Estimating Software Fault-Proneness for Tuning Testing Activities

Giovanni Denaro
Politecnico di Milano
Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci, 32
20133 Milano (Italy)
+39-02-2399-3638
denaro@elet.polimi.it

1 INTRODUCTION

Quality of software is increasingly important and testing related issues are becoming crucial for software. Methodologies and techniques for predicting the testing effort, monitoring process costs, and measuring results can help in increasing efficacy of software testing. Being able to measure the fault-proneness of software can be a key step towards steering the software testing and improving the effectiveness of the whole process.

In the past, several metrics for measuring software complexity and testing thoroughness have been proposed. Static metrics, e.g., the McCabe's cyclomatic number [1] or the Halstead's Software Science [2], statically computed on the source code, try to quantify software complexity. Dynamic metrics, e.g., structural and data flow coverage [3, 4], measure the thoroughness of testing as the amount of elements of the program covered by test executions. Such metrics only partially reflect the many aspects that influence the software fault-proneness, and thus provide limited support for tuning the testing process. Some works try to combine existing metrics to better estimate costs and quality and thus better support the testing process [5, 6, 7]. However, none of the work presented so far provides experimental evidence of the correlation that may exists among software metrics, testing coverage, software fault-proneness, testing costs and software quality.

This thesis aims at investigating whether a correlation exists between the fault-proneness of the software and the measurable attributes of the code (i.e. the static metrics) and of the testing (i.e. the dynamic metrics). The thesis aims also at studying how to use such data for tuning the testing process. Goal of the thesis is not to find a general solution to the problem (a solution may not even exists), but to investigate the scope of specific solutions, i.e., to what extent homogeneity of the development process, organization, environment and application domain allows data computed on past projects to be projected on new projects. A suitable variety of case studies is selected to investigate a methodology applicable to classes of homogeneous products, rather than investigating if a specific solution exists for few cases.

2 THE APPROACH

We define *fault-proneness* of a software module as the probability that the module contains faults. This work investigates whether a correlation exists between such probability parameter, which is generally not directly measurable, and the static and dynamic metrics, which are measurable. We try to compute such correlation on data from past projects, where most faults have been finally identified and documented. The correlation on existing data is computed using logistic regression techniques [8]. In the following, the analytic formulas to express the relationships investigated are referred to as *models* of the software fault-proneness. The metrics involved with a given model are the independent variables of that model and the fault-proneness is the dependent variable of the model. The produced models, that describe software fault-proneness as a function of static and dynamic metrics are then used on new projects. Results on new projects are later compared with the predicted values to estimate the stability of the models and their scope. We will plan a set of experiments with different classes of software and different models to investigate whether the divergence of the estimated values, with respect to the actual data computed on different products, can identify the class of products for which the model is valid.

Building Initial Models

We perform a first set of experiments to identify models that relate software fault-proneness with static measures and can thus be applied before testing. The experiments apply to past projects where faultiness data are available. Software modules are classified as faulty or non-faulty depending on the documented faults. Statistical techniques, and in particular logistic regression, are applied to correlate the values of static metrics to module faultiness. These results are later applied to

other projects for validation and valid models are then related to estimation of costs of testing. A second set of experiments aims at correlating the fault-proneness of a software module with both the static and dynamic metrics. Such models are proposed for estimating residual faultiness, i.e., the probability of a software module with given characteristics to contain unrevealed faults once achieved a given coverage. These experiments are performed on data of faults recorded on the field. As intermediate step for this experiments, we study the ratio between the number of test cases that reveal a given fault and the number of test cases that traverse the module. Such variable represents the reliability of a given coverage, i.e., the probability that a test that satisfies the coverage reveals a potential fault.

A preliminary set of experiments is executed with many different sets of metrics, to identify candidates for the role of independent variables. The goal of these first experiments is to find models presenting high significance. We expect to be able to formulate more precise hypotheses based on the obtained preliminary results.

Validating The Models

The accuracy of the computed models needs to be measured on several projects and tuned accordingly. The initial models are thus applied on several case studies with different characteristics. These case studies should provide feedback about the validity, the applicability, the precision and the usefulness of the models. In particular, the validity is assessed by examining the goodness of models in the new contexts. Since parameters of the models need to be tuned on the data related to the new contexts, the significance indexes of the models will be correspondingly recalculated. High indexes confirm the existence of a correlation between the chosen metrics and the fault-proneness of the software. Otherwise, a more in-depth investigation for models needs to be carried out on the available data. Applicability is measured as the difficulty to use the models. Since the application of the models is fully automatizable and only marginally affects the normal testing process, we expect marginal overheads. The precision of the models is computed by examining data about faults identified during and after testing. These data have to be compared with the predictions based on the models. The evaluated differences could confirm the validity or enable new hypothesis about the shape of the relationships among the variables involved in the process. The effectiveness of the actual process with respect to past testing sessions should provide feedback about the usefulness of the models.

A New Process Component

The approach investigated in the thesis to the problem of ameliorating a generic industrial testing process has an adaptive nature. First, by analyzing data documenting past testing experiences, it support the definition of statistical models of software fault-proneness which take into account context specific parameters. Second, by incorporating a tuning mechanism on the process, it allows for learning by experience and maintaining the process and the models up to date with respect to ongoing achieved results. Parameters of the models employed and, sometimes, the models themselves may vary for different industrial processes. However, the approach defines general guidelines for identifying, applying and maintaining such models. The proposed methodology identifies a new component of the testing process that support monitoring of process costs and results. The new component can be easily integrated into an existing testing process without significant overheads.

3 FIRST EXPERIMENTS

The experiments carried out so far demonstrate the existence of a correlation between static metrics and software fault-proneness in the class of generalized linear models [9]. The first experiments have been carried out on an antenna configuration software application named Space in use at the European Space Agency, chosen due to the large amount of available testing documentation. In particular for the program Space, we have 38 documented faults identified during testing, and more than 13,000 tests that have been executed for the program. The first set of experiments have been designed to measure the correlation between static metrics and fault-proneness. We considered 37 different metrics collected with several commercial and prototype tools. Faultiness of module m is modeled with a dependent variable Y_m defined as follows:

$$Y_m = \begin{cases} 1 & \text{if the module } m \text{ is known to be faulty} \\ 0 & \text{otherwise} \end{cases}$$

The correlation between metrics and fault-proneness has been studied using logistic regression, that, given a set of independent (subsets of static metrics) and dependent variables (the module faultiness Y_m), provides the following data:

- a value b_i for each independent variable. The b_i values represent the coefficients of the linear combination of independent variables that constitutes the model for estimating the value of the dependent variable;
- a coefficients p_i for each independent variable. Each p_i ranges between 0 and 1, and indicates the significance of the referred variable in the model. The lower the p_i value, the lower the probability we have detect the correlation between the dependent and the independent variable just by chance;
- a value R^2 that indicates the goodness of fit of the model. The closer the value of R^2 to one, the higher

the significance of the model. Logistic regression R^2 is not to be confused with the R^2 of least-square regression. Very high values of R^2 are rare for logistic regression. Values of R^2 greater then 0.4 are considered quite good in the context of logistic regression.

The following equations sample the quality of the results achieved so far:

$\ln \frac{E[Y]}{1-E[Y]} = -4.927 - 0.57eLOC - 0.545Comments + 0.53Lines + 0.36IC - 0.46CONTROL$

$\ln \frac{E[Y]}{1-E[Y]} = -4.767 + 0.479FP - 0.682CONTROL + 0.664V(G) - 0.502LOC + 0.451Lines - 0.425Comments$

In these models the notation E[Y] refers to the expected value of Y. The independent variables that occur in these two equations are: *eLOC*, the number of executable lines of code; *Lines*, the total number of lines of code; *Comments*, the number of lines containing comments; *IC*, the interface complexity of the module given as the sum of the number of the input parameters and the number of the exit points; *CONTROL*, the number of control statements; *FP*, the number of formal parameters; *V(G)*, the cyclomatic number. The coefficients p_i provided by logistic regression are very low for all the independent variables of both models, meaning that all the chosen variable have a significant impact on the dependent variable. The R^2 values of both models (0.5176 and 0.5513, respectively) are high in the context of logistic regression, meaning that both models have a high statistical significance. These preliminary results show that we can build statistically significant models based on a suitable combination of metrics. The experiments that we are currently performing aim at investigating the reliability of these models for predicting faultiness of new software.

Less encouraging, up to now, the experiments performed for estimating the ratio between tests that reveal a fault and tests that traverse the faulty modules. In this case we use both logistic regression and least-squares analysis. Analysis based on logistic regression results in modules with too many independent variables, which are not statistically relevant. Analysis based on least-squares regression results in models with low significance indexes. Although not encouraging, the few experiments executed so far do not allow to conclude that the investigated relation does not exist. We are currently working on a large set of experiments for tuning the results and identify different dependencies that can be studied.

4 FUTURE PLANS

The experiments performed so far suggest the existence of a correlation between a reasonable set of static metrics and software fault-proneness. We plan to execute a larger set of experiments to tune the early models and to identify the parameters that indicate the class of software that preserve the validity of a given model. The identified ratio between tests that reveal a fault and tests that traverse the faulty elements seems to be an interesting indicator of the significance of the coverage, although results obtained so far need confirmation. We plan to design and perform a set of experiments to study the correlation between this variable, static characteristics of software, classes of faults, and residual faultiness. We will than study the correlation between estimated fault-proneness, testing costs and software quality.

5 ACKNOWLEDGEMENTS

I would like to thank Mauro Pezzè, my advisor for this thesis, for the time he spent both in brainstorming on the described concerns and in reviewing the manuscript.

REFERENCES

[1] T. J. McCabe, *A Complexity Measure.* IEEE Transactions on Software Engineering, SE-2(4), pp. 308-320, Dec. 1976.

[2] M. H. Halstead, *Elements of Software Science.* North Holland, New York, 1977.

[3] E. Miller and W. E. Howden, *Tutorial: Software Testing and Validation Techniques.* IEEE Computer Society Press, 1981.

[4] L. Clarke, A. Podgurski, D. Richardson and S. Zeil, *A Formal Evaluation of Data-Flow Path Selection Criteria.* IEEE Transaction on Software Engineering, vol. 15, num. 11, pag. 1318-1332, Nov. 1989.

[5] J.P. Cavano and J.A. McCall, *A Framework for the Measurement of Software Quality.* Proceedings of the Software Quality and Assurance Workshop, San Diego, CA, 133–139, ACM SIGMETRICS and SIGSOFT, Nov. 1978.

[6] F. Brito and Abreu, *The MOOD Metrics Set.* Proceedings of the ECOOP'95 Workshop on Metrics, 1995.

[7] P. Liggesmeyer, *Selecting Test Methods, Techniques, Metrics, and Tools Using Systematic Decision Support.* In Proceedings of Software Quality Week, San Francisco, CA, May, 1996.

[8] D. Hosmer and S. Lemeshow, *Applied Logistic Regression.* Wiley-Interscience Publication, 1989.

[9] P. McCullagh and J. A. Nelder, F. R. S., *Generalized Linear Models.* Monographs on Statistics and Applied Probability, Chapman and Hall, 1983.