Attracting the Community's Many Eyes: an Exploration of User Involvement in Issue Tracking

Lars Grammel
University of Victoria
Victoria, BC, Canada
Lars.Grammel@gmail.com

Holger Schackmann
RWTH Aachen University
Aachen, Germany
schackmann@swc.rwth-aachen.de

Adrian Schröter
University of Victoria
Victoria, BC, Canada
schadr@uvic.ca

Christoph Treude
University of Victoria
Victoria, BC, Canada
ctreude@uvic.ca

Margaret-Anne Storey
University of Victoria
Victoria, BC, Canada
mstorey@uvic.ca

ABSTRACT
A community of users who report bugs and request features provides valuable feedback that can be used in product development. Many open source projects provide publicly accessible issue trackers to facilitate such feedback. We compare the community involvement in issue tracker usage between the open source project Eclipse and the closed source project IBM Jazz to evaluate if publicly accessible issue trackers work as well in closed source projects. We find that IBM Jazz successfully receives user feedback through this channel. We then explore the differences in work item processing in IBM Jazz between team members, project members and externals. We conclude that making public issue trackers available in closed source projects is a useful approach for eliciting feedback from the community, but that work items created by team members are processed differently from work items created by project members and externals.

Categories and Subject Descriptors
K.6.3 [**Software Management**]: Software process

General Terms
Human Factors

Keywords
Open source, users, issue tracking

1. INTRODUCTION AND MOTIVATION
Involvement of the user community during software development can lead to creating higher quality software that is more appropriately tailored to the users' needs [16, 18]. For software companies that are seeking to improve their products, it is therefore important to gain more feedback from their users. Such community involvement can work well in major open source projects (e.g. Eclipse [8]). This raises a question: Can a similar degree of community involvement also be achieved for closed source offerings?

In order to solicit input from the user community, IBM has introduced a new development approach in the Jazz[1] project, that they call "open commercial development" [7]. Jazz is a team collaboration platform for integrating work across the phases of the software development lifecycle. While it is based on the open source project Eclipse, Jazz components, such as the Rational Team Concert IDE, are commercial and closed source. In contrast to a closed source approach, however, the Jazz community can download integration builds, see the development process, and participate in forums and issue trackers. Also, community members have insights into the release and iteration plans, as well as access to work items, such as bug reports, tasks, or change requests. Work items can be created and discussed by community members. Every user has access to the team wikis, blogs and tutorials, and can discuss issues or request help via forums. In other words, the community members, especially beta-testers, are aware of the project progress at any time, even though they are not allowed to modify source code. The goal of the open commercial development is to leverage some of the strengths of the open source model, e.g. using the comments and recommendations from the community to influence development.

It is not well understood yet whether the community embraces this type of projects and whether companies are successful in benefiting from the community's many eyes. Our study aims to close this gap by analyzing the degree of community involvement in the issue tracking systems of Jazz [7]. In order to draw comparisons to an open source project, we use the Eclipse project as a baseline. Eclipse is not only one of the largest open source projects, but it is also one of the most intensively studied [2, 4, 5]. Eclipse is used as a baseline since both projects are in the domain of development environments. Moreover many Jazz developers are former Eclipse developers. This reduces threats to validity to this study given by different working styles of the development teams.

The remainder of this paper is structured as follows: Section 2 provides an overview of the related work. It is followed by a description of our research questions and methodology (Section

Conference'10, Month 1–2, 2010, City, State, Country.

[1] http://jazz.net

3). In Section 4, we present our results. We then discuss the implications of these results in Section 5. Next, we describe threats to validity and how we mitigate them (Section 6). Finally, we present our conclusions and ideas for future work (Section 7).

2. RELATED WORK

Issue trackers in open source projects such as Eclipse, Mozilla, and Apache represent a well studied source of information. Studies leveraging this archive of community knowledge [14] had different goals such as assessing the quality of and evaluating the relations between different issue reports [2, 19].

Bettenburg et al. [2] measured the quality of issued bug reports and studied the difference between the expectations of developers and reporters on its quality. Hooimeijer and Weimer [12] considered how resolution time can be used as a quality indicator for bug reports. Sandusky [19] investigated the connections between different issue reports and how the community creates these connections (e.g. dependencies or duplicate relations) over time.

Several studies investigated the role changes of open source community members using data from issue trackers and mailing lists. They found that people enter the community and evolve from users to developers, but also become inactive and drop out of the ranks of developers [25, 24, 6]. The frequency of changes depends on the type of community structure. Structures ranging from "monarchies" to "democracies" [17] and from hierarchical to dynamic [18, 3] have been observed within open software communities.

Recent studies [1, 21, 13, 10] focused on the role of users in open source communities. This research concludes that users are a vital part of open source software communities because they drive the software project and its evolution as a whole [13, 1]. Besides influencing the project evolution towards their needs [1], users are also rewarded by fast replies to support requests. Developers also acknowledge the important role of users [21]. Although this research has shown interest in open source users and developers, we still need to improve our understanding of the interaction between developers and users within a project.

While the general importance of involving the community in software projects is clear, their participation in issue tracking is more controversial. For the same software system (Firefox web browser), Van Liere finds that large communities of bug reporters reduce the time needed for defect resolution [22], but Ko and Chilana report evidence that the user community reports "non-issues that devolved into technical support, redundant reports with little new information, or narrow, expert feature requests" [15]. Further research is necessary to understand when and how the community should be involved in issue tracking and how it should be coordinated with technical support.

Many software companies, including major vendors such as IBM, Sun, and Microsoft, support various open source projects [23]. Despite their familiarity with the transparency of open source development, Jazz is, to our knowledge, one of the first major attempts to bring a similar transparency in issue tracking to commercial software development. We are not aware of any other studies that compare the community involvement in issue tracking between open source and commercial development.

3. METHODOLOGY

This section identifies our research questions as well as the methods we used to answer these questions.

3.1 Research Questions

The research questions below cover the main themes of our study: the community involvement in Jazz compared to the community involvement in the open source project Eclipse, and potential differences in the processing of Jazz work items created by members outside of IBM. In this study, we measure *community involvement in issue tracking* as the number of comments made by community members, the number of work items created by community members, and the number of involved community members. Our detailed research questions are:

1. What is the degree of community involvement in issue tracking in the open commercial project Jazz?
   a. What is the absolute community involvement in Jazz?
   b. What is the community involvement in Jazz relative to the developer involvement?
   c. What is the average involvement of an active community member in Jazz?
2. Is there a similar degree of community involvement in Eclipse and Jazz?
3. Are work items created by community, project and team members (See 3.2 community status distinction) processed differently in the open commercial project Jazz?
   a. Do the amounts of communication differ?
   b. Do the assignment and processing times differ?
   c. Do the reopened rates differ?

We focus on Jazz for the last research question because there are differences in the issue tracking between Jazz and Eclipse that prevent a cross-comparison, and our main focus is to understand the community involvement in issue tracking in open commercial development.

3.2 Methodology

Our study follows a case study design. We divide the individuals active in development and issue tracking into team members, project members, and community members. Then, we compare the amounts of involvement in both issue creation and commenting on issues for these three groups. This analysis also enables us to compare the community involvement in the open source project Eclipse to the open commercial project Jazz. Moreover, we analyze the average assignment and processing times of work items as well as the average amount of communication related to work items.

Community Status Distinction. Measuring the community involvement requires determining the size of the community and distinguishing participants according to their role in the community. As our focus lies on issue tracking, we only count individuals who are actively participating in the issue trackers for the selected components during the selected time periods. For these individuals, we distinguish between 3 groups:

- Team Members (TM): members of the component development team.
- Project Members (PM): members of the project (i.e. Jazz, Eclipse) who are not team members of the component under evaluation.
- Community Members (CM): those who actively participate in issue tracking and who are neither team nor project members.

For Eclipse, lists with developers[2] and former developers[3] are available on eclipse.org. Moreover, the project dash[4] provides lists of committers for each Eclipse component based on automatically collected information from the source code repository. We found that the latter information is more accurate. Therefore, we identified team members by manually resolving the source code repository usernames to Eclipse Bugzilla user IDs. Alias names were also resolved manually. Resolving the Bugzilla user IDs of all Eclipse project members was not feasible due to inconsistencies of real user names in the database. Therefore, we considered Eclipse Bugzilla users with "ibm.com" or "oti.com" email addresses as project members. This can be justified since 95% of the lines of change in the Eclipse project in the years 2001-2003 have been contributed by IBM and former OTI developers (according to analysis results of Eclipse dash). For Jazz, lists of all project members as well as of the team members of each component are available in the Jazz repository.

Time Period Selection. We have chosen the following evaluation time periods with the goal of selecting work items from comparable project phases for analysis. This reduces the risk that the project phases influence our results in an unanticipated way.

For Jazz, we evaluated the time period from the month after the Jazz project became open commercial to the month in which Rational Team Concert 1.0 was released. This is the time period from December 2006 to June 2008 (19 months).

For Eclipse, we analyzed two different time intervals, both beginning when Eclipse became open source and Eclipse 1.0 was released in November 2001. The first time interval ends in June 2002 (7 months), the month in which Eclipse 2.0 was released. The reason for choosing this interval was to have intervals that end at releases and thus do not include work item activity that is caused by post-release bugs. The second time interval ends in June 2003 (19 months), the month in which Eclipse 2.1.1 was released. This time interval was chosen because it has the same length as the time interval that was evaluated for Jazz.

Component Selection. In order to reduce the risk that project size, topic or other project-specific peculiarities influence our results, we have chosen several components from the Eclipse and Jazz projects which are comparable in terms of numbers of contributors, maturity and goal. Each component has its development team.

For Eclipse, we have chosen the components from the Eclipse project list[5] and the component status in 2002. We excluded components that started after 2001, that were too small, or that had less than 10 contributors. We also excluded components that had more than 30 contributors, because the components in Jazz have fewer contributors. JDT was included as it had 30 team members active in the issue tracker between December 2001 and June 2003. Because there were no components in Jazz with comparable development topics, we also excluded modeling, runtime, and technology. The following three Eclipse components met our requirements:

- Java Development Tools (JDT)
- Plugin Development Environment (PDE)
- C/C++ Development Tooling (CDT)

For Jazz, we excluded all components that have fewer than 10 contributors, that are too small or new, that are too generic, or that are not focused on development. The following four Jazz components met our requirements:

- Work Item (WI): support for managing defect reports, feature requests, and other development tasks
- Source Control (SC): version control system
- Process (P): process support foundations
- Repository (R): server-side architecture, low-level client-side API

Data Retrieval and Analysis. Instead of developing custom scripts for retrieving the data from the Eclipse Bugzilla database and the Jazz database respectively, we applied a more general approach that is supported by the open source tool BugzillaMetrics[6]. The tool offers a generic evaluation algorithm for metrics on change request data and is adaptable to different underlying data sources [9]. Metrics can be specified in a declarative language. Basic language constructs are filters for attributes and events in the life cycle of a change request. These filters can then be combined with calculations like counting certain events or determining the length of a time interval in the life cycle of an issue report. Hence, metrics can be specified at a higher abstraction level, which simplifies development and validation of metrics [20]. Moreover, this approach made it easier to ensure comparability of the metric definitions used for Jazz and for Eclipse.

We analyzed the Eclipse data based on a snapshot of the Bugzilla database taken in August 2008. All change requests reported for the selected Eclipse components in the considered time periods since November 2001 contained valid historic entries.

Mining the Jazz data with BugzillaMetrics required a preparatory data extraction step. We extracted the data from the Jazz server using a program that accessed both the Jazz API and the Jazz database. We stored the extracted data in a separate database. This separate database was analyzed using BugzillaMetrics. This data preparation step was necessary for an efficient analysis of the Jazz work item data using BugzillaMetrics. Due to inconsistencies in the underlying Jazz database, extracting the history was not possible for 63 work items (out of 47,669 work items, 0.13%).

Comparability of Work Items in Jazz and Eclipse. In order to draw comparisons between the handling of work items in Jazz and Eclipse, it must be considered whether the work items in the selected components of both projects are comparable in terms of size and complexity.

Unfortunately, this analysis cannot be based simply on an attribute of the work items. Eclipse does not collect effort values for the work items. It was shown in a case study of the Eclipse project that severity and priority are poor indicators of the complexity of a work item [11], as not every reporter is capable of using the given severity classifications, and developers do not use priorities in a consistent manner.

However, we believe that the work items for the selected components are comparable for a number of reasons. We selected components with similar development topics for both projects.

[2] http://www.eclipse.org/projects/lists.php?list=allbyproject
[3] http://www.eclipse.org/projects/committers-alumni.php?sort=name
[4] http://www.eclipse.org/dash/
[5] http://www.eclipse.org/projects/lists.php?list=allbyproject
[6] http://www.bugzillametrics.org

Moreover, a considerable number of Jazz developers had been former developers of Eclipse. Due to this personnel continuity, we assume that there is a common understanding of the usage scheme of the issue tracking system by team and project members.

The resolution categories can be inspected with respect to the work items reported by community members. The resolution *Fixed* indicates that some change of the source code had been made related to the work item. For the analyzed Eclipse components, the proportion of community work items with resolution *Fixed* ranges from 36.5% to 47.2%, while for the Jazz components, the proportion ranges from 36.1% to 70.0%. We could not detect any significant difference, $t(3)=1.09$, $p=.33$. Similar ranges in both Jazz and Eclipse can also be observed for other resolution categories like *Invalid* or *Duplicate*.

Relationship between Research Questions and Methodology. To answer our first two research questions regarding the degree of community involvement in issue tracking in both Eclipse and Jazz, we compared the absolute and relative numbers of created work items and comments for the different components, time periods and community roles, normalized per month. We also compared the number of distinct individuals who were active in the issue trackers, again split by time period, component and community role. Furthermore, we looked at the average number of created work items and comments per active individual. We again compared between the different components, time periods and roles, normalized per month.

Besides the community involvement in change request management, we also investigated the processing differences and similarities between work items created by community, project and team members (Research Question 3). In particular, we examined if there are differences in the amount of communication, in the reopened rate, and in the assignment and processing speed. We evaluated the following three measurements with that goal in mind:

1. The number of reopened work items relative to the number of resolved work items (reopened rate).
2. The median age in days when a work item is resolved for the first time (resolution age).
3. The percentage of work items still unassigned 7 days after being created.

For each of the three measurements, the work items were divided into work items created by community members, work items created by project members, and work items created by team members. The measurements were then carried out for the four evaluated Jazz projects. For number of comments and median age of resolution, we also measured the corresponding Eclipse projects to get a frame of reference.

4. RESULTS

4.1 What is the degree of community involvement in issue tracking in the open commercial project Jazz?

Metric results related to the community involvement in the selected Jazz components are given in Table 1. The Source Control (SC) and Work Item (WI) components are the Jazz components with the most issue tracking activity of the four evaluated components. They have over 1.5 times more created work items and about 2-3 times more comments than the smallest of the evaluated Jazz components, the Process (P) component. P also exhibits the lowest absolute community activity in issue tracking. Most community comments per month (78.8) were made in the Repository (R) component, and most community work items were created in the WI component. Table 1(a) shows the average number of work item that were created per month, split into community, project and team member created work items. Table 1(b) shows a similar table for the average number comments per month.

Table 1: Metrics calculated on Jazz components

	WI	SC	P	R
Community	30.5	24.8	10.3	25.6
Project	138.6	106.4	92.4	64.8
Team	244.2	268.1	152.8	116.7

(a) average number of created work items per month

	WI	SC	P	R
Community	40.7	50.6	14.9	78.7
Project	240.4	226.8	163.4	215.9
Team	773.4	1294.8	396.1	563.3

(b) average number of comments per month

	WI	SC	P	R
Community	150	116	90	139
Project	108	105	97	88
Team	16	12	9	23

(c) number of active contributors

	WI	SC	P	R
Community	0.20	0.21	0.11	0.18
Project	1.28	1.01	0.95	0.74
Team	15.26	22.34	16.98	5.08

(d) average number of created work items per month and active contributor

	WI	SC	P	R
Community	0.27	0.44	0.17	0.57
Project	2.23	2.15	1.68	2.45
Team	48.34	107.90	44.01	24.49

(e) average number of comments per month and active contributor

The results for the distribution of issue tracking activity between team, project and community members indicate that the community share in work item creation is significantly lower than the share of team and project members (see Figure 1; Paired t-tests: community vs. team: $t(3)=-14.78, p<0.01$; community vs. project: $t(3)=-8.06, p<0.01$). Similarly, the community share in commenting is significantly lower (see Figure 2; Paired t-tests: community vs. team: $t(3)=-14.52, p<.01$; community vs. project: $t(3)=-5.91, p<.01$). In detail, the community involvement was 4-12% in work item creation and 2-9% in commenting. Team members dominate the work item creation (56-67%) and contribute the most comments (65-82%).

Community members are the biggest group of distinct active people in issue tracking. For two of the evaluated Jazz components (SC and P), the number of active community members is similar to the number of active project members. For the others, about 1.5 times as many community members are active as project members. When we summarize over all four evaluated Jazz components, the number of active community members is twice as large as the number of active project members, because project members are more likely to contribute work items and comments to several components. The number of active team members is much lower. In terms of shares,

community members are about 45-60% of the active issue tracking contributors, project members are about 28-50% and team members are between 4-10% (including summarized results of all 4 components).

The average community member contributes less comments (paired t-tests: community vs. team: $t(3)=-4.08, p=.0265$; community vs. project: $t(3)=-7.56, p<.01$) and work items (community vs. team: $t(3)=-3.31, p=.0454$; community vs. project: $t(3)=-18.19, p<.01$) compared to project and team members. Team members are the most active contributors, both in terms of created work items and in terms of commenting. They contribute between 25-155 times as many work items as community members, and between 7-22 times as many work items as project members. Team members contribute between 42-265 times as many comments as community members and between 10-50 times as many as project members. Table 1(d) shows the number of work items per month and active contributor, split by community, project and team members. Table 1(e) shows the comments.

4.2 Is there a similar degree of community involvement in Eclipse and Jazz?

The metric results for the selected projects of Eclipse are given in Table 2. The three evaluated Eclipse projects exhibit large differences in absolute issue tracking activity as well as in community issue tracking activity.

Table 2: Metrics calculated for Eclipse components

	Dec. '01 – Jun. '02			Dec. '01 – Jun. '03		
	PDE	JDT	CDT	PDE	JDT	CDT
Community	21.0	201.1	1.4	20.7	227.2	12.7
Project	54.9	205.4	38,4	41.9	128.2	15.0
Team	35.4	409.3	4.4	30.8	293.0	10.5

(a) Average number of created work items per month

	Dec. '01 – Jun. '02			Dec. '01 – Jun. '03		
	PDE	JDT	CDT	PDE	JDT	CDT
Community	69.3	649.6	6.0	60.1	681.3	38.3
Project	239.6	903.6	193.1	158.4	546.4	73.1
Team	527.4	5956.1	36.7	405.1	4102.3	82.4

(b) Average number of comments per month

	Dec. '01 – Jun. '02			Dec. '01 – Jun. '03		
	PDE	JDT	CDT	PDE	JDT	CDT
Community	41	469	6	164	1409	94
Project	85	147	15	126	252	23
Team	12	27	1	16	30	16

(c) Number of active contributors

	Dec. '01 – Jun. '02			Dec. '01 – Jun. '03		
	PDE	JDT	CDT	PDE	JDT	CDT
Community	0.51	0.43	0.24	0.13	0.16	0.14
Project	0.65	1.40	2.56	0.33	0.51	0.65
Team	2.95	15.16	4.43	1.92	9.77	0.66

(d) Average number of created work items and active contributor

	Dec. '01 – Jun. '02			Dec. '01 – Jun. '03		
	PDE	JDT	CDT	PDE	JDT	CDT
Community	1.7	1.4	1.0	0.4	0.5	0.4
Project	2.8	6.2	12.9	1.3	2.2	3.2
Team	43.9	220.6	36.7	25.3	136.7	5.2

(e) Average number of comments per month and active contributor

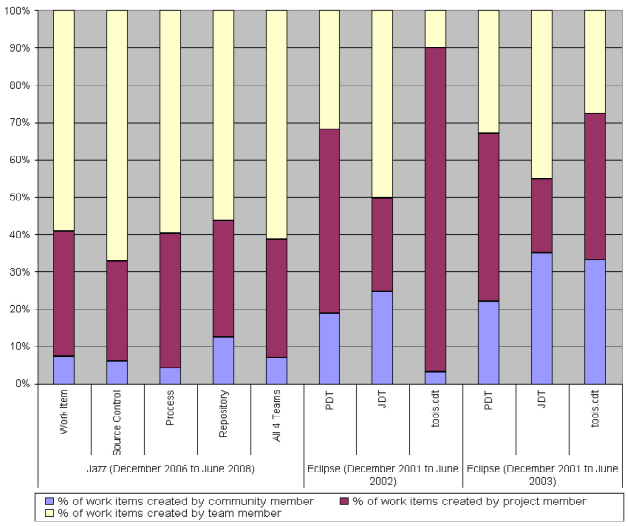

Figure 1: Distribution of reporters by community role.

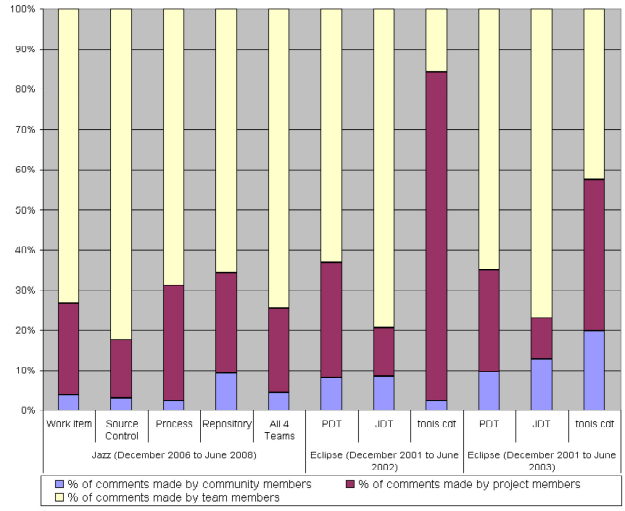

Figure 2. Distribution of comments by community role

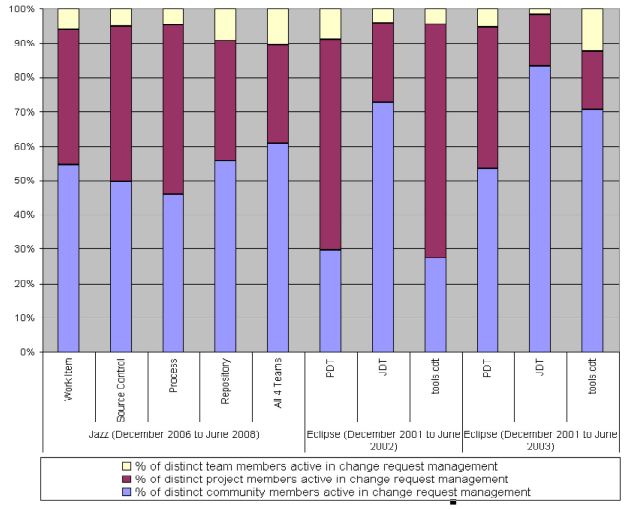

Figure 3. Distribution of people active by community role

Although the results for the distribution of issue tracking activity between team, project and community members seem to indicate that the community share in work item creation and commenting is lower than the share of team and project members (see Tables 2(a) and 2(b)), we could not detect a statistically significant effect. This might have been due to the low number of components and the high variance that stems from the results for CDT. The relative community involvement in issue tracking seems to increase when comparing the time interval until June 2002 to the time interval until June 2003 (see Figures 1 and 2). In detail, the community involvement increased from 18-25% to 22-35% in work item creation, and from 8% to 9-13% in commenting. However, these differences are not statistically significant. The rest of the work item creation is split fairly evenly between team members (25-50%) and project members (20-50%), whereas the team members contribute the most comments (60-80%).

A large number of community members contribute to issue tracking in Eclipse (see Table 2(c)), compared to a small number of team members and an intermediate number of project members. The community share of distinct people active in issue tracking increases from 25-73% for the time interval until June 2002 to 53-84% for the time interval until June 2003. This might be caused by a substantial increase of the Eclipse user base during this period.

We could not find any significant differences between the Eclipse and Jazz components with regards to the community share in commenting. For work item reporting, the community share was significantly lower in the Jazz components compared to the Eclipse components for Dec'01 – Jun'03, $t(2)=-5.15$, $p=0.0168$. However, there was no evidence of this effect when comparing the Jazz components to the Eclipse components for Dec'01-Jun'02, $t(2)=-1.22$, $p=.3388$.

Also, in Eclipse projects community members contribute relatively more work items than comments for Dec'01-Jun'03 ($t(2)=5.30$, $p=.03376$), whereas such a difference cannot be observed for the Eclipse components during the time period from Dec'01-Jun'02 ($t(2)-2.03$, $p=.1793$). Jazz community members also contribute relatively more work items than comments ($t(3)=5.91$, $p<.01$). The data for the Jazz components regarding user distribution are similar to the evaluated Eclipse components. Except for the Process component, the community is the largest group of active issue tracking users.

4.3 Are Work Items created by non-project members processed differently in the open commercial project Jazz?

We found that work items created by community members typically involve more communication than work items created by team members and they are resolved more slowly. Work items created by community members are also more likely to be unassigned after 7 days.

There is more communication on work items created by project and community members compared to work items created by team members (see Table 3(a)). We did a Wilcoxon rank sum test with continuity correction for each pair of different reporter groups in each Jazz component. We found significant differences ($p < 0.01$) in the community/team and project/team pairs.

No significant differences between the different reporter groups have been detected for the percentage of reopened work items.

Table 3: Work item processing in Jazz components

	WI	SC	P	R
Community	3.44	4.49	4.73	5.13
Project	3.42	4.61	3.02	5.70
Team	2.42	3.70	2.11	3.43

(a) Average number of comments for resolved, verified and closed work items divided by reporter and Jazz component

	WI	SC	P	R
Community	38.76	5.82	23.05	14.39
Project	6.09	3.43	3.34	5.88
Team	5.93	4.86	3.96	5.88

(b) Median age in days at resolution divided by reporter and Jazz component

	WI	SC	P	R
Community	16.58	6.78	32.31	12.11
Project	9.79	7.08	19.49	11.17
Team	6.92	2.55	21.35	8.39

(c) % of work items that are unassigned 7 days after creation

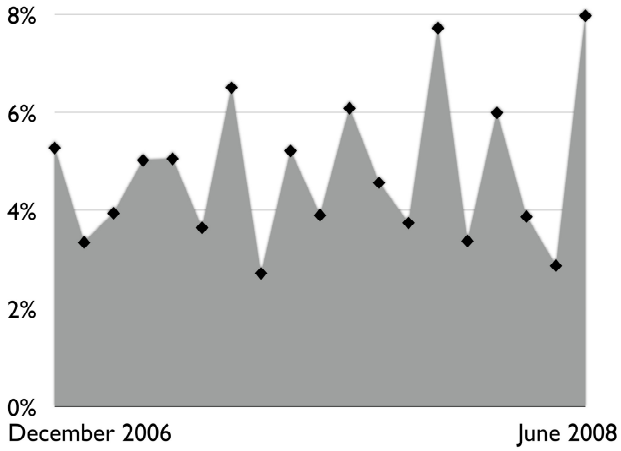

Figure 4: Percentage of comments from community members in the 4 Jazz components over time.

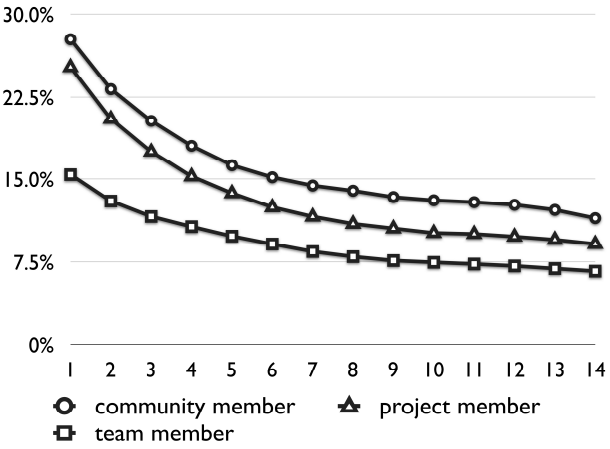

Figure 5: Percentage of unassigned work items in the 4 Jazz components for the first 14 days after creation

We applied a chi-square test to the different reporter groups and reopened vs. not-reopened work items. Work items reported by community members in Jazz were resolved slower than other work items (see Table 3(b)). We did a Wilcoxon rank sum test with continuity correction for each pair of different reporter groups in each Jazz component, as well as on the community reported work items vs. the other work items. There are significant differences between the work items reported by community members and both the work items created by team and project members ($p < 0.02$).

The work items created by community members are more likely to be unassigned 7 days after they have been created (see Table 3(c)). We divided the work items based on their assignment status after 7 days into assigned and unassigned, and by their reporter groups (community, project, team). We then applied a chi-square test to the number of work items in the 6 buckets. This showed that the distribution was not due to chance ($p < 0.01$).

5. DISCUSSION

This paper has two main contributions. First, we investigated community involvement in issue tracking in open commercial development. Second, we explored the differences in work item processing in open commercial development between non-developer and developer work items. The following discussion elaborates those contributions and their implications.

Community involvement in issue tracking. Both the Jazz and Eclipse projects have been able to attract considerable participation of the community in issue tracking. However, the community work item reporting in the Jazz components is significantly lower than in the Eclipse components for Dec'01-Jun'03. The difference in work item creation can be explained by the fact that there was no Jazz Release until June 2008, and thus post-release defects that are found by community members are missing. This hypothesis is also supported by the fact that such a difference between Jazz and Eclipse could not be observed before the first Eclipse release.

For Eclipse, an increase of the relative community involvement in work item reporting and commenting could be observed when comparing the time periods from December 2001 until June 2002 and from December 2001 until June 2003. It is not clear if such an increase had also happened in Jazz after its first release. For the time between December 2006 and June 2008, an increase of community involvement in Jazz could not be observed (compare Figure 4). The community involvement could be influenced by how the team reacts on user contributions. Hence, we analyzed how work items contributed by community members are processed.

Differences in work item processing depended on reporter group. We found that the community status of the reporter influences how much communication there is on work items, how fast they are resolved, and how long they stay unassigned.

Work items created by community and project members have significantly more comments on the average than work items reported by team members. One hypothesis that could explain this is that team members create small work items for themselves, which do not have any discussion. To further investigate this idea, we measured the percentage of work items that were fixed without any comments, split by the different author groups. We found that the number of work items that were resolved without comments was between 6 and 14 percent higher for work items created by team members compared to work items created by community members. However, when we restricted the measurement of the number of comments to work items with the resolution Fixed and with at least one comment, the differences in number of comments still prevailed. Compared to Eclipse, the average number of comments per work item in Jazz is much lower. In the evaluated Eclipse components, the average ranges between 6.4 and 10.2 for the same reporter groups.

We also found that work items created by community members are more likely to be unassigned after 7 days. This could be explained by the awareness of the project and team members who could work on a work item. We further investigated how the percentage of unassigned work items changes over the first 14 days in the evaluated Jazz components (see Figure 5). The work items created by project and community members are more likely to be unassigned in the beginning, and although the difference between these and work items created by team members gets lower over time, the work items created by team members are always more likely to be assigned. This supports the hypotheses that this difference is due to whether the work items are assigned on creation or not, which is more likely for work items created by team members because they are aware who could work on a specific work item.

We also found significant differences in the age at resolution between work items created by community members and work items created by team and project members. These could be due to differences in granularity of the work items. However, we also found no significant differences in the reopening rate between community and other work items. This measurement was not applicable for Eclipse, because work items cannot be unassigned in the Eclipse Bugzilla. A possible alternative, using the work item state "assigned" in Eclipse, is not comparable because the owner has to accept a bug manually, which is different from Jazz.

6. LIMITATIONS

Eclipse and Jazz are two different projects and thus there are limitations in their comparison. We selected the components and time periods in a way that many of those problems such as major differences in team size, component age, and type of software system were mitigated. Moreover, threats to validity can be caused by different usage schemes of the issue tracking system. However, a considerable number of Jazz developers were former Eclipse developers. Thus, we can assume at least some common understanding of the usage scheme.

We looked at the early stages of IDE development projects. Our results apply to this specific domain and it is difficult to generalize them beyond it. The work item data for Eclipse from December 2001 to June 2003 was compared to work item data for Jazz from December 2006 to June 2008. Eclipse became open source with the release of version 1.0. For Jazz, the first release was IBM Rational Team Concert in June 2008. The public availability of an Eclipse release could have influenced the likelihood of finding defects by the community, which could have impacted the community involvement in issue tracking.

The assignment of people to the community, project and team groups is not exact. For Jazz, we distinguished between team, project and community members based on their current team assignment. This does not necessarily reflect past team or project assignment, as people could have switched between teams or could have left the project. For Eclipse, we used the IBM and OTI email addresses to distinguish between project and community members, because a complete reconstruction of the

project members was not feasible. This could mean that we included people who were not active in the Eclipse project, and it could also mean that we excluded project members who were not using an IBM or OTI email address.

7. CONCLUSION

While both the Eclipse and Jazz projects have attracted significant contributions from the community, our results indicate that there is a lower community work item reporting in Jazz compared to Eclipse. Work items created by community members are processed differently from work items created by team members in Jazz.

Possible future work could investigate these issues in detail, e.g. using interviews or observational field studies. Looking at the quality of work items and at other community involvement artifacts such as newsgroups might be worthwhile. Studying these issues could shed further light on how user communities engage in issue tracking.

8. ACKNOWLEDGMENTS

We would like to thank the members of the CHISEL group and the reviewers for all the helpful comments and editing suggestions. This work was funded by IBM CAS Canada.

9. REFERENCES

[1] R. Bagozzi and U. Dholakia. Open source software user communities: A study of participation in linux user groups. *Management Science*, 52(7), 2006.

[2] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug report? In *SIGSOFT '08/FSE-16*, pp. 308–318, ACM, NY, 2008.

[3] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu. Latent social structure in open source projects. In *SIGSOFT '08/FSE-16*, pp. 24–35, ACM, New York, 2008.

[4] M. Burke et al. Eclipse technology exchange workshop. In *OOPSLA '06*, ACM, New York, 2006.

[5] Li-Te Cheng, Alessandro Orso, and Martin P. Robillard, editors. *Proc. of the 2007 OOPSLA workshop on Eclipse Technology eXchange, ETX 2007, Montreal, Canada, 2007*.

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

[7] Randall Frost. Jazz and the Eclipse Way of Collaboration. *IEEE Software*, 24(06):114–117, 2007.

[8] E. Gamma. Agile, open source, distributed, and on-time – inside the eclipse development process. Keynote Talk, ICSE 2005, St.Louis, Missouri, 2005.

[9] L. Grammel, H. Schackmann, and H. Lichter. Bugzillametrics: an adaptable tool for evaluating metric specifications on change requests. In *IWPSE '07*, pages 35–38, ACM, New York, 2007.

[10] David G. Hendry. Public participation in proprietary software development through user roles and discourse. *International Journal of Human-Computer Studies*, 7, 2008.

[11] I. Herraiz, D. Germán, J. González-Barahona, and G. Robles. Towards a simplification of the bug report form in eclipse. In Ahmed E. Hassan, Michele Lanza, and Michael W. Godfrey, editors, *MSR*, pages 145–148. ACM, 2008.

[12] P. Hooimeijer and W. Weimer. Modeling bug report quality. In *ASE '07*, pp. 34–43, ACM, New York, NY, USA, 2007.

[13] N. Iivari. Empowering the users? a critical textual analysis of the role of users in open source software development. *AI SOCIETY*, 23:511–528(18), July 2009.

[14] T. Kakimoto, Y. Kamei, M. Ohira, and K.-I. Matsumoto. Social network analysis on communications for knowledge collaboration in oss communities. In *Second intl. Workshop on Supporting Knowledge Collaboration in Software Development, Tokyo, Japan*, September 2006.

[15] A. Ko and P. Chilana. How power users help and hinder open bug reporting. In *CHI '10*, pages 1665–1674, ACM, New York, NY, USA, 2010.

[16] A. Mockus, R. Fielding, and J. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Trans. Softw. Eng. Methodol.*, 11(3):309–346, 2002.

[17] K. Nakakoji, Y. Yamamoto, Y. Nishinaka, K. Kishida, and Y. Ye. Evolution patterns of open-source software systems and communities. In *IWPSE '02*, pp. 76–85, ACM, New York, USA, 2002.

[18] E. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary (O'Reilly Linux)*. O'Reilly, October 1999.

[19] R. Sandusky. Software Problem Management as Information Management in a F/OSS Development Community. In *Proceedings of the First International Conference on Open Source Systems*, pages 44–49, 2005.

[20] H. Schackmann and H. Lichter. Comparison of process quality characteristics based on change request data. In *IWSM/Metrikon/Mensura*, volume 5338 of *Lecture Notes in Computer Science*, pages 127–140. Springer, 2008.

[21] V. Singh, M. Twidale, and D. Nichols. Users of open source software - how do they get help? *Hawaii International Conference on System Sciences*, 2009.

[22] D.W. van Liere. How Shallow is a Bug? Why Open Source Communities Shorten the Repair Time of Software Defects. *ICIS 2009 Proceedings*, 2009.

[23] A.I. Wasserman and E. Capra. Evaluating software engineering processes in commercial and community open source projects. *Emerging Trends in FLOSS Research and Development,* May 2007.

[24] M. Weiss, G. Moroiu, and P. Zhao. Evolution of open source communities. In *OSS*, volume 203 of *IFIP*, pp. 21–32. Springer, 2006.

[25] Y. Ye and K. Kishida. Toward an understanding of the motivation open source software developers. In *ICSE '03*, pp. 419–429, IEEE, Washington, DC, USA, 2003.