Automatic Defect Classification: An Experience Applying Natural Language Processing

Santiago Matalonga.[1] Eng., Tomás San Feliu. Phd.[2], Vasile Rus. Phd.[3]

[1]Cuariem 1141, 11100. Montevideo Uruguay. Facultad de Ingeniería, Universidad ORT
[2]Campus de Montegancedo. Boadilla del Monte. España. Facultad de Informática, Universidad Politécnica de Madrid.
[3]373 Dunn Hall, Memphis TN. Department of Computer Science, University of Memphis

smatalonga@uni.ort.edu.uy
tomas.sanfeliu@upm.es
vrus@memphis.edu

Abstract. This paper presents initial results of a research effort that merges two approaches on Quality Management. On one hand, while working with process improvement initiatives involving defect causal analysis, the researcher were faced with the problem of measuring inter-rater reliability for a defect classification taxonomy in a software development organization. On the other hand, the researchers were looking for ways to guide defect correction effort by using information retrieval and natural language processing to cluster defect reports. This paper presents the results of applying the later approach to the inter-rater reliability problem. The results indicate that the machine learning approach scores at rates only achieved by senior members of the organization.

Keywords: Defect Classification, Natural Language Processing, Defect Causal Analysis, Process Improvement

1 Introduction

Causal analysis lies at the core of process improvement initiatives. From Deming Process Improvement Cycle[1] to CMMI[2], causal analysis is a fundamental tool used to find root causes of production defects and to initiate process improvement proposals[3]. Being such an important tool for process improvement, it has recently been argued [4, 5] that CMMI has no requirement for causal analysis at lower maturity levels. CMMI is therefore, missing a fundamental tool for process improvement. This year, the CMMI steward, the Software Engineering Institute at Carnegie Mellon, has announced that its upcoming update to the CMMI will include requirements for causal analysis at maturity level 3[6]. These will likely result in a number of lower maturity organizations to strive for a cost effective alternative to achieve causal analysis capabilities within their processes if they want to retain their maturity level rating.

Even though it is not explicitly defined by any process improvement model (for instance ISO 9001:2000 or CMMI), most causal analysis implementations are based on the analysis of defects[7, 8]. Previous research has already established the value of the information that a software development organization can learn from their defects[7-9], nevertheless, the analysis of each individual defect can be tedious, error prone, and time consuming[10]. As a result, most organizations rely on defect taxonomies. In its most basic form, organizations would classify defects in order to establish a priority for tackling defects at correction time. This basic classification will take the form of Low-Medium-High[11, 12]. A classification taxonomy which has been extensively used in conjunction with defect causal analysis[13-15] is IBM's Orthogonal Defect Classification[13].

A common ground for all these approaches is their reliance on manual classification of defects by groups of individuals, usually the testers who discover the defect and/or developers who correct them, in order to produce a set of classified defects. For an organization to trust the validity of its classified defect database it must ensure that classifiers are sufficiently trained to produce consistent taxonomies[14].

We present in this paper the result of the fusion of two research initiatives for the improvement of testing effectiveness. On one hand, we were interested in defect taxonomies and defect classification from a process improvement perspective [14, 16]. On the other hand, we explored classifying defects using machine learning techniques [10, 17].

The remainder of this paper briefly describes both research approaches and presents the findings of our joint venture. The following section describes the two research initiatives. Section 3 details our research objectives while section 4 presents the results we have obtained so far. Finally, in *Conclusion and future work*, we conclude the paper and present our goals for future research.

2 Related work

This section presents a summary of the two research projects and the results we have obtained prior to this joint research effort. The research described in section 2.1 focuses on return of investment of process improvement techniques, more specifically in how training can be used to enhance the return of investment of process improvement techniques.

The research described in Section 2.2, focuses on improving the effectiveness of the software testing and defect removal process. More importantly it shows how the application of technology can improve the performance of the defect correction process.

By merging the two approaches we are in fact proposing an approach that takes two of the sides of the people-tools-process triangle.

2.1 Impact of training on developers' classification ability

As we described in [14, 16, 18], we implemented a process that takes a project's defect data and uses it as input to the training department. Our hypothesis is that given the right set of conditions, training should be more cost effective than process improvement. In short, in the described process, developers classify defects found during different phases and activities of the project's life cycle (inspections, peer review, and testing). At every delivery milestone, the developers are asked to conduct causal analysis meetings. The objective of these meetings is to answer the question of what they would have needed to know in order to prevent injecting the defects into the code again. This knowledge is stored in the Defect Tracking System and the Training department uses the information as input to plan new training interventions. This process combines organizational training with defect classification and causal analysis.

This led to a search for case studies which showed that Causal Analysis has already been implemented at maturity level 3 organizations [4, 5]. As a result, in [14] we investigated further the problem of defect classification and the impact of training in the classification ability of the individual. In that work, we described how an organization must invest in training in order to assure that defects are classified consistently across the organizations' development project. We show an experiment with four subjects and evaluate their classification ability by applying the Cohen's[19] and Fleiss'[20] Kappa for inter-rater reliability. Both Kappa's test the agreement level between independent classifiers. Cohen's Kappa is used for testing the level of agreement between two subjects, while Fleiss' Kappa is used for testing groups.

The following tables present our results, which will be used as the baseline to evaluate the results of the experiment presented in this paper. Table 1 shows the significance levels for the results of both Kappa's.

Cohen's Kappa	**Significance**
< 0	Poor Agreement
0.00 – 0,20	Slight Agreement
0,21 – 0,40	Fair Agreement
0,41 – 0,60	Moderate Agreement
0,61 – 0,80	Substantial Agreement
0,81 – 1	Almost Perfect Agreement

Table 1. Kappa significance table for Cohen and Fleiss

Table 2 shows the result the four subjects obtained with the Fleiss' Kappa. Subject letters represent the seniority of each of them within the organization. E, stands for an expert in the classification taxonomy. The expert was involved in the development of the taxonomy. C1-C3, are developers who have received training in the taxonomy, and O represents an outsider, a new hire of the organization.

Subject Group	K agreement
E-C1-C2-C3	0,53
E-C1-C2-C3-O	0,44

Table 2. Fleiss'Kappa clasification results

We concluded that agreement is just moderate, and that the outsider has no substantial impact in the overall group classification ability.

2.2 Applying Natural language processing to defect classification

In [10], we presented an method that relies on Natural Language Processing (NLP) and Information Retrieval (IR) to generate defect clusters on defect reports from the Mozilla Foundation's *Bugzilla*[1] defect tracking system. Defect clustering is the unsupervised classification of patterns (usually represented as a vector of measurements, or a point in a multidimensional space) into groups (clusters) based on similarity. Typically, clustering involves the following steps[21]:

1. Data Representation
2. Definition of a similarity measure appropriate to the domain
3. Clustering
4. Assessment of output

A lot of effort was spent on finding a suitable representation of the defect reports in Bugzilla. In the experiment, we chose a vectorial representation [17, 22] of the most significant fields of the defect. In addition to this, the textual descriptions of defects had to be preprocessed before it could be used as an input for the clustering algorithm. Preprocessing of the defects reports included transforming the English language into vectorial maps and removing stop words for the English language[2]. The last preprocessing step includes the lemmatization, which is the transformation of each morphological variation of a word into its base form (i.e., *go, going*, *went, gone* are all lemmatized to *go*).

We selected to cluster the reports based on their describing the underlying the same bug. Clustering was performed using the K-means algorithm[23]. In particular, we used the K-means implementation in WEKA, a JAVA toolkit for machine learning algorithms[3].

Clustering accuracy was calculated by dividing the corrected clustered defects by the total number of defects. **Table 3** summarizes our best results (a full discussion of accuracy results can be found in [10])

[1] bugzilla.mozilla.org
[2] Standard list of English language STOP words ftp>//ftp.cs.cornell.edu/pub/smart/English.stop
[3] WEKA www.cs.waikato.ac.nz/ml/weka/

Input field	Accuracy		
	Max (seed)	Min (seed)	average
Description	48% (33)	7,3% (175)	29%
Summary	60% (825)	34% (275)	44%

Table 3. K-means classification of Bugzilla Hot Bug List Results

3 Automatic defect classification

The objective of this work is to evaluate if processing defects by means of NLP can match or outperform results achieved by the organization in section 2.1. In order to achieve the objective, we take the clustering machine described earlier and turn it into a classification machine. We then apply it to a sample of defects that came from the organization.

Our first challenge was to train the machine to process the Spanish language as in the previous section the dataset was based on defects from the Mozilla project which were written in English. As it proved to be hard to find preprocessing tools for Spanish, such as lemmatizers, we decided to use the words as they were. That is, we have treated each word as a token without reducing it to their base form. The subject organization uses a custom defect tracking system. Defects were exported in a Coma Separated File so as to use them as input to the analyzer. A defect report in the organization has a short title, from which individual defects are identified, a rich text description field, were the testers provide the steps for reproducing the defect, and a field that determines the category of the taxonomy in which the defect has been classified. Remaining fields like priority, project name, and version number are internal fields that are used for tracking purposes.

Table 4 shows the results we have obtained. We have used three models to represent the defect reports: using only the title field of the report, descriptions only, and both title and description fields. Each results has been validated by applying Fleiss Kappa against the classification of each defect in the sample dataset, using both the 10-fold cross validation technique (which in our dataset stands for about 34 defects) and a training-test method (66%-34% split).

When analyzing this result, it must first be observed that we have used a sample consisting of 236 defects. Nevertheless, these results are impressive when compared with the experimental results presented in section 2.1. Using the Naïve Bayes method, agreement scored similarly or worse that the experimental group. Decision Trees clearly outperforms the group, and ranks in the highest confidence interval for the Kappa’s. A broader defect sample will probably allow us to make conclusions about which field would better serve as a predictor. For instance, in the 10xfold validation method, all field scored equal, which is not the case when using the training-test method.

Input Field	Classification algorithm		Validation Method
	Naïve Bayes	Decision Trees	
Title	0.58/0.47	0,94/0.92	10x Fold
Description	0.48/0.31	0.94/0.92	
Title+Description	0.48/0.34	0.94/0.92	
Title	0.61/0.50	0.92/0.91	66-34% Split
Description	0.44/0.28	0.81/0.77	
Title+Description	0.44/0.25	0.81/0.80	

Table 4. Results from Automatic classification by Natural language Processing. Both accuracy and Kappa values are shown (accuracy/Kappa).

4 Conclusion and future work

The results we have presented are promising results for our research. First of all, with a 90% confidence on the classification, a software organization can do without training and rely entirely on automatic defect classification to classify its defects database. This is extremely important when defects are used as input for causal analysis. In a software factory like the organization we are working with, with 4-5 concurrent development projects, we favor reliable and consistent classification to correct classification of each defect. A 90% confidence rating would give the organization reasonable grounding to base its causal analysis decisions on the results of this tool. On the downside, it can be argued that completely relinquishing the classification knowledge to an automatic classifier can have a long term issue. For instances, the organization´s staff might eventually lose the learning feedback gained when classifying defects. Nevertheless, this is one of the many interesting lines of future research we can explore.

In the near future, our future efforts will include the validation of these initial results with a broader defect dataset. Another important line of future research is to apply defect clustering[10], which is the unsupervised classification of patterns into groups. The objective being to evaluate how process improvement based on defect clusters performs compared to process improvement based on taxonomies that were developed in alignment with the organizational business goals[24].

Practical application can also include the use of defect clustering together with contextual information from the organization and standard estimation techniques as a proxy to estimate remaining defect density in delivered software. As mentioned in [10], clustering can also be used to prioritize which defects to address/fix first.

Our next step will involve further development of the clustering algorithm, including an improved front end in order to make it more usable for third parties so we can obtain a bigger defect sample that would confirm our results. We are also rallying the support of a target software factory to try out our proposed method.

5 Bibliography

1 Deming, W.E.: ‘Out of the Crisis’ MIT Press, 2000, 1 st edn.
2 Chrissis, M.B., Konrad, M., and Shrum, S.: ‘CMMI : guidelines for process integration and product improvement’ Addison-Wesley, 2007.
3 Card, D.N.: ‘Defect-causal analysis drives down error rates’, IEEE Software, 1993, 10, (4), pp. 98-99
4 Buglione, L.: ‘Strengthening CMMI Maturity Levels with a Quantitative Approach to Root-Cause Analysis’. Proc. R5th Software Measurement European forum, 2008, Milan. pp.
5 Buglione, L., and Abran, A.: ‘Introducing Root-Cause Analysis and Orthogonal Defect Classification at Lower CMMI Maturity Levels ’. Proc. MENSURA, 2006, Spain. pp. 29
6 CMMI Version 1.3 Product Suite. Available at http://www.sei.cmu.edu/collaborating/spins/021009webinar.html, accessed 3/03/2008
7 Card, D.N.: ‘Learning from Our Mistakes with Defect Causal Analysis’, IEEE Software, 1998, 15, (1), pp. 7
8 Card, D.N.: ‘Managing Software Quality with Defects’. Proc. Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment, 2002. pp. 472-474
9 Mizukami, D.: ‘Analyzing Defects Can Tell a LOT About a Company’. Proc. SEPG Conference 2007, March 26 - 29, 2007 2007 pp. Pages
10 Rus, V., Sajjan, S.G., and Mohammed, S.: ‘Automatic Clustering of Defect Reports’. Proc. International Conference on Software Engineering and Knowledge Engineering. , July 2008 2008, Redwood City, USA. pp. 291-296
11 Diane, K., and Terry, S.: ‘A case study in the use of defect classification in inspections’. Proc. Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research, Toronto, Ontario, Canada2001 pp. Pages
12 Microsoft Solution Framework for CMMI Process Guideance Available at http://msdn2.microsoft.com/en-us/teamsystem/aa718802.aspx
13 Chillarege, R., Bhandari, I., Chaar, J., Halliday, M., Moebus, D., and Ray, B.: ‘Orthogonal Defect Classification-A Concept for In-Process Measurements’, IEEE Computer, 1992, 18, (11)
14 Matalonga, S., and SanFeliu, T.: ‘Linking Return on Training Investment with Defects Causal Analysis’. Proc. 20th Conference of Software Engineering Theory and Practice, July 1-3 2008, Redwood City. pp. 42-47
15 Kelly, D., and Shepard, T.: ‘A case study in the use of defect classification in inspections’. Proc., 2001. pp.
16 Matalonga, S., and San Feliu, T.: ‘Defect Driven Organizational training’. Proc. Europe SEPG 2008, 2008, Munich. pp.
17 Rus, V., and Shiva, S.G.: ‘A general framework for quantitative software testing’. Proc. First International Workshop on advances and Innovations in software Testing, 2007, Memphis, USA. pp.

18 Matalonga, S., and Feliu, T.S.: 'Using defect data to drive Organizational Training efforts'. Proc. International Conference on Software Engineering Theory and Practice, July 7-10 2008, Orlando, Florida. pp. 61-68

19 Cohen, J.: 'A Coefficient of Agreement for Nominal Scales', Educational and Psychological Measurement, 1960, 20, (1), pp. 37

20 Fleiss, J.L.: 'Measuring nominal scale agreement among many raters', Psychological Bulletin, 1971, 76, (5), pp. 378-382

21 Jain, A.K., Murty, M.N., and Flynn, P.J.: 'Data clustering: a review', ACM computing surveys, 1999, 31, (3)

22 Runeson, P., Alexandersson, M., and Nyholm, O.: 'Detection of duplicate defect reports using natural language processing'. Proc. 29th international conference on Software Engineering, 2007, Minneapolis, USA. pp. 499-510

23 Kaufmann, M.: 'Data Mining Practical Machine Learning Tools and Techniques', 2005

24 Freimut, B., Denger, C., and Ketterer, M.: 'An industrial case study of implementing and validating defect classification for process improvement and quality management', Software Metrics, 2005. 11th IEEE International Symposium, 2005, pp. 10