Issue Reports Analysis in Enterprise Open Source Systems

Lerina Aversano
Department of Engineering, University of Sannio, Benevento, Italy

Keywords: Enterprise Systems, Software Evolution, Software Maintenance, Issue Report.

Abstract: In many organizations Enterprise Resource Planning (ERP) systems can be considered the backbone to managing business processes. Therefore, understanding their maintenance processes is a relevant topic for practitioners. As occur for many open source projects change requirements for ERP software are managed trough Issue Tracker systems, that, collect requests for change in form of Issue Reports. However, very often issue reports have relevant lack of information. Consequently, the time to resolution is strongly influenced by the quality of the reporting. In this paper, we investigate the quality of issue reports for enterprise open source systems. We examined some relevant metrics impacting the quality of issue reports, such as the presence of itemization, presence of attachments, comments, and readability. Then, the evaluation of the quality of the issue reports has been conducted on enterprise open source software.

1 INTRODUCTION

Despite the extensive knowledge about Enterprise resource planning (ERP) projects, research on their maintenance effects is still limited. ERP systems in most enterprises can be considered the backbone to managing business processes. Therefore, understanding their evolution processes is a relevant topic. As happen for many open source projects the ERP evolution is managed trough Issue Tracker that collect requests for change in form of Issue Reports. Specifically, the Issues management is a very crucial aspect that influences the quality of an ERP system.

Issue reports about a software system could be about a failure that produces an incorrect or unexpected behavior, therefore it causes numerous effects. In some cases, an issue has a low impact on the functionalities of the software system and consequently may remain unknown for a long time. In others cases the issue could impact quality aspects, such as security, for example it could allow an user to bypass access controls, in order to gain unauthorized privileges.

In any case issue reports are essential for the maintenance and evolution of most software systems. These allow final users of a software to inform maintainers about the problems encountered during the system usage. Typically issue reports contain a detailed description of a failure, sometimes in the report there is the indication to the involved code fragment (in the form of patches or stack traces). The quality of the issue reports can be different according to their content, however, very often they provide incorrect or inadequate information.

The consequence is that the understanding of a problem requires an effort higher than the effort required to solve the problem. To address this difficulty many guidelines on how to write a good bug report have been defined (Goldmerg, 2010) (Breu et al., 2010).

The quality of a issue report could impact the entire software system life cycle. In fact, it is a common practice in many software project, to discard issue reports unclear or having a severe lack of information.

In the context of ERP – Enterprise Resource Planning, the relevance of the good quality issue reports is even more important due to the complexity of such a kind of software systems and the strategic role they have within operative organizations.

This paper focuses on the evaluation of the quality of issue reports respect to the main features of the considered software system and an reports about results of an analysis that has been performed to detected the features categories of systems mainly impacted by issues.

In particular, categories of features are extracted by investigated ERP systems documentation through a manual inspection of the software main functionalities.

Aversano, L.
Issue Reports Analysis in Enterprise Open Source Systems.
DOI: 10.5220/0007757803370344
In *Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019)*, pages 337-344
ISBN: 978-989-758-372-8

In particular, this paper investigates the quality of issue reports from the perspective of software maintainers. Several attributes impacting the quality of issue reports have been considered, such as the presence of stack traces and attachments (such as screenshots). However, in particular, the authors investigate the presence of comments and the waiting time to resolution.

The analysis focus on enterprise open source systems (ERP or CRM used by small and medium-sized companies). The systems selected for this study are: Dolibarr, ERPNext, and SuiteCRM.

The paper is structured as follows: Section 2 describes the state of the art and provides information about some relevant research work related to the quality of a issue report; Section 3 describes the plan of study followed for the for the evaluation; Section 4 describes the obtained results. Finally, Section 5 section outlines the conclusions and future work.

2 BACKGROUND

The literature reports different studies addressing topics related to the quality of an issue report, but in few cases propose approaches methods for the evaluation of the quality of the report.

Breu et al., have identified the information that developers consider necessary within a bug report (Breu *et al.*, 2010) and suggest, on the basis of the investigations carried out, improvements to the bug tracking systems.

Another work describes an adaptive model for the life cycle of a bug report identifying in the time to resolution a good measure of its quality (Hooimeijer and Weimer, 2007). The authors highlight how writing a good bug report is complicated, and have to deal with poorly written report increases the resolution time. Knowing how the quality of an Issue impacts the overall lifecycle encourages users to submit better reports (Hooimeijer and Weimer, 2007).

Aranda and Venoila (Aranda and Venolia, 2009) examined the communication between the developers of bug reports in Microsoft and observed that many bugs are discussed before they are reported and this information is not stored within the Issue Tracker. However, in open source projects, many bugs are discussed in the bug tracking systems (or mailing list) to ensure transparency and to encourage developers who are geographically distant.

Different works in the literature use bug reports to automatically assign a bug to the developers (Anvik et al., 2006), identify duplicate bugs (Jalbert and Weimer, 2008) while others define guidelines for assessing the severity of a bug (Menzies and Marcus, 2008). Schroter et al. (Schroter et al., 2010) showed the importance of the Stack Trace for developers when they have to fix a bug.

Antoniol et al. (Antoniol et al., 2004) (Antoniol et al., 2008) indicate the lack of integration between the system of versioning and bug tracking system which makes it difficult the location of the fault within the system software, also in (Antoniol et al., 2008) it is discussed that not all the bugs are software problems but many indicate requests for improvements.

Ko et al. (Ko et al., 2006) in order to design new systems for reporting bugs have conducted a linguistic analysis on the securities of the bug report. They observed numerous references to software entities, physical devices or user actions, suggesting that the future system of systems Bug Tracking will be to collect data in a very structured way.

Not all bug reports are generated by humans, many systems of auto-detection of the bugs can report safety violations and annotate them with counter examples. Weimer (Weimer, 2006) presents an algorithm to build patches automatically as it shows that the report accompanied by patches have three times more likely to be localized within the code with respect to a standard report. Users can also help developers fix bugs without depositing the bug report, for example, many products automatically report information on the crash such as Apple CrashReporter, Windows Error Reporting, Gnome BugBuddy.

Hooimejer and Weimer (Hooimeijer and Weimer, 2007) proposed a descriptive model of quality bug reports based on statistical analysis of over 27,000 reports related to the open source project Mozilla Firefox. The model is designed to predict if a bug is fixed within a time limits in order to reduce the cost of bug triage. It leads the implications on the bug tracking system highlighting the features to be added when creating a bug report. The model proposed by Hooimejer and Weimer (Hooimeijer and Weimer, 2007) classifies bug reports based on the characteristics that can be extracted by the same bug report excluding features that require to compare the report with earlier reports, such as the similarity of the text. The features of the model include the Severity, the Readability Measures, and Submitter Reputation.

Finally, the authors consider the number of comments made in response to the bug and the number of attachment. The results presented show that the bug with high number of comments are resolved in less time. Furthermore, the measure of readability indicated that the bugs fixed in a short

time are easy to understand and highly readable. Finally, the results of Hooimejer Weimer and (Hooimeijer and Weimer, 2007) show that some characteristics, contrary to what is believed, have no significant effect on the model, such as the severity of the bug.

A significant contribution to the quality of bug reports was provided by the work of Zimmermann et al. (Zimmermann et al., 2010), where is defined a quality model of a bug report.

Zimmermann et al. (Zimmermann et al., 2010) propose a quality model for bug reports in order and implemented a prototype that helps users to insert the appropriate information while reporting a bug. The work is based on a survey involving developers and users. The survey carried out by the authors shows clearly a mismatch between what the developers believe important to fix a bug and what they consider important reporters. On the other hand, the developers point out that the real problem for the resolution of a bug is not wrong information but rather the lack thereof. Moreover, the difference in perspective between developers and reporters leads to knowledge of different quality.

The model adopted in this paper is even composed of a number of attributes each associated to a score that can be binary (for example the attachment is present or not) or a scalar (such as readability): itemization; Completeness.

3 PLAN OF THE STUDY

This section describes the process used to analyse the Issues of different Enterprise and CRM systems. In particular, provides an overview of the different phases of the analysis, and outlines the tools and techniques used for its achievement.

The general steps of the analysis are described in the following.

3.1 Definition of the Objective

The goal of the study is to perform an analysis of the Issue Reports of three Open Source enterprise software systems, focusing on ERP Enterprise Resource Planning systems as they are relevant software systems which integrates all the relevant business processes of a company (warehouse management, sales, purchases, accounting, etc.), and CRM Customer Relationship Management, as these software systems manage the relationship with the client, and support enterprises to offer the best product, the best service and the best possible sales assistance. In particular, the aim of the study is to understand if different features of the systems lead to different quality issue reports, and if, this entails a different treatment of the issue. To this aim, the analysis focused, for each issue report analysed, on number of comments about the issue; presence of screenshot, presence of item; waiting time.

3.2 System Selection

In this step the systems to consider for the analysis have been selected. Among the numerous ERP and CRM open source systems available, the following ones have been considered for this analysis: Dolibarr, ERPNext and SuiteCRM. In the following there is a brief description of the three systems, while some descriptive data are reported in Table 1.

- DOLIBARR: Dolibarr is an open source software system for the management of enterprise business processes. Dolibarr is both an ERP and CRM (depending on the activated modules). Dolibarr is mostly written in PHP using the MySQL database.
- ERPNEXT: is an Open Source software designed for small and medium enterprises. This system is particularly used by people with few skills in the field of business management systems, as it is presented as a simple app, so easy to use, configure and manage.
- SuiteCRM: SuiteCRM is an open source Customer Relationship Management application. It is often used as an alternative to proprietary CRM software from major corporations. It is a fork of SuiteCRM and started when SuiteCRM decided to stop development of its open source version.

3.3 Data Extraction

In this step the type of data source to obtain the information required for the analysis has been selected.

The analysis of the Issues was conducted starting from GitHub, a hosting service for software projects, based on the Git system distributed version control software. GitHub provides an Issue tracking system, pull request and comments that allows to improve the code of the repository by solving bugs or adding functionality. In this study Github has been used to obtain the Issues of the analysed systems and the source codes of the Java classes used to obtain additional data.

The data have been extracted from the Issue Tracking system. In particular, Issues can be in two

main different states, that are Open and Closed. Each Issue consists of several parameters:

- Id: unique number that identifies an Issue;
- Title: description of the problem treated by the Issue;
- Labels: labels used to organize problems and retrieve requests in a repository in categories based on priority, category or any other information deemed useful;
- Assignee: username of the programmers responsible for the resolution of the Issue;
- Milestone: they are containers of Issue able to collect more Issues in based on a specific characteristic;
- Comments: feedback entered by users accessing a repository.

Table 1: Descriptive data of the selected software system

General	Dolibarr	ERPNext	SuiteCRM
Homepage	dolibarr.org	erpnext.com	Suitecrm.com
Project License	GPL-3.0+	GPL-3.0+	GPL-3.0
All Time Statistics at September 2018			
Contributors	335	1185	5
Commits	66539	92822	6

The download of issues data has been performed through a Java tool for automatically downloading the issue and preparing the data set for the analysis. The data set contains some main attributes, among the ones available:

- Id, title, assignee, milestone, labels: parameters previously introduced;
- Creator: username of the author of the issue;
- State: state in which an issue may be found. It can be both Closed and Open;
- Comments: number of comments written as feedback to a report;
- Follower: number of users who follow the author of the issue;
- Following: number of users who are followed by the author of the issue;
- Creation date: the date on which the report was created;
- Closing date: date on which the report was closed (with a value different from NULL only in case of issue Closed);
- Body text: description of the problem encountered by the user during the use of the software.

Moreover, additional attributes have been computed and added to the dataset, including specific attributes to investigate quality aspect of the issue reports.

The additional considered attributes are the following:

- Itemization: Boolean attribute. It can take TRUE or FALSE value based on the presence or not, in the body of the issue, of the so-called "step to" reproduce ", i.e. the description, step by step, of the reproduction of the problem, to better identify and correct problems;
- Screenshots: Boolean attribute. It can take the value of TRUE or FALSE based on the presence or not, in the body of the issue, of images, gif or video.
- Current Waiting Time: waiting time for an issue before being examined and resolved by a developer, expressed in days. The formula used to calculate this parameter is:

  Current Waiting Time = Current Date () - Creation Date ()

- Category: this parameter represents the categories of a system. These are detected based on the characteristic modules of that system.
- Completeness: represents the completeness of the issues.

3.4 Inspection of Feature Categories

To investigate on the topic of the issue reports they have been associated to one of the feature categories of the software system analysed. To this aim different feature categories have been identified for each system. The issues therefore have been related to these features. The analysis was conducted manually, looking for: the "key words" within the issue titles; the description and on the basis of available online documentation. Then a java tool has been used for associating the issue reports to the identified categories. This tool analyses the body of the issues to obtain new attributes, such as Itemization, Screenshots, and Feature Category.

4 RESULTS AND DISCUSSIONS

The analysis starts from the dataset created from the issues of the considered case studies. Only the issues in ‘Open’ state have been considered, in order to be able to carry out an analysis of the waiting time of an issue after its creation. The issues of the Dolibarr taken into consideration amount to 636; those of the

ERPNext system are 1352; while 776 issues were analysed for the SuiteCRM system.

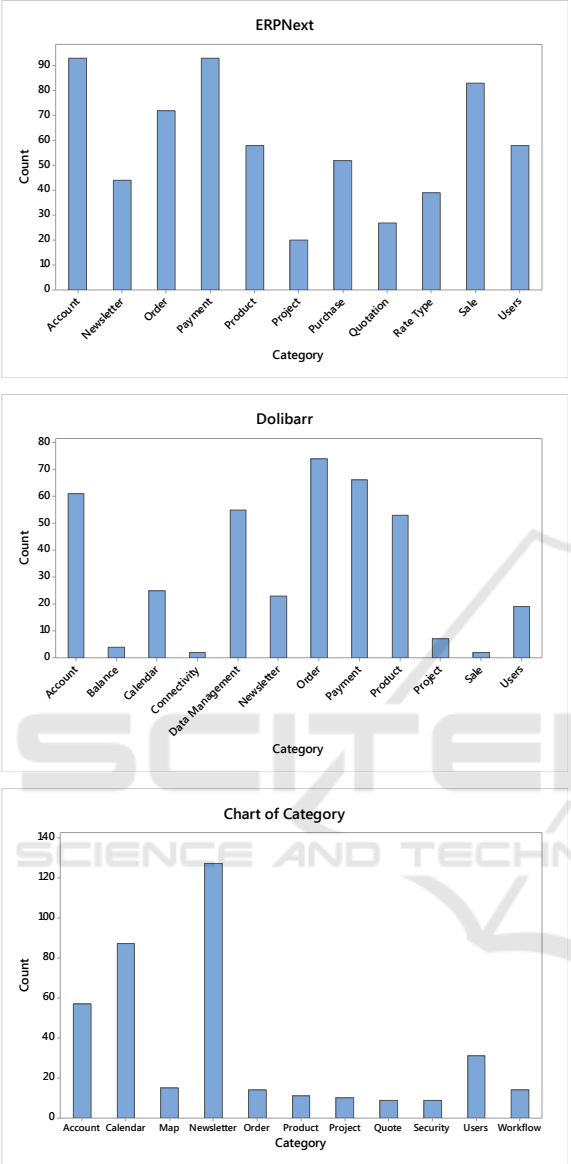


Figure 1: Number of Issue Vs Feature Categories.

Figure 1 shows the number of open issue for each feature category. It can be observed that for the Dolibarr system the categories mainly affected by issue are: Order and Payment while for the ERPNext system the category mainly affected are: Account, Payment and Order. The results are easy to comprehend, indeed Dolibarr and ERPNext two ERP systems, therefore aimed at business management.

On the other hand, in the case of the CRM systems it is expected that the categories most impacted are those relating to communication and customer management. From the analysis conducted on the SuiteCRM system issue reports it emerged that the categories mainly impacted categories are: Newsletter and Calendar.

Then, the analysis focused on distribution of issue reports among the categories, evidencing for each category the number of issue containing Itemization element in the description. Indeed, as described in the previous section, the presence of itemization is a element for the quality of an issue report, as it indicates the presence, or less, of the "Step to Reproduce". The results of this analysis are shown in Figure 2.

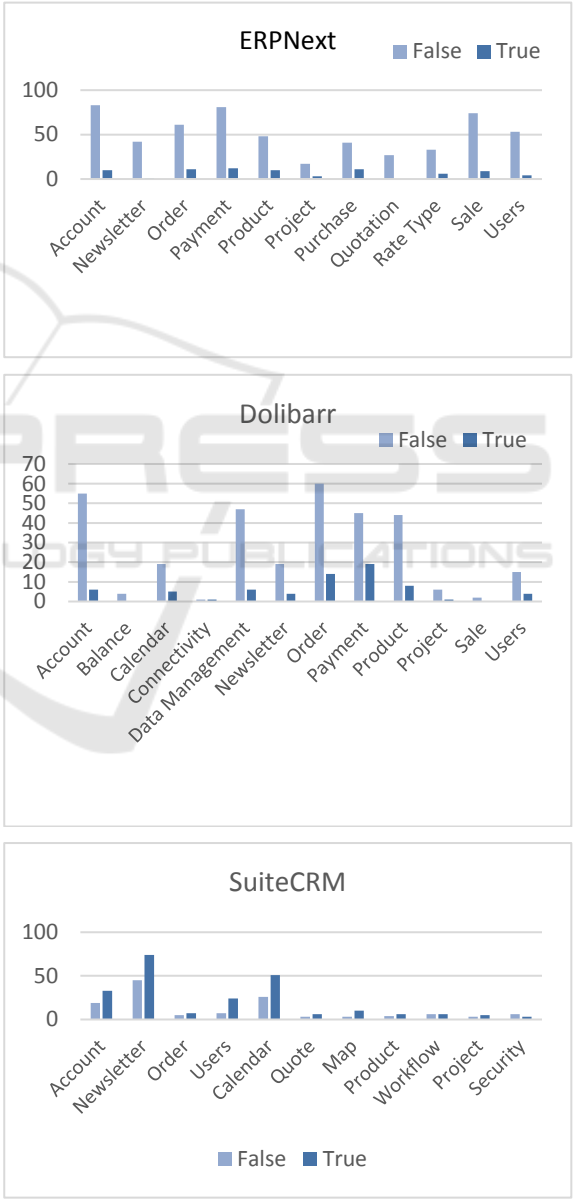


Figure 2: Number of Issue with Itemization Vs Feature Categories.

From the graphs it is possible to observe that the large part of issues are without Itemization, which does not guarantee a good evaluation of the issue because, as already explained, the presence of the steps to reproduce to detect and, later, solve the problem, allows to improve the resolution time of an issue and, moreover, allows greater clarity.

However, it is possible to highlight that in the case of SuiteCRM the presence of issues with Itemization is greater than those without. In this system, therefore, the problems detected by users, are easily understood by developers and the resolution time of an issue will certainly be less than that of other systems.

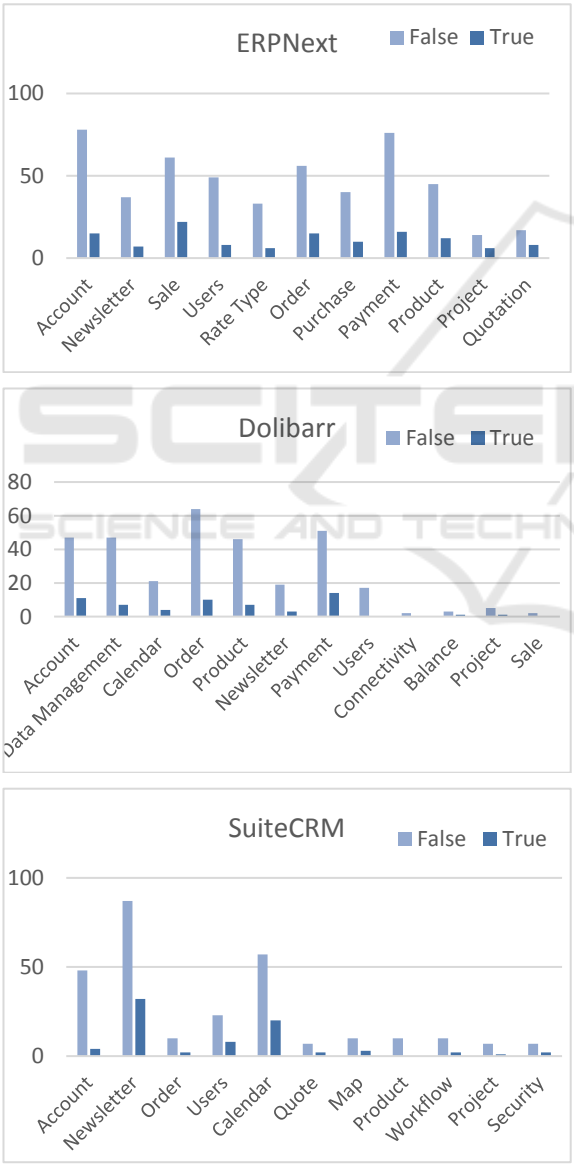


Figure 3: Number of Issue with screenshot Vs Feature Categories.

Similarly, Figures 3 show the results of the analysis related to the presence of Screenshots in the issue reports distributed respect to the different categories. The presence of screenshots is considered important in order to make the problem described in the issue report clearer and more comprehensible.

Even in this case it is possible to notice that, for all the considered systems, the number of issue without screenshots is greater than those with. This result compromises the quality of the issue report analysed.

To investigate more in details this aspect Figure 4 reports a scatter plots relating the number of screenshots and the number of comments. This analysis aims to understand if the presence (or the absence) of screenshot lead to a higher number of comments.

In particular, the analysis has been conducted by using a linear graph to represent the degree of correlation (ie, linear dependence) between the two variables.

In the graph it is possible to notice that the absence of the screenshots corresponds to a higher number of comments.

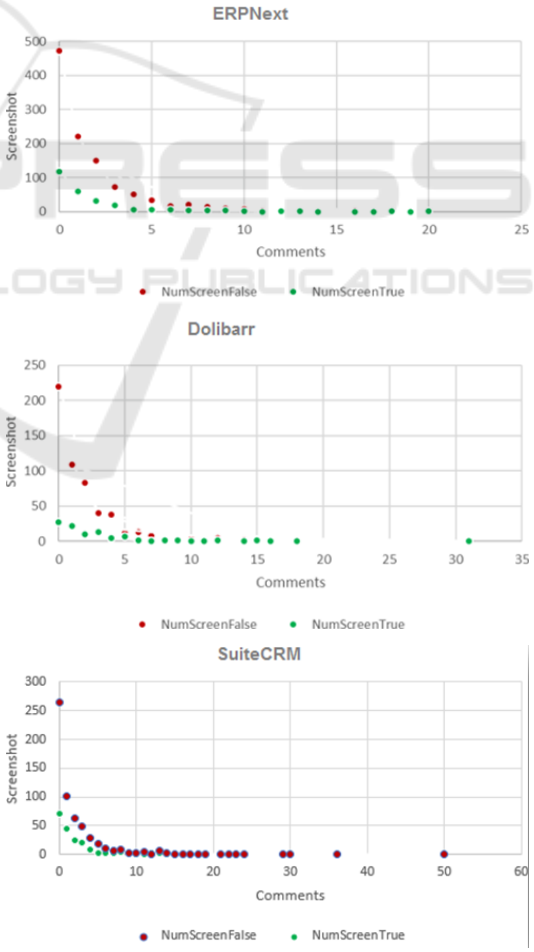
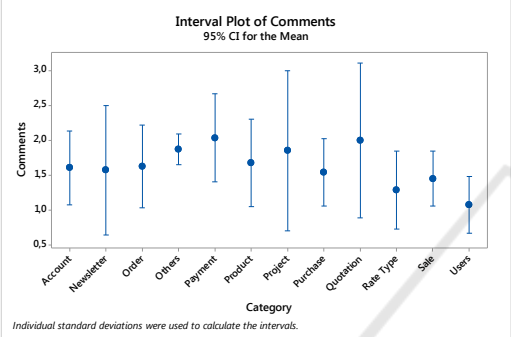
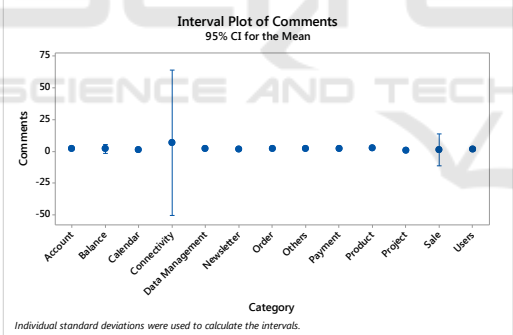


Figure 4: Number of comments on the issues Vs presence of screenshot in the Issue Report.

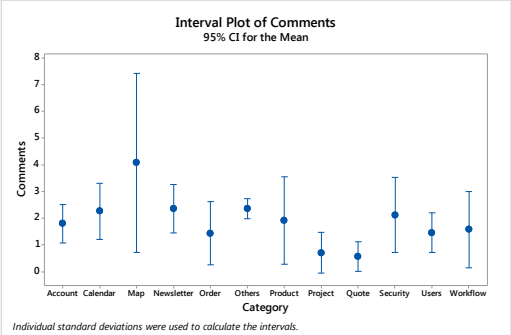
This result is interesting as a high number of comments could influence the time to resolution of the issue.vIn particular, Figure 5 shows the distribution of Comments about the submitted issues respect to the features categories identified. It is possible to observe that in Dolibarr and SuiteCRM there are some important differences among the categories, while in ERPNext the distribution of the comments is mainly similar. For the SuiteCRM system it is possible to note that Map is the category with the greatest variability and comment while for the ERPNext system there is a greatest variability and Payment is the category with the highest number of submitted comments.



(a) ERPPNext.



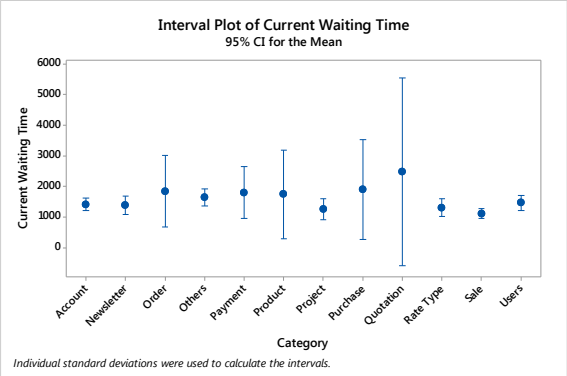
(b) Dolibarr



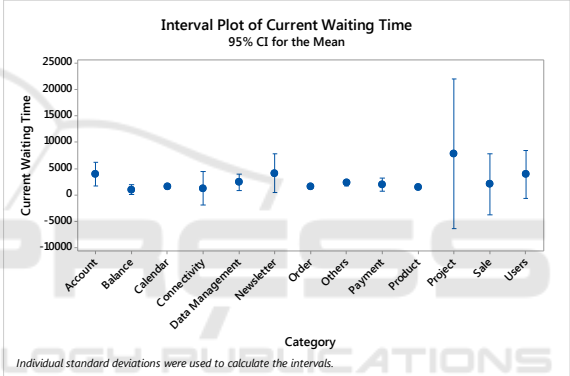
(c) SuiteCRM

Figure 5: Comments on issues Vs Feature Categories.

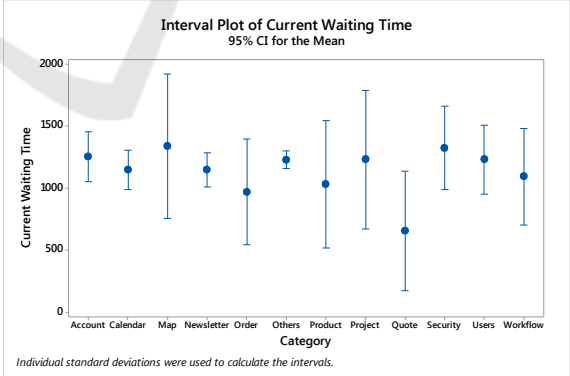
Then, the analysis focused on the waiting time. To this aim, Interval plots have been used to understand how the waiting time of the issues change respect to the categories of the system.



(a) ERPNext.



(b) Dolibarr



(c) SuiteCRM

Figure 6: Waiting time of the issues Vs Feature Categories.

Figure 6 shows that in the SuiteCRM system, the issues open with a higher waiting time are those related to the Map and Security category. This can be related to the fact that many issues of these categories are without Item or Screenshot in the Issue Report,

and therefore, the resolution time is greater respect to the one of categories in which, these parameters are satisfied. In the Dolibarr system, it emerged that the categories with the longest waiting time are: Newsletter and Project, even if the same waiting time exhibit few change among the different categories.

Finally, in the ERPNext system the categories with higher waiting time are: Account and Users.

5 CONCLUSIONS

ERP systems are relevant support for business processes performances. They are subject to continuous change requests submitted by the users by using Issue reports. Therefore, Issue reports have a significant role in the evolution of software systems. They should be able to provide, if accurately, the precise steps to reproduce the problem encountered. The objective of this paper consists in the analysis of Open Issue Report, that is, still under resolution, of some Enterprise software including: Dolibarr, ERPNext, and SuiteCRM. In summary, for each system different categories have been identified, through which the issues were grouped. The analyses were carried out using data collected and organized from issue tracker.

From the analysis, it was possible to note that only SuiteCRM presents "Complete" Issue Reports, including images related to the part of the code / program in which the error occurred and the description of the steps to reproduce the problem.

REFERENCES

Anvik J., Hiew L., Murphy G. C., 2006, Who Should Fix This Bug?, *IEEE Proceedings. 28th Int'l Conf. Software Eng., pp. 361-370.*

Antoniol G., Gall H., Di Penta M., Pinzger M., 2004 Mozilla Closing the Circle, *Technical Report TUV-1841-2004-05 Technical Univ. of Vienna*.

Antoniol G., Di Penta M., Ayari K., Khomh F., Guéhéneuc Y.G., 2008, Is It a Bug or An Enhancement? A Text-Based Approach to Classify Change Requests., *Proceedings of Conference for Advanced Studies on Collaborative Research, pp. 304-318.*

Aranda J., Venolia G., 2009, The secret life of Bugs: Going Past the Errors and Omissions in Software Repositories, *Proceedings of the 31st International Conference on Software Engineering*.

Breu S., Premraj R., Sillito J., and Zimmerman T., 2010, Information Needs in Bug Reports: Improving Cooperation between Developers and Users, *ACM Proceedings Conf. Computer Supported Cooperative Work, pp. 301-310*.

Goldmerg E, 2010, Bug writing guidelines, https://issues.apache.org/bugwritinghelp.html.

Hooimeijer P. and Weimer W., 2007, Modeling Bug Report Quality, *IEEE/ACM Proceedings of the International Conference Automated Software Eng, pp. 34-43*.

Jalbert N., and Weimer W., 2008, Automated Duplicate Detection for Bug Tracking System, *Proceedings. Conference Dependable System and Networks, pp. 52-61*.

Menzies T., Marcus A., 2008, Automated Severity Assessment of Software Defect Reports, *IEEE Proceedings of 24th International Conference Software Maintenance, pp. 346-355*.

Schroter A., Bettenburg N., Premraj R., 2010, Do Stack Trace Help Developpers Fix Bugs?, *IEEE Proceedings of International Working Conference Mining Software Repositories*.

Ko A., Myers B. A., Chau D. H., 2006, A Linguistic Analysis of How People Desribe Software Problems, *IEEE Proceedings Symposium Visual Languag and Human-Centric Computing, pp. 127-134*.

Weimer W., 2006, Patches as Better Bug Reports, *Proceedings Fifth International Conference Generative Programming and Component Eng., pp. 181-190*.

Zimmermann T., Premraj R., Bettenburg N., Just S., Schroter A., Weiss C., 2010, What Makes a Good Bug Report?, *IEEE Transactions on Software Engineering*.