

A peer-reviewed version of this preprint was published in PeerJ on 18 July 2016.

[View the peer-reviewed version](https://peerj.com/articles/cs-73) (peerj.com/articles/cs-73), which is the preferred citable publication unless you specifically need to cite this preprint.

Destefanis G, Ortu M, Counsell S, Swift S, Marchesi M, Tonelli R. 2016. Software development: do good manners matter? PeerJ Computer Science 2:e73 <https://doi.org/10.7717/peerj-cs.73>

Software development: do good manners matter?

Giuseppe Destefanis, Marco Ortu, Steve Counsell, Michele Marchesi, Roberto Tonelli

A successful software project is the result of a complex process involving, above all, people. Developers are the key factors for the success of a software development process, not merely as executors of tasks, but as protagonists and core of the whole development process. This paper investigates social aspects among developers working on software projects developed with the support of Agile tools. We studied 22 open source software projects developed using the Agile board of the JIRA repository. All comments committed by developers involved in the projects were analyzed and we explored whether the politeness of comments affected the number of developers involved and the time required to fix any given issue. Our results showed that the level of politeness in the communication process among developers does have an effect on the time required to fix issues and, in the majority of the analysed projects, it had a positive correlation with attractiveness of the project to both active and potential developers. The more polite developers were, the less time it took to fix an issue. In the majority of the analysed cases, the more developers wanted to be part of a project, the more they were willing to continue working on the project over time.

1 Software Development: Do Good Manners 2 Matter?

3 Giuseppe Destefanis¹, Marco Ortu², Steve Counsell³, Michele Marchesi⁴,
4 and Roberto Tonelli⁵

5 ¹Brunel University, London, giuseppe.destefanis@brunel.ac.uk

6 ²DIEE, Cagliari, marco.ortu@diee.unica.it

7 ³Brunel University, London, steve.counsell@brunel.ac.uk

8 ⁴DIEE, Cagliari, michele@diee.unica.it

9 ⁵DIEE, Cagliari, roberto.tonelli@diee.unica.it

10 ABSTRACT

11 A successful software project is the result of a complex process involving, above all, people. Developers are the key factors for the success of a software development process, not merely as executors of tasks, but as protagonists and core of the whole development process. This paper investigates social aspects among developers working on software projects developed with the support of Agile tools. We studied 22 open source software projects developed using the Agile board of the JIRA repository. All comments committed by developers involved in the projects were analyzed and we explored whether the politeness of comments affected the number of developers involved and the time required to fix any given issue. Our results showed that the level of politeness in the communication process among developers *does* have an effect on the time required to fix issues and, in the majority of the analysed projects, it had a positive correlation with attractiveness of the project to both active and potential developers. The more polite developers were, the less time it took to fix an issue. In the majority of the analysed cases, the more developers wanted to be part of a project, the more they were willing to continue working on the project over time.

12 Keywords: social and human aspects, politeness, mining software repositories, issue fixing time, software development

13 1 INTRODUCTION

14 High-level software development is a complex activity involving a range of people and activities; ignoring
15 human aspects in the software development process or managing them in an inappropriate way can,
16 potentially, have a huge impact on the software production process and team effectiveness. Increasingly,
17 researchers have tried to quantify and measure how social aspects affect software development. Bill Curtis
18 claimed that “the creation of a large software system must be analyzed as a behavioural process (Curtis
19 et al. (1988)).” Coordinating and structuring a development team is thus a vital activity for software
20 companies and team dynamics have a direct influence on group successfulness. Open-source development
21 usually involves developers that voluntarily participate in a project by contributing with code-development.
22 In many senses, the management of such developers is more complex than the management of a team
23 within a company - developers are not in the same place at the same time and coordination therefore
24 becomes more difficult. Additionally, the absence of face-to-face communication mandates the use of
25 alternative technologies such as mailing lists, electronic boards or issue tracking systems. In this context,
26 being rude or aggressive when writing a comment or replying to a contributor can affect the cohesion of
27 the group, its membership and the successfulness of a project. On the other hand, a respectful environment
28 provides an incentive for new contributors to join the project and could significantly extend the lifetime
29 and usefulness of a project to the community.

30 According to VersionOne (2013): “more people are recognising that agile development is beneficial
31 to business, with an 11% increase over the last 2 years in the number of people who claim that agile
32 helps organisations complete projects faster”. A main priority reported by users was to accelerate time
33 to market, manage changing priorities more easily and better align IT and business objectives. Agile

34 project management tools and Kanban boards experienced the largest growth in popularity of all agile tool
35 categories, with use or planned use increasing by 6%. One of the top five ranked tools was Atlassian JIRA¹,
36 with an 87% recommendation rate. Agile boards represent the central aspect of communication in the
37 Agile philosophy. According to Perry (2008) “the task board is one of the most important radiators used
38 by an agile team to track their progress.” The JIRA board is a good solution for bridging the gap between
39 open-source software development and the Agile world. It is the view of many that agile development
40 requires a physical aspect, i.e. developers working together in the same room or building, or at the same
41 desk; the pair programming paradigm, for example, requires at least two people working simultaneously
42 on the same piece of code. By using tools such as the JIRA board it is possible to use an agile board for
43 development of a project by developers in different physical places. Working remotely, in different time
44 zones and with different time schedules, with developers from around the world, requires coordination
45 and communication. When a new developer joins a development team, the better the communication
46 process works, the faster the new developer can become productive and the learning curve reduced. The
47 notion of an agile board therefore places emphasis on the know-how and shared-knowledge of a project
48 being easily accessible for the development team throughout the development process. Fast releases,
49 continuous integration and testing activities are directly connected to the knowledge of the system under
50 development. The potential for agile boards to simplify development across geographically disparate
51 areas is in this sense relatively clear. In a similar vein, the social and human aspects of the development
52 process are becoming more and more important. The Google work style has become a model for many
53 software start-ups - a pleasant work environment is important and affects the productivity of employees.
54 One important contributor to a healthy work environment is that each employee is considerate and polite
55 towards their fellow employees. More specifically “Politeness is the practical application of good manners
56 or etiquette. It is a culturally defined phenomenon and what is considered polite in one culture can
57 sometimes be quite rude or simply eccentric in another cultural context. The goal of politeness is to make
58 all of the parties relaxed and comfortable with one another.”² The last part of this definition is what we
59 consider in our analysis. In this specific work, we do not take different cultures into account (although
60 developers involved in a specific project could be from all around the world); we focus on the politeness
61 of the comment-messages written by the developers. The research aims to show how project management
62 tools such as agile boards can directly affect the productivity of a software development team and the
63 health of a software project.

64 Our research focuses around the concepts developed by Yamashita et al. (2014) who introduced the
65 concepts of magnetism and stickiness for a software project. A project is classified as *Magnetic* if it has
66 the ability to attract new developers over time. *Stickiness* is the ability of a project to keep its developers
67 over time. We measured these two metrics by considering the period of observation of one year. Figure 1
68 shows an example of the evaluation of Magnet and Sticky metrics. In this example, we were interested in
69 calculating the value of Magnetism and Stickiness for 2011. From 2010 to 2012, we had a total of 10
70 *active*³ developers. In 2011, there were 7 active developers and 2 of them (highlighted with black heads)
71 were new. Only 3 (highlighted with grey heads) of the 7 active developers in 2011 were also active in
72 2012. We can then calculate the Magnetism and Stickiness as follows:

- 73 • *Magnetism* is the fraction of new active developers during the observed time interval, in our example
74 $2/10$ (*dev_6* and *dev_7* were active in 2011 but not in 2010).
- 75 • *Stickiness* is the fraction of active developers that were also active during next time interval, in our
76 example $3/7$ (*dev_1*, *dev_2*, *dev_3* were active in 2011 and in 2012).

77 We considered 22 open source projects from one of the largest datasets of issues reports openly
78 available (Ortu et al. (2015d)). This paper aims to answer the following research questions:

- 79 • **Does politeness among developers affect issues fixing time?**
80 Issue fixing time for polite issues was found to be faster than issue fixing time for impolite issues
81 for 17 out of 22 analysed projects.

¹<https://www.atlassian.com/software/jira>

²en.wikipedia.org/wiki/Politeness

³We consider active all developers that posted/commented/resolved/modified an issue during the observed time (from dev_1 to dev_10)

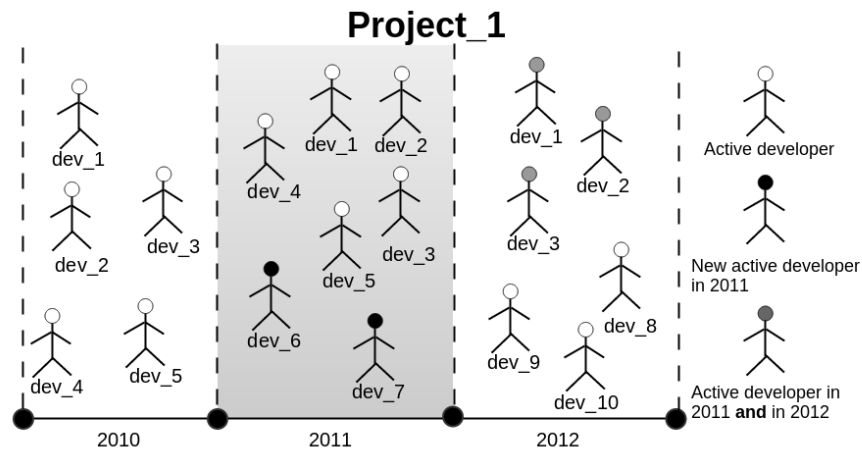


Figure 1. Example of Magnet and Sticky metrics computation in 2011

- 82 • **Does politeness among developers affect the attractiveness of a project?**
 83 We concluded that politeness is positively correlated with Magnetism and Stickiness metrics in
 84 subsequent years for the systems in our corpus.
- 85 • **Does politeness among developers vary over time?**
 86 Average politeness does vary over time and in some cases it changes from negative values (impolite)
 87 to positive (polite) from two consecutive observation intervals. Regarding average politeness over
 88 individual work days, we did not find great differences.
- 89 • **How does politeness vary with respect to JIRA maintenance types and issue priorities?**
 90 Comments related to issues with maintenance Bug, priority Minor and Trivial, tend to have a higher
 91 percentage of impolite comments. Issues with maintenance New Feature, priority Blocker and
 92 Critical tend to have a higher percentage of polite comments.

93 This paper is an extended version of earlier work by the same authors (Ortu et al. (2015b)). We added
 94 8 new systems to the original corpus analyzed in Ortu et al. (2015b), and two new research question
 95 (RQ3 and RQ4). The remainder of this paper is structured as follows: In the next section, we provide
 96 related work. Section 3 describes the dataset used for this study and our approach/rationale to evaluate
 97 the politeness of comments posted by developers. In Section 4, we present the results and elaborates on
 98 the research questions we address. Section 5 discusses the threats to validity. Finally, we summarize the
 99 study findings and present plans for future work in Section 6.

100 2 RELATED WORK

101 A growing body of literature has investigated the importance and the influence of human and social
 102 aspects, emotions and mood both in software engineering and software development. Research has
 103 focused on understanding how the human aspects of a technical discipline can affect final results (Brief
 104 and Weiss (2002), Capretz (2003), Cockburn and Highsmith (2001), Erez and Isen (2002), Kaluzniacky
 105 (2004)), and the effect of politeness (Novielli et al. (2014), Tan and Howard-Jones (2014), Winschiers and
 106 Paterson (2004), Tsay et al. (2014)).

107 Feldt et al. (2008) focused on personality as a relevant psychometric factor and presented results
 108 from an empirical study about correlations between personality and attitudes to software engineering
 109 processes and tools. The authors found that higher levels of the personality dimension “conscientiousness”
 110 correlated with attitudes towards work style, openness to changes and task preference.

111 IT companies are also becoming more conscious of social aspects. Ehlers (2015) evaluated the
 112 efforts of IT companies in acquiring software engineers by emphasizing socialness in their job ads. The
 113 research analyzed 75,000 jobs ads from the recruiting platform Indeed and about 2,800 job ads from
 114 StackoverflowCareers to investigate correlations between social factors and the employee satisfaction of a

115 work place. The findings showed that many companies advertise socialness explicitly. The Manifesto for
116 Agile Development indicates that people and communications are more essential than procedures and
117 tools (Beck et al. (2001)).

118 Steinmacher et al. (2015) analyzed social barriers that obstructed first contributions of newcomers (new
119 developers joining an open-source project). The study indicated how impolite answers were considered as
120 a barrier by newcomers. These barriers were identified through a systematic literature review, responses
121 collected from open source project contributors and students contributing to open source projects.

122 Roberts et al. (2006) conducted a study which revealed how the different motivations of open-source
123 developers were interrelated, how these motivations influenced participation and how past performance
124 influenced subsequent motivations.

125 Rigby and Hassan (2007) analyzed, using a psychometrically-based linguistic analysis tool, the five
126 big personality traits of software developers in the Apache httpd server mailing list. The authors found
127 that the two developers that were responsible for the major Apache releases had similar personalities
128 and their personalities were different from other developers. Bazelli et al. (2013) analyzed questions and
129 answers on stackoverflow.com to determine the developer personality traits, using the Linguistic Inquiry
130 and Word Count (Pennebaker et al. (2001)). The authors found that the top reputed authors were more
131 extroverted and expressed less negative emotions than authors of down voted posts.

132 Tourani et al. (2014) evaluated the use of automatic sentiment analysis to identify distress or happiness
133 in a team of developers. They extracted sentiment values from the mailing lists of two mature projects of
134 the Apache software foundation, considering developers and users. The authors found that an automatic
135 sentiment analysis tool obtained low precision on email messages (due to long size of the analyzed text)
136 and that users and developers express positive and negative sentiment on mailing lists. Murgia et al.
137 (2014b) analyzed whether issue reports carried any emotional information about software development.
138 The authors found that issue reports contain emotions regarding design choices, maintenance activity or
139 colleagues. Gómez et al. (2012) performed an experiment to evaluate whether the level of extraversion
140 in a team influenced the final quality of the software products obtained and the satisfaction perceived
141 while this work was being carried out. Results indicated that when forming work teams, project managers
142 should carry out a personality test in order to balance the amount of extraverted team members with
143 those who are not extraverted. This would permit the team members to feel satisfied with the work
144 carried out by the team without reducing the quality of the software products developed. Acuña et al.
145 (2008), performed empirical research examining the work climate within software development teams.
146 The authors attempted to understand if team climate (defined as the shared perceptions of team work
147 procedures and practices) bore any relation to software product quality. They found that high team vision
148 preferences and high participative safety perceptions of the team were significantly related to better
149 software. In a study conducted by Fagerholm et al. (2014), it was shown that software teams engaged in
150 a constant cycle of interpreting their performance. Thus, enhancing performance experiences requires
151 integration of communication, team spirit and team identity into the development process.

152 **3 EXPERIMENTAL SETUP**

153 **3.1 Dataset**

154 We built our dataset collecting data from one of the JIRA⁴ largest public dataset available (Ortu et al.
155 (2015d)). An Issue Tracking System (ITS) is a repository used by software developers to support the
156 software development process. It supports corrective maintenance activity like Bug Tracking systems,
157 along with other types of maintenance requests. We mined the dataset (Ortu et al. (2015d)) collecting
158 issues from October 2002 to December 2013. To create our corpus, we selected projects for which the
159 JIRA Agile board contained a significant amount of activity (e.g., projects with the highest number of
160 comments). Table 1 shows the corpus of 22 projects selected for our analysis, highlighting the number of
161 comments recorded for each project and the number of developers involved.

162 **3.2 Comment Politeness**

163 Danescu-Niculescu-Mizil et al. (2013) proposed a machine learning approach for evaluating the politeness
164 of a request posted in two different web applications: Wikipedia⁵ and Stackoverflow⁶. Stackoverflow is

⁴<https://www.atlassian.com/software/jira>

⁵https://en.wikipedia.org/wiki/Main_Page

⁶<http://stackoverflow.com>

Project	# of comments	# of developers
HBase	91016	951
Hadoop Common	61958	1243
Derby	52668	675
Lucene Core	50152	1107
Hadoop HDFS	42208	757
Cassandra	41966	1177
Solr	41695	1590
Hive	39002	850
Hadoop Map/Reduce	34793	875
Harmony	28619	316
OFBiz	25694	578
Infrastructure	25439	1362
Camel	24109	908
ZooKeeper	16672	495
GeoServer	17424	705
Geronimo	18017	499
Groovy	18186	1305
Hibernate ORM	23575	4037
JBoss	23035	453
JRuby	22233	1523
Pig	21662	549
Wicket	17449	1243
Tot	737572	18144

Table 1. Selected Projects Statistics

well known in the software engineering field and is largely used by software practitioners (Rekha and Venkatapathy (2015), Choi et al. (2015), Rosen and Shihab (2015)); hence, the model that authors used in Danescu-Niculescu-Mizil et al. (2013) was suitable for our domain based on Jira issues, where developers post and discuss technical aspects of issues; the authors provide a Web application⁷ and a library version of their tool. Given some text, the tool calculates the politeness of its sentences, providing one of two possible labels: *polite* or *impolite*. Along with the politeness label, the tool provides a level of confidence related to the probability of a comment being labeled as *polite* or *impolite*. We thus considered comments whose level of confidence was less than 0.5 as neutral (namely, the text did not convey either politeness or impoliteness). Tables 2 and 3 show some examples of polite and impolite comments as classified by the tool⁸, respectively.

We evaluated the average politeness *per* month considering all comments posted in a certain month. For each comment, we assigned a value according to the following rules:

- Value of +1 for those comments marked as polite by the tool (confidence level > 0.5);
- Value of 0 for those comments marked as neutral (confidence level < 0.5);
- Value of -1 for those comments marked as impolite (confidence level > 0.5);

Finally, we averaged the assigned values for a certain month. In total, we analyzed the politeness of about 500K comments.

3.3 Issue Politeness

We inferred the politeness of issues from the knowledge of comments politeness and grouped issues together as follows:

- by dividing comments into two sets: polite and impolite, ignoring neutral comments;

⁷<http://www.mpi-sws.org/cristian/Politeness.html>

⁸User's names are reported as `<dev_name.a>` for the sake of privacy.

Comment	Confidence Level
Hey <dev_name_a>, Would you be interested in contributing a fix and a test case for this as well? Thanks, <dev_name_b>	0.7236
<dev_name>, can you open a new JIRA for those suggestions? I'll be happy to review.	0.919
<dev_name>, the latest patch isn't applying cleanly to trunk – could you resubmit it please? Thanks.	0.806
<dev_name>, Since you can reproduce, do you still want the logs? I think I still have them if needed.	0.803

Table 2. Examples of polite comments.

Comment	Confidence Level
Why are you cloning tickets? Don't do that.	0.816
shouldn't it check for existence of tarball even before it tries to allocate and error out ???	0.701
<dev_name_a>, why no unit test? <dev_name_b>, why didn't you wait for +1 from Hudson???	0.942
> this isn't the forum to clarify Why not? The question is whether this is redundant with Cascading, so comparisons are certainly relevant, no?	0.950

Table 3. Examples of impolite comments.

- 186 • by dividing issues into two sets: polite issues, commented only with polite comments and impolite
187 issues, commented only by impolite comments.
- 188 • by ignoring issues with both polite and impolite comments and ignoring issues with neutral
189 comments.

190 For each issue, we evaluated the politeness expressed in its comments (removing neutral comments as
191 discussed in Section 3.2) and then divided issues in two groups: polite issues containing polite comments
192 and impolite issues containing impolite comments. For each of this two groups of issues, the issue fixing
193 time as the difference between resolution and creation time was evaluated.

194 4 RESULTS AND DISCUSSION

195 4.1 Does politeness among developers affect issues fixing time?

196 **Motivation.** Murgia et al. (2014a) demonstrated the influence of maintenance type on issue fixing time,
197 while Zhang et al. (2013) developed a prediction model for bug fixing time for commercial software.
198 There are many factors able to influence issue fixing time; in this case, we were interested in finding out if
199 politeness, expressed by developers in comments, had an influence on that time.

200 **Approach.** To detect differences among the fixing time of polite and impolite issues, we used the
201 Wilcoxon rank sum test. This test is non-parametric and unpaired (Siegel (1956), Wilcoxon and Wilcox
202 (1964), Weiss et al. (2007)) and can be used without restriction on the statistical distribution of the sample
203 populations. The test is suitable for comparing differences among the averages or the medians of two
204 populations when their distributions are not gaussian. For the analysis, we used the one-sided Wilcoxon
205 rank sum test using the 5% significance level (i.e., $p\text{-value} < 0.05$) and we compared issue fixing time
206 between polite and impolite issues.

207 **Findings. Issue fixing time for polite issues is faster than issue fixing time for impolite issues for 17**
208 **out of 22 analysed projects.**

209 Figure 3 shows the box-plot of the issues fixing time for the two groups of issues considered (polite
210 and impolite) for all the projects analyzed. The issue fixing time is expressed in hours on a logarithmic
211 scale. It can be seen that the median of issues fixing time for polite issues is shorter than that for impolite
212 issues, for the majority of considered projects. Table 4 shows the Wilcoxon test results. The Test column
213 indicates whether the median of the first group (polite issues containing polite comments) is larger or
214 smaller than the second group (impolite issues containing impolite comments). Table 4 shows that for 17
215 of the 22 projects analysed, issue fixing time for polite issues is faster than that for issue fixing time for
216 impolite issues.

217 There are however two projects, *Camel* and *Hibernate ORM*, which behave differently. In these cases
218 issue fixing time for impolite issues is faster than the issue fixing time for polite issues. Furthermore,
219 for Infrastructure, Lucene Core and Cassandra projects, the *Test* failed, indicating that polite issue fixing
220 time is less than the impolite issue fixing time; however, with a $p\text{-value} > 0.05$ for these projects, we
221 cannot conclude that the two distributions are statistically different. We also considered the effect size, a
222 quantitative measure of the strength of a phenomenon, finding that is generally small with a maximum of
223 0.19 for Hadoop HDFS and a minimum of 0.007 for Infrastructure.

224 4.2 Does politeness among developers affect the attractiveness of a project?

225 **Motivation.** Magnetism and Stickiness are two metrics capable of describing the general health of a
226 project; namely, if a project is able to attract new developers and keep them over time, we can conclude
227 that the project is healthy. On the other hand, if a project is not magnetic and is not sticky we can conclude
228 that the project is losing developers and is not attracting new developers over time. Although there may be
229 many factors influencing magnetism and stickiness values, we were interested in analysing the correlation
230 between politeness expressed by developers in their comments and these two metrics.

231 **Approach.** To detect if there was a direct correlation between magnetism and stickiness of a project
232 and politeness, we considered an observation time of one year. During this time interval, we measured
233 magnetism, stickiness and percentage of comments classified by the tool as polite. Politeness in the
234 observed time could affect magnetism and stickiness in the subsequent observation time. This fact causes
235 a form of “reputation” for a given project. Developers can share an idea about other developers involved
236 in a project and newcomers can decide whether or not to join the project. To understand if a correlation

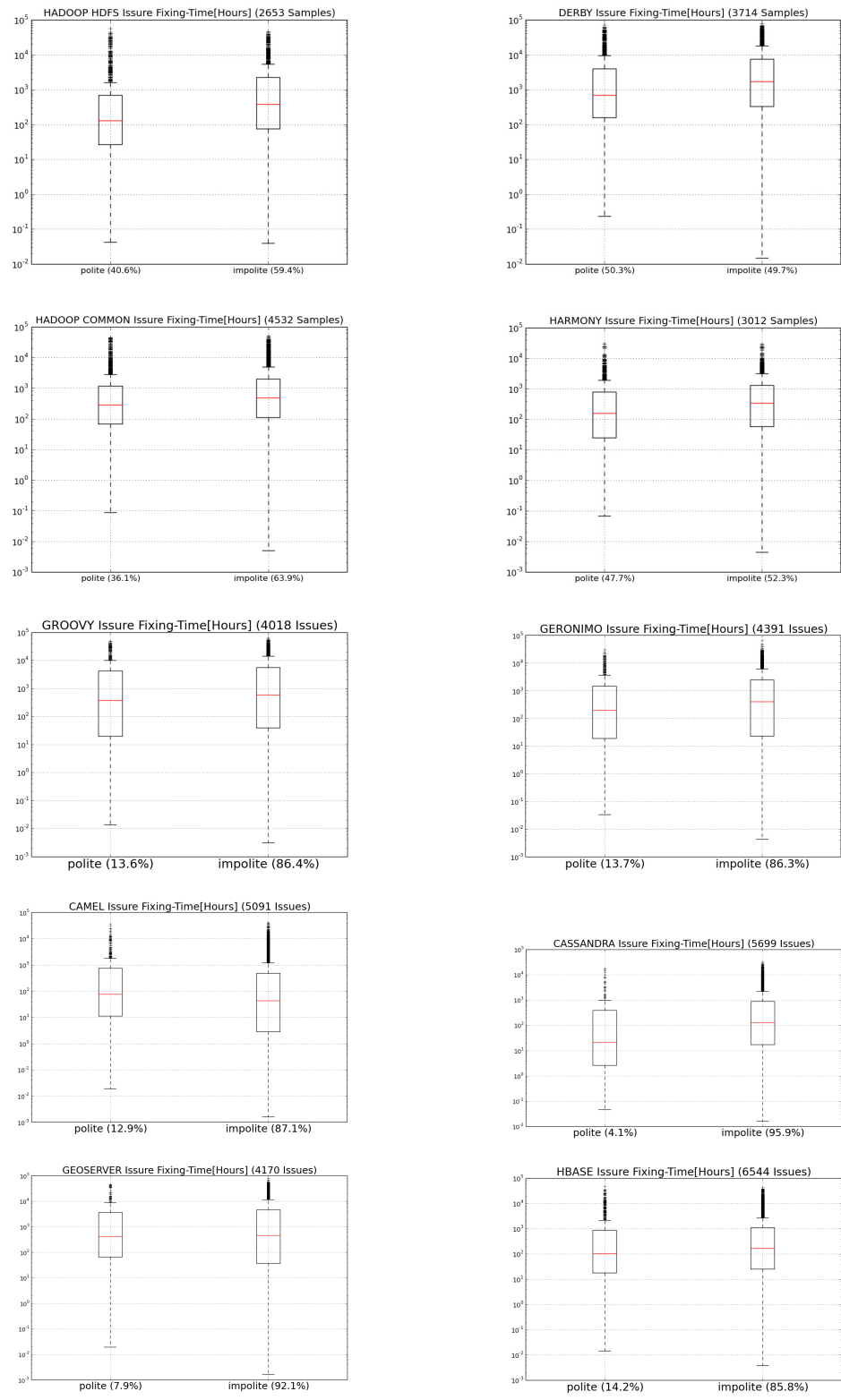


Figure 2. Box-plot of the fixing-time expressed in hours. The number in parentheses next to polite/impolite indicates the percentage of impolite and polite issues.

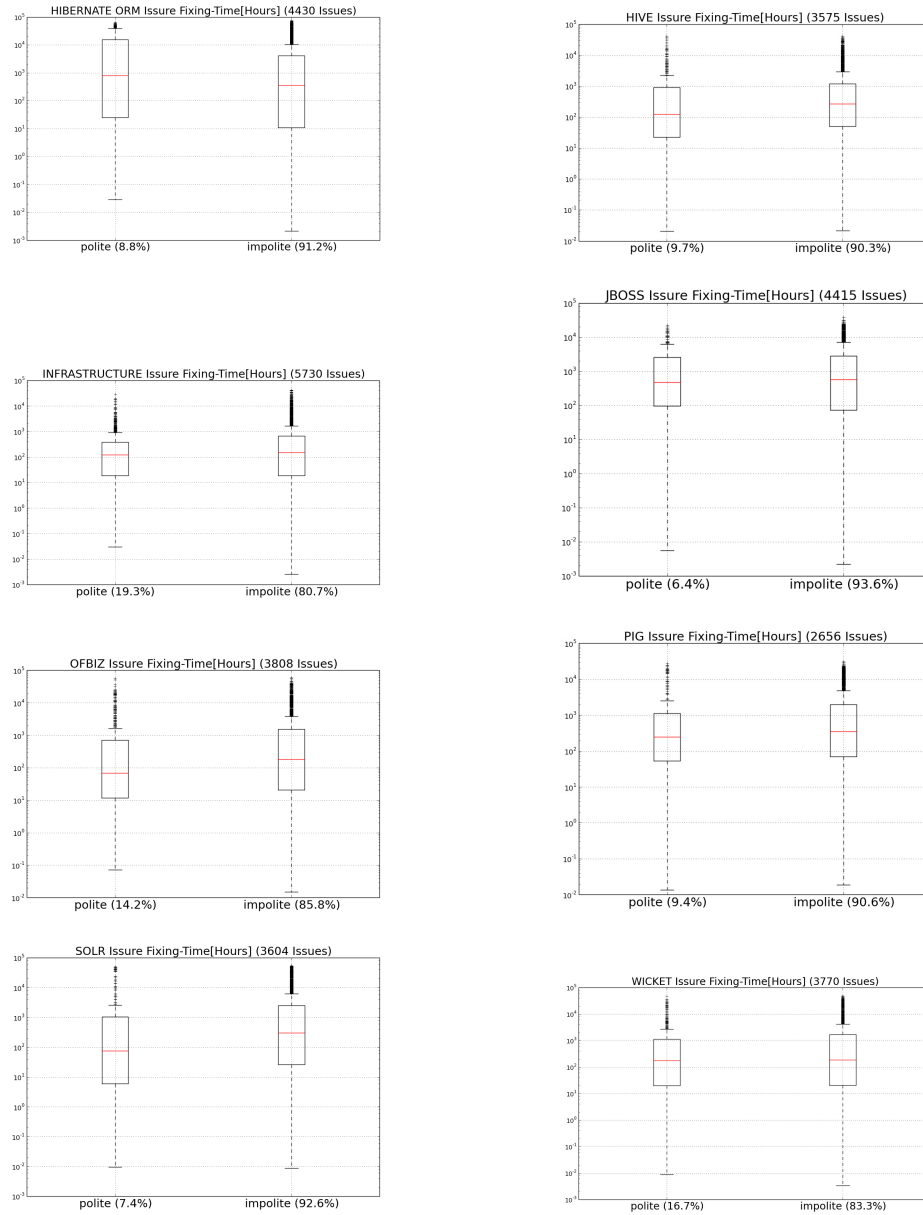


Figure 3. Box-plot of the fixing-time expressed in hours. The number in parentheses next to polite/impolite indicates the percentage of impolite and polite issues.

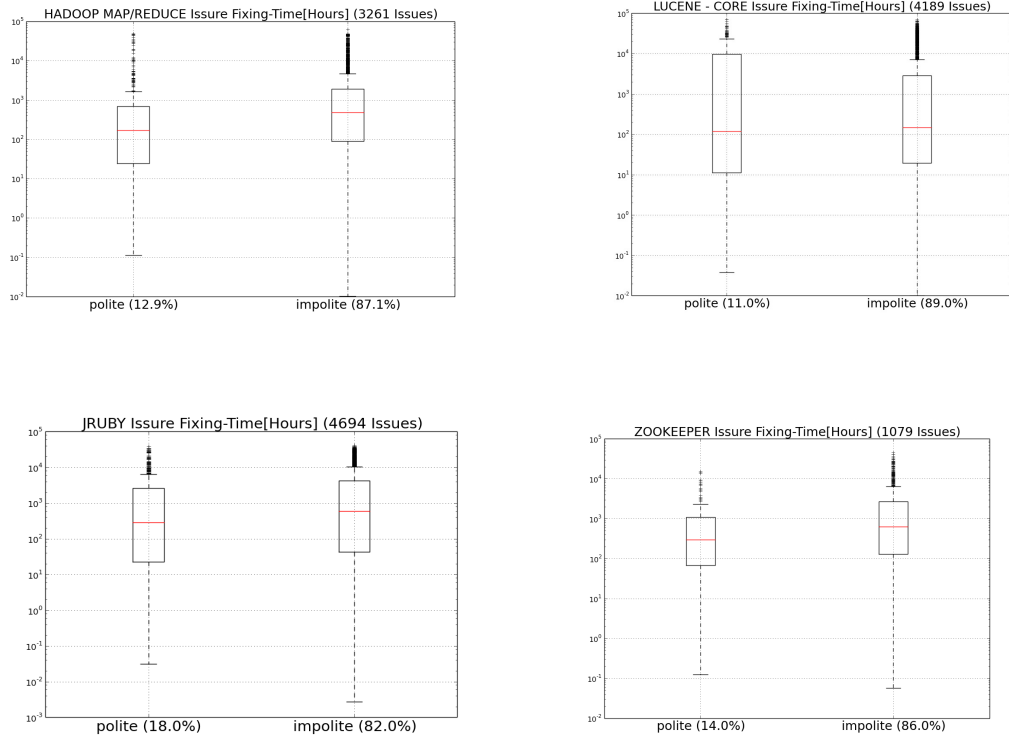


Figure 4. Box-plot of the fixing-time expressed in hours

Project	Test	p-value	effect size
ZooKeeper	lesser	***	0.14
Camel	greater	***	0.089
Infrastructure	lesser	0.67	0.007
OFBiz	lesser	***	0.15
Harmony	lesser	***	0.133
Hive	lesser	***	0.061
Solr	lesser	***	0.089
Cassandra	lesser	0.51	0.012
Hadoop HDFS	lesser	***	0.192
Lucene Core	lesser	0.492	0.01
Derby	lesser	***	0.15
Hadoop Common	lesser	***	0.11
HBase	lesser	***	0.144
Hadoop Map/Reduce	lesser	***	0.11
GeoServer	lesser	***	0.11
Geronimo	lesser	***	0.04
Groovy	lesser	***	0.04
Hibernate ORM	greater	***	0.07
JBoss	lesser	***	0.007
JRuby	lesser	***	0.06
Pig	lesser	***	0.03
Wicket	lesser	***	0.01

Table 4. Wilcoxon test results

237 exists between Magnet, Sticky and Politeness, we evaluated the cross-correlation coefficient.

238 **Findings. In the majority of projects, Magnet and Sticky were positively correlated with politeness.**

239 Table 5 shows the cross-correlation coefficient between the percentage of polite comments and magnetism
240 and stickiness during the observation period. The two columns represent the cross-correlation coefficient
241 between Magnetism and Stickiness and the percentage of politeness comments during the same observation
242 time. The correlation values are positive in all cases.

243 Considering the results obtained with cross correlation, we conclude that Politeness is positively
244 correlated, for the majority of the systems in our corpus with Magnetism and Stickiness metrics in
245 subsequent years.

Project	Cross-Correlation	
	Magnet	Sticky
HBase	0.581	0.667
Hadoop Common	0.848	0.641
Derby	0.126	0.240
Lucene Core	0.494	0.705
Hadoop HDFS	0.716	0.627
Cassandra	0.876	0.631
Solr	0.602	0.773
Hive	0.714	0.802
Hadoop Map/Reduce	0.631	0.697
Harmony	0.142	0.372
OFBiz	0.692	0.498
Infrastructure	0.479	0.610
Camel	0.120	0.293
ZooKeeper	0.319	0.497
GeoServer	0.7	0.5
Geronimo	0.43	0.42
Groovy	0.35	0.4
JBoss	0.54	0.52
Hibernate ORM	0.40	0.37
JRuby	0.58	0.46
Pig	0.49	0.45
Wicket	0.28	0.69

Table 5. Politeness Vs Magnet and Sticky Cross-Correlation Coefficient

246 4.3 Does politeness among developers vary over time?

247 **Motivation.** Politeness has an influence on the productivity of a team (Ortu et al. (2015b), Ortu et al.
248 (2015a), Ortu et al. (2015c)). Thus, it is interesting to understand if there are periods of time in which the
249 level of politeness decreases (potentially affecting the productivity of a team).

250 **Approach.** We calculated the level of politeness for any given issue and then plotted the average politeness
251 *per* month grouping issues *per* project. For politeness over week days (Figure 6), we grouped all the
252 comments *per* day of the week (checking the posting date); we then averaged politeness *per* day.

253 **Findings. Average politeness does vary over time and in some cases it changes from negative values**
254 **(impolite) to positive (polite) from two consecutive observation intervals. Regarding average polite-**
255 **ness over week days, we did not find great differences.** This fact could be related to the composition
256 of our corpus. We considered only open source systems, hence there are no strict deadlines or particular
257 busy days (such as Fridays, as suggested by Silwersky et al. Sliwerski et al. (2005)). Figure 5 shows the
258 the average politeness *per* month for Hadoop Common.

259 It is interesting to note that there are variations in the average politeness over time. This is by no
260 means a representation of a time dynamics, but simply the representation of random variation of average
261 politeness over time. In Hadoop HDFS for example, we see how the average politeness is negative (the
262 majority of comments are impolite) for some time interval and positive for some others. As we have

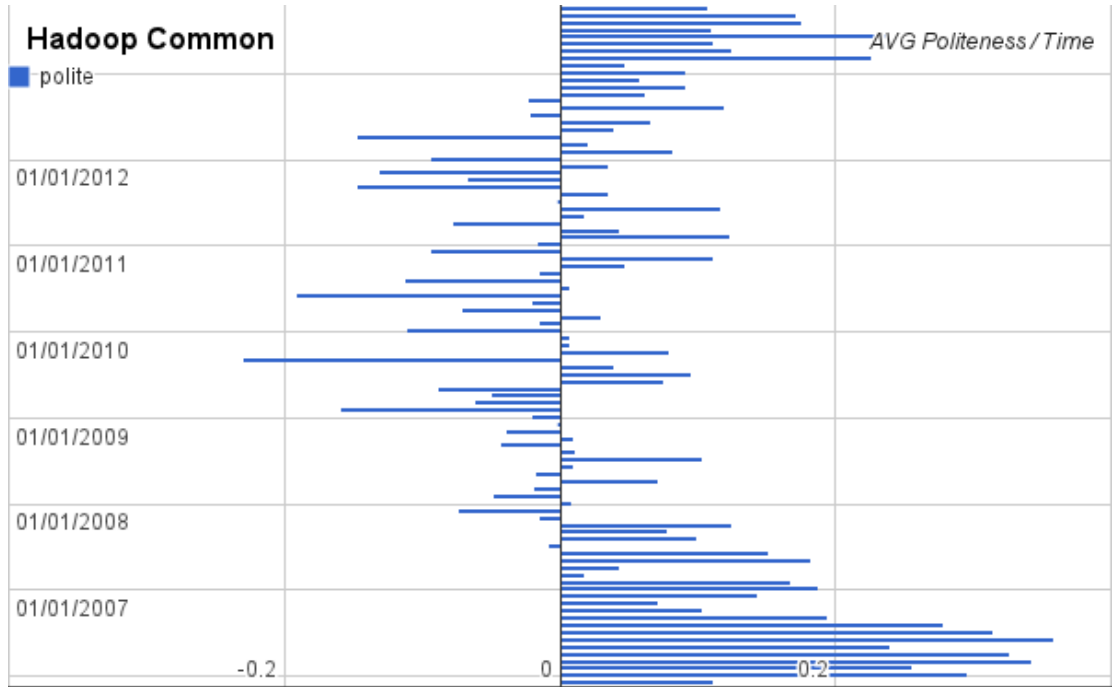


Figure 5. Average Politeness per month

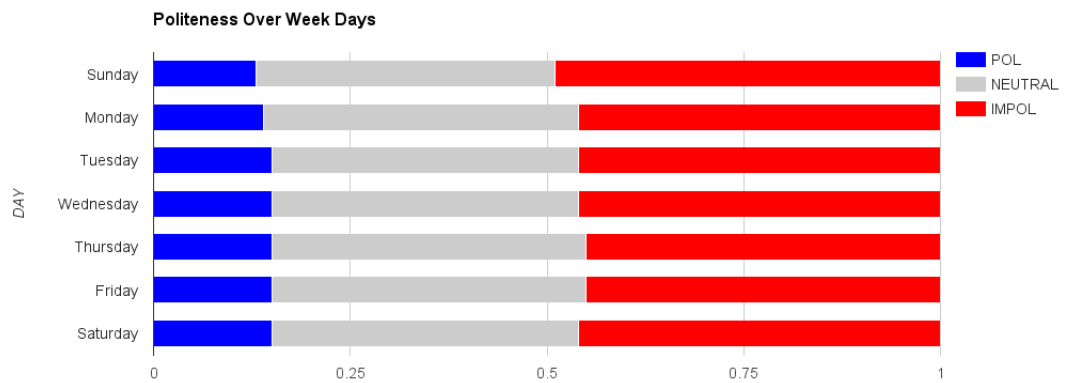


Figure 6. Average Politeness Over Week Days

263 seen, for those projects, polite issues are solved faster, so monitoring the average politeness over time
264 could help during software development. If there is a time period with a negative politeness, then the
265 community may take action to drive the average politeness back to positive values.

266 **4.4 How does politeness vary with respect to JIRA maintenance types and issue priori-** 267 **ties?**

268 **Motivation.** Understanding which typology of issue attracts more impolite comments could help man-
269 agers better understand the development process and take action to better manage the distribution of issues
270 within development teams. A classification of the type of issues, is provided on the JIRA wiki⁹. The
271 following list gives a brief introduction:

- 272 • Bug: this type of issue indicate a defect in the source code, such as logic errors, out-of-memory
273 errors, memory leaks and run-time errors. Any failure of the product to perform as expected and
274 any other unexpected or unwanted behaviour can be registered as type Bug.
- 275 • SubTask: this type of issue indicates that a task must be completed as an element of a larger and
276 more complex task. Subtask issues are useful for dividing a parent issue into a number of smaller
277 tasks, more manageable units that can be assigned and tracked separately.
- 278 • Task: this type of issue indicates a task that it is compulsory to complete.
- 279 • Improvement: this type of issue indicates an improvement or enhancement to an existing feature of
280 the system.
- 281 • New Feature: this type of issue indicates a new feature of the product yet to be developed.
- 282 • Wish: this type of issue is used to track general wishlist items, which could be classified as new
283 features or improvements for the system under development.
- 284 • Test: this type of issue can be used to track a new unit or integration test.
- 285 • New JIRA Project: this type of issue indicates the request for a new JIRA project to be set up.
- 286 • Brainstorming: this type of issue is more suitable for items in their early stage of formation not yet
287 mature enough to be labelled as a Task or New Feature. It provides a bucket where thoughts and
288 ideas from interested parties can be recorded as the discussion and exchange of ideas progresses.
289 Once a resolution is made, a Task can be created with all the details defined during the brainstorming
290 phase.
- 291 • Umbrella: this type of issue is an overarching type comprised of one or more sub-tasks.

292 Issues on JIRA are also classified, considering the level of priority, as Major, Minor, Blocker (e.g., an
293 issue which blocks development and/or testing work), Critical and Trivial.

294 **Approach.** To detect the level of politeness for each kind of issue, we grouped the issue comments for
295 type of maintenance and priority. We calculated for each group, the percentage of polite, impolite and
296 neutral comments.

297 **Findings. Comments related to issues with maintenance Bug, priority Minor and Trivial, tended**
298 **to have a higher percentage of impolite comments. Issues with maintenance New Feature, priority**
299 **Blocker and Critical, tend to have a higher percentage of polite comments.**

300 Figures 7 and 8 show the percentage of polite, impolite and neutral comments for each type of issue
301 maintenance and priority. Issues with maintenance Bug are related to defects and software failures. This
302 category presents the higher percentage of impolite comments. Issues with maintenance New Feature are
303 proposals made by developers and it is interesting to see that when proposing something new, developers
304 tend to be more polite.

305

⁹<https://cwiki.apache.org/confluence/display/FLUME/Classification+of+JIRA+Issues>

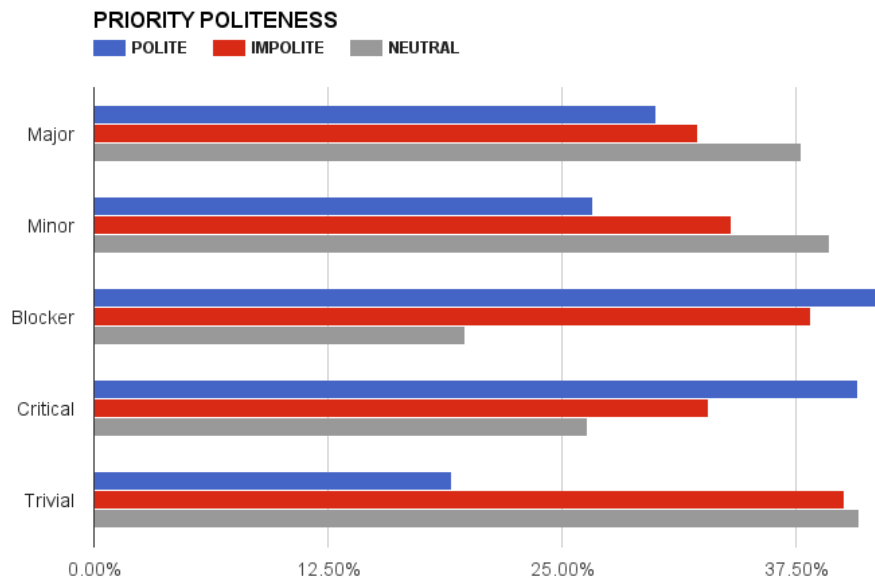


Figure 7. Priority

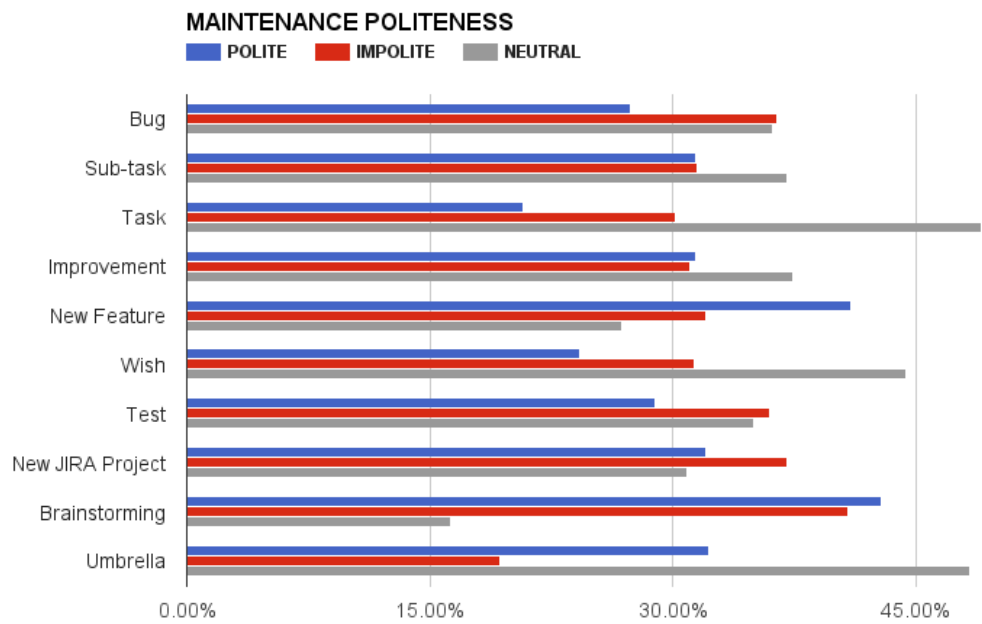


Figure 8. Maintenance

306 5 THREATS TO VALIDITY

307 This work is focused on sentences written by developers for developers. To illustrate the influence of
308 these comments, it is important to understand the language used by developers. We believe that the tool
309 used for measuring politeness Danescu-Niculescu-Mizil et al. (2013) is valid in the software engineering
310 domain, since the developers also used requests posted on Stackoverflow to train the classifier.

311 Threats to external validity correspond to the generalization of our results (Campbell and Stanley
312 (1963)). In this study, we analyzed comments from issue reports from 22 open source projects. Our results
313 cannot be representative of all environments or programming languages, we considered only open-source
314 systems and this could affect the generality of the study. Commercial software is usually developed using
315 different platforms and technologies, by developers with different knowledge and background, with strict
316 deadlines and cost limitation. Replication of this work on other open source systems and on commercial
317 projects are needed to confirm our findings. Also, the politeness tool can be subject to bias due the domain
318 used to train the machine learning classifier. We used a threshold of 0.5 for the confidence level to label a
319 comment as neutral, but other values of this threshold may lead to different results.

320 Threats to internal validity concern confounding factors that can influence the obtained results. Based
321 on empirical evidence, we suppose a causal relationship between the emotional state of developers and
322 what they write in issue reports (Pang and Lee (2008)). Since the main goal of developer communication is
323 the sharing of information, the consequence of removing or camouflaging emotions *may* make comments
324 less meaningful and cause misunderstanding. The comments used in this study were collected over an
325 extended period from developers unaware of being monitored. For this reason, we are confident that the
326 emotions we analyzed were genuine. We do not claim any causality between politeness and the issue
327 resolution time, but we built an explanatory model to understand the characteristics of issues with short
328 and long fixing time.

329 Threats to construct validity focus on how accurately the observations describe the phenomena
330 of interest. The detection of emotions from issue reports presents difficulties due to vagueness and
331 subjectivity. The politeness measures are approximated and cannot perfectly identify the precise context,
332 given the challenges of natural language and subtle phenomena like sarcasm.

333 6 CONCLUSIONS AND FUTURE WORK

334 Software engineers have been trying to measure software to gain quantitative insights into its properties
335 and quality since its inception. In this paper, we present the results about politeness and attractiveness on
336 22 open-source software projects developed using the Agile board of the JIRA repository. Our results
337 show that the level of politeness in the communication process among developers does have an effect
338 on both the time required to fix issues and the attractiveness of the project to both active and potential
339 developers. The more polite developers were, the less time it took to fix an issue. In the majority of cases,
340 the more the developers wanted to be part of project, the more they were willing to continue working on
341 the project over time. This work is a starting point and further research on a larger number of projects is
342 needed to validate our findings especially, considering proprietary software developed by companies and
343 different programming languages. The development of proprietary software follows different dynamics
344 (e.g., strict deadlines and given budget) and this fact could lead to different results. The takeaway message
345 is that politeness can only have positive effect on a project and on the development process.

347 REFERENCES

- 348 Acuña, S. T., Gómez, M., and Juristo, N. (2008). Towards understanding the relationship between
349 team climate and software quality—a quasi-experimental study. *Empirical software engineering*,
350 13(4):401–434.
- 351 Bazelli, B., Hindle, A., and Stroulia, E. (2013). On the personality traits of stackoverflow users. In
352 *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*, pages 460–463. IEEE.
- 353 Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J.,
354 Highsmith, J., Hunt, A., Jeffries, R., et al. (2001). Manifesto for agile software development.
- 355 Brief, A. P. and Weiss, H. M. (2002). Organizational behavior: Affect in the workplace. *Annual review of*
356 *psychology*, 53(1):279–307.

- 357 Campbell, D. T. and Stanley, J. C. (1963). *Experimental and quasi-experimental designs for generalized*
358 *causal inference*. Houghton Mifflin.
- 359 Capretz, L. F. (2003). Personality types in software engineering. *International Journal of Human-*
360 *Computer Studies*, 58(2):207–214.
- 361 Choi, E., Yoshida, N., Kula, R. G., and Inoue, K. (2015). What do practitioners ask about code clone? a
362 preliminary investigation of stack overflow. *9th Internl Work. on Soft. Clones, IWSC*.
- 363 Cockburn, A. and Highsmith, J. (2001). Agile software development: The people factor. *Computer*,
364 (11):131–133.
- 365 Curtis, B., Krasner, H., and Iscoe, N. (1988). A field study of the software design process for large
366 systems. *Communications of the ACM*, 31(11):1268–1287.
- 367 Danescu-Niculescu-Mizil, C., Sudhof, M., Jurafsky, D., Leskovec, J., and Potts, C. (2013). A computa-
368 tional approach to politeness with application to social factors. In *Proceedings of ACL*.
- 369 Ehlers, J. (2015). Socialness in the recruiting of software engineers. In *Proceedings of the 12th ACM*
370 *International Conference on Computing Frontiers*, page 33. ACM.
- 371 Erez, A. and Isen, A. M. (2002). The influence of positive affect on the components of expectancy
372 motivation. *Journal of Applied Psychology*, 87(6):1055.
- 373 Fagerholm, F., Ikonen, M., Kettunen, P., Münch, J., Roto, V., and Abrahamsson, P. (2014). How do
374 software developers experience team performance in lean and agile environments? In *Proceedings*
375 *of the 18th International Conference on Evaluation and Assessment in Software Engineering*, page 7.
376 ACM.
- 377 Feldt, R., Torkar, R., Angelis, L., and Samuelsson, M. (2008). Towards individualized software engi-
378 neering: empirical studies should collect psychometrics. In *Proceedings of the 2008 international*
379 *workshop on Cooperative and human aspects of software engineering*, pages 49–52. ACM.
- 380 Gómez, M. N., Acuña, S. T., Genero, M., and Cruz-Lemus, J. A. (2012). How does the extraversion of
381 software development teams influence team satisfaction and software quality?: A controlled experiment.
382 *International Journal of Human Capital and Information Technology Professionals (IJHCITP)*, 3(4):11–
383 24.
- 384 Kaluzniacky, E. (2004). *Managing psychological factors in information systems work: An orientation to*
385 *emotional intelligence*. IGI Global.
- 386 Murgia, A., Concas, G., Tonelli, R., Ortu, M., Demeyer, S., and Marchesi, M. (2014a). On the influence
387 of maintenance activity types on the issue resolution time. In *Proceedings of the 10th International*
388 *Conference on Predictive Models in Software Engineering*, pages 12–21. ACM.
- 389 Murgia, A., Tourani, P., Adams, B., and Ortu, M. (2014b). Do developers feel emotions? an exploratory
390 analysis of emotions in software artifacts. In *Proceedings of the 11th Working Conference on Mining*
391 *Software Repositories*, pages 262–271. ACM.
- 392 Novielli, N., Calefato, F., and Lanubile, F. (2014). Towards discovering the role of emotions in stack
393 overflow. In *Proceedings of the 6th International Workshop on Social Software Engineering*, pages
394 33–36. ACM.
- 395 Ortu, M., Adams, B., Destefanis, G., Tourani, P., Marchesi, M., and Tonelli, R. (2015a). Are bullies more
396 productive? empirical study of affectiveness vs. issue fixing time. In *Proceedings of the 12th Working*
397 *Conference on Mining Software Repositories, MSR 2015*.
- 398 Ortu, M., Destefanis, G., Kassab, M., Counsell, S., Marchesi, M., and Tonelli, R. (2015b). Would you
399 mind fixing this issue? In *Agile Processes, in Software Engineering, and Extreme Programming*, pages
400 129–140. Springer.
- 401 Ortu, M., Destefanis, G., Kassab, M., and Marchesi, M. (2015c). Measuring and understanding the
402 effectiveness of jira developers communities. In *Proceedings of the 6th International Workshop on*
403 *Emerging Trends in Software Metrics, WETSoM 2015*.
- 404 Ortu, M., Destefanis, G., Murgia, A., Marchesi, M., Tonelli, R., and Adams, B. (2015d). The jira
405 repository dataset: Understanding social aspects of software development. In *Proceedings of the 11th*
406 *International Conference on Predictive Models and Data Analytics in Software Engineering*, page 1.
407 ACM.
- 408 Pang, B. and Lee, L. (2008). Opinion Mining and Sentiment Analysis. *Foundations and Trends in*
409 *Information Retrieval*, 2(1-2):1–135.
- 410 Pennebaker, J. W., Francis, M. E., and Booth, R. J. (2001). Linguistic inquiry and word count: Liwc 2001.
411 *Mahway: Lawrence Erlbaum Associates*, 71:2001.

- 412 Perry, T. (2008). Drifting toward invisibility: The transition to the electronic task board. In *Agile, 2008.*
413 *AGILE'08. Conference*, pages 496–500. IEEE.
- 414 Rekha, V. S. and Venkatapathy, S. (2015). Understanding the usage of online forums as learning platforms.
415 *Procedia Computer Science*, 46:499–506.
- 416 Rigby, P. C. and Hassan, A. E. (2007). What can oss mailing lists tell us? a preliminary psychometric text
417 analysis of the apache developer mailing list. In *Proceedings of the Fourth International Workshop on*
418 *Mining Software Repositories*, page 23. IEEE Computer Society.
- 419 Roberts, J. A., Hann, I.-H., and Slaughter, S. A. (2006). Understanding the motivations, participation,
420 and performance of open source software developers: A longitudinal study of the apache projects.
421 *Management science*, 52(7):984–999.
- 422 Rosen, C. and Shihab, E. (2015). What are mobile developers asking about? a large scale study using
423 stack overflow. *Empirical Software Engineering*, pages 1–32.
- 424 Siegel, S. (1956). Nonparametric statistics for the behavioral sciences.
- 425 Sliwinski, J., Zimmermann, T., and Zeller, A. (2005). Don't program on fridays! how to locate fix-inducing
426 changes. In *Proceedings of the 7th Workshop Software Reengineering*.
- 427 Steinmacher, I., Conte, T. U., Gerosa, M., and Redmiles, D. (2015). Social barriers faced by newcomers
428 placing their first contribution in open source software projects. In *Proceedings of the 18th ACM*
429 *conference on Computer supported cooperative work & social computing*, pages 1–13.
- 430 Tan, S. and Howard-Jones, P. (2014). Rude or polite: Do personality and emotion in an artificial
431 pedagogical agent affect task performance? In *2014 Global Conference on Teaching and Learning*
432 *with Technology (CTLT 2014)*, page 41.
- 433 Tourani, P., Jiang, Y., and Adams, B. (2014). Monitoring sentiment in open source mailing lists —
434 exploratory study on the apache ecosystem. In *Proceedings of the 2014 Conference of the Center for*
435 *Advanced Studies on Collaborative Research (CASCON)*, Toronto, ON, Canada.
- 436 Tsay, J., Dabbish, L., and Herbsleb, J. (2014). Let's talk about it: Evaluating contributions through
437 discussion in github. *FSE. ACM*.
- 438 VersionOne (2013). 8th annual state of agile survey report.
- 439 Weiss, C., Premraj, R., Zimmermann, T., and Zeller, A. (2007). How long will it take to fix this bug?
440 In *Proceedings of the Fourth International Workshop on Mining Software Repositories*, page 1. IEEE
441 Computer Society.
- 442 Wilcoxon, F. and Wilcox, R. A. (1964). *Some rapid approximate statistical procedures*. Lederle
443 Laboratories.
- 444 Winschiers, H. and Paterson, B. (2004). Sustainable software development. In *Proceedings of the*
445 *2004 annual research conference of the South African institute of computer scientists and information*
446 *technologists on IT research in developing countries*, pages 274–278. South African Institute for
447 Computer Scientists and Information Technologists.
- 448 Yamashita, K., McIntosh, S., Kamei, Y., and Ubayashi, N. (2014). Magnet or sticky? an oss project-by-
449 project typology. In *MSR*, pages 344–347.
- 450 Zhang, H., Gong, L., and Versteeg, S. (2013). Predicting bug-fixing time: an empirical study of commercial
451 software projects. In *Proceedings of the 2013 International Conference on Software Engineering*, pages
452 1042–1051. IEEE Press.