dr Łukasz Radliński
Institute of Information Technology in Management
Faculty of Economics and Management
University of Szczecin

Predicting Defect Types in Software Projects

Abstract

Predicting software defects has been one of the most demanding tasks for software engineering researchers and practitioners. The work in this area resulted in producing various defect prediction models. Their common weakness is that they typically treat all defects equally. However, software companies need to categorize defects found in their products to estimate user satisfaction and to prioritize which defects are supposed to be fixed first. Several classifications are supported by popular defect tracking systems, such as Bugzilla. The Defect Types Model introduced in this paper estimates proportions of defects of various types categorized by their severity for users. This model contains two groups of factors: controllable (process quality) and uncontrollable (features of software). It incorporates results from statistical analysis of the ISBSG project dataset adjusted by other reported results and expert knowledge. This model can be used either as a standalone model (predicting proportions of defects) or in combination with other defect prediction model (predicting number of defects of different types).

Keywords: defect prediction, defect types, analysis of empirical data, Bayesian net

Introduction

Most of current efforts in software defect prediction field have been focused on predicting total number of defects in software. Although software companies are strongly interested in obtaining such predictions they also need additional information on defects. Predicting types of defects is one of such extensions. Since defects can be at different levels of severity for future users, software companies need to prioritize defects and plan the testing and rework stage accordingly.

This paper introduces a Defect Types Model (DTM) which is a Naïve Bayesian Classifier – a simple Bayesian net [13]. DTM predicts proportions of defects categorized by their severity. It incorporates various sources of data: results of statistical analysis of empirical data (mainly using ISBSG dataset [7]), other published results and the expert knowledge. DTM contains both the controllable process factors and uncontrollable project factors. This model can be either used in isolation or in combination with other defect prediction models. The latter enables predicting both proportions and amount of defects of different types.

This paper is organized as follows. First the definition of dependent variables is provided. Then the steps and brief results of statistical analysis are discussed. The next section discusses the structure of the DTM which is followed by examples of its usage.

Dependent variables

ISBSG defines a defect as “a failure of some part of an application” [8]. They distinguish three types of defects in the dataset depending on their severity:

- Minor – “A minor defect does not make the application unusable in any way, (e.g. a modification is required to a screen field or report)”.
- Major – “A major defect causes part of the application to become unusable”.
- Extreme – “A failure of some part of an application that causes the application to become totally unusable”.

The aim of this study was to predict proportions of defects of these three types. The ISBSG dataset does not explicitly provide the values of these proportions but provides the amounts of defects of each type and the total number of defects. Using these data the first step was to define the three predictive variables: proportion of minor defects (*prop minor*), proportion of major defects (*prop major*), proportion of extreme defects (*prop extreme*) as shown in Equations 1-3.

$$prop\,minor = \frac{minor\,defects}{total\,defects\,delivered} \tag{1}$$

$$prop\,major = \frac{major\,defects}{total\,defects\,delivered} \tag{2}$$

$$prop\,extreme = \frac{extreme\,defects}{total\,defects\,delivered} \tag{3}$$

Statistical analysis of the dataset

The aim of statistical analysis was to identify variables influencing the proportions of defects of different types and to establish the nature (direction and strength) of these influences. This analysis was performed on a publicly available ISBSG dataset [7] containing data on 3024 software projects described by 99 variables. However, data on only 119 projects could be used in this analysis since only they met the following criteria:

- *data quality* had either ‘A’ or ‘B’ assigned (meaning respectively ‘very good’ and ‘good’ quality) as suggested in [9],
- cases where *total defects delivered* was either 0 or missing were excluded,
- cases with no detailed data on proportions of defects were excluded.

This statistical analysis involved the following sequence of steps previously performed in [14] when the same dataset was used:

1. Analyzing dependent variables and applying transformations to logged distributions;
2. Preparing list of potential predictors based on experience and availability of data:
    a. continuous: *average team size*, *defect rate*, *functional size*, *max team size*, *productivity rate*, *project elapsed time*, *summary work effort* and *value adjustment factor*,
    b. nominal, including boolean: *activity build*, *activity design*, *activity implement*, *activity planning*, *activity specification*, *activity test*, *application type*, *architecture*, *business area type*, *case tool used*, *client-server*, *development platform*, *development type*, *how methodology acquired*, *intended market*, *language type*, *organisation type*, *package customization*, *primary programming language* and *used methodology,*
    c. ordinal: *user base – business units*, *user base – concurrent users*, *user base – locations*.
3. Analyzing categories of potential nominal predictors which resulted in removing one predictor (*how methodology acquired*) from further analysis due to unclear interpretation of its values and setting new categories grouping many similar low-frequency states for some variables.
4. Identifying relationships between predictors and dependent variables using Spearman’s rank correlation coefficient, Kruskal-Wallis analysis of variance, categorized histograms and box-plots as well as analyzing if identified relationships were valid from a causal perspective. This step resulted in keeping 3 predictors for *prop minor* and *prop major*: *defect rate*, *activity test* and *activity implement*, and 8 predictors for *prop extreme*: *defect rate*, *functional size*, *productivity rate*, *activity test*, *activity implement*, *intended market*, *package customization* and *used methodology*.
5. Identifying correlations and associations among predictors to keep only these predictors which are independent on each other – using the same measures as in the previous stage and additionally: Phi, Cramer’s V and contingency coefficients. After this step the only predictor remaining for *prop minor* and *prop major* was *activity test*,

and for *prop extreme* there were only 3 predictors: *functional size*, *activity test* and *package customization*.

Bayesian net for estimating types of defects

Statistical analysis of data resulted in identifying very few predictors for dependent variables. This was caused by a general problem of predicting defect severity which is highly subjective: the same defect may be categorized as *minor* by one person but as *major* by another. As a result the dataset becomes noisy and identifying most influential predictors from such a dataset may not be possible.

Furthermore, this dataset did not contain additional variables which may influence the proportions of types of defects. However, not all of them (e.g. code metrics) can be included in the model since the model is intended to be used at an early stage of software development when little data describing project structure are available.

In many cases Naïve Bayesian Classifiers (NBC) are generated automatically from the data. Due to the reasons discussed above in this case it was not sensible to build a NBC with such unsupervised learning using the ISBSG dataset but to build a model incorporating the mixture of expert knowledge and the results from this data analysis. This enabled to add more predictors which were either not included in the dataset or which were included in the dataset but not found correlated/associated with dependent variables but which we believe influence the dependent variables. To extend model functionality Boolean variables have been converted to a ranked scale.

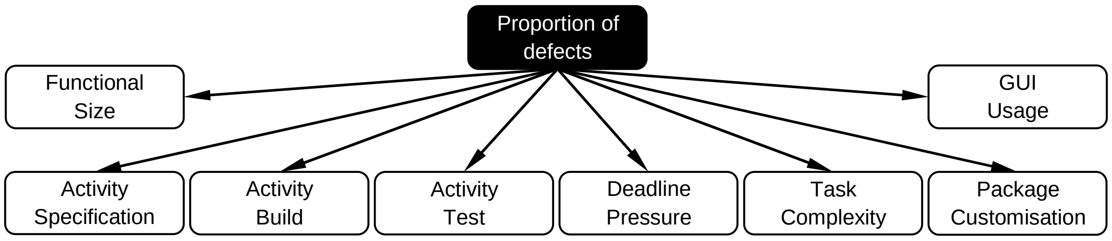
The DTM model contains one labelled node representing the dependent variable which has the following three states:

- 'prop minor' – reflecting proportion of minor defects,
- 'prop major' – reflecting proportion of major defects,
- 'prop extreme' – reflecting proportion of extreme defects.

The probability of each state reflects the expected value of proportion of each type of defects. Such a structure can be easily calibrated because the sum of probabilities of these three states is always equal to 1. This reflects the situation where the sum of proportions of all types of defects also sums up to 1.

There is one drawback in such a model structure – information on the probability distribution of the proportion of each type of defect is lost. Only the overall expected value is available. However, even with this limitation the user can benefit from obtaining predictions from such a model.

Figure 1 illustrates the structure of the DTM model and also summarizes predictors in this model. It is beyond the scope of this paper to present detailed probability tables for all variables therefore only the direction of the conditional distribution is presented.



Predictor	Predictor identified from*	Description	Scale	Impact of predictors on proportions of defects**		
				minor	major	extreme
Activity Specification	A, E	Effectiveness of specification process (spec. effort + process and people quality)	Ranked: from 'Very Low' to 'Very High'	+	–	–
Activity Build	E	Effectiveness of coding process (coding effort + process and people quality)	Ranked: from 'Very Low' to 'Very High'	+	–	–
Activity Test	A	Effectiveness of testing process (testing effort + process and people quality)	Ranked: from 'Very Low' to 'Very High'	+	–	–
Deadline Pressure	E	Extent to which there is a pressure on delivering software faster than they normally would like to deliver such software	Ranked: from 'Very Low' to 'Very High'	+	–	–
Functional Size	A	Size of the project	Numeric: 2-36316 function points	–	–	+
GUI usage	E	Extent at which graphical user interface (GUI) is used in the developed software	Ranked: from 'None' to 'High'	+	–	–
Package Customisation	A	Extent at which the project involves package customisation	Ranked: from 'None' to 'High'	–	–	+
Task complexity	A, E	How complex is the task/problem	Ranked: from 'Very Low' to 'Very High'	–	+	+

* A – results of statistical analysis, E – expert opinion
** '+' indicates positive impact – as predictor increases, dependent also increases
'–' indicates negative impact – as predictor increases, dependent decreases

Figure 1 Structure of DTM model with a summary of its predictors

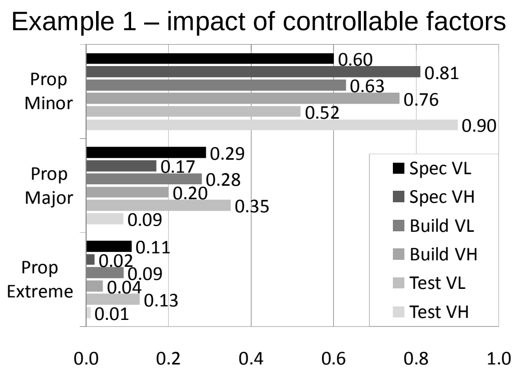
The prior proportions of number of defects of each type are mixtures from ISBSG, Apache [1] and Mozilla [12] datasets (as analyzed previously in [15]) based on the frequencies of observations and adjusted by expert judgement. They are the following:

- for 'prop minor' – 0.68,
- for 'prop major' – 0.25,
- for 'prop extreme' – 0.07.

Examples of model usage

To validate the DTM several scenarios have been applied. The aim of this validation was to analyse if the DTM correctly captures results of empirical data and expert knowledge. No validation in terms of accuracy of predictions purely against empirical data set has been performed because of different scales for some of variables in the model and in the dataset, and because the model contains additional variables missing in the dataset. This section discusses two examples: one for using DTM in isolation and another for using DTM in combination with other defect prediction model. Other examples have been discussed in [17].

In the first example let us analyze the impact of controllable factors on proportions of defects. The DTM contains three controllable factors: *activity specification*, *activity build* and *activity test*. Figure 2 (top) illustrates predictions for most ('very high') and least desired ('very low') states of these process factors. We can observe that *activity test* has the greatest impact on reducing *prop major* and *prop extreme* and increasing *prop minor* – highest spread in proportions of defects between 'very low' and 'very high' states of process factors. *Activity specification* and *activity build* have a lower impact on proportions of defects. This correctly reflects results of statistical analysis in which *activity test* was identified as the strongest predictor for proportions of defects.



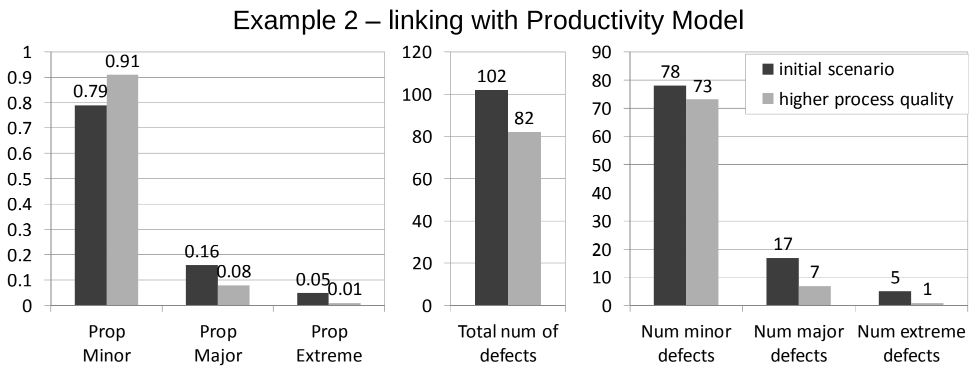


Figure 2 Predictions from DTM in Example 1 (top) and Example 2 (bottom)

Apart from predicting proportions of defects it is also possible to predict the number of defects of each type. This requires linking DTM with other defect prediction model such as: parametric [2], [5], [11], neural network [10], decision tree [6], Bayesian net [3], [4], [16] or other. This second example shows predictions obtained from linking DTM with a causal Bayesian net model – the Productivity Model [16]. Let us assume that a company has to deliver software of 1000 function points by using 3000 person-hours which indicates 'medium' *deadline pressure*. The software about to be developed is of 'medium' *complexity*, involves 'low' *package customisation* and where GUI *usage* is 'high'. We know that in the past this company delivered similar type of software by using 2500 person-hours, reaching *productivity rate* of 0.3 function points per person-hour and *defect rate* of 0.1 defects per function point.

With these observations the Productivity Model predicts that we should expect about 102 defects (median) left in software after release, and DTM predicts that we should expect the following proportions of defects: 0.79 for prop minor, 0.16 for prop major and 0.05 for prop extreme (Figure 2 - bottom). We use these predictions to estimate the number of defects of specific type. In this case we get predicted probability distributions with medians around 78 for minor, 17 for major and 5 for extreme defects.

Let us now assume that these numbers are too high to be accepted by a project manager and future users. We analyze what impact would it make if process quality on all activities (specification, coding and testing) is higher. Now, the Productivity Model predicts 82 residual defects and DTM predicts 0.91 for *prop minor*, 0.08 for *prop major* and 0.01 for *prop extreme*. After combining these predictions we obtain the following predictions: 73 minor, 7 major and 1 extreme defect. Note that, although *prop minor* is higher with increased process quality, the actual number of minor defects is lower. This is because the increased process quality leads to decrease in the total number of defects.

Summary

The Defect Types Model predicts the proportions of defects categorized by their severity. This model incorporates the results from statistical analysis of ISBSG dataset, Mozilla and Apache projects and our expert knowledge. Performed internal validation shows that predictions provided by this model can be a useful extension to traditional defect prediction models which typically provide only the total number of defects. Since this model has an NBC structure it is easy to extend the model by adding new predictors, for example code metrics. However adding such predictors shifts the time when the model can be used to a stage of development when the values of such metrics are available.

Acknowledgements

I would like to acknowledge the following individuals from Queen Mary, University of London: Prof. Norman Fenton for his overall support in developing DTM, Dr. David Marquez for his help in performing statistical analysis, Dr. Peter Hearty for his ideas in developing earlier version of the model. I am indebted to Agena, Ltd. for providing Agenarisk – a software tool for building Bayesian nets and performing simulations.

Bibliography:

1. Apache Software Foundation, ASF Bugzilla, issues.apache.org/bugzilla/report.cgi, 2008.
2. Chulani S., Boehm B., Modelling Software Defect Introduction and Removal: COQUALMO (COnstructive QUAlity MOdel), Technical Report USC-CSE-99-510, University of Southern California, Center for Software Engineering, 1999.
3. Fenton N., Neil M., Marsh W., Hearty P., Radliński Ł., Krause P., On the effectiveness of early life cycle defect prediction with Bayesian Nets, *Empirical Software Engineering*, vol. 13 no. 5, pp. 499–537, Oct. 2008.
4. Fenton N.E., Neil M., Marsh W., Krause P., Mishra R., Predicting Software Defects in Varying Development Lifecycles using Bayesian Nets, *Information and Software Technology*, vol. 43 no. 1, pp. 32–43, Jan. 2007.
5. Gaffney J.R., Estimating the Number of Faults in Code, *IEEE Transactions on Software Engineering*, vol. 10 no. 4, 1984, pp. 141–152.
6. Gokhale S.S., Lyu M.R., Regression Tree Modelling for the Prediction of Software Quality, *Proc. ISSAT Int. Conf. on Reliability and Quality in Design*, Anaheim, 1997, pp. 31–36.
7. ISBSG, Estimating, Benchmarking & Research Suite Release 9, International Software Benchmarking Standards Group, www.isbsg.com, 2005.
8. ISBSG, Glossary of Terms, V5.9.1, International Software Benchmarking Standards Group, www.isbsg.org, 2006.
9. ISBSG, ISBSG Comparative Estimating Tool V4.0 – User Guide, International Software Benchmarking Standards Group, www.isbsg.org, 2005.

10. Khoshgoftaar T., Pandya A., Lanning D., Application of neural networks for predicting program faults, *Annals of Software Engineering*, vol. 1 no. 1, pp. 141–154, Dec. 1995.
11. Lipow M., Number of Faults per Line of Code, *IEEE Transactions on Software Engineering*, vol. 8 no. 4, pp. 437–439, Jul. 1982.
12. Mozilla.org, Bugzilla@Mozilla, bugzilla.mozilla.org, 2008.
13. Pearl J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Francisco, 1988.
14. Radliński Ł., Fenton N., Marquez D., Estimating Productivity and Defect Rates Based on Environmental Factors, *Information Systems Architecture and Technology: Models of the Organisation's Risk Management*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2008, pp. 103-113.
15. Radliński Ł., Fenton N., Marquez D., Hearty P.: Empirical Analysis of Software Defect Types, *Information Systems Architecture and Technology. Information Technology and Web Engineering: Models, Concepts & Challenges*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2007, pp. 223-231.
16. Radliński Ł., Fenton N., Neil M., Marquez D., Improved Decision-Making for Software Managers Using Bayesian Networks, *Proc. 11th IASTED Int. Conf. Software Engineering and Applications*, Cambridge, MA, 2007, pp. 13–19.
17. Radlinski L., Improved Software Project Risk Assessment Using Bayesian Nets, Ph.D. Thesis, Queen Mary, University of London, 2008.