

Using Process Metrics to Enhance Software Fault Prediction Models

Greg Kaszycki gjkaszyc@NortelNetworks.com

Abstract:

There are several research papers that attempt to predict risk by analyzing the source code and applying a quantitative model. The accuracy of these models is significantly enhanced if process data is included in the development of the quantitative model.

Introduction

EMERALD is a software risk assessment tool that was developed for internal use within Nortel Networks and is now commercially available. Like other risk assessment models or software engineering tools, it examines the code to look for complexity and potentially problematic constructs to estimate risk. Unlike other tools, it compliments the complexity data with process data and customizes the model to the historical data from a specific development environment. The data for this paper is drawn from the models that have been created for the EMERALD product. The companies for which these models have been created are private and are omitted from this paper.

Definitions

- A "field fault" is a software fault that goes undetected in the development debugging and testing, and is identified in the customer's operational system.
- A "risk model" is a formula or set of rules to predict whether a software module is more likely to have a field fault.
- A "module" refers to the component of code that is managed by the CM system. It is the entity at which faults are tracked and it is the target of the software risk model.
- A "release" is a version of the entire software package that is delivered to a customer. If it is not released to the customer, then there is no field fault data

"process data" includes the data that is gathered in and by the problem tracking system and the

configuration management system. It is likely to include things like:

- Number of changes since last release
- Number of faults found since last release
- Number of different developers who turned over versions of this module since last release
- Number of features that were added that affected this module

Process data may also include things like experience levels of the developers, the amount of time that the module spent in review, the number of defects found in reviews, the number of test cases (and unique test cases) run that touched the module.

As the software process varies from customer to customer, so does the amount and quality of the process data that is available. Therefore, it is difficult to construct a generic model that will utilize process data and be available to all customers. This is one reason that a customized model is necessary to properly include the process data.

Statistical Significance

One simple aspect of process data that greatly affects the likelihood of a fault is whether or not the module has been recently modified. While this may seem simplistic, a parser or analysis tool that only looks at the code cannot determine this on its own. Historically, we find that a module that has been modified is 10 times more likely to have a field fault than a module of equal complexity that has not been modified in the given release.

Additional process metrics are often statistically significant (and therefore contribute to a more accurate model). The following table shows the statistical significance of various factors in a logistic regression analysis used for field fault prediction. This is a specific example from a specific set of software data, but it is representative of our general experience with risk models using process data.

Since the statistical significance of an attribute can change in the presence of other factors, the table shows the significance of the factor by itself and in the full model. The full model includes seven factors (four of which are aspects of the code alone).

For our example data, we found the following significance levels for process data:

Process metric	Significance level by itself Chi-squared & significance	Significance level in the full model (including complexity attributes)
Modified lines of source code	53.40 / .0000	4.02 / .0449
Updates	86.07 / .0000	15.86 / <.0001
Programmer experience level	28.07 / .0000	9.59 / .0020

Since the process metrics are significant, even in the presence of other factors, it is expected that the total accuracy of the model is improved by including them.

Since the cost of a software fault in the field is high, software risk models should be constructed to err on the side of caution. This means that some modules will be falsely accused, but coverage will be increased. The models used in the following tables were constructed with this bias for the cost function in mind.

Both of these models are logistic regression models. It is important to note that the model without process metrics is a full analysis that was restricted to not use process metrics. It is not simply the same model with the process metrics removed.

Without process metrics		Predicted		
		Low risk	High risk	Total
Actual	Fault free	71.6%	16.8%	88.4%
	Faulty	3.2%	8.4%	11.6%
	Total	74.8%	25.2%	100%

So 33.3% of the modules predicted "high risk" and 4.3% of the modules predicted "low risk" have faults.

With process metrics		Predicted		
		Low risk	High risk	Total
Actual	Fault free	72.4%	16%	88.4%
	Faulty	2.5%	9.1%	11.6%
	Total	74.9%	25.1%	100%

While identifying about the same number of modules as "high risk", coverage was increased by 8.3%. Thus 21.9% fewer faults escape identification.

Conclusion

Including process data and customizing the risk model significantly increases the accuracy of the risk model. The utility of a risk model is allowing the organization to focus its review, inspection and test efforts on a smaller set of high-risk modules. A model with higher accuracy translates into even more efficient allocation of resources and more effective fault detection earlier in the development process.