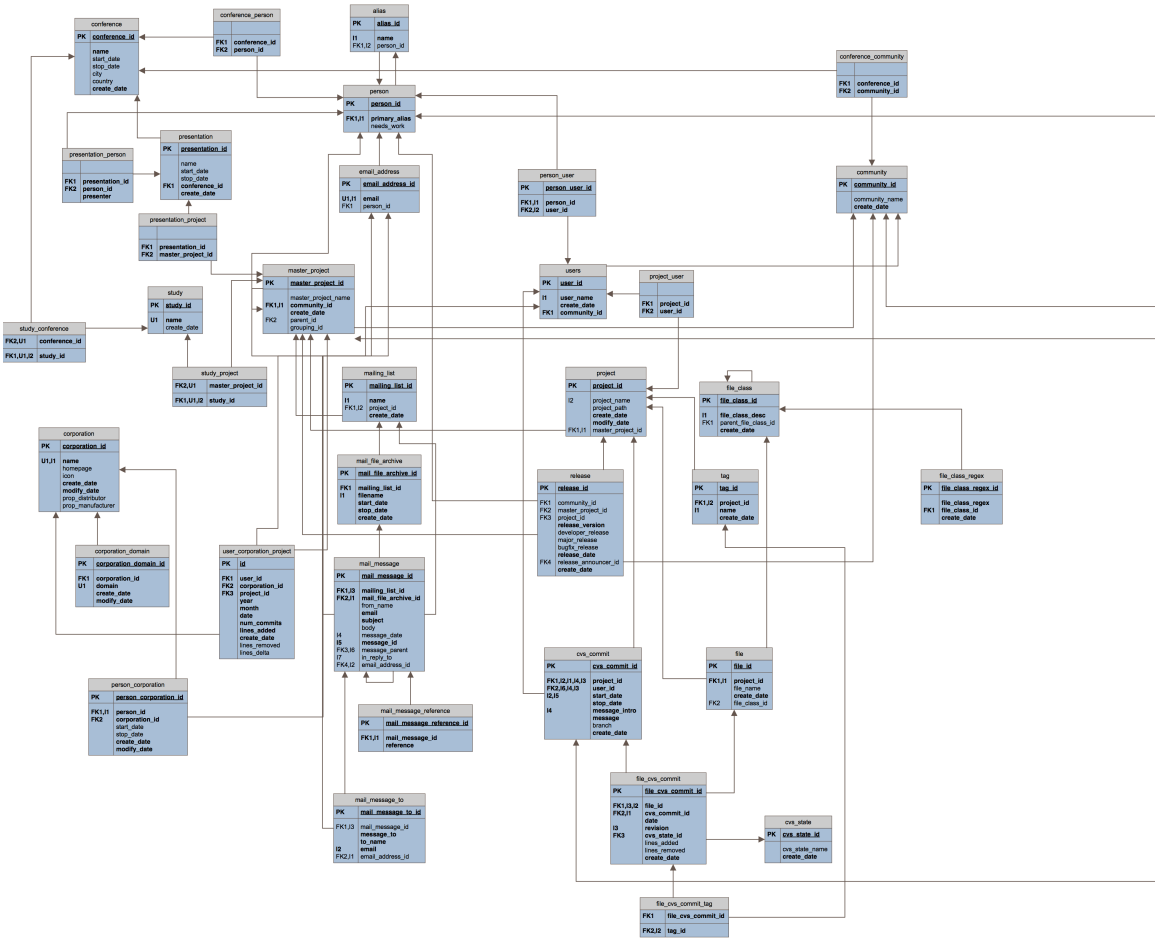

Figure B.1: Schema diagram for GNOME community data.

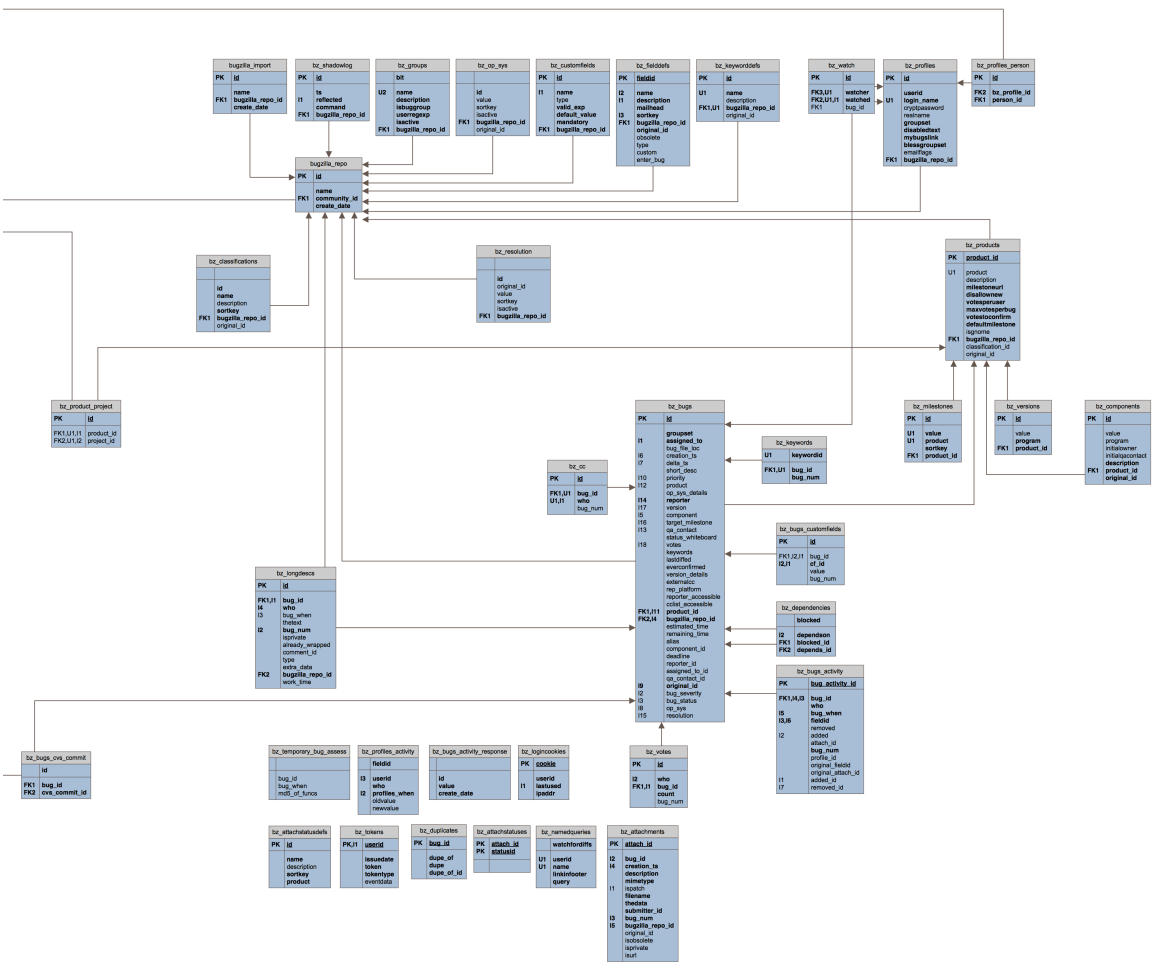

Figure B.2: Schema diagram for GNOME community data.

Bibliography

[1] M. S. Ackerman and T. W. Malone. Answer garden: a tool for growing organizational memory. *SIGOIS Bull.*, 11(2-3):31–39, March 1990.

[2] Lada A. Adamic, Jun Zhang, Eytan Bakshy, and Mark S. Ackerman. Knowledge sharing and yahoo answers: everyone knows something. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 665–674, New York, NY, USA, 2008. ACM.

[3] Jeff Atwood. Creative commons data dump sep 11. http://blog.stackoverflow.com/2011/09/creative-commons-data-dump-sep-11. [Online; accessed 10-July-2013].

[4] Gerard Beenen, Kimberly Ling, Xiaoqing Wang, Klarissa Chang, Dan Frankowski, Paul Resnick, and Robert E. Kraut. Using social psychology to motivate contributions to online communities. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, CSCW '04, pages 212–221, New York, NY, USA, 2004. ACM.

[5] Christian Bird, Alex Gourley, and Prem Devanbu. Open borders? immigration in open source projects. In *In MSR 07: Proceedings of the Fourth International Workshop on Mining Software Repositories*, page 6. IEEE Computer Society, 2007.

[6] Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, and Anand Swaminathan. Mining email social networks. In *Proceedings of the 2006 international workshop on Mining software repositories*, MSR '06, pages 137–143, New York, NY, USA, 2006. ACM.

[7] Christian Bird, David Pattison, Raissa D'Souza, Vladimir Filkov, and Premkumar Devanbu. Latent social structure in open source projects. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, SIGSOFT '08/FSE-16, pages 24–35, New York, NY, USA, 2008. ACM.

[8] Cornelia Boldyreff, Kevin Crowston, Björn Lundell, and Anthony I. Wasserman. *Open Source Ecosystems: Diverse Communities Interacting 5th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2009, Skvde, Sweden, in Information and Communication Technology*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[9] Phillip Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92(5):1170–1182, Mar 1987.

[10] David Constant and Sara Sproull, Leeand Kiesler. The kindness of strangers: The usefulness of electronic weak ties for technical advice. *Organization Science*, 7(2):119–135, 1996.

[11] Kevin Crowston and James Howison. The social structure of open source software development teams. In *First Monday*, 2003.

[12] Kevin Crowston, Kangning Wei, James Howison, and Andrea Wiggins. Free/libre open source software development: What we know and what we do not know. *ACM Computing Surveys*, 44, 02/2012 2012.

[13] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, CSCW '12, pages 1277–1286, New York, NY, USA, 2012. ACM.

[14] Barthélémy Dagenais, Harold Ossher, Rachel K. E. Bellamy, Martin P. Robillard, and Jacqueline P. de Vries. Moving into a new software project landscape. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 275–284, New York, NY, USA, 2010. ACM.

[15] David Dearman and Khai N. Truong. Why users of yahoo!: answers do not answer questions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 329–332, New York, NY, USA, 2010. ACM.

[16] Nicolas Ducheneaut. Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work (CSCW)*, 14(4):323–368, 2005.

[17] Benjamin Edelman. Earnings and ratings at google answers. *Economic Inquiry*, 50(2):309–320, 04 2012.

[18] Rosta Farzan, Laura A. Dabbish, Robert E. Kraut, and Tom Postmes. Increasing commitment to online communities by designing for social presence. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, CSCW '11, pages 321–330, New York, NY, USA, 2011. ACM.

[19] H. Gall, K. Hajek, and M. Jazayeri. Detection of logical coupling based on product release history. In *Software Maintenance, 1998. Proceedings., International Conference on*, pages 190–198, 1998.

[20] Daniel M. German. The gnome project: a case study of open source, global software development. *Software Process: Improvement and Practice*, 8(4):201–215, 2003.

[21] Zoltan Gyongyi, Georgia Koutrika, Jan Pedersen, and Hector Garcia-Molina. Questioning yahoo! answers. Technical Report 2007-35, Stanford InfoLab, 2007.

[22] F. Maxwell Harper, Daniel Moy, and Joseph A. Konstan. Facts or friends?: distinguishing informational and conversational questions in social q&a sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 759–768, New York, NY, USA, 2009. ACM.

[23] F. Maxwell Harper, Daphne Raban, Sheizaf Rafaeli, and Joseph A. Konstan. Predictors of answer quality in online q&a sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 865–874, New York, NY, USA, 2008. ACM.

[24] Guido Hertel, Sven Niedner, and Stefanie Herrmann. Motivation of software developers in open source projects: an internet-based survey of contributors to the linux kernel. *Research Policy*, 32:1159–1177, 2003.

[25] Gary Hsieh, Robert E. Kraut, and Scott E. Hudson. Why pay?: exploring how financial incentives are used for question & answer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 305–314, New York, NY, USA, 2010. ACM.

[26] Chris Jensen and Walt Scacchi. Role migration and advancement processes in ossd projects: A comparative case study. In *in Proceedings 29th International Conference Software Engineering, ACM*, pages 364–374, 2007.

[27] Corey Jergensen, Anita Sarma, and Patrick Wagstrom. The onion patch: migration in open source ecosystems. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, ESEC/FSE '11, pages 70–80, New York, NY, USA, 2011. ACM.

[28] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[29] Robert E. Kraut, Robert S. Fish, Robert W. Root, Barbara L. Chalfonte, I S. Oskamp, Robert E. Kraut, Robert S. Fish, Robert W. Root, and Barbara L. Chalfonte. Informal communication in organizations: Form, function, and technology. In *Human reactions to technology*, 1990.

[30] Karim R Lakhani and Eric von Hippel. How open source software works: free user-to-user assistance. *Research Policy*, 32(6):923 – 943, 2003.

[31] Josh Lerner and Jean Triole. The simple economics of open source. Working Paper 7600, National Bureau of Economic Research, March 2000.

[32] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. Design lessons from the fastest q&a site in the west. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2857–2866, New York, NY, USA, 2011. ACM.

[33] Andrew Meneely, Laurie Williams, Will Snipes, and Jason Osborne. Predicting failures with developer networks and social network analysis. In *Proceedings of*

the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, SIGSOFT '08/FSE-16, pages 13–23, New York, NY, USA, 2008. ACM.

[34] A. Mockus, R.T. Fielding, and J. Herbsleb. A case study of open source software development: the apache server. In *Software Engineering, 2000. Proceedings of the 2000 International Conference on*, pages 263–272, 2000.

[35] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering Methodology*, 11(3):309–346, July 2002.

[36] Kevin Kyung Nam, Mark S. Ackerman, and Lada A. Adamic. Questions in, knowledge in?: a study of naver's question answering community. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 779–788, New York, NY, USA, 2009. ACM.

[37] B. Nonnecke and J. Preece. Shedding light on lurkers in online communities. *Ethnographic Studies in Real and Virtual Environments: Inhabited Information Spaces and Connected Communities, Edinburgh*, page 123128, 1999.

[38] Blair Nonnecke and Jenny Preece. Lurker demographics: counting the silent. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '00, pages 73–80, New York, NY, USA, 2000. ACM.

[39] P. Norris. *Digital Divide: Civic Engagement, Information Poverty, and the Internet Worldwide*. Communication, Society and Politics. Cambridge University Press, 2001.

[40] Wonseok Oh and Sangyong Jeon. Membership herding and network stability in the open source community: The ising perspective. *Management Science*, 53(7):1086–1101, July 2007.

[41] Hüseyin Oktay, Brian J. Taylor, and David D. Jensen. Causal discovery in social media using quasi-experimental designs. In *Proceedings of the First Workshop on Social Media Analytics*, SOMA '10, pages 1–9, New York, NY, USA, 2010. ACM.

[42] Katherine Panciera, Aaron Halfaker, and Loren Terveen. Wikipedians are born, not made: a study of power editors on wikipedia. In *Proceedings of the ACM 2009 international conference on Supporting group work*, GROUP '09, pages 51–60, New York, NY, USA, 2009. ACM.

[43] Chris Parnin, Christoph Treude, Lars Grammel, and Margaret-Anne Storey. Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow. Technical report, Georgia Institute of Technology, 2012.

[44] S. Rafaeli, G. Ravid, and V. Soroka. De-lurking in virtual communities: a social communication network approach to measuring the effects of social and cultural capital. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pages 10 pp.–, 2004.

[45] Walt Scacchi. Free/open source software development: recent research results and emerging opportunities. In *The 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering: companion papers*, ESEC-FSE companion '07, pages 459–468, New York, NY, USA, 2007. ACM.

[46] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. How do programmers ask and answer questions on the web? (nier track). In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 804–807, New York, NY, USA, 2011. ACM.

[47] Georg von Krogh, Sebastian Spaeth, and Karim R. Lakhani. Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, 32(7):1217–1241, July 2003.

[48] Patrick Wagstrom. Gnome archive data. http://academic.patrick.wagstrom.net/research/gnome. [Online; accessed 10-July-2013].

[49] Alexander Wolfe. Eclipse: A platform becomes an open-source woodstock. *Queue*, 1(8):14–16, November 2003.

[50] Bo Xu and Donald R. Jones. Volunteers' participation in open source software development: a study from the social-relational perspective. *SIGMIS Database*, 41(3):69–84, August 2010.

[51] J. Yang, L. A Adamic, and M. S Ackerman. Competing to share expertise: the taskcn knowledge sharing community. In *International conference on weblogs and social media*, 2008.

[52] Yunwen Ye and Kouichi Kishida. Toward an understanding of the motivation open source software developers. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, pages 419–429, Washington, DC, USA, 2003. IEEE Computer Society.

[53] Jie Yu, Zhenhui Jiang, and Hock Chuan Chan. Knowledge contribution in problem solving virtual communities: the mediating role of individual motivations. In

Proceedings of the 2007 ACM SIGMIS CPR conference on Computer personnel research: The global information technology workforce, SIGMIS CPR '07, pages 144–152, New York, NY, USA, 2007. ACM.