Empir Software Eng (2016) 21:1437–1475
DOI 10.1007/s10664-015-9391-7

Using text clustering to predict defect resolution time: a conceptual replication and an evaluation of prediction accuracy

Saïd Assar[1] · **Markus Borg**[2] · **Dietmar Pfahl**[3]

Published online: 27 June 2015

Abstract Defect management is a central task in software maintenance. When a defect is reported, appropriate resources must be allocated to analyze and resolve the defect. An important issue in resource allocation is the estimation of Defect Resolution Time (DRT). Prior research has considered different approaches for DRT prediction exploiting information retrieval techniques and similarity in textual defect descriptions. In this article, we investigate the potential of text clustering for DRT prediction. We build on a study published by Raja (2013) which demonstrated that clusters of similar defect reports had statistically significant differences in DRT. Raja's study also suggested that this difference between clusters could be used for DRT prediction. Our aims are twofold: First, to conceptually replicate Raja's study and to assess the repeatability of its results in different settings; Second, to investigate the potential of textual clustering of issue reports for DRT prediction with focus on accuracy. Using different data sets and a different text mining tool and clustering technique, we first conduct an independent replication of the original study. Then we design a fully automated prediction method based on clustering with a simulated test scenario to check the accuracy of our method. The results of our independent replication are comparable to those of the original study and we confirm the initial findings regarding significant differences in DRT between clusters of defect reports. However, the simulated test scenario used to assess our prediction method yields poor results in terms of DRT prediction accuracy. Although our replication

Communicated by: Thomas Zimmermann

✉ Saïd Assar
said.assar@telecom-em.eu

Markus Borg
markus.borg@cs.lth.se

Dietmar Pfahl
dietmar.pfahl@ut.ee

[1] Ecole de Management, Institut Mines-Telecom, 9, rue C. Fourier, 91011 Evry, France

[2] Department of Computer Science, Lund University, Box 118, SE-221 00 Lund, Sweden

[3] Institute of Computer Science, University of Tartu, J. Liivi 2, Tartu 50409, Estonia

confirms the main finding from the original study, our attempt to use text clustering as the basis for DRT prediction did not achieve practically useful levels of accuracy.

Keywords Defect resolution time · Prediction · Text mining · Data clustering · Independent replication · Simulation

1 Introduction

Defect management is a central aspect in software maintenance. The detection and elimination of software defects require significant effort (Boehm and Basili 2001); software defects' cost to the US economy was in 2002 estimated to $59.5 billion annually (Tassey 2002). When a defect is detected, appropriate resources must be allocated to identify and resolve the defect. The allocation is based on estimates of defect severity and potential impact of the defect, as well as the availability of resources. An important aspect to consider in the allocation of resources is the expected Defect Resolution Time (DRT), i.e., the estimated time required to resolve the defect. The DRT also reflects how long end users have to wait for the corrected piece of software, especially in software development applying continuous deployment. Thus, predicting the DRT of newly submitted defects is central to project managers, as it supports both resource allocation and release planning.

DRT prediction has been investigated intensively in the last decade (Strate and Laplante 2013). The basic underlying idea is to use certain defect attributes, e.g., severity and number of comments (Panjer 2007), and to leverage historical data in order to derive prediction models. Multiple techniques have been experimented, e.g., decision trees (Giger et al. 2010), univariate and multivariate regression analysis (Bhattacharya and Neamtiu 2011), textual similarity (Weiss et al. 2007), Markov model (Zhang et al. 2013), but fully conclusive results have not yet been obtained. Of particular interest for us in this article is the method proposed by U. Raja for DRT prediction based on textual clustering of defect reports (Raja 2013). Raja used data clustering, a well-known method for unsupervised learning. She showed that different clusters of defect reports have significantly different mean DRT. Based on this observation, Raja claims that "*text based classification of defect reports can be a useful early indicator of defect resolution times*" (p.135), implying that her approach could be applied to predict the DRT of incoming defect reports. For example, the mean (or median) DRT of the cluster a newly incoming defect report belongs to could be a plausible predictor of the time required to resolve that specific defect. It should be noted that Raja provides no instructions on how to design such a prediction method in practice. She also did not check the applicability and quality of such a prediction method.

The contribution of this paper is twofold. First, we replicate the experiment reported in Raja (2013) and check whether our replication yields comparable results with regards to the difference of DRTs of clustered defect reports. We structure our replication according to the framework proposed by González-Barahona and Robles (2012). Our concern is to check whether the results of the proposed technique hold when a different data mining tool is used and a clustering technique is applied that is fully automated. Thus, the replication we conduct is an external, non-exact (conceptual) replication. The value of replication has been emphasized in empirical software engineering and its importance to allow firm conclusions has been stressed in the community (Brooks et al. 2008) (Shull et al. 2008) (Menzies and Shepperd 2012). Replication is motivated by the necessity to demonstrate repeatability of experimental results and to understand the effects of a study's characteristics on its reproducibility

(González-Barahona and Robles 2012). Thus, there is a need for both increasing the number of replications as well as for better acknowledging the value of replicated studies in software engineering (Carver et al. 2014).

Second, we go beyond the goal of the original experiment (i.e., testing whether clusters have significantly different mean DRTs), and operationalize a method to actually predict DRTs of (sequential chunks of) incoming defect reports. As such, we test the claim made by Raja concerning the applicability of clustering for DRT prediction under industrial working conditions. To this end, we design simulation scenarios in which we assess the prediction accuracy of our prediction method, if applied repeatedly over time.

Accordingly, the research questions we aim to answer are the following:

> RQ1: Are the results of the experiment reported in Raja (2013) replicable using a fully automated clustering technique?
> RQ2: How accurate is a method for DRT prediction based on automated clustering of defect reports?

Our study is not an exact replication as there are three main variation points. First, while in the original experiment five open source data sets were used, we use two data sets from open source repositories and one data set from a project conducted in a private company. Second, we used a different software tool for the textual clustering, i.e., RapidMiner instead of SAS Text miner, and a different clustering technique (k-means clustering instead of clustering using entropy minimization). Third, we apply automated clustering. This is different from the original experiment which required human intervention and expert knowledge to optimize the obtained clusters. Our replication can thus be classified as non-exact and conceptual, i.e., an external replication undertaken by a different research team with several experimental parameters changed.

The structure of the paper is as follows: The next section discusses defect management in software engineering, reviews prediction approaches based on textual similarity and clustering used in defect management in general and applied to DRT in particular. Also, the next section describes Raja's original experiment in detail (Raja 2013). In section 3, we present our replication of Raja's baseline experiment, including a description of the data sets and the experimental setup. We put special emphasis on the differences between the original experiment and our replication. Section 4 presents the results of the replicated experiment, in particular the statistical tests concerning the difference in DRT among clusters. Also, section 5 introduces the design of a possible operationalization of the clustering approach for the purpose of DRT prediction. This operationalization is tested in a simulation scenario to evaluate the prediction accuracy of the method and to explore the possible effect of certain confounding factors. Section 7 presents the results of the simulation scenario in terms of prediction accuracy. Section 8 discusses our results in relation to the research questions, and gives possible explanations to the obtained results. Section 9 analyzes threats to validity. Finally, section 10 concludes the paper and outlines plans for future work.

2 Background and Related Work

Figure 1 describes the general lifecycle of a defect report. Defect reporting systems allow users to report, track, describe, comment on and classify defect reports. A defect report comes with a number of pre-defined fields, including categorical information such as the relevant product,

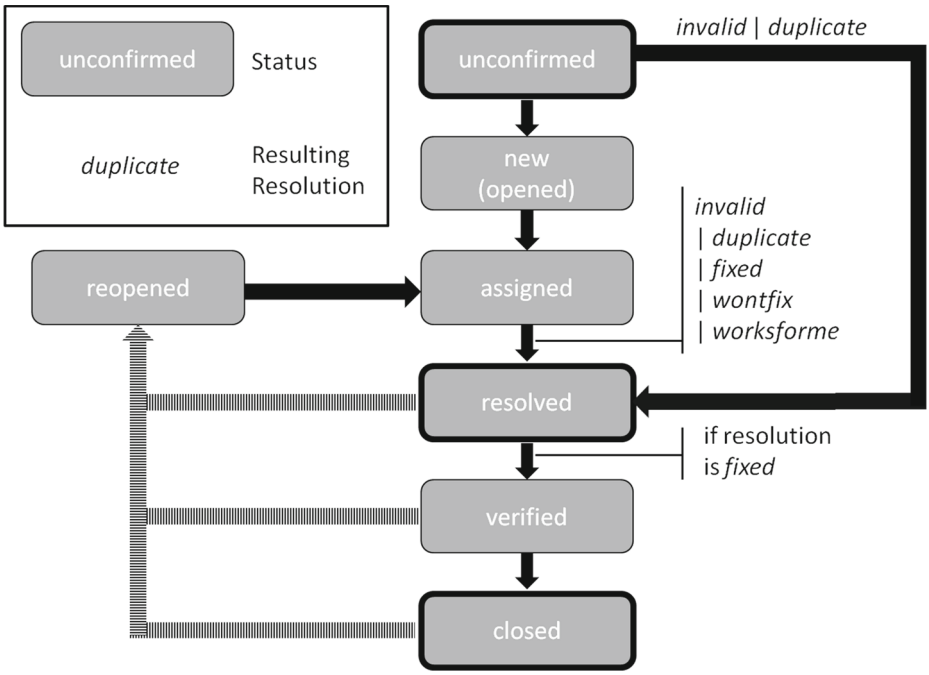

Fig. 1 Lifecycle of a defect report (adapted from (Zeller 2009))

version, operating system and self-reported incident severity, as well as free-form text fields such as defect title and description. In addition, users and developers can leave comments and submit attachments, which often take the form of patches or screenshots. When initially declared, a defect starts out in the *unconfirmed* pending state until a triager makes a first evaluation to see if the defect report corresponds to a valid defect, and that the defect is not already known, i.e., the submitted defect report is a duplicate of another defect report already stored in the defect reporting system. Defect reports can pass through several different stages before finally being resolved. Defect reports that are resolved receive one of the following designators: *duplicate, invalid, fixed, wontfix*, or *worksforme*. These indicate why the report was closed; for example, *worksforme* and *invalid* both indicate that quality assurance was unable to reproduce the issue described in the report. Sometimes a defect report needs to be reopened (cf. gray arcs in Fig. 1). In this case, the normal defect lifecycle (cf. black arcs) starts with status 'reopened'.

Defects are inherent to software development; producing defect-free code is a fundamental challenge in software engineering (Boehm and Basili 2001). The automated prediction and detection of defects in code is an important issue which has received much attention. This topic is, however, beyond the scope of this study and we refer interested readers to literature reviews on the subject (Lessmann et al. 2008) (D'Ambros et al. 2010). We will focus here on newly submitted defect reports and the problem of predicting their resolution time.

2.1 Analysis and Prediction of Defect Resolution Time

One of the first studies on defect resolution times is by Kim and Whitehead Jr (2006). The authors use descriptive statistics to analyze several aspects: the distribution of defect resolution

times, identification of who resolved which defect, the modules that have the highest number of defects, and the modules that have the most corrective fixes. The authors' goal is to demonstrate how defect resolution time could be used as a factor for defect related analysis. The median value of defect resolution times for source code files in the two open source projects that were analyzed, ArgoUML and PostgreSQL, is about 200 days; however, for certain files, the defect resolution time extended into years.

In Panjer (2007), the author explores how different methods and different attributes, e.g., severity and the number of comments (excluding textual descriptions), can be used in a prediction model. The experiments are conducted using a historical portion of the Eclipse Bugzilla database. Five different algorithms are tested, and logistic regression yielded the best results and provided the best prediction accuracy, i.e., 34.9 %, for the defect reports in the test set. The author concludes that "*there are other attributes or metrics that may have greater influence of the resolution time of bugs*". This study has been replicated in Bougie et al. (2010) using a different data set, and as the findings are inconsistent, they conclude that both the predictive accuracy, and the most accurate prediction method, may vary significantly between software projects.

Another study that applies regression analysis to predict defect resolution time has been published by Anbalagan and Vouk (2009). The authors use the number of persons concerned with the defect report as a predictor. They conclude that the more people that are involved in a defect report, the longer its resolution time.

More recently, several studies tried to understand the effects of different defect attributes by applying various mining approaches. In Giger et al. (2010), decision tree analysis is applied to classify incoming defects from three open source projects into fast or slowly fixed. Results show that *assignee*, *reporter*, and *monthOpened* are the attributes that have the strongest influence on defect resolution time (DRT) prediction. Furthermore, post-submission data contained in defect reports, e.g., comments attached within a few weeks after the submission, improve the performance of prediction models by 5 to 10 %. The study in Marks et al. (2011) goes further and identifies 49 metrics according to three dimensions (location, reporter and description) and use them in a Random Forest approach to predict DRT in simulation scenarios. Results show that the location dimension of a defect, e.g., component name, operating system, or the number of open and closed bugs, have the highest effect on the prediction accuracy. In Bhattacharya and Neamtiu (2011), the authors perform univariate and multivariate regression analysis to capture the significance of the features selected in building prediction models, and to assess their predictive power. The results of the multivariate regression analysis show that most of the post-submission features of a defect report, such as the number of attachments or defect severity, contribute significantly to the prediction model, however they also report significant variations in prediction power among different data sets Finally, the study by Lamkanfi and Demeyer (2012) has similar goals in terms of better understanding the predictive power of applied techniques. As outliers may negatively influence the performance of the data mining technique chosen, the authors experiment with removing outliers and obtain a significant improvement.

An interestingly different approach for DRT prediction is proposed by Zhang et al. (2013). The authors develop a Markov-based method for predicting the number of bugs that will be fixed in the future. For a given number of defects, their method estimates the total amount of time required to fix them based on the empirical distribution of bug-fixing time derived from historical data. Their results confirm previous findings, e.g., Panjer (2007), concerning the attributes that have the strongest influence on the bug fixing time, i.e., the assigned developer, the bug reporter, and the month when the bug was reported.

In fact, there is only one single study (Weiss et al. 2007) that relies exclusively on the textual content of a 'bug report' for resolution time prediction. Therefore, this study is an important reference point for the second part of our study, i.e., investigating RQ2, and we report the results of this study with more detail. Weiss et al. (2007) use 567 'bug reports' (including types 'defect', 'feature request', 'task' and 'sub-task') from the JBoss open source dataset and apply their prediction approach to each newly incoming report. For the N-th incoming report, the prediction is based on a training set composed of the previously arrived N-1 reports. The k-nearest-neighbor (kNN) approach is used to find a set of the k most similar defects, k varying from 1 to 3, 5, 7, 9 and ∞ (the whole set). Text similarity between reports is measured using the open source search engine Apache Lucene (McCandless et al. 2010). Parameter α is used to set a threshold for the level of similarity, ranging from 0 (no similarity at all) to 1 (identical texts). For higher values of α often no similar reports could be found and the approach returns *Unknown*. Applicability is measured as the percentage of responses different from *Unknown*. Accuracy is calculated as the percentage of predictions having a relative error not greater than 25 and 50 %, respectively. For kNN (without threshold), the accuracy is poor. Only 30 % of the predictions lie within the ±50 % range of the actual resolution time for any choice of k. The setting of threshold α creates a tradeoff between applicability and accuracy. For $k=\infty$ (searching all previous defects) and $\alpha=0.9$ (restricted to very similar defects), accuracy is excellent: the average prediction is off by only 7 h and almost every second prediction lies with ±50 % of the actual resolution time. However, in this particular configuration, predictions for only 13 % of the data set are made, i.e., for 87 % of the defects, the approach yields no results as there are no defects with 0.9 level of similarity to be compared with and to obtain estimates from. For lower values of α and $k=\infty$, accuracy decreases; for $\alpha=0.5$, only 30 % of predicted values are within the 25 % error margin, and for $\alpha=0$, it drops to 20 %. In fact, with $k=\infty$ and $\alpha=0$, the approach corresponds to a naïve estimation approach, i.e., the average resolution time of all previous reports is taken as predictor. Finally, the results vary a lot depending on which JBoss subproject is analyzed, and on the type of report. The authors explain their results by the diversity of textual descriptions in the reports and suggest that additional fields, i.e., version information, stack traces and attachments could be exploited to improve the prediction accuracy.

2.2 Replications in Software Engineering

Replication is a complex issue in empirical software engineering (Miller 2005); different terminology is used and little consensus has emerged concerning replication processes and procedures. Moreover, conclusion instability is being raised as a serious threat: effect X discovered in a certain situation does not hold in other situations, and there can be as much evidence for effect X as against it (Menzies and Shepperd 2012). Variance and bias can come from different sources such as sampling, data pre-processing or languages and tools used in the experiment. Hence, there are many variation points to consider when replicating a study (Shull et al. 2008), and different replication strategies are possible (Brooks et al. 2008). An *internal* replication is undertaken by the original experimenters who repeat their own experiment, while an *external* replication is conducted by independent researchers and is considered a critical verification step in experimental research. Furthermore, *exact* (or dependent) replications, i.e., following as closely as possible the procedures of the original experiment, should be distinguished from *conceptual* (or independent) replications, i.e., the same research question is evaluated by using a different experimental procedure where independent variables are changed (Shull et al. 2008) (Juristo and Gómez 2012).

2.3 The Original Study to be Replicated: Clustering Defect Reports

One of the goals of our research is to conduct an external, conceptual replication of the study by Raja (2013) in which she suggests clustering as a technique with potential to predict DRT. Clustering is an unsupervised machine learning technique for identifying homogenous subgroups of elements in a set of data (Jain 2010). Clustering is applied for solving problems such as image segmentation, object and pattern recognition, and information retrieval (Jain et al. 1999). The goal of Raja's study was, as stated by the author, to "*examine the suitability of textual content of defect reports as an early indicator of defect resolution time*" (Raja 2013, p.125). Raja uses the textual descriptions of defect reports in a text mining approach and applies clustering techniques to organize similar defect reports into clusters. The obtained categories, i.e., clusters, differ from project to project, and accordingly, the mean DRT of the defects in each category. Raja thus assumes that "*project specific categorization can significantly distinguish defect resolution time patterns*" (Raja 2013, p.126). Furthermore, she discovers a significant difference in mean DRT of defect reports in different clusters, and proposes that a cluster's mean DRT could be used as an early indicator of the effort required to resolve a defect report belonging to this cluster.

Raja studies five data sets containing defect reports originating from the open source domain, with a total of approximately 14,000 closed defect reports. First, she applies standard data pre-processing for information retrieval: tokenization, elimination of stop words and stemming. When applying a clustering approach, two stages are essential: pattern representation and similarity measurement (Jain et al. 1999). Raja represents each defect report as a word frequency matrix (WFM), and she uses Latent Semantic Indexing (LSI) to reduce the dimension of the WFM. LSI uses singular value decomposition to reduce the size of the sparse WFM (Deerwester et al. 1990). The goal of LSI is to transform individual terms to "concepts" by capturing higher order term co-occurrence (e.g., synonyms). The reduced matric requires much less computational power, and sometimes the retrieval or clustering accuracy improves as well, but not necessarily (Borg and Runeson 2013). A disadvantage of using LSI is that the number of dimensions to keep must be explicitly set, a parameter that is highly dependent on the text collection and thus difficult to configure (Kontostathis 2007).

Raja uses the tool SAS Text Miner to cluster defect reports. She applies automatic clustering based on entropy minimization together with some human intervention (using an interactive feature of SAS Text Miner) to reduce and optimize the obtained clusters. These manual interventions were not explicitly detailed. The final result is three to five clusters of DRs for each project with, for each cluster, a set of the most descriptive terms extracted from their textual content.

Raja performs statistical processing in three steps: first, she performs descriptive statistics on the obtained clusters and notices that "*the average time taken to resolve a reported defect varies significantly across projects and across clusters within the same project*" (Raja 2013, p. 127). Thus, as the second step, she applies one-way ANOVA analysis of variance to test the significance of inter-cluster variation. Beyond independence of sample observations, ANOVA relies on assumptions of normality of the sample and homogeneity of variance. Raja performs the Kolmogorov-Smirnov test of normality and Levene's test of homogeneity. She interprets the results as globally positive, i.e., assumption of normality is established although it was violated in certain clusters. Although Levene's test revealed unequal variance, Raja argues that ANOVA could be performed – because ANOVA can be considered robust enough despite unequal variance – but Raja explains that additional testing with Brown-Forsythe test is

required. Finally, the last step is post-hoc testing using Games-Howell's test to identify the exact pattern of differences among the clusters' means DRT. The results indicate that only some combinations of clusters show statistically significant variation in DRT means.

Table 1 presents a synthesis of the results of the data analysis performed in Raja's study. As the post-hoc testing showed inconsistent results among clusters, Raja seeks an explanation by analyzing two possible factors. First, she analyses key terms appearing in the clusters and their relation to the nature of the software component to which the defect report could be related. For example, if key terms in a cluster are related to interface development, while key terms in another cluster (in the same project) are related to server code, the differences in resolution time can then be related to differences in complexity among the related code. The other factor that Raja considered is the number of software downloads, and thus, the popularity of the software and its potential impact on the maintenance team in terms of priority given to solving the defect.

2.4 Replicability of the Original Study

Support for replications in software engineering has been developed. In González-Barahona and Robles (2012), the authors present a method for assessing the reproducibility of a study, i.e., a way to evaluate a study's ability to be reproduced in whole or parts. The authors propose a reproducibility assessment method based on eight elements of interest extracted from the empirical study to be replicated. In Table 2, we have evaluated the reproducibility of Raja's original study according to this method. This assessment is based on information extracted from the original paper (Raja 2013) and on email exchanges with the author of the paper.

As can be seen from Table 2, Raja's study is difficult to replicate. Concerning the data, although the source is easily identifiable (sourceforge.com software repository), the raw data set, i.e., the set of closed defect reports, are not directly available for retrieval. Furthermore, although we contacted the author and requested the raw data sets, she was unable to provide a set identical to what was used in the baseline experiment. At last, the processed data set (i.e., the set of defect reports after manual cleaning) and the corresponding retrieval and extraction methods are difficult to reproduce because they are poorly documented in the original study

Table 1 Synopsis of data analysis in Raja's study

Statistical test	Goal of the test	Results
Kolmogorov-Smirnov test	Normality distribution of DRT among clusters	Partially positive – for each project data, "*there was a slight violation of normality assumption*" for certain clusters (p.129)
Levene's test	Equal variance of DRT among clusters	Negative – unequal variance for all projects' data (p.129)
One-way ANOVA test	Analysis of DRT variance among clusters	Positive – for each project, DRT mean of at least one cluster differs from all other clusters (p.130)
Brown-Forsythe test	Equality of RT means among clusters	Positive – the results were consistent, across all projects, with the findings of ANOVA (p.130)
Games-Howell's post-hoc test	Identify the exact pattern of differences among clusters' RT means	Partially positive – some of the clusters do not have significant differences in their DRT (p.130)

Table 2 Evaluation of Raja's study reproducibility according to González-Barahona and Robles' method (2012)

	Identification	Description	Availability	Persistence	Flexibility	Assessment	Comments
Data source	Yes	Detailed	Public	Likely	–	Usable	The data source is the public repository Sourceforge; however, the defect reports are not directly downloadable
Retrieval method	Partial	Partial	No	No	–	Difficult	Parameters for extraction queries for are not provided
Raw data set	Yes	Detailed	No	No	–	Difficult	The collection of defect report is not made available by the author
Extraction meth.	Yes	Partial	No	No	–	Difficult	Complex text pre-processing steps briefly described
Parameters	Partial	No	No	No	–	Difficult	The parameters for text pre-processing are not described
Processed data set	Partial	Partial	No	No	–	Difficult	The pre-processed defect reports are not available
Analysis method	Yes	Partial	No	No	–	Usable	Relies on proprietary software tool and specific parameters
Results data set	Yes	Detailed	Yes	Yes	No	Usable	The results are presented in the paper

and have not been made publicly available. Concerning the data analysis method, it is also very difficult to reproduce as it relies on a proprietary software tool (i.e., SAS Text Miner) using specific non-documented processing parameters (i.e., interactive clustering and cluster reduction).

3 Design of the Replication

In this section, we present the design of our conceptual replication of Raja's baseline experiment. In the rest of this paper, we will use the term *baseline experiment* to refer to the set-up of the experiment itself, and the term *original study* to the research results published in (Raja 2013).

In light of the evaluation of replicability presented in the previous section, to be exactly replicated, the original experiment reported in (Raja 2013) requires using the same data mining tool and clustering technique, as well as close collaboration with its author in order to set the various processing parameters consistently. However, we are not interested in an exact replication of this study. Indeed, beyond replication, our goal is to assess the accuracy of the suggested method and explore its potential applicability in real world settings. From this perspective, the clustering step should rely on non-proprietary software tools and – more importantly – avoid any manual steps.

We chose to conduct an independent, conceptual, non-exact replication of Raja's study. Dependent (exact) replications, i.e., all the conditions of the experiment are the same or very similar, are recommended for understanding all the factors that influence a result (Shull et al. 2008). A successful exact replication confirms the experimental results from the baseline experiment and provides support to the theoretical results. In other words, an exact replication consolidates the internal and conclusion validity of the baseline experiment (Rosenthal 1991, p.5). On the other hand, an independent, conceptual, non-exact replication, i.e., when one or more major aspects of the experiment are varied, is more convincing in demonstrating that an effect or a technique is robust to changes with subjects, settings, and materials (Kitchenham 2008) (Juristo and Gómez 2012). Indeed, we seek to understand how sensitive the results are to different operationalization, i.e., "*variation in results in response to changes to treatment application procedures*" (Gómez et al. 2014, p. 1040). When successful, a non-exact conceptual replication extends the potential validity of the theoretical knowledge gained from the baseline experiment and contributes to its external validity (Gómez et al. 2014; Rosenthal 1991).

Table 3 compares the settings for the baseline experiment and for the replication we have designed. In the baseline experiment, Raja constructed the sample by selecting data sets from projects "*with significant activity and usage over an extended period of time"* and that had *"at least 3 years of historical data available"* (Raja 2013, p. 122). Beyond the availability of the defect reports, the author did not mention other conditions for the inclusion of a data set in the sample. In line with Raja's selection, we selected two publicly available open source data sets from the Android and the Eclipse development projects. These two data sets are considerably larger than any of the five projects selected in the baseline experiment. The rationale for our selection is two fold: first, as our goal is to test the clustering approach using a simulation scenario, we considered that it is important to have large data sets so that multiple scenarios can be explored. Second, in accordance with principles for conceptual and non-exact replications (Kitchenham 2008) (Juristo and Gómez 2012), varying the data sets and their size contributes further to evaluating the external validity of the baseline experiment. Also, to

Table 3 Comparison of the baseline experiment and the replication we conducted

	Baseline experiment	Conceptual replication
Studied projects (#closed defects)	OSS: FileZilla (956), jEdit (1682), phpMyAdmin (1521), Pidgin (2689), Slash (3580)	OSS: Android (4684), Eclipse (4158) Proprietary: Company A (6790)
Tool support	SAS Text Miner	RapidMiner
Manual preprocessing	Removal of report clones, manual examination of issues closed in <1 h	Removal of report clones
Automatic preprocessing	Stop words, stemming, term weighting (entropy weights), Latent Semantic Indexing (LSI)	Stop words, stemming, term weighting (TF-IDF[a])
Clustering algorithm	Entropy minimization	K-means clustering
Manual cluster tuning	Several labor-intensive steps	None
# of clusters	3-5	4

[a] TF-IDF (Term Frequency-Inverse Document Frequency) is a standard weighting scheme from information retrieval, based on the length of the document and the frequency of the term, both in the document and in the entire document collection (Singhal 2001).

further challenge the external validity of the baseline experiment, we also extracted defect reports from a large proprietary project at an industrial partner, referred to as Company A. Indeed, comparison to a proprietary development context is highlighted as an important research endeavor (Robinson and Francis 2010).

3.1 Description of the Data Sets

The data sets are presented in Table 4. Android is an operating system designed for touchscreen mobile devices such as smartphones and tablet computers. The Android Open Source project, led by Google, develops and maintains the system. The defect reports were collected from the public issue tracker on Dec 6, 2011, and made available as a 60 MB XML file containing 20,176 reports by Shihab et al. as part of the mining challenge at the 9th Working Conference on Mining Software Repositories (Shihab et al. 2012).

The Eclipse project consists of several subprojects with their own respective defect reports. The defect reports of the Eclipse project, managed in Bugzilla databases, have been used for DRT prediction in several previous studies (Panjer 2007), (Giger et al. 2010), (Lamkanfi and Demeyer 2012), which motivates inclusion in this study. We extracted 4158 defect reports with "closed" status from the subproject "Eclipse Platform" in March 2012. The raw data sets for both projects are available for download on a publicly available website.[1]

Company A is a large multinational company active in the power and automation sector. The software development context is safety-critical embedded development in the domain of industrial control systems, governed by IEC 61511 (IEC 2014). The number of developers is in the magnitude of hundreds a project has typically a length of 12 to 18 months and follows an iterative stage gate project management model. At Company A, DRs are managed in Merant Tracker (part of Merant Professional, acquired by Serena Software in 2004, http://www.serena.com/). We extracted 7697 defect reports corresponding to a single development program, which in turn is divided into several development projects.

3.2 Overview of the Study

Figure 2 is a synopsis of the main steps in our study. The prediction simulation part is presented in sections 5 and 6. For all defect reports, we extracted and merged the titles (one line summaries of defect reports) and the descriptions. Similar to the pre-processing in the baseline experiment by Raja, we manually cleaned the collected defect reports from the three projects prior to running the clustering process. In cases of defect reports with identical textual descriptions, i.e., report clones, we only included the first submitted defect report in the experiment. Also, we excluded all defect reports that were still open at the time of the data collection. However, while the baseline experiment manually investigated all closed defect reports with DRTs of less than 1 h to ensure that they were not invalid, we did not perform this cleaning step as all data sets did not provide such fine-granular time information. Moreover, as argued by Menzies et al. (2011a), while some data cleaning should be performed before using mining tools, perfect data often comes with a too high cost. Menzies et al. (2011a) also claim that most data mining approaches can handle a limited amount of noise. Thus, we only conducted data cleaning that can be performed automatically (i.e., removal of duplicates) and otherwise took the data unfiltered as input to the clustering step.

[1] http://serg.cs.lth.se/research/experiment-packages/clustering-defect-reports/

Table 4 Overview of the data sets used in the study

	Android	Eclipse	Company A
Product domain	Telecom, operating system	Application development, IDE	Process automation, industrial control system
Project characteristics	Open source, managed by Google	Open source, managed by non-profit organization	Proprietary, iterative stage-gate process, safety-critical
Initial release	2008	2004	2003
#Users / #Downloads	~500 million activated devices	~5 million downloads	Limited user base
Total #extracted defect reports	20,176	9,0691	7697
Sanitized # defect reports	4684	4158	6790
Average #characters / defect report	821	3699	1208
Main author of defect reports	OSS community	OSS community	Internal developers/testers

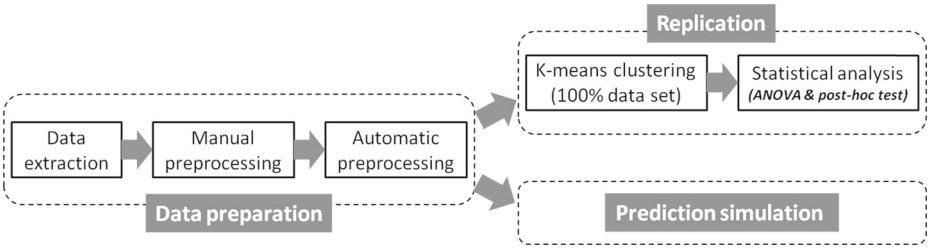

Fig. 2 Main steps in our study (the prediction simulation part is presented in sections 5 and 6)

We used RapidMiner, an open source data mining tool (Hofmann and Klinkenberg 2014), for clustering the defect reports based on their textual content. Before executing the clustering algorithm, the following sequence of preprocessing steps was performed on the defect reports in RapidMiner:

a. Defect reports were tokenized, using non-letters for splitting.
b. Defect reports were transformed to lower case.
c. Stop words were removed from the defect reports, using the filter for English provided by RapidMiner.
d. Defect reports were stemmed using Porter's stemmer.
e. Defect reports were filtered to include only tokens between 2 and 999 characters.
f. Document vectors were created from the remaining terms with TF-IDF weights.

Determining the number of clusters (K) in unknown data, referred to as "*the fundamental problem of cluster validity*" by Dubes (1993), is acknowledged as a difficult problem. In a survey of clustering algorithms, Xu and Wunsch II (2005) identified a large number of proposals for estimating an appropriate K. They organize the proposals as either 1) *Visualization* of the data set, 2) Construction of *stop rules*, 3) Optimization under *probabilistic mixture-model frameworks*, and 4) Other *heuristic* approaches. Unfortunately, the various estimation methods are often unreliable (Frost and Moore 2014). On the other hand, there exist clustering algorithms that adaptively adjust the number of clusters rather than use a fixed number. However, for these algorithms the challenge of setting the number of clusters is converted to parameter tweaking of other parameters (Xu and Wunsch II 2005). Consequently, to determine a feasible number of clusters, the researchers must use their skills to explore the data and motivate the selection of K.

In Raja's baseline experiment, the author manually determined K by iterative clustering using entropy minimization, a process she described as labor intensive. Raja reported that the number of clusters was iteratively refined (i.e., until a stop criterion was reached) using an interactive feature of SAS Text Miner, allowing easy creation of start and stop lists through a manual analysis of the preliminary results. Raja also conducted a manual filtering of terms, to ensure that only relevant terms were included in the final clusters, e.g., removing misspelled words and terms related to source code. This process resulted in decreasing the number of clusters from 40, the default maximum number of clusters in SAS Text Miner, to between three and five clusters of defect reports in the five data sets.

In line with our aim to automate the clustering step, we did not want to perform any manual tuning of the clusters produced by RapidMiner (using K-means clustering with default

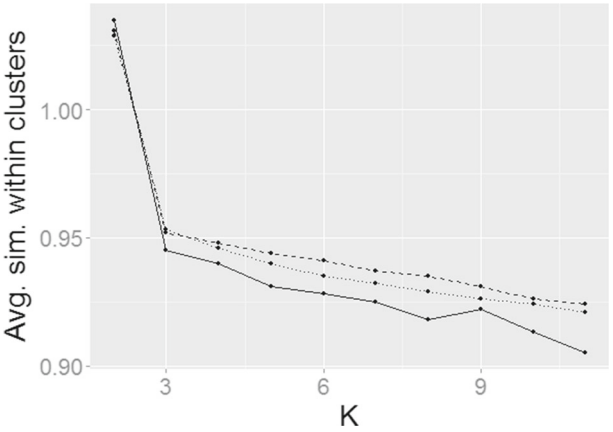

Fig. 3 The average within centroid distance as a function of the number of clusters (K). The curves corresponding to the three data sets display similar behavior (*solid line* = Android, *dashed line* = Eclipse, *dotted line* = Company A). The elbow method suggests that K should be at least 3 for all data sets

settings[2]). Instead, we based our selection of K on our pre-understanding gained from Raja's original study, complemented by the heuristic *elbow method*. Raja concluded the number of clusters to between three and five for all her five data sets, indicating that a feasible number of clusters would be around a handful. The elbow method means to visually identify the point on a curve with the maximum curvature (Tibshirani et al. 2001). Figure 3 shows the average within centroid distance, an internal validity measure for clusters, as a function of the number of clusters (Hofmann and Klinkenberg 2014). The figure shows, for all three data sets, that the distinct decrease appears as K goes from 2 to 3. Thus, the elbow method suggests that at least three clusters should be identified among the defect reports from Android, Eclipse, and Company A. Finally, we set K to 4 for all our three data sets, corresponding to the average of what Raja reported as optimal for her five data sets, a value that is also higher than the lower bound suggested by the elbow method. However, the best validation is to test if the choice of K actually leads to clusters with statistically significant differences.

In line with the baseline experiment, we conduct the same types of statistical tests (cf. Table 1), using R software packages: normality distribution test for DRT (Kolmogorov-Smirnov test), DRT homogeneity of variance among clusters (Levene's test), ANOVA analysis of variance together with Brown-Forsyth robust test, and post-hoc inter-cluster analysis of variance (Games-Howell test).

4 Results of the Replication

In this section, we present the results of our replication study and compare them with those of the baseline experiment.

We first present descriptive statistics about the data sets used in our study. Figure 4 shows how DRT is distributed for each project. All data sets are clearly left-skewed and, obviously, they are not normally distributed.

[2] The RapidMiner process is exported to a file available in the same repository as the raw data (http://serg.cs.lth.se/research/experiment-packages/clustering-defect-reports/)

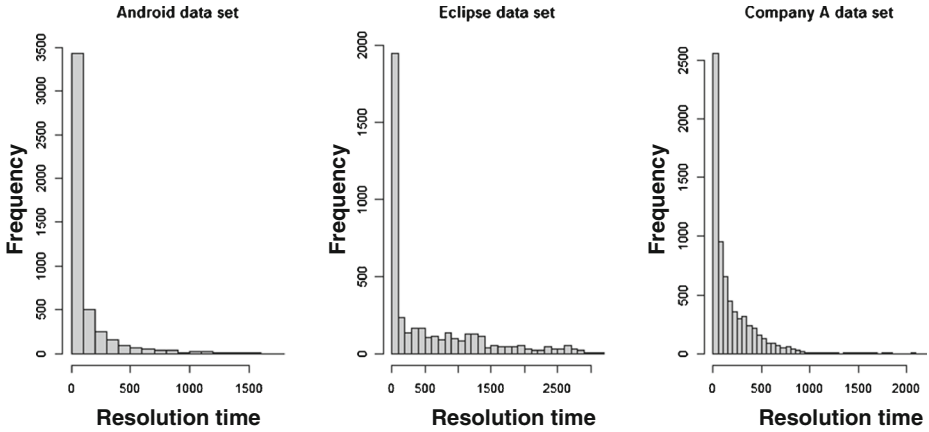

Fig. 4 Distribution of DRT for the three data sets (resolution time expressed in days)

Table 5 summarizes the three data sets. Skew and kurtosis values are very high, although Eclipse seems slightly less heterogeneous. To confirm the non-normality of the data sets, we applied the Kolmogorov-Smirnov test with parameters estimated from the sample and D threshold values estimated at 0.05 level of significance according to the formula in (Lilliefors 1967). The results consistently show D statistic larger than the threshold with p-values very close to 0 (i.e., smaller than 0.05) which implies a rejection of the null hypothesis, i.e., the DRT in each data set is not normally distributed.

The next step is clustering the set of defect reports according to textual similarity. This step was a complex and sophisticated process in the baseline experiment. Raja (2013) conducted several iterations of the clustering step. In the tool she used, i.e., SAS Text Miner, the default number of maximum permissible clusters is 40; it allows for outliers to be treated as individual clusters. Although none of the 5 projects processed in the original study did actually yield more than ten clusters, the author used an interactive feature of the SAS Text Miner tool to create start and stop word lists and, for each project, to optimize these two lists by identifying irrelevant terms and discrepancies such as misspelled words and code presence. This labor intensive manual selection of relevant terms minimized noise and helped in obtaining a small number of clusters that could each be described by a set of highly relevant keywords. Thus, the clusters obtained in the baseline experiment tend to be a collection of defect reports strongly related to each other through common use of terms, and can therefore be considered as exhibiting a certain form of semantic unity.

Table 5 Descriptive statistics of the data sets (significance codes for p-value: <0.001'***' <0.01'**' <0.05'*')

Data set	#Defect reports	DRT mean (days)	DRT std. dev.	Skew	Kurtosis	Normality assumption (Kolmogorov-Smirnov test)
Android	4684	108	208	3.34	13.33	$D_{threshold}$=0.013, D=0.303 p-value=0.0000 ***
Eclipse	4158	551	726	1.4	1.12	$D_{threshold}$=0.014, D=0.224 p-value=0.0000 ***
CompA	6790	177	215	2.04	6.16	$D_{threshold}$=0.011, D=0.205 p-value=0.0000 ***

In this study, we use the RapidMiner tool for text mining and clustering. We deliberately fixed the number of clusters to four, as explained in section 3. We did not intervene manually in the clustering step as we opted for a fully automatic process that goes beyond replication of the baseline experiment, by also evaluating the applicability and predictive quality of using the clustering approach for DRT prediction, as suggested by Raja (2013, pp. 134). The boxplots for DRT for the clusters in each data set are shown in Fig. 5.

The detailed results of the clustering are shown in Table 6. The average DRT varies across the clusters in each data set. In line with the original study, we performed the normality Kolmogorov-Smirnov test on each cluster. The D statistic is always larger than the threshold value and the p-values are almost always close to 0; thus, in all clusters, the DRT values are not normally distributed.

Raja suggested in the original study that the differences in DRT means across clusters can provide some insights into the patterns of defect resolution time, and that the patterns can be used to predict the DRT of future defect reports (Raja 2013; p. 127). However, she continues, it needs to be established if the differences between the mean DRT for each cluster are statistically significant. For this end, a one-way analysis of variance ANOVA test is suitable. Three assumptions need to be met: (i) independence of the sample observations, (ii) normality of the sample, and (iii) approximately equal variance (homogeneity) for each group. The first assumption is met as reports are randomly ordered and each report is independent. According to the results of the Kolmogorov-Smirnov test, the second assumption is not fully met in our data. However, the normality of the sample was violated in the original study as well. Raja (2013) indicates that certain clusters she obtained did not satisfy the Kolmogorov-Smirnov test (p. 129). However, she considers that "since the sample size is very large", the assumption of normality is established. Indeed, the ANOVA test is considered as reasonably robust to deviations from normality. Even fairly skewed distributions might not be problematic, as long as the groups are similarly skewed (Sawilowsky and Blair 1992).

Concerning the homogeneity of variance assumption, like in the baseline experiment, we conducted the Levene's test. Table 7 shows the obtained results. Our results are identical to the baseline experiment, i.e., the variance is unequal among the clusters in each data set and the null hypothesis is rejected.

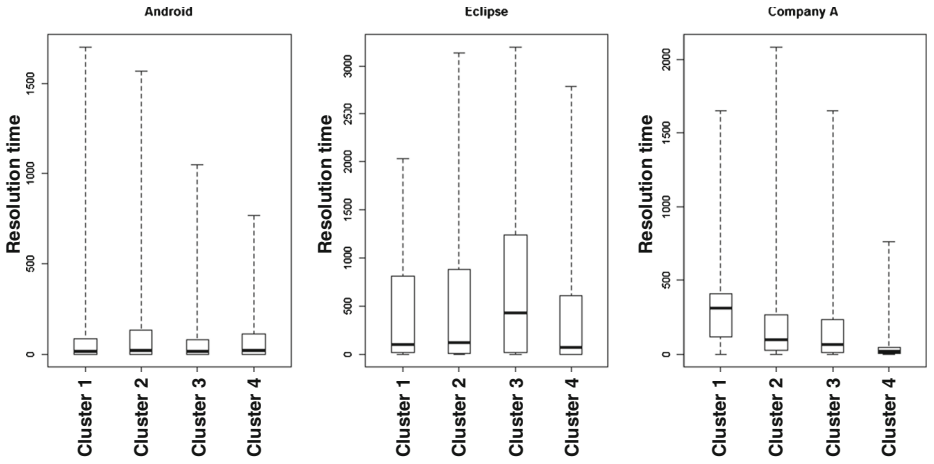

Fig. 5 DRT for the four clusters in each data set

Table 6 Descriptive statistics for the four clusters in the three data sets (significance codes for p-value: <0.001'***' <0.01'**' <0.05'*').

	Android				Eclipse				Company A			
	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 1	Cluster 2	Cluster 3	Cluster 4
# defect	999	2876	734	75	160	2944	704	350	399	4408	1542	441
RT mean	93	122	73	107	415	537	721	385	284	187	158	53
RT std. dev.	191	229	132	182	514	735	775	540	189	224	201	98
$D_{threshold}$ (1)	0.028	0.017	0.033	0.102	0.07	0.016	0.033	0.047	0.044	0.013	0.023	0.42
$D^{(1)}$ =	0.313	0.298	0.291	0.316	0.254	0.233	0.188	0.238	0.070	0.202	0.216	0.294
p-value$^{(1)}$ =	0.0000***				0.0000***				0.038*	0.0000***		

(1) $D_{threshold}$, D and p-value for Kolmogorov-Smirnov test ($D_{threshold}$ calculated according to formula in (Lilliefors 1967))

Table 7 Test of unequal variance among the four clusters in the three data sets. The variance of DRT is not homogeneous

	Levene's test
Android	F=34.247 p-value=0.0000 ***
Eclipse	F=32.228 p-value=0.0000 ***
CompA	F=80.873 p-value=0.0000 ***

Because of the violation of the third assumption, i.e., equal variance, the baseline experiment required additional testing beyond the ANOVA test. Raja performed the Brown-Forsythe robust test of equality of means. Both the ANOVA and the Brown-Forsyth tests in the baseline experiment exhibited consistent results (Raja 2013; p.130).

In line with the baseline experiment, we conducted both one-way ANOVA and Brown-Forsythe tests on the data sets, as presented in Table 8. The results of both tests are consistent, and corroborate the results obtained in the baseline experiment: in each data set, the mean DRTs in each cluster are significantly different from each other.

The ANOVA test establishes that the differences in the means of the various clusters are statistically significant. A post-hoc comparison is performed to identify how the means DRTs of the clusters in each data set differ from each other. Post-hoc testing matches all data in each cluster to data in every other cluster, calculates the differences in means and estimates the level of statistical significance, i.e., the p-value. As the data is not normally distributed and homogeneity of variance is not assumed, the original study applied the Games-Howell post-hoc test.

The results of the post-hoc testing are presented in Table 9. We report only the p-value as we are mainly interested in knowing if the mean DRT in each pair of clusters are significantly different. For the Company A data set, all clusters are significantly different from each other, while for the other two data sets, one or two pairs are not significantly different from each other. This is, again, consistent with the baseline experiment's results in which, for all data sets, certain pairs of clusters were not significantly different from each other.

The knowledge gained from a non-exact, conceptual replication enhances the external validity of the results and extends the potential generalizability. In the replication that we have conducted, three important factors were varied: the data sets, the mining tool and the clustering technique. The results we obtained are consistent with those obtained by Raja (2013): the results of the ANOVA and Games-Howell tests for the three data sets are almost identical to those obtained in the baseline experiment, i.e., there is a statistically significant difference in the mean of the DRT for the various report clusters in all data sets. The statistical validity of these tests, the ANOVA test in particular, is constrained by two important conditions: homogeneity of variance and normal distribution. The data sets used in our replication did not fully satisfy these constraints. However, this was the case also in the baseline experiment (Raja 213, p. 129). Together with Raja, we consider that, since the sample size is very large,

Table 8 Analysis of variance for the four clusters in the three data sets

	One-way ANOVA test	Brown-Forsythe test
Android	F=13.47 p-value=0.0000 ***	F=14.03 p-value=0.0000 ***
Eclipse	F=21.44 p-value=0.0000 ***	F=18.78 p-value=0.0000 ***
CompA	F=92.99 p-value=0.0000 ***	F=48.5 p-value=0.0000 ***

Table 9 P-values for Games-Howell post-hoc test (bold values are not significant at α=0.05 threshold)

Android	Cluster 2	Cluster 3	Cluster 4
Cluster 1	0.0004 ***	0.0472 *	**0.9127**
Cluster 2		0.0000 ***	**0.8930**
Cluster 3			**0.3840**
Eclipse			
Cluster 1	0.0250 *	0.0000 ***	**0.9308**
Cluster 2		0.0000 ***	0.0000 ***
Cluster 3			0.0000 ***
Company A			
Cluster 1	0.0000 ***	0.0000 ***	0.0000 ***
Cluster 2		0.0000 ***	0.0000 ***
Cluster 3			0.0000 ***

the assumption of normality is established and ANOVA analysis is robust enough to be applied to data with slightly unequal variance.

Moreover, some of the results we obtained are more distinct than those in the baseline experiment. In the baseline experiment, the post-hoc test was globally successful for all pairs of clusters in the studied projects (Raja 213, p. 131). In other words, for all projects, there was at least one pair of clusters for which the result of post-hoc test was not significant. In our replication, for the data set from company A, the results of the post-hoc are indeed significant for all pairs of clusters.

5 Design of the Simulation Scenario

To test the suggestion made in (Raja 2013), i.e., that variations of DRT in clusters of defect reports can be used as an early indicator of the effort required for their resolution, we proceed in a deductive manner. Starting out from the theoretical claim, we formulate a hypothesis and design an operational context to gather observations, analyze the results and seek to confirm – or refute – the hypothesis. The hypothesis is that clusters of defect reports with significantly different DRTs can be used to predict DRTs of incoming defect reports. To define an operational context, we adopt a simulation approach (Müller and Pfahl 2008). A simulation involves imitating the behavior of some process (Walker and Holmes 2014), in our case the inflow of defect reports. The challenge lies in designing a plausible and feasible operational scenario that provides ground for gathering relevant data. A possible approach, for example, would be to cluster the defect reports each time a new defect report arrives. The clustered data would then include both closed and open defect reports. A possible prediction of the DRT of a defect report would then be the mean DRT of the closed defect reports in the cluster to which the incoming defect report belongs. Such an *event-driven* scenario is however barely feasible as clustering is a computationally intensive operation. Furthermore, it can hardly be redone each time a single defect report is submitted. The clustering time can easily extend the average time between incoming defect reports.

We opted instead for a scenario that simulates the application of the proposed prediction method at various points in time (Fig. 6). The evaluation scenario we implement is inspired by the k-tail simulation technique in which data after a given point are used for the test set (Walker and Holmes 2014). In our context, the test set is the subset of defect reports for which DRT is

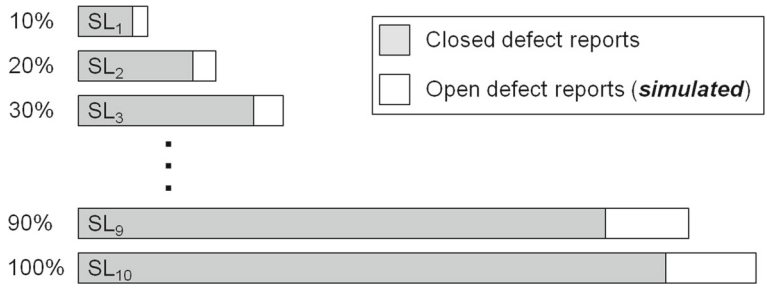

Fig. 6 Cumulatively growing data slices

predicted. As our simulation was not linked to a real-world software development context, we did not have a practical criterion to decide what points in time would be suitable to split the data set into cumulatively growing subsets of the total data set available. Such time-related criterion could be defined, for example, by a specific day of the week, or an hour of the day. For example, it might make sense to estimate new defect reports once a week. Then we would do weekly estimations based on the number of closed defect reports available at a specific day of the week. Another criterion could be to estimate once a certain amount of new defect reports is waiting for prediction, or there could be a combination of criteria. In our simulation, with lacking criteria from a real-world context, we were mainly interested in demonstrating the mechanics of the approach and thus chose a somewhat arbitrary way of splitting the available data so we could simulate the repeated DRT prediction of sequential chunks of newly incoming defect reports along the timeline. We simply divided each data set of defect reports into ten subsets, the first corresponds to a 10 % slice of the whole data set ordered by the submission time, the second to 20 %, the 3rd to 30 %, etc. (cf. Fig. 6). These subsets represent different points in time when the defect reports could have been clustered to predict DRTs. Thus, larger subsets of defect reports imply that more historical data are available.

Theoretically, one could argue that a prediction should be made whenever a new (open) defect report arrives. In order to simulate this situation, we would have had to repeat the clustering and prediction steps thousands of times for each data set. This could become, even when fully automated, rather time consuming. On the other hand, in practice, it might not even be so that there is a need to predict every newly incoming defect report immediately. Instead, estimation might happen at a certain fixed point in time, e.g., in the evening (or morning) of a work day, or at the end (start) of a week. In those cases, there could be already several defect reports be waiting for prediction. In our simulation, we tried to find a reasonable balance between predictive power (which is highest when only one new defect report is to be predicted) and saving of resources (which is lower the less predictions have to be made).

At each point in time when a prediction needs to be made, the whole data subset is clustered and the DRTs of newly incoming defect reports uniformly receive the mean value of the cluster to which they belong as predictor. This is followed by an evaluation step in which we compare the predicted DRT with the actual DRT. Recall that in our simulation we only do as if the defect reports were newly incoming. In reality, of course, we know their actual DRTs. The whole process is shown in Fig. 7.

The simulated number of open defect reports is defined as the 10 % of the subset at its tail (cf. the white regions in Fig. 6). For example, if we consider the Android data set (Table 4), the 40 % slice contains 1874 defect reports ordered according to their time of arrival. The

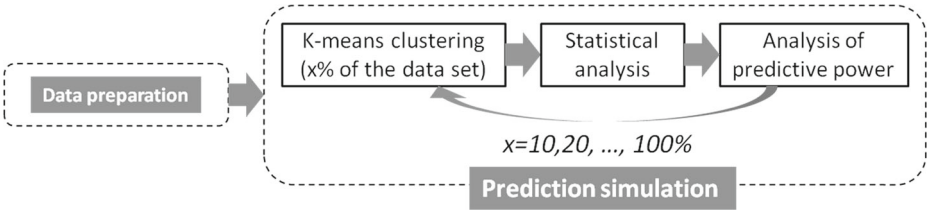

Fig. 7 Synopsis of the test part of the study (cf. sections 3 for data preparation steps)

simulated subset of open defect reports would then contain 187 defect reports. The relative size of the simulated open defects subset is to be considered as a parameter in the simulation scenario. We will refer to this parameter as the *Simulation Size Factor* or SSF. It is possible to envision different values for this parameter. For the Android data set and the 40 % slice for example and for SSF=0.3, the size of the simulated open defects subset would be equal to 562 defect reports, and for SSF=0.5, equal to 937 defect reports. We consider 0.5 as the largest possible value for SSF. This is summarized in the following definitions:

$Size(SL_i)$ = Size of the i^{th} slice SL_i, (i =1, 2, 3, …, 10)
SSF = *Simulation Size Factor*, a parameter to define the size of the subset of simulated open defects reports, (0.1 < = SSF < =0.5)
$Size(SimODR_i) = Size(SL_i) * SSF$, (size of the subset of simulated open defects reports for slice i)

To enact the simulation scenario, at each point in time and for each data slice, we define a fixed number of defect reports as a *test set* (cf. Fig. 8). The goal of the simulation scenario is to evaluate the predictive quality of the clustering approach at all ten simulation points. For this reason, to make results comparable, the number of predicted defect reports in the test set should be constant for all data slices. Accordingly, and in order to have a fixed size test set, we set the size of the test set based on the size of the subset of simulated open defect reports for the smallest slice (i.e., the 10 % slice). At SSF=0.1, 0.3 and 0.5, this correspond to 1, 3 and 5 % of the whole data set respectively.

$nb\_PR = Size(SimODR_1) * SSF$, number of predicted defects (i.e., size of test set)

In case of the Android data set for example, at SSF=0.1, nb_PR is equal to 46. For the Eclipse and Company A data sets at SSF=0.1, the size of the test set is 42 and 68 defect reports

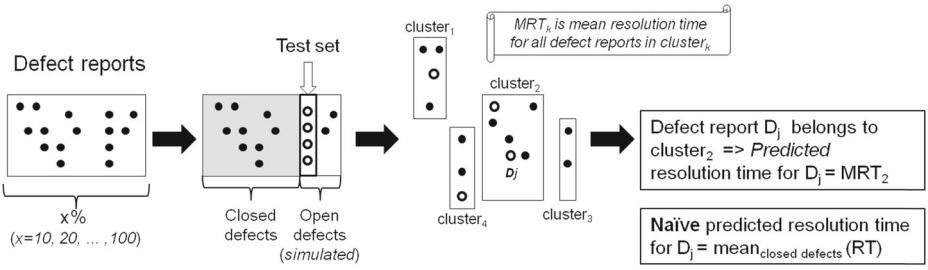

Fig. 8 Overview of the test scenario for DRT prediction

respectively. For larger values of the SSF parameter, the size of the test set grows proportionally. For the Android data set at SF=0.5, nb_PR is 234, i.e., 5 % of the size of the entire data set.

At each point in time, we use the information made available from the clustering as a prediction of the DRT for the new defect report (cf. Fig. 8). In this way, we replay the history of the issue management system and simulate an inflow of defect reports for our clustering based approach. Our test scenario constitutes 10 experimental runs similar to a k-tail evaluation (Matejka et al. 2009), a setup retaining the order of sequential data. For each run, a fixed number of recent defect reports (1 % of the entire data set at SSF=0.1) are simulated as being open and used for evaluation, and all defect reports submitted before that point in time are considered training examples.

To measure the predictive quality of the prediction approach at each point in time, we use the Magnitude of Relative Error (MRE) accuracy indicator (Shepperd and Kadoda 2001):

$$\mathrm{MRE(d)} = \frac{|e - \hat{e}|}{e}$$

We assess the predictive power of the clustering approach by reporting the fraction of predicted DRTs where MRE are below 25 and 50 %. This corresponds to the fraction of predicted DRTs that are within ±25 and ±50 % of the true DRTs. These two indicators are referred to as Pred(0.25) and Pred(0.5) respectively. We only consider absolute differences, i.e., we assume that both over and underestimation is equally unfortunate, in line with previous evaluations of DRT predictions (Weiss et al. 2007). Pred(x) is calculated in the following manner:

N(k) = Number of defects to be predicted belonging to cluster *k*
∑N(k) = 1 % of the total number of defects in a data set
RT(d, k) = Actual resolution time for defect *d* in cluster *k*
RTpred(d, k) = Predicted resolution time for defect *d* in cluster *k*

$$\mathrm{Res(d,\ k,\ x)} = 1 \text{ if } \frac{|\mathrm{RTpred(d,\ k)} - \mathrm{RT(d,k)}|}{\mathrm{RTpred(d,k)}} < x,\ 0 \text{ otherwise}$$

$$\mathrm{Pred(x)} = \frac{\sum \mathrm{Res(d,k,\ x)}}{\sum \mathrm{N(k)}} \quad ,\mathrm{k} = 1,\ \ldots,\ 4$$

Measuring Pred(x) at ten points in time enables us to assess how an increased number of available defect reports affect the predictive power. To enable a comparison with a less complex prediction approach, we also predict the resolution times of newly arrived defect reports using the mean DRT of all available closed defect reports, i.e., the average resolution time for all closed defects (cf. Fig. 8) . We refer to this approach as naïve predictions, in line with related work by (Weiss et al. 2007). Consequently, the same formula for predictive power measurement will be applied for naïve predictions, and will be used for comparison purposes, i.e., to evaluate the clustering based prediction approach.

6 Results of the Simulation Scenario

We present the results from the simulation study that applies the clustering approach to predict defect resolution time (DRT). The underlying idea of the approach, as suggested in Raja's study, is the significant variation in differences between mean DRTs of clusters of defect

reports. Accordingly, the first step in the simulation study is to analyze these conditions for all the data sets used in the simulation scenario. We conduct the two most important statistical tests to check these conditions: the ANOVA test for the analysis of variance, which we consider as robust enough with large data sets even if the data is not normally distributed (cf. section 3); and the Games-Howell post-hoc test to check the significance of variation of DRT among clusters. Table 10 and Fig. 9 present the results of the ANOVA tests and the post-hoc analysis respectively, for the 10 data slices from each data set.

The results are slightly different for the three data sets: for the Company A data set, the ANOVA test is positive for all 10 data slices; for the other two data sets, Android and Eclipse, the test tends to be negative for smaller slices. The post-hoc analysis confirms a certain difference among the data sets. In terms of inter-cluster variation in DRT, the data slices from Company A are more compliant to the statistical conditions when compared with the other two data sets (cf. Fig. 9).

Thus, the theoretical assumptions for the DRT prediction method suggested in Raja (2013) are not systematically met by all the data slices in the simulation scenario. However, although the ANOVA test is not positive for certain Android and Eclipse subsets, we decided to include them anyway in the simulation scenario. Indeed, it is interesting to see whether the violation of the statistical assumption has any effect on the results of the simulation scenario, i.e., whether any differences can be observed in terms of prediction accuracy.

Figures 10 and 11 visualize the results of the test scenario. In these figures, the solid red line corresponds to the prediction accuracy using the clustering approach, while the dashed blue line corresponds to the accuracy of the "naïve" prediction (c.f. section 5). Each figure presents three accuracy graphs, one for each data set: Android, Eclipse and Company A respectively. Furthermore, each figure corresponds to a different accuracy threshold, i.e., Pred(0.25) and Pred(0.5) respectively.

Figure 10 presents the accuracy graphs at 25 % error threshold, i.e. the percentage of DRT predictions where the error margin from the actual DRT does not exceed 25 %. The graphs look very irregular and the accuracy is globally below 20 %, i.e., in only 20 % of the DRT predictions did the prediction accuracy result in an estimate within the 25 % error margin. At the smallest data slice, i.e. the 10 % slice, the accuracy is better with the Eclipse data set, but it falls to 0 for other slices. Finally, when compared with the "naïve" approach, the clustering

Table 10 ANOVA test of analysis of variance for the clusters in each data slice of the simulation scenario (significance codes for p-value: <0.001'***' <0.01'**' <0.05'*', bold values are not significant)

Subset	Android	Eclipse	Company A
10 %	(*F*=**1.551) 0.201**	(*F*=**0.383) 0.765**	(*F*=11.45) 0.0000 ***
20 %	(*F*=3.903) 0.0087 **	(*F*=**2.392) 0.0673**	(*F*=27.12) 0.0000 ***
30 %	(*F*=2.791) 0.0393 *	(*F*=8.157) 0.0000 ***	(*F*=36.6) 0.0000 ***
40 %	(*F*=**0.176) 0.912**	(*F*=18.43) 0.0000 ***	(*F*=28.42) 0.0000 ***
50 %	(*F*=**2.441) 0.0625**	(*F*=19.74) 0.0000 ***	(*F*=33.9) 0.0000 ***
60 %	(*F*=8.381) 0.0000 ***	(*F*=20.82) 0.0000 ***	(*F*=30.95) 0.0000 ***
70 %	(*F*=**2.745**) 0.0416 *	(*F*=46.72) 0.0000 ***	(*F*=43.21) 0.0000 ***
80 %	(*F*=9.374) 0.0000 ***	(*F*=13.8) 0.0000 ***	(*F*=32.42) 0.0000 ***
90 %	(*F*=10.9) 0.0000 ***	(*F*=39.58) 0.0000 ***	(*F*=89.25) 0.0000 ***
100 %	(*F*=13.47) 0.0000 ***	(*F*=21.44) 0.0000 ***	(*F*=92.99) 0.0000 ***

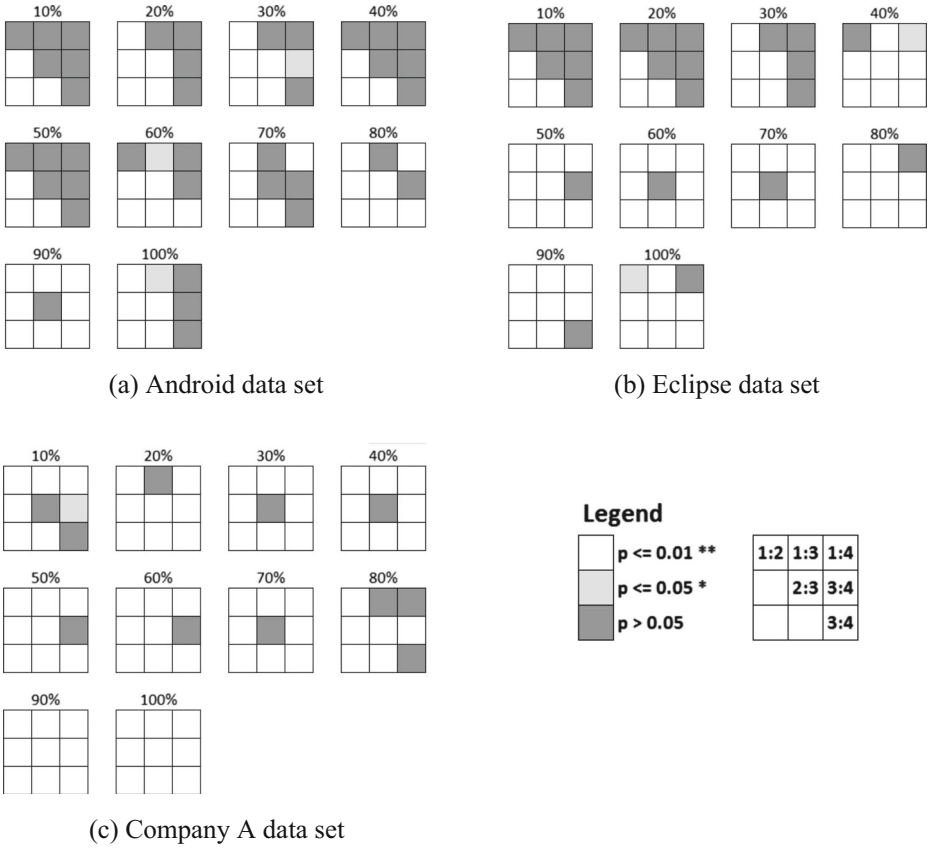

Fig. 9 Games-Howell post-hoc test for the clusters in each data slice of the simulation scenario (**a**) Android data set (**b**) Eclipse data set (**c**) Company A data set

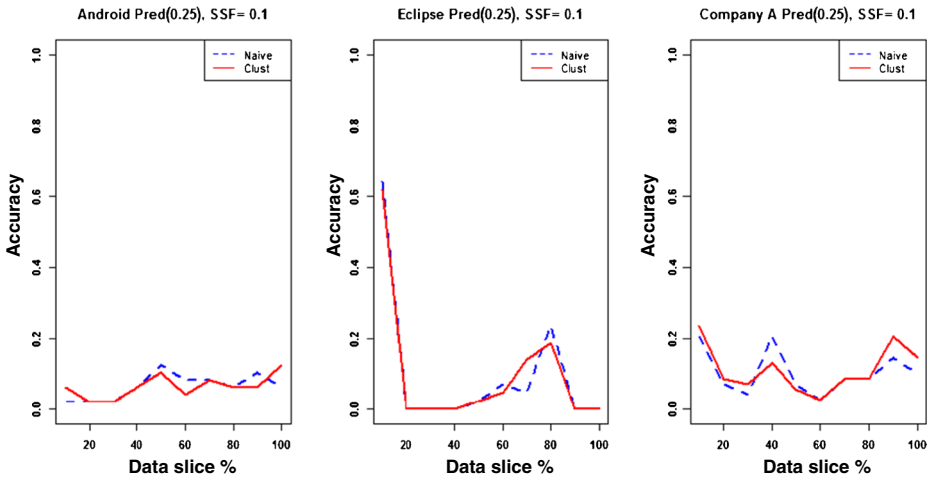

Fig. 10 The DRT prediction accuracy for the three data sets, error threshold = 25 %

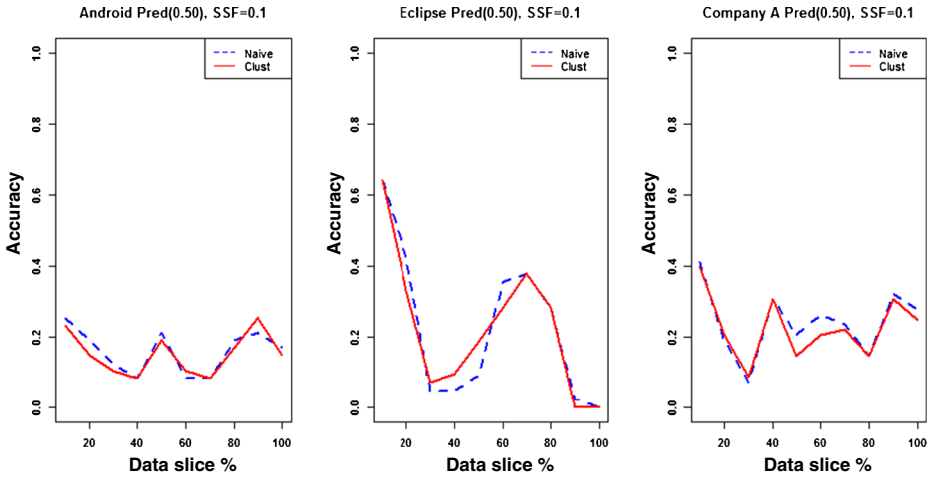

Fig. 11 The DRT prediction accuracy for the three data sets, error threshold = 50 %

approach to DRT prediction seldom provides more accurate results. This is true for the three data sets, and moreover, the "naïve" prediction is sometimes even slightly more accurate.

The accuracy graphs at 50 % error threshold are presented in Figure 11. At this relatively high level of error margin, the prediction accuracy improves. The prediction accuracy reaches 40 to 60 % for the 10 and 20 % slices in the Eclipse data. The same irregular patterns are however still observed; The prediction accuracy for the Company A data set is again slightly more regular than for the Eclipse data set, but the naïve approach is in all cases almost as accurate as the clustering approach, an observation that applies also for the Android data set.

Analyzing the accuracy figures and the possible relation with the results of the ANOVA and the post-hoc tests (cf. Table 10 and Fig. 9), there seems to be no clear pattern of effect. For those slices where the ANOVA test failed, i.e., Android slices 10, 40, 50 % and Eclipse 10 and 20 %, the prediction accuracy is as irregular as for the rest. For the Android data set, the prediction accuracy at 40 and 50 % is around the average, i.e., ~10 at Pred(0.25) and ~15 % at Pred(0.50), and the prediction accuracy does not seem to be impacted by the negative result of the ANOVA test. The case of the Eclipse data set seems particularly chaotic: for the 10 % slice, the prediction accuracy is high at both the 25 and 50 % thresholds, although the ANOVA test is negative, and the accuracy is null for the slices 90 and 100 %, although the test is positive. Indeed, the chaotic shape of the Eclipse accuracy graph is difficult to interpret. Figure 12 presents the prediction accuracy of the simulation scenario from another perspective, highlighting the variation for each data set. Indeed, the Eclipse data presents a particularly irregular shape, in particular at Pred(0.5), with prediction accuracy in a range from 0 to 60 % .

6.1 Changing the Size of the Test Set

As stated earlier (cf. section 5), the simulation scenario relies on a fixed number of simulation points in time, corresponding to 10 cumulatively growing data slices (cf. Fig. 6). However, the size of the subset of defects that are considered for prediction can be varied. This variation in the simulation procedure is implemented through the parameter SSF, *Simulation Size Factor*. The results concerning prediction accuracy showed so far were obtained with SSF=0.1, which sets the number of predicted defects equal to 1 % of the whole data set (i.e., 0.1 * Size(SL_1)).

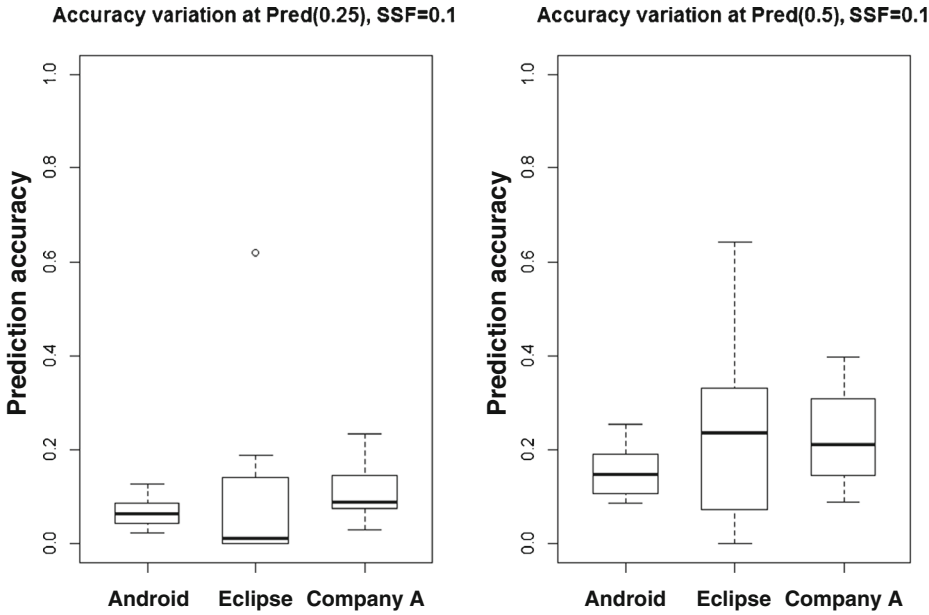

Fig. 12 Variation in DRT prediction accuracy for the three data sets

Next, we report the results from simulations with larger values for SSF. Larger SSF values imply that a larger number of predictions are made at a time, thus increasing the statistical power of the evaluation steps in our simulation procedure.

We ran the simulation procedure with SSF values 0.3 and 0.5. In Fig. 13, we combined the previous prediction accuracy results obtained with SSF=0.1 with the new results obtained with SSF=0.3 and SSF=0.5. However, for the sake of readability, the naïve prediction is not presented. Looking at each data set separately, it is interesting to see for the Android and Company A data sets, the shape of the curves do not diverge and tend to be flattened as SSF increases. However, results for the Eclipse data set are similarly irregular for all values of SSF.

From this exploration of the data, we can establish that, approximately, the average prediction accuracy is between 10 and 15 % for Pred(0.25) and 15 and 20 % with the three data sets used in the experiment. There is a visible variation in this result, with the Eclipse data set being less homogenous than the others, and the Company A data being slightly more homogenous that the others two. For the rest of this section, we will focus on the Company A data set and explore the potential effect of the number of clusters on the prediction accuracy.

6.2 Increasing the Number of Clusters (K=6, 8 and 10)

An important issue in both the baseline experiment and in our paper is the number of clusters K. As discussed earlier (cf. section 3), Raja applied a sophisticated and complicated clustering technique in the baseline experiment, a technique that includes several iterations and human interventions. She indicated that she started with a default maximum number of clusters, and that "*several iterations lead to the optimal solution*" (Raja 2013, p.126), with a final number of clusters between 3 and 5. The clustering technique we use in the replication, and in the subsequent simulation scenario, is intended to be fully automated with no human intervention for practical reasons, i.e., daily use for making DRT predictions. In our conceptual replication,

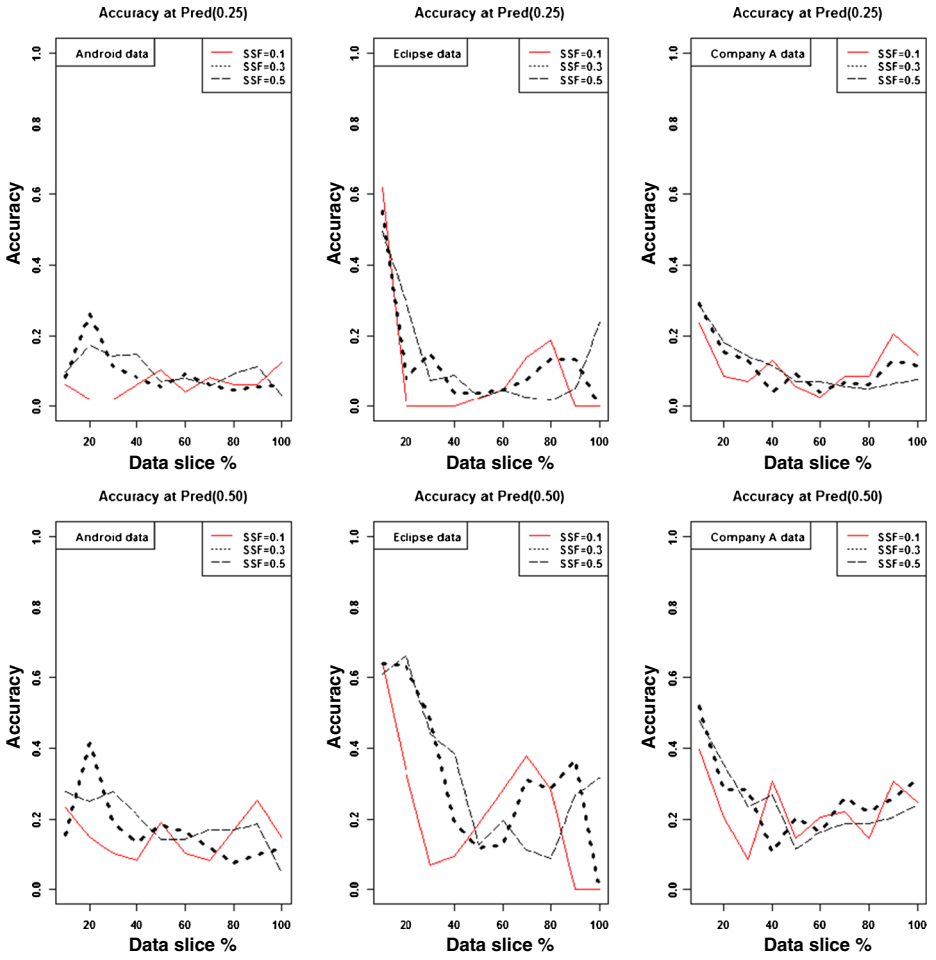

Fig. 13 The DRT prediction accuracy for the three data sets with different simulation size factors (0.1, 0.3 and 0.5) and at error thresholds 25 and 50 %

we fixed the number of clusters to 4 using the well known heuristic "Elbow technique" (cf. section 3). However, fixing the number of clusters can eventually be a confounding factor in the measured prediction accuracy.

To explore this issue, we run the clustering step with larger numbers of clusters. Concerning the number of clusters, we relied on the indication given by Raja in the description of the baseline experiment, indicating that, during the clustering she conducted, "*none of the projects yielded more than ten clusters*" (p. 126). So we considered K=10 as a significantly large number for clustering defect reports, and we decide to explore three additional values for K: 6, 8, and 10. We chose to explore the effect of K on the Company A data set. This data set is the largest, and it yielded the best prediction accuracy with K=4 for Pred(0.25). Regarding the statistical assumptions, the ANOVA and the post-hoc test were both positive for all slices of the Company A data set.

We conducted the ANOVA test on the Company A data for each data slice, with K set to 6, 8 and 10. The results of these tests were positive in all cases, and they are not presented here. However, the results of the post-hoc analysis were not as clear as for the initial clustering with

K=4 (cf. c in Fig. 9). In Fig. 14 we present the post-hoc results for K=6 and K=8. The results get better with larger slices, and for the 80 to 100 % slices, the test is almost fully positive. For sake of space, we don't show the post-hoc test for K=10, the results were very similar to the K=8 case.

For the simulation part, as there are more clusters to make predictions from, the test set needs to be sufficiently large. Thus, we decide to run the simulation scenario with a Simulation Size Factor SSF=0.3 and 0.5. This would set the size of the test set to 204 and 340 respectively.

The results of the simulation scenario are presented in Fig. 15. At the 25 % error threshold, i.e., Pred(0.25), the prediction accuracy plot shows a clear and homogenous pattern that drops from ~30 % (i.e., 30 % of predictions within the range set by the threshold) at early points in time (i.e., smaller data slices) to ~20 % at later points in time. Comparing the shape of the accuracy plot for K=4 (Fig. 13) and K=6, 8 and 10 (Fig. 15), in particular for SSF=0.5, the results are very comparable and thus seem to be independent from the number of clusters K. At the 50 % error threshold, i.e., Pred(0.5), the figures are less regular but the prediction accuracy is the highest, i.e., ~45 %, for early points in time.

7 Discussion

In the previous section, we presented the results regarding RQ1 (replication) and RQ2 (test scenario). Regarding RQ1, we demonstrated that the results obtained by Raja (2013) are fully reproducible with different data sets and using a different mining technique. The success of the conceptual, non-exact replication we conducted contributes to the external validity of the results published in the original study. We thus confirm Raja's claim: clustering defect reports leads to clusters with significantly different mean defect resolution time (DRT). The significantly different DRTs support Raja's suggestion that "*text based classification of defect reports can be a useful early indicator of defect resolution times*" (p. 135).

The idea behind our second research question RQ2 is to go beyond the replication, and to explore a method for predicting DRT based on clustering. Raja gave no indication of how such method could be designed. We choose a simulation approach for exploring this issue. We run a

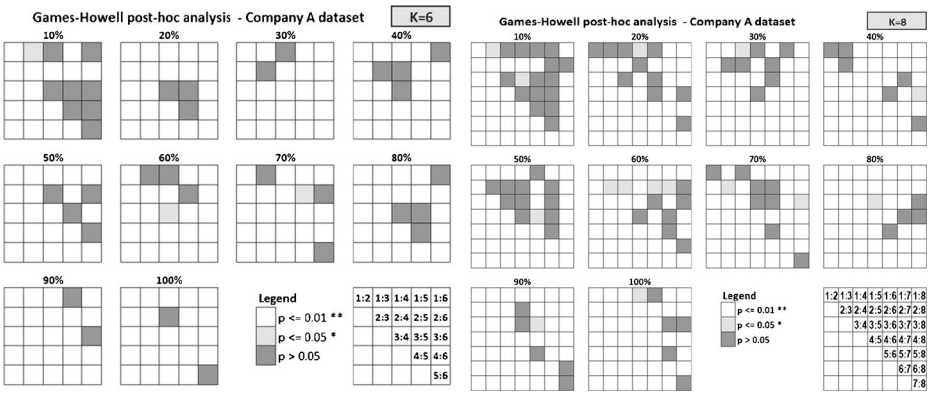

Fig. 14 Games-Howell post-hoc test for the Company A data slices with K=6 and K=8 clusters

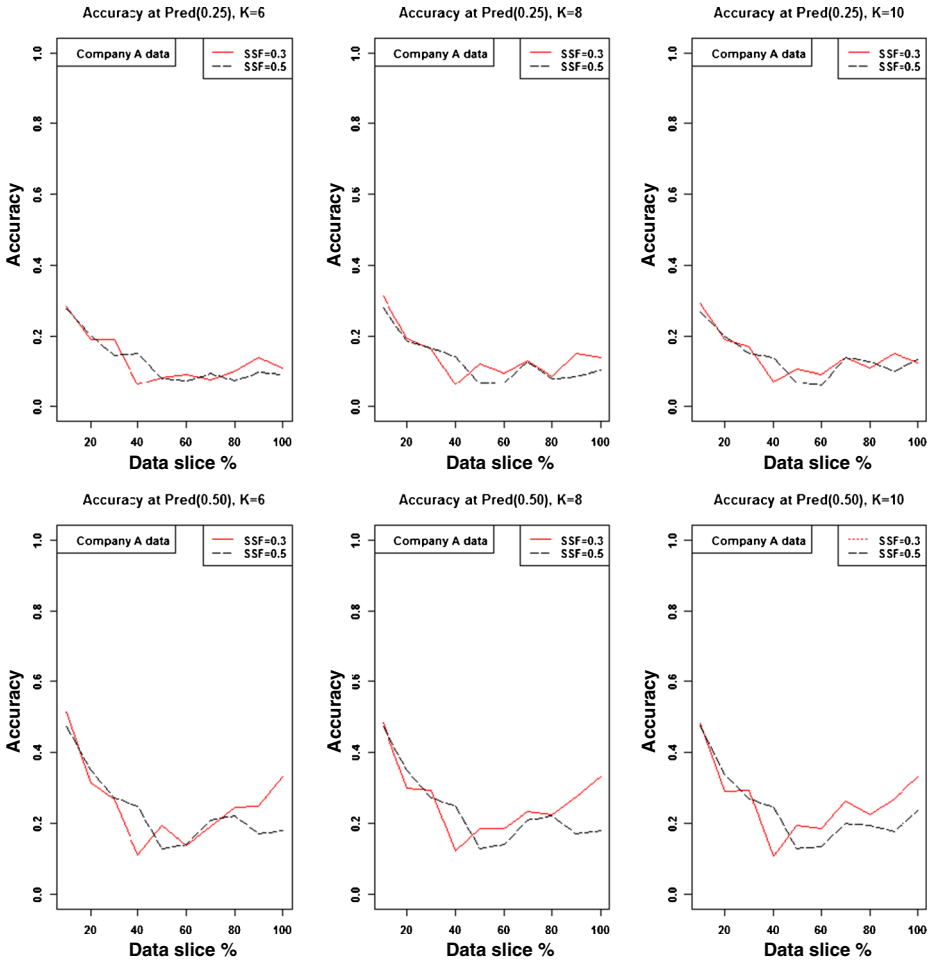

Fig. 15 The DRT prediction accuracy for the different number of clusters, using two simulation size factors (0.3 and 0.5), and at error thresholds 25 % and 50 %

set of simulation scenarios in which we simulate the arrival of new open defects at ten points in time. The predicted DRT for an incoming defect is the mean resolution time of the cluster to which it belongs. In industrial settings, this method would be applied in a similar manner, i.e., open and closed defect reports are clustered either each time a new defect arrives, or as batch jobs on a regular basis, e.g., nightly or every weekend. To evaluate the quality of the method, we measure the prediction accuracy, i.e., the percentage of predictions with an error within 25 and 50 %. Looking at the results, not only was the predictive quality generally low (even for Pred(0.5)), also there was no visible pattern that would help indicate in which situation predictions are of higher or lower quality. In particular, there was no indication that larger sets of closed DRs are better or worse for the purpose of predicting DRT of incoming defect reports. Interestingly, in many cases, clustering did not improve predictive quality over simply using the average DRT of all closed defect reports as a naïve prediction approach (Figs. 10 and 11). To check if this result is influenced by certain parameters of the experiment, we run the simulation with larger prediction sets and with larger number of clusters, and the results are

globally consistent (Figs. 13 and 15), i.e., the average accuracy for Pred(25) is between 10 and 15 %, and for Pred(50) between 15 and 30 %.

While these results are discouraging, they seem to be consistent with what others have found. For example, Weiss et al. (2007), who applied the k-nearest neighbor (kNN) method to predict the defect resolution effort for various software systems, did not get better results than we did. On average, only 30 % of their predictions lay within a ±50 % range of the actual effort. Interestingly, their data were more fine-granular (person-hours) than ours, which reconfirms our observation that different data granularity does not seem to have an impact on accuracy of predicted DRT. Weiss et al. explain this by the diversity of the defect reports. Others, like Panjer (2007) and Bougie et al. (2010) suggest that there are other attributes that may have greater influence on the DRT than the pure lexical information in defect reports, and they speculate that predictive accuracy and optimal methods for predicting DRTs may vary significantly between software projects.

Moreover, the theoretical assumption about significant differences in mean DRT between clusters (ANOVA and post-hoc test) did not seem to have an effect on the results. In the case of the Eclipse data set, when the ANOVA test is negative (cf. Table 10), the prediction accuracy is higher than average, and in others, when the test is positive, the prediction is very low or even null. This is also confirmed when exploring the Company A data with larger number of clusters, there is no visible pattern of correlation between the results of the post-hoc test (Fig. 14) and the prediction accuracy (Fig. 15). From this conclusion, we would tend to deduce that Raja's suggestion about the usefulness of text based classification as an early indicator of DRT cannot be applied in practice, and more importantly, do not produce an accurate prediction of DRT. However, we are aware that we designed a method that uses a different clustering approach and we applied it to different data sets. Thus, all we can confirm here is that significantly different mean DRT among clusters is insufficient for making an accurate prediction. Nonetheless, we cannot affirm that the specific clustering approach applied by Raja in the baseline experiment would produce a similarly negative result. We indeed made modifications in the clustering approach we applied in our method for two reasons: (1) Raja did not disclose all details of the manual steps she applied to improve the data clustering; (2) in industrial contexts, engineers need simple, robust, and fully automated prediction methods to be applicable in a non-intrusive and effortless fashion.

Generally, in the literature, the opinion that DRT prediction shall be based on more than just the description of the defect report is popular, e.g. Keung and Kitchenham (2008); however, what features of a defect report are the most relevant is unclear. In Giger et al. (2010), where a decision-tree algorithm was applied to classify incoming defect reports into those with fast (less than median DRT) and those with slow (more than median DRT) resolution time, the best performing prediction models were obtained when using additional information added to defect reports (e.g., a milestone, a comment) between 14 to 30 days after defect reports submission. Menzies et al. (2011b) and Bettenburg et al. (2012) suggest to first cluster defect reports according to severity and domain information, and then apply the text based clustering approach. Furthermore, another problem with basing DRT prediction on the defect reports description might be insufficient quality of the defect report itself; for a discussion on defect report quality see, e.g., Wang et al. (2011) and Laukkanen and Mäntylä (2011). Indeed, our analysis relies on a strong assumption: a defect report is a correct, complete and relevant description of a defect; the same defect reported by two different users would use the same terms and thus, would have a high degree of similarity. The validity of this assumption might be questionable.

8 Threats to Validity

Threats to validity in experimental studies are categorized into construct, internal, external and conclusion validity (Perry et al. 2000) (Wohlin et al. 2012). Construct validity means that the studied phenomenon is adequately conceptualized with the dependent and independent variables in the model, and whether the experimental setup correctly measures the variables. Internal validity points to the explanatory power of the model and the extent to which changes in the dependent variables can be safely attributed to changes in the independent variables, i.e., the absence of any confounding factor. External validity means that the results can be generalized to other settings similar to those of the study. Conclusion validity concerns the relationship between the treatment and the outcome, and the risk of type I and II errors occurring. As this study is of type theory testing, i.e., checking the existence or the extent of a relationship or an effect claimed by a theory, we present the validity threats in the decreasing priority mentioned by Wohlin et al. (2012): internal, construct, conclusion and external. We will discuss each validity threat for RQ1 first and then for RQ2.

Concerning RQ1, we successfully reproduced the results of the original study. Our results are subject to the same threats to internal validity as the baseline experiment. Raja (2013) discussed three threats concerning the data sets she used for her experiment: 1) the selected sample of defect reports from five successful OSS projects, 2) that all defects were considered equal, neglecting the possible effect of various defect attributes on DRT, and 3) that DRT was computed without considering the complexity of the problem. The same threats should also be mentioned also for our study. Moreover, concerning our replication, the clustering approach we applied (i.e., k-means clustering using default settings in RapidMiner) might have influenced the results for RQ1. Indeed, in the baseline experiment, Raja exploited specific features of the software tool she used (i.e., SAS Text Miner) and made subsequent efforts to optimize the results of the clustering to reduce possible sources of "noise", e.g., presence of code fragments, misspelled words, and terms that appear rarely (Raja 2013, p. 126). In our experimental setting, and for reasons related to RQ2, we deliberately limited the manual preprocessing and restrained ourselves to features available in the open-source software tool we used for clustering, i.e., RapidMiner. Finally, when we applied clustering on some smaller data slices, the ANOVA test was negative (cf. Table 10). Thus, the size of the data sample could be a confounding factor in both the baseline experiment and in the reproduction we made. There is indeed a risk that, for large data sets, any clustering technique, or even random grouping, might produce clusters for which mean DRTs are significantly different.

Concerning RQ2, we have analyzed two possible confounding factors : the size of the test set (the SSF parameter), i.e., the number of defects used for prediction, and the number of clusters in the clustering approach. Our analysis showed that the results are robust, i.e., there is no notable impact on the prediction accuracy. Another possible confounding factor is the non-determinism of k-means clustering (Su and Dy 2004), i.e., another run of the clustering could in theory produce clusters for which the simulation results would be better. However, this threat is minimized by both the size of our data sets, and the multiple runs conducted for each data slice according to the simulation protocol (cf. section 5).

In terms of construct validity regarding replication and RQ1, the threats concern the experimental process we applied and the data sets we used. We have exclusively relied on the description of the original experiment as reported in the published papers as no supplementary material was available (cf. Table 2). As mentioned above, there are indeed certain sophisticated details concerning the clustering technique in the baseline experiment that were

not fully described. Concerning the data sets, we could not reuse the data sets from the baseline experiment. Correspondence with the author did not result in the raw data needed; it was lost after the baseline experiment was conducted, highlighting the importance of uploading experimental data sets to online repositories. On the other hand, using the same data sets would have reduced the significance of the replication aspect of the study as there would have been too few variation points as compared with the original study, i.e., only the software tools used for clustering and statistical analysis would have been different.

Regarding RQ2, construct validity spans two main issues. The first is again related to the data sets we used, i.e., it could be questioned why the data sets of the original study were not directly used for prediction in the test scenario. Beyond the unavailability of the original data sets, we argue that diversifying the data sets, e.g., by including a proprietary data set in the experiment, enriches our experiment. The three data sets we studied contribute to the external validity of our findings, in line with recommendations for evaluating mining techniques on non-open source systems (Hassan 2008). Indeed, the results for the proprietary data set we studied are better than for any of the data sets in the original study in terms of the ANOVA analysis, i.e., the difference in DRT is significant for all pairs of clusters for Company A data (Table 8), while this is not the case for any of the data sets in the original study (Raja 2013, p. 131, Table 7). The second construct validity issue related to RQ2 is the simulation approach we use in the test scenario. We replayed the inflow of defect reports and chose to cluster the data at ten points in time. It would have been possible to cluster the data sets more frequently, but that would have been considerably more computationally expensive. Also, choosing clustering points based on the time dimension rather than a fixed number of defect reports might better correspond to a real world usage scenario. However, keeping the number of additional defect reports between every clustering point constant helps exploring the potential value of more data. Finally, cumulatively increasing the number of defect reports at each clustering point is perhaps not feasible. As shown for the Eclipse data set (cf. Figs. 10 and 11), the Pred(X) graphs drops twice, suggesting that the performance of the DRT predictions must be carefully monitored if the approach is deployed. At certain times the time locality of the training data is critical, i.e.,. patterns in the old defect reports do no longer apply for newly submitted defect reports. An alternative experimental setup would be to instead use the sliding window method (Dietterich 2002) to only evaluate DRT prediction based on recent defect reports.

Regarding the conclusion of RQ2, i.e., the low level of prediction accuracy, the main threats to its validity are again related to the data sets and the pre-processing steps. Although the prediction accuracy is low for the three data sets, the proprietary data showed patterns slightly more regular than the patterns for the Android and Eclipse data sets. We tend to believe that defect reports originating from proprietary contexts are authored more carefully. Thus, the result of our test scenario could be threatened by the quality of the textual descriptions in the defect reports. Furthermore, a better filtering of the data sets could have improved the results. Previous work showed that filtering out outliers can support DRT prediction (Lamkanfi and Demeyer 2012) (AbdelMoez et al. 2013). However, we decided to rely on as few preprocessing steps as possible.

The main threat to conclusion validity is that the three data sets we studied might not have fulfilled the normality assumptions. The author of the original study assumed normality for the DRT distribution in each cluster, although she reported that the Kolmogorov-Smirnov test indicated “slight violations” (Raja 2013, p. 129). We made the same assumption even though both skewness and kurtosis of our data samples were rather high. Another assumption made in

the original study, related to the ANOVA tests, concerned the independence of defect reports. We assumed such independence although we suspect that defects in large software systems often appear in complex networks and thus are not always independent of each other (Borg et al. 2013a, b).

At last, concerning threats to external validity, the issue is slightly different for RQ1 and RQ2. The successful replication of the original study positively confirmed its results using different data sets and a fully automated clustering approach. Accordingly, this work contributes to a better generalization of the original study result concerning the significant differences in mean resolution time for defects in different clusters. Concerning RQ2 and the test scenario, and in the light of the threats to internal and construct validity as presented above, it is difficult to draw a positive conclusion concerning the clustering based approach to DRT prediction. The obtained patterns are very irregular, either for a single data set at different clustering points in the test scenario, or for the three data sets all together, i.e., the shapes of the prediction accuracy graphs are different for each data set. However, while our results suggest that clustering based DRT prediction is not promising, confirming whether our negative results extend to other data sets requires further replications.

9 Conclusion

In conclusion, we must clearly say that using a simple, fully automated clustering approach based on term-frequency in defect report descriptions cannot predict defect resolution time (DRT) with sufficient accuracy, no matter whether the acceptable error margin is set to 25 %, or 50 %. Moreover, the predictions are in most cases not better than simply using the average DRT of the whole available data set as a predictor for the DRT. On the other hand, our replication confirmed the results achieved by Raja (2013) with regards to the significant difference of average DRTs in defect report clusters. Indeed, the value of replication lies in analyzing the results in conjunction with the baseline experiment. If the results are compatible, they can be considered additive, increasing confidence in the original hypothesis (Miller 2005).

However, as suggested by our findings, the practical value of this observation regarding the prediction of DRTs is rather limited. Given the results of other researchers, this finding is not surprising. While according to Weick (1995), the "*differential 'responsiveness' of data to changes in a treatment is frequently an informative precursor to theorizing*", we must admit that in our case Barbara Kitchenham is right when saying that she is "*not convinced that we can find theories by simply varying experimental conditions*" (Kitchenham 2008), and that – as was true in our case – "*a replication without the framework of a theory, whether independent or dependent, is by far the most risky type of replication*". Indeed, our study is grounded on two theoretical principles: the first is the similarity principle, i.e., similar defects require similar time to be resolved, the second is the clustering principle, i.e., similar defects are grouped in clusters. These principles seem insufficient to fully explain our empirical findings, and further investigations are necessary to better understand the relationships between multiple factors that can affect defect similarity (e.g., severity) and DRT (e.g., developers' background knowledge or project specific settings).

Thus, our perspectives for future work will be guided by these observations. We first intend to question the assumption of similarity using a controlled experiment: do different persons describe defects that are alike – or even identical – with similar textual reports? Further, we are

interested in exploring any confounding factors in software projects that impact defect likeness beyond textual similarity. This would enhance the theoretical understanding of how a defect is best described and how this information can be leveraged in software repositories for navigation, recommendation and prediction purposes (Borg 2014). At last, automatic clustering techniques for grouping similar defects needs further questioning and exploration. As a cluster is an automatically derived group of similar elements, to enhance usefulness when used for prediction, a cluster should exhibit a certain level of *semantic* unity. Indeed, in the baseline experiment we replicated, the author manually exploited interactive features of the text mining tool to obtain more homogenous clusters in terms of domain concepts and keywords. Moreover, a recent investigation leverages the observation that certain components of a software are more error-prone than others (Chen et al. 2012); this would suggest that, beyond textual similarity, words appearing in a defect report can be related to a software topic and, thus, to a certain group of similar defects. Therefore, we believe that these ideas should be developed further, for example, how automatic clustering can exploit domain knowledge, i.e., a domain ontology, and create categories related to the domain of the software to which the defect is linked. Indeed, ontology-based semantic clustering has been experimented recently in other research domains and showed promising results (Batet 2011).

Acknowledgments This research was partly funded by the institutional research grant IUT20-55 of the Estonian Research Council and the Industrial Excellence Center EASE – Embedded Applications Software Engineering, Sweden.

References

AbdelMoez W, Kholief M, Elsalmy FM (2013) Improving bug fix-time prediction model by filtering out outliers. Proceedings of the Int'l Conf. on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE). IEEE Computer Society, pp 359–364. doi:10.1109/TAEECE.2013.6557301

Anbalagan P, Vouk M (2009) On predicting the time taken to correct bug reports in open source projects. Proceedings of the IEEE Int'l Conf. on Software Maintenance (ICSM'09). IEEE Computer Society, pp 523–526. doi:10.1109/ICSM.2009.5306337

Batet M (2011) Ontology-based semantic clustering. AI Commun 24:291–292. doi:10.3233/AIC-2011-0501

Bettenburg N, Nagappan M, Hassan AE (2012) Think locally, act globally: Improving defect and effort prediction models. Proceedings of the 9th IEEE Working Conf. on Mining Software Repositories (MSR'12). IEEE Computer Society, pp 60–69. doi:10.1109/MSR.2012.6224300

Bhattacharya P, Neamtiu I (2011) Bug-fix time prediction models: can we do better? Proceedings of the 8th Working Conf. on Mining Software Repositories (MSR'11). ACM, New York, NY, USA, pp 207–210. doi:10.1145/1985441.1985472

Boehm B, Basili VR (2001) Software defect reduction top 10 list. Computer 34:135–137

Borg M (2014) Embrace your issues: compassing the software engineering landscape using bug reports. Presented at the Doctoral Symposium, 29th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE'14), Sept. 15th, 2014, Västerås, Sweden

Borg M, Runeson P (2013) IR in software traceability: from a bird's eye view. Proceedings of the 7th International Symposium on Empirical Software Engineering and Measurement (ESEM'13), pp. 243–246

Borg M, Gotel OCZ, Wnuk K (2013) Enabling traceability reuse for impact analyses: a feasibility study in a safety context. Proceedings of the Int'l Workshop on Traceability in Emerging Forms of Software Eng. (TEFSE'13). IEEE Computer Society, pp 72–78. doi:10.1109/TEFSE.2013.6620158

Borg M, Pfahl D, Runeson P (2013) Analyzing networks of issue reports. Proceedings of the 17th European Conf. on Software Maintenance and Reengineering (CSMR'13). pp 79–88

Bougie G, Treude C, German DM, Storey M (2010) A comparative exploration of FreeBSD bug lifetimes. Proceedings of the 7th IEEE Working Conference on Mining Software Repositories (MSR'10). IEEE Computer Society, pp 106–109. doi:10.1109/MSR.2010.5463291

Brooks A, Roper M, Wood M et al (2008) Replication's role in software engineering. In: Shull F, Singer J, Sjøberg DIK (eds) Guide to advanced empirical software engineering. Springer, London, pp 365–379

Carver JC, Juristo N, Baldassarre MT, Vegas S (2014) Replications of software engineering experiments. Empir Softw Eng 19:267–276. doi:10.1007/s10664-013-9290-8

Chen T-H, Thomas SW, Nagappan M, Hassan A (2012) Explaining software defects using topic models. Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR'12). pp 189–198. doi:10.1109/MSR.2012.6224280

D'Ambros M, Lanza M, Robbes R (2010) An extensive comparison of bug prediction approaches. Proceedings of the 7th IEEE Working Conference on Mining Software Repositories (MSR'10). pp 31–41. doi:10.1109/MSR.2010.5463279

Deerwester S, Dumais S, Furnas G, Landauer T, Harschman R (1990) Indexing by latent semantic indexing. J Am Soc Inf Sci 41(6):391–407

Dietterich T (2002) Machine learning for sequential data: a review. structural, syntactic, and statistical pattern recognition – Proceedings of the Joint IAPR International Workshops SSPR 2002 and SPR 2002, pp. 15–30

Dubes R (1993) Cluster analysis and related issues. In: Chen C, Pau L Wang P (eds) Handbook of pattern recognition and computer vision, Chen C, Pau L Wang P (Eds.), World Scientific Publishing, pp. 3–32

Frost HR, Moore JH (2014) Optimization of gene set annotations via entropy minimization over variable clusters (EMVC). Bioinformatics btu110:1–9. doi:10.1093/bioinformatics/btu110

Giger E, Pinzger M, Gall H (2010) Predicting the fix time of bugs. Proceedings 2nd Int. Workshop on Recommendation Systems for Software Eng. (RSSE'10). ACM, New York, NY, USA, pp 52–56

Gómez OS, Juristo N, Vegas S (2014) Understanding replication of experiments in software engineering: a classification. Inf Softw Technol 56(8):1033–1048. doi:10.1016/j.infsof.2014.04.004

González-Barahona J, Robles G (2012) On the reproducibility of empirical software engineering studies based on data retrieved from development repositories. Empir Softw Eng 17:75–89. doi:10.1007/s10664-011-9181-9

Hassan A (2008) The road ahead for mining software repositories. Frontiers of Software Maintenance (FoSM 2008). IEEE Computer Society, pp 48–57.

Hofmann M, Klinkenberg R (2014) RapidMiner: data mining use cases and business analytics applications. CRC Press

IEC (2014) IEC 61511–1 ed1.0. http://webstore.iec.ch/webstore/webstore.nsf/artnum/031559

Jain AK (2010) Data clustering: 50 years beyond K-means. Pattern Recogn Lett 31:651–666. doi:10.1016/j.patrec.2009.09.011

Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31:264–323. doi:10.1145/331499.331504

Juristo N, Gómez OS (2012) Replication of software engineering experiments. In: Meyer B, Nordio M (eds) Empirical software engineering and verification. Springer, Berlin/Heidelberg, pp 60–88

Keung J, Kitchenham B (2008) Experiments with analogy-X for software cost estimation. Proceedings of the 19th Australian Conf. on Software Engineering (ASWEC'08). pp 229–238

Kim S, Whitehead,Jr. EJ (2006) How long did it take to fix bugs? Proceedings of the IEEE Int. Workshop on Mining Software Repositories (MSR'06). ACM, New York, USA, pp 173–174

Kitchenham BA (2008) The role of replications in empirical software engineering—a word of warning. Empir Softw Eng 13:219–221. doi:10.1007/s10664-008-9061-0

Kontostathis A (2007) Essential dimensions of Latent Semantic Indexing (LSI), Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS '07), pp. 73–80. doi:10.1109/HICSS.2007.213

Lamkanfi A, Demeyer S (2012) Filtering bug reports for fix-time analysis. Proceedings of the 16th European Conf. on Software Maintenance and Reengineering (CSMR'12). IEEE Computer Society, pp 379–384. doi:10.1109/CSMR.2012.47

Laukkanen EI, Mäntylä MV (2011) Survey reproduction of defect reporting in industrial software development. Proceedings of the Int'l Symposium on Empirical Software Engineering and Measurement (ESEM'11). pp 197–206

Lessmann S, Baesens B, Mues C, Pietsch S (2008) Benchmarking classification models for software defect prediction: a proposed framework and novel findings. IEEE Trans Softw Eng 34:485–496. doi:10.1109/TSE.2008.35

Lilliefors HW (1967) On the Kolmogorov-Smirnov test for normality with mean and variance unknown. J Am Stat Assoc 62:399–402

Marks L, Zou Y, Hassan AE (2011) Studying the fix-time for bugs in large open source projects. Proceedings of the 7th Int. Conf. on Predictive Models in Software Engineering. ACM, New York, NY, USA, pp 11:1–11:8

Matejka J, Li W, Grossman T, Fitzmaurice G (2009) CommunityCommands: command recommendations for software applications, Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology (UIST'09), pp. 193–202.

McCandless M, Hatcher E, Gospodnetic O (2010) Lucene in action, second edition. Manning Publications Co., Greenwich, CT, USA. http://www.manning.com/hatcher3/

Menzies T, Shepperd M (2012) Special issue on repeatable results in software engineering prediction. Empir Softw Eng 17:1–17. doi:10.1007/s10664-011-9193-5
Menzies T, Bird C, Zimmermann T, et al. (2011) The inductive software engineering manifesto: principles for industrial data mining. Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering. ACM, New York, NY, USA, pp 19–26
Menzies T, Butcher A, Marcus A, et al. (2011) Local vs. global models for effort estimation and defect prediction. 26th IEEE/ACM Int. Conf. on Automated Software Engineering (ASE). pp 343–351. doi:10.1109/ASE.2011.6100072
Miller J (2005) Replicating software engineering experiments: a poisoned chalice or the Holy Grail. Inf Softw Technol 47:233–244. doi:10.1016/j.infsof.2004.08.005
Müller M, Pfahl D (2008) Simulation methods. In: Shull F, Singer J, Sjøberg DIK (eds) Guide to Advanced Empirical Software Engineering. Springer, London, pp 117–152
Panjer LD (2007) Predicting eclipse bug lifetimes. Proceedings of the 4th IEEE Working Conf. on Mining Software Repositories (MSR'07). IEEE Computer Society, p 29. doi:10.1109/MSR.2007.25
Perry DE, Porter AA, Votta LG (2000) Empirical studies of software engineering: a roadmap. Proceedings of the Conference on The Future of Software Engineering. ACM, New York, NY, USA, pp 345–355
Raja U (2013) All complaints are not created equal: text analysis of open source software defect reports. Empir Softw Eng 18:117–138. doi:10.1007/s10664-012-9197-9
Robinson B, Francis P (2010) Improving industrial adoption of software engineering research: a comparison of open and closed source software. Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. ACM, New York, NY, USA, pp 21:1–21:10. doi:10.1145/1852786.1852814
Rosenthal R (1991) Replication in behavioral research. In: Neuliep JW (ed) Replication research in the social sciences. SAGE Publications Inc., Newbury Park, pp 1–30
Sawilowsky SS, Blair RC (1992) A more realistic look at the robustness and Type II error properties of the t test to departures from population normality. Psychol Bull 111:352–360. doi:10.1037/0033-2909.111.2.352
Shepperd M, Kadoda G (2001) Using simulation to evaluate prediction techniques [for software]. Proceedings of the 7th Int'l Software Metrics Symposium (METRICS 2001). IEEE Computer Society, pp 349–359
Shihab E, Kamei Y, Bhattacharya P (2012) Mining challenge 2012: the android platform. Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR'12). IEEE Computer Society, pp 112–115
Shull F, Carver J, Vegas S, Juristo N (2008) The role of replications in empirical software engineering. Empir Softw Eng 13:211–218. doi:10.1007/s10664-008-9060-1
Singhal A (2001) Modern information retrieval: a brief overview. Data Eng Bull 24(2):1–9
Strate JD, Laplante PA (2013) A literature review of research in software defect reporting. IEEE Trans Reliab 62:444–454. doi:10.1109/TR.2013.2259204
Su T, Dy J (2004) A deterministic method for initializing K-means clustering. 16th IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI 2004). pp 784–786. doi:10.1109/ICTAI.2004.7
Tassey G (2002) The economic impacts of inadequate infrastructure for software testing. National Institute of Standards and Technology (NIST), USA
Tibshirani R, Walther G, Hastie T (2001) Estimating the number of clusters in a data set via the gap statistic. J R Stat Soc Ser B (Stat Methodol) 63:411–423. doi:10.1111/1467-9868.00293
Walker R, Holmes R (2014) Simulation - a methodology to evaluate recommendation systems in software engineering. In: Robillard M, Maalej W, Walker R, Zimmermann T (eds) Recommendation systems in software engineering. Springer, London, pp 301–327
Wang D, Wang Q, Yang Y, et al. (2011) "Is it really a defect?" An empirical study on measuring and improving the process of software defect reporting. Proceedings of the Int'l Symposium on Empirical Software Engineering and Measurement (ESEM'11). IEEE Computer Society, pp 434–443. doi:10.1109/ESEM.2011.62
Weick KE (1995) What theory is not, theorizing is. Adm Sci Q 40:385–390
Weiss C, Premraj R, Zimmermann T, Zeller A (2007) How long will it take to fix this bug? Proceedings of the 4th Int. Workshop on Mining Software Repositories (MSR'07). IEEE Computer Society. doi:10.1109/MSR.2007.13
Wohlin C, Runeson P, Höst M et al (2012) Experimentation in software engineering. Springer, Berlin/Heidelberg
Xu R, Wunsch D (2005) Survey of clustering algorithms. IEEE Trans Neural Netw 16(3):645–678. doi:10.1109/TNN.2005.845141
Zeller A (2009) Why programs fail: a guide to systematic debugging, 2nd ed. Morgan Kaufmann Publishers.
Zhang H, Gong L, Versteeg S (2013) Predicting bug-fixing time: an empirical study of commercial software projects. Proceedings of the 2013 International Conf. on Software Eng. (ICSE'13). IEEE Computer Society, Piscataway, NJ, USA, pp 1042–1051. doi:10.1109/ICSE.2013.6606654

Saïd Assar is an Associate Professor of Information Systems at Institut Mines-Telecom, Ecole de Management, and is associate researcher at Informatics Research Center CRI at Sorbonne University, Paris, France. In 2012, he was a visiting scholar at Lund University, Sweden, with the Software Engineering Research Group (SERG). His research interests include methods and tools for information systems analysis and design, empirical software engineering and e-learning. Saïd is particularly concerned in exploring the contribution of interdisciplinarity to research in information systems. He received his MSc (1988) and PhD (1995) degrees in computer science from Pierre & Marie Curie University, Paris.

Markus Borg received a MSc degree in Computer Science and Engineering (2007) and a PhD degree in software engineering (2015) from Lund University, where he is a member of the Software Engineering Research Group. His research interests are related to alleviating information overload in large-scale software development, with a focus on information retrieval and recommendation systems. Prior to his PhD studies, he worked three years as a development engineer at ABB in safety-critical software engineering. He is a student member of the IEEE.

Dietmar Pfahl is an Associate Professor of the Software Engineering Group in the Institute of Computer Science at the University of Tartu, Estonia. In addition, he is an Adjunct Professor with the Schulich School of Engineering at the University of Calgary, Canada. His research interests include software project management, software process improvement, software product management, software testing, and empirical software engineering. Dietmar has more than 20 years of experience in conducting and leading national and international research and transfer projects with software industry, including organizations such as Bosch, DaimlerChrysler, Dräger, Ericsson, Siemens, and Telenor. He is a senior member of the ACM and the IEEE.