# 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.

PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

# Software Development: Do Good Manners Matter?

**Giuseppe Destefanis[1], Marco Ortu[2], Steve Counsell[3], Michele Marchesi[4], and Roberto Tonelli[5]**

[1]**Brunel University, London, giuseppe.destefanis@brunel.ac.uk**
[2]**DIEE, Cagliari, marco.ortu@diee.unica.it**
[3]**Brunel University, London, steve.counsell@brunel.ac.uk**
[4]**DIEE, Cagliari, michele@diee.unica.it**
[5]**DIEE, Cagliari, roberto.tonelli@diee.unica.it**

## ABSTRACT

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.

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

## 1 INTRODUCTION

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

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

project management tools and Kanban boards experienced the largest growth in popularity of all agile tool categories, with use or planned use increasing by 6%. One of the top five ranked tools was Atlassian JIRA[1], with an 87% recommendation rate. Agile boards represent the central aspect of communication in the Agile philosophy. According to Perry (2008) "the task board is one of the most important radiators used by an agile team to track their progress." The JIRA board is a good solution for bridging the gap between open-source software development and the Agile world. It is the view of many that agile development requires a physical aspect, i.e. developers working together in the same room or building, or at the same desk; the pair programming paradigm, for example, requires at least two people working simultaneously on the same piece of code. By using tools such as the JIRA board it is possible to use an agile board for development of a project by developers in different physical places. Working remotely, in different time zones and with different time schedules, with developers from around the world, requires coordination and communication. When a new developer joins a development team, the better the communication process works, the faster the new developer can become productive and the learning curve reduced. The notion of an agile board therefore places emphasis on the know-how and shared-knowledge of a project being easily accessible for the development team throughout the development process. Fast releases, continuous integration and testing activities are directly connected to the knowledge of the system under development. The potential for agile boards to simplify development across geographically disparate areas is in this sense relatively clear. In a similar vein, the social and human aspects of the development process are becoming more and more important. The Google work style has become a model for many software start-ups - a pleasant work environment is important and affects the productivity of employees. One important contributor to a healthy work environment is that each employee is considerate and polite towards their fellow employees. More specifically "Politeness is the practical application of good manners or etiquette. It is a culturally defined phenomenon and what is considered polite in one culture can sometimes be quite rude or simply eccentric in another cultural context. The goal of politeness is to make all of the parties relaxed and comfortable with one another." [2] The last part of this definition is what we consider in our analysis. In this specific work, we do not take different cultures into account (although developers involved in a specific project could be from all around the world); we focus on the politeness of the comment-messages written by the developers. The research aims to show how project management tools such as agile boards can directly affect the productivity of a software development team and the health of a software project.

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

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

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

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

[1] https://www.atlassian.com/software/jira

[2] en.wikipedia.org/wiki/Politeness

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



PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

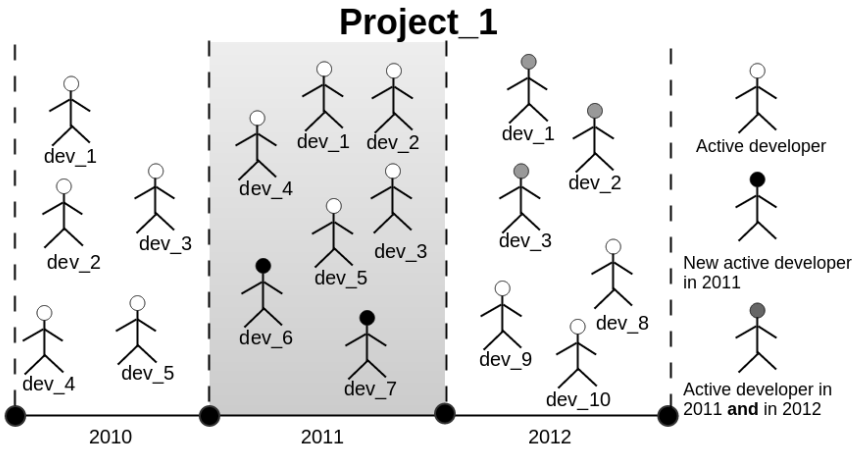


**Figure 1.** Example of Magnet and Sticky metrics computation in 2011

- **Does politeness among developers affect the attractiveness of a project?**
  We concluded that politeness is positively correlated with Magnetism and Stickiness metrics in subsequent years for the systems in our corpus.

- **Does politeness among developers vary over time?**
  Average politeness does vary over time and in some cases it changes from negative values (impolite) to positive (polite) from two consecutive observation intervals. Regarding average politeness over individual work days, we did not find great differences.

- **How does politeness vary with respect to JIRA maintenance types and issue priorities?**
  Comments related to issues with maintenance Bug, priority Minor and Trivial, tend to have a higher percentage of impolite comments. Issues with maintenance New Feature, priority Blocker and Critical tend to have a higher percentage of polite comments.

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

## 2 RELATED WORK

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

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

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



PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

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

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

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

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

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

# 3 EXPERIMENTAL SETUP

## 3.1 Dataset

We built our dataset collecting data from one of the JIRA[4] largest public dataset available (Ortu et al. (2015d)). An Issue Tracking System (ITS) is a repository used by software developers to support the software development process. It supports corrective maintenance activity like Bug Tracking systems, along with other types of maintenance requests. We mined the dataset (Ortu et al. (2015d)) collecting issues from October 2002 to December 2013. To create our corpus, we selected projects for which the JIRA Agile board contained a significant amount of activity (e.g., projects with the highest number of comments). Table 1 shows the corpus of 22 projects selected for our analysis, highlighting the number of comments recorded for each project and the number of developers involved.

## 3.2 Comment Politeness

Danescu-Niculescu-Mizil et al. (2013) proposed a machine learning approach for evaluating the politeness of a request posted in two different web applications: Wikipedia[5] and Stackoverflow[6]. Stackoverflow is

[4]https://www.atlassian.com/software/jira
[5]https:en.wikipedia.orgwikiMain_Page
[6]http:stackoverflow.com



PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

| 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 application[7] 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 tool[8], 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

[8] User's names are reported as $<dev\_name\_a>$ for the sake of privacy.



PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

| Comment | Confidence Level |
|---|---|
| Hey <dev_name_a>,<br>Would you be interested in contributing<br>a fix and a test case for this as well?<br>Thanks,<br><dev_name_b> | 0.7236 |
| <dev_name>, can you open a<br>new JIRA for those suggestions?<br>I'll be happy to review. | 0.919 |
| <dev_name>, the latest patch isn't applying<br>cleanly to trunk – could you resubmit it please?<br>Thanks. | 0.806 |
| <dev_name>,<br>Since you can reproduce, do you still want<br>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?<br>Don't do that. | 0.816 |
| shouldn't it check for existence of<br>tarball even before it tries to allocate<br>and error out ??? | 0.701 |
| <dev_name_a>, why no unit test?<br><dev_name_b>, why didn't you wait<br>for +1 from Hudson??? | 0.942 |
| > this isn't the forum to clarify<br>Why not? The question is whether<br>this is redundant with Cascading,<br>so comparisons are certainly relevant, no? | 0.950 |

**Table 3.** Examples of impolite comments.



PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

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

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

# 4 RESULTS AND DISCUSSION

## 4.1 Does politeness among developers affect issues fixing time?

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

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

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

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

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

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

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

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



PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

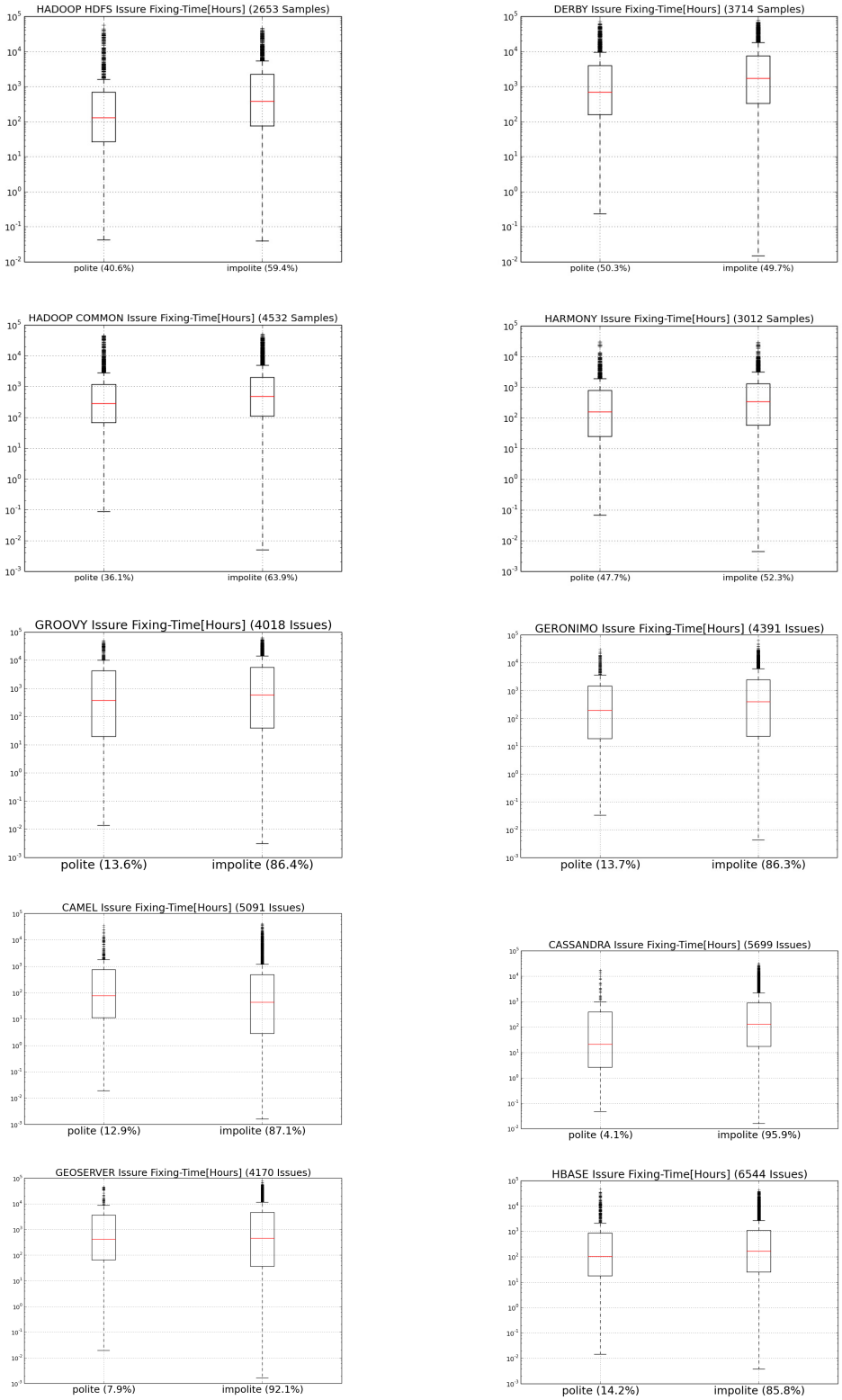


**Figure 2.** Box-plot of the fixing-time expressed in hours. The number in parentheses next to polite/impolite indicates the percentage of impolite and polite issues.



PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

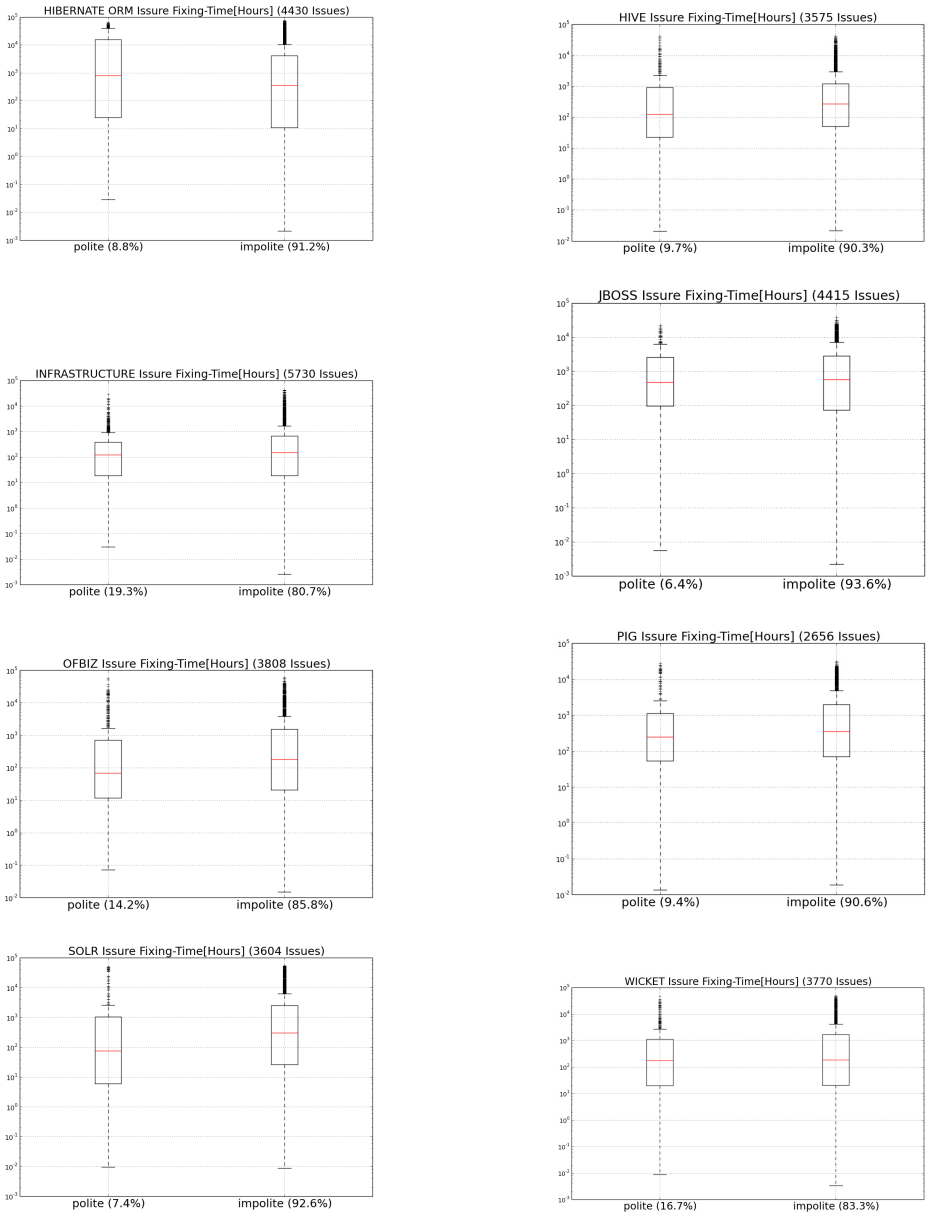


**Figure 3.** Box-plot of the fixing-time expressed in hours. The number in parentheses next to polite/impolite indicates the percentage of impolite and polite issues.



PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

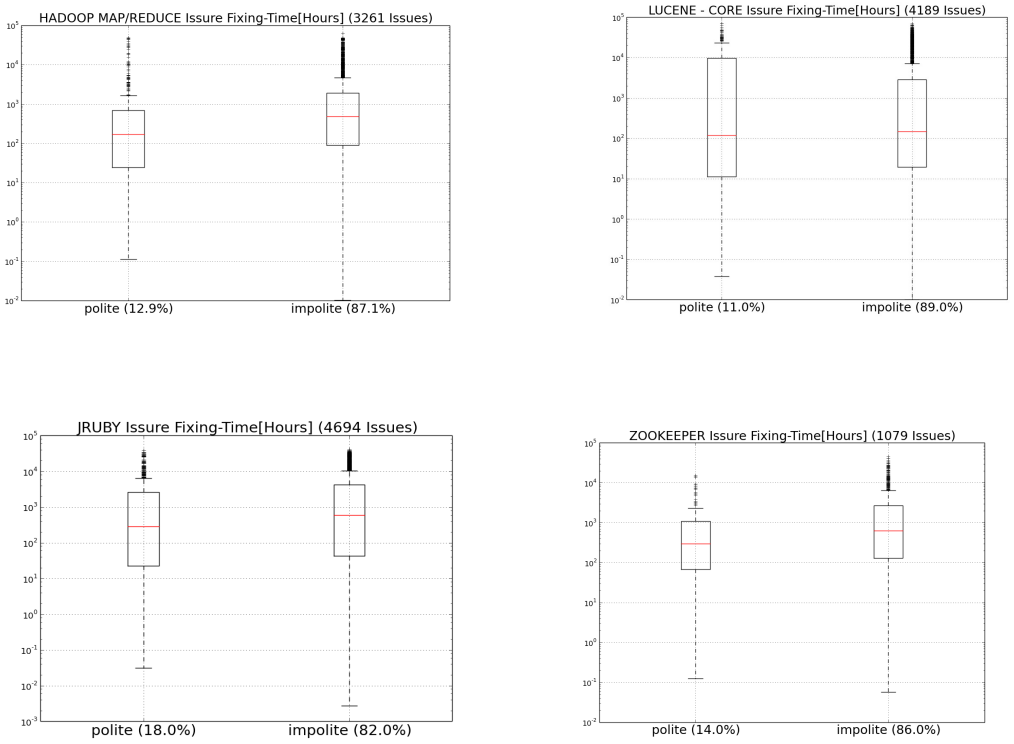


**Figure 4.** Box-plot of the fixing-time expressed in hours

| 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



PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

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

**Findings**. **In the majority of projects, Magnet and Sticky were positively correlated with politeness.** Table 5 shows the cross-correlation coefficient between the percentage of polite comments and magnetism and stickiness during the observation period. The two columns represent the cross-correlation coefficient between Magnetism and Stickiness and the percentage of politeness comments during the same observation time. The correlation values are positive in all cases.

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

### 4.3 Does politeness among developers vary over time?

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

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

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

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



PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

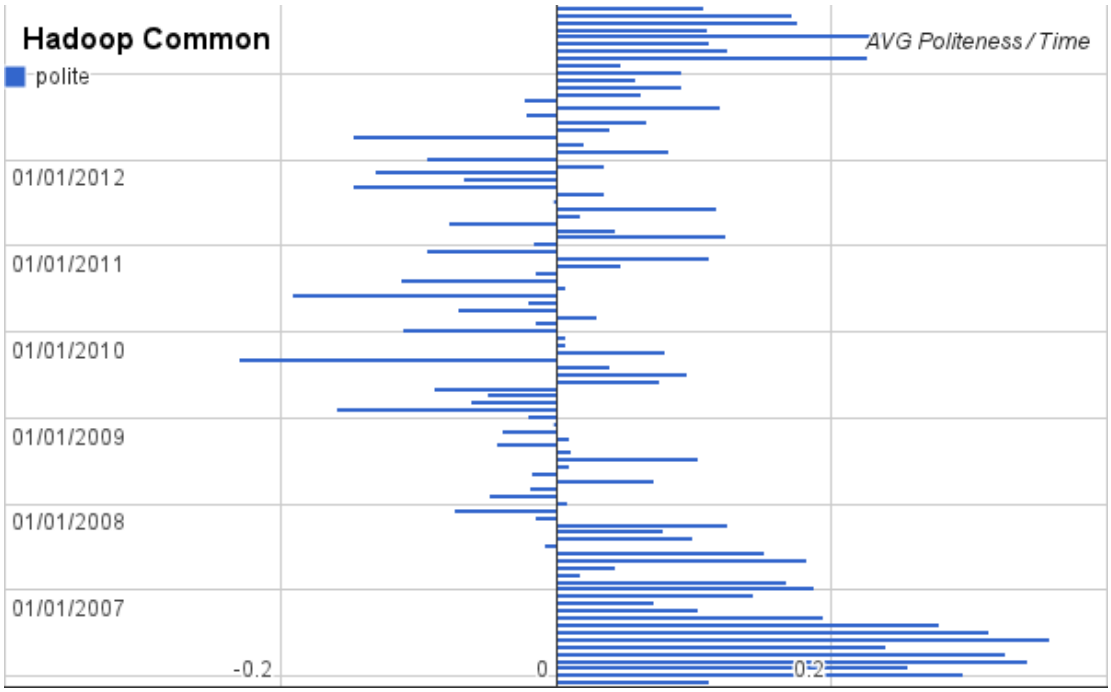


**Figure 5.** Average Politeness per month

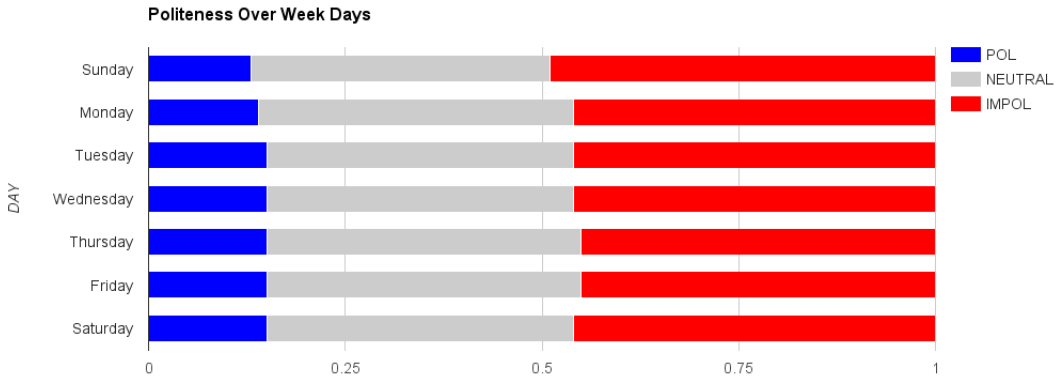


**Figure 6.** Average Politeness Over Week Days



PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

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

### 4.4 How does politeness vary with respect to JIRA maintenance types and issue priorities?

**Motivation**. Understanding which typology of issue attracts more impolite comments could help managers better understand the development process and take action to better manage the distribution of issues within development teams. A classification of the type of issues, is provided on the JIRA wiki[9]. The following list gives a brief introduction:

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

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

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

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

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

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



PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

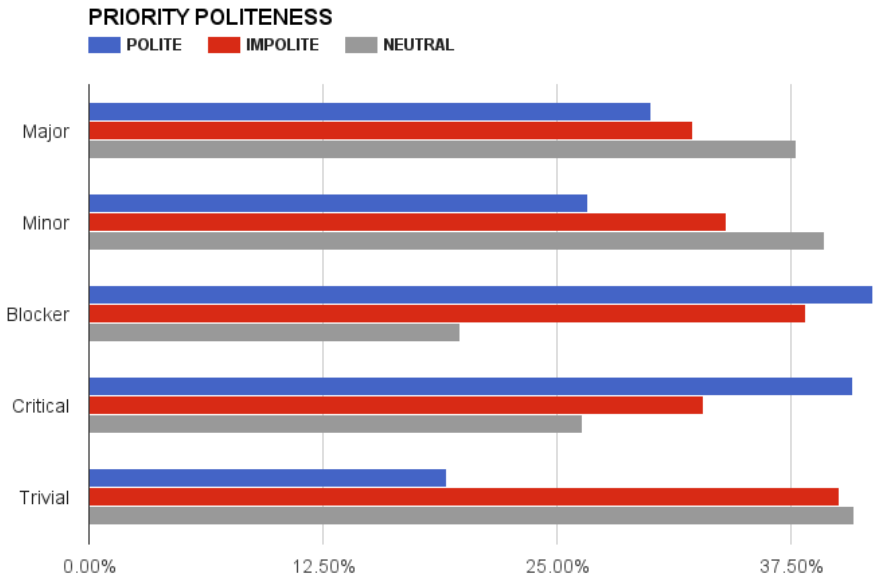


**Figure 7.** Priority

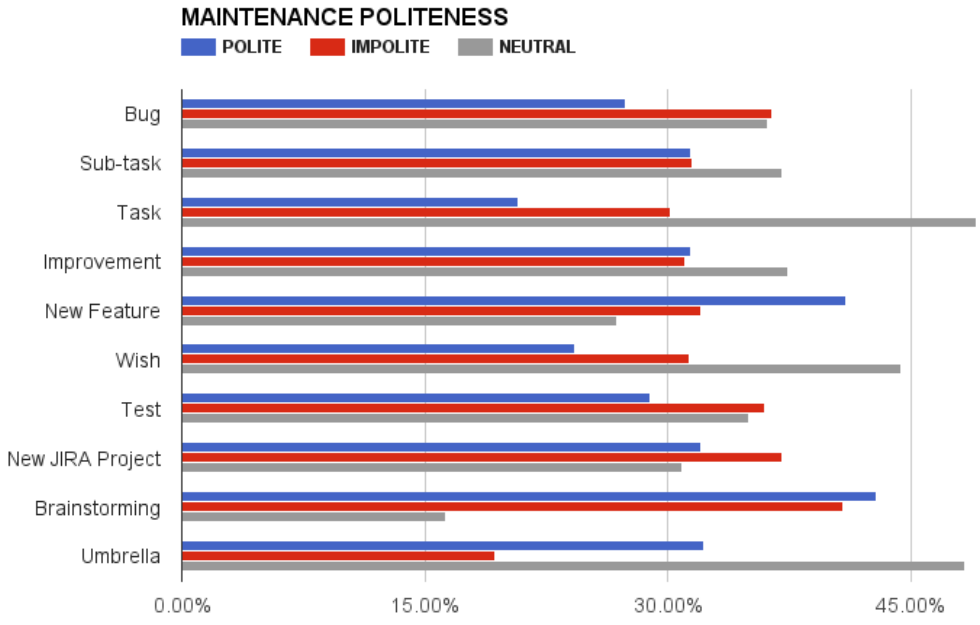


**Figure 8.** Maintenance



PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

## 5 THREATS TO VALIDITY

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

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

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

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

## 6 CONCLUSIONS AND FUTURE WORK

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

## REFERENCES

Acuña, S. T., Gómez, M., and Juristo, N. (2008). Towards understanding the relationship between team climate and software quality—a quasi-experimental study. *Empirical software engineering*, 13(4):401–434.

Bazelli, B., Hindle, A., and Stroulia, E. (2013). On the personality traits of stackoverflow users. In *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*, pages 460–463. IEEE.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001). Manifesto for agile software development.

Brief, A. P. and Weiss, H. M. (2002). Organizational behavior: Affect in the workplace. *Annual review of psychology*, 53(1):279–307.



PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

Campbell, D. T. and Stanley, J. C. (1963). *Experimental and quasi-experimental designs for generalized causal inference*. Houghton Mifflin.

Capretz, L. F. (2003). Personality types in software engineering. *International Journal of Human-Computer Studies*, 58(2):207–214.

Choi, E., Yoshida, N., Kula, R. G., and Inoue, K. (2015). What do practitioners ask about code clone? a preliminary investigation of stack overflow. *9th Internl Work. on Soft. Clones, IWSC*.

Cockburn, A. and Highsmith, J. (2001). Agile software development: The people factor. *Computer*, (11):131–133.

Curtis, B., Krasner, H., and Iscoe, N. (1988). A field study of the software design process for large systems. *Communications of the ACM*, 31(11):1268–1287.

Danescu-Niculescu-Mizil, C., Sudhof, M., Jurafsky, D., Leskovec, J., and Potts, C. (2013). A computational approach to politeness with application to social factors. In *Proceedings of ACL*.

Ehlers, J. (2015). Socialness in the recruiting of software engineers. In *Proceedings of the 12th ACM International Conference on Computing Frontiers*, page 33. ACM.

Erez, A. and Isen, A. M. (2002). The influence of positive affect on the components of expectancy motivation. *Journal of Applied Psychology*, 87(6):1055.

Fagerholm, F., Ikonen, M., Kettunen, P., Münch, J., Roto, V., and Abrahamsson, P. (2014). How do software developers experience team performance in lean and agile environments? In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, page 7. ACM.

Feldt, R., Torkar, R., Angelis, L., and Samuelsson, M. (2008). Towards individualized software engineering: empirical studies should collect psychometrics. In *Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering*, pages 49–52. ACM.

Gómez, M. N., Acuña, S. T., Genero, M., and Cruz-Lemus, J. A. (2012). How does the extraversion of software development teams influence team satisfaction and software quality?: A controlled experiment. *International Journal of Human Capital and Information Technology Professionals (IJHCITP)*, 3(4):11–24.

Kaluzniacky, E. (2004). *Managing psychological factors in information systems work: An orientation to emotional intelligence*. IGI Global.

Murgia, A., Concas, G., Tonelli, R., Ortu, M., Demeyer, S., and Marchesi, M. (2014a). On the influence of maintenance activity types on the issue resolution time. In *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, pages 12–21. ACM.

Murgia, A., Tourani, P., Adams, B., and Ortu, M. (2014b). Do developers feel emotions? an exploratory analysis of emotions in software artifacts. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 262–271. ACM.

Novielli, N., Calefato, F., and Lanubile, F. (2014). Towards discovering the role of emotions in stack overflow. In *Proceedings of the 6th International Workshop on Social Software Engineering*, pages 33–36. ACM.

Ortu, M., Adams, B., Destefanis, G., Tourani, P., Marchesi, M., and Tonelli, R. (2015a). Are bullies more productive? empirical study of affectiveness vs. issue fixing time. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, MSR 2015.

Ortu, M., Destefanis, G., Kassab, M., Counsell, S., Marchesi, M., and Tonelli, R. (2015b). Would you mind fixing this issue? In *Agile Processes, in Software Engineering, and Extreme Programming*, pages 129–140. Springer.

Ortu, M., Destefanis, G., Kassab, M., and Marchesi, M. (2015c). Measuring and understanding the effectiveness of jira developers communities. In *Proceedings of the 6th International Workshop on Emerging Trends in Software Metrics*, WETSoM 2015.

Ortu, M., Destefanis, G., Murgia, A., Marchesi, M., Tonelli, R., and Adams, B. (2015d). The jira repository dataset: Understanding social aspects of software development. In *Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering*, page 1. ACM.

Pang, B. and Lee, L. (2008). Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Pennebaker, J. W., Francis, M. E., and Booth, R. J. (2001). Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71:2001.




PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015

Perry, T. (2008). Drifting toward invisibility: The transition to the electronic task board. In *Agile, 2008. AGILE'08. Conference*, pages 496–500. IEEE.

Rekha, V. S. and Venkatapathy, S. (2015). Understanding the usage of online forums as learning platforms. *Procedia Computer Science*, 46:499–506.

Rigby, P. C. and Hassan, A. E. (2007). What can oss mailing lists tell us? a preliminary psychometric text analysis of the apache developer mailing list. In *Proceedings of the Fourth International Workshop on Mining Software Repositories*, page 23. IEEE Computer Society.

Roberts, J. A., Hann, I.-H., and Slaughter, S. A. (2006). Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the apache projects. *Management science*, 52(7):984–999.

Rosen, C. and Shihab, E. (2015). What are mobile developers asking about? a large scale study using stack overflow. *Empirical Software Engineering*, pages 1–32.

Siegel, S. (1956). Nonparametric statistics for the behavioral sciences.

Sliwerski, J., Zimmermann, T., and Zeller, A. (2005). Don't program on fridays! how to locate fix-inducing changes. In *Proceedings of the 7th Workshop Software Reengineering*.

Steinmacher, I., Conte, T. U., Gerosa, M., and Redmiles, D. (2015). Social barriers faced by newcomers placing their first contribution in open source software projects. In *Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing*, pages 1–13.

Tan, S. and Howard-Jones, P. (2014). Rude or polite: Do personality and emotion in an artificial pedagogical agent affect task performance? In *2014 Global Conference on Teaching and Learning with Technology (CTLT 2014)*, page 41.

Tourani, P., Jiang, Y., and Adams, B. (2014). Monitoring sentiment in open source mailing lists -– exploratory study on the apache ecosystem. In *Proceedings of the 2014 Conference of the Center for Advanced Studies on Collaborative Research (CASCON)*, Toronto, ON, Canada.

Tsay, J., Dabbish, L., and Herbsleb, J. (2014). Let's talk about it: Evaluating contributions through discussion in github. *FSE. ACM*.

VersionOne (2013). 8th annual state of agile survey report.

Weiss, C., Premraj, R., Zimmermann, T., and Zeller, A. (2007). How long will it take to fix this bug? In *Proceedings of the Fourth International Workshop on Mining Software Repositories*, page 1. IEEE Computer Society.

Wilcoxon, F. and Wilcox, R. A. (1964). *Some rapid approximate statistical procedures*. Lederle Laboratories.

Winschiers, H. and Paterson, B. (2004). Sustainable software development. In *Proceedings of the 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, pages 274–278. South African Institute for Computer Scientists and Information Technologists.

Yamashita, K., McIntosh, S., Kamei, Y., and Ubayashi, N. (2014). Magnet or sticky? an oss project-by-project typology. In *MSR*, pages 344–347.

Zhang, H., Gong, L., and Versteeg, S. (2013). Predicting bug-fixing time: an empirical study of commercial software projects. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 1042–1051. IEEE Press.




PeerJ PrePrints | https://dx.doi.org/10.7287/peerj.preprints.1515v1 | CC-BY 4.0 Open Access | rec: 17 Nov 2015, publ: 17 Nov 2015