N. Ohlsson and C. Wohlin, "Identification of Failure-Prone Modules in Two Software System Releases", Proceedings Twenty-First Annual Software Engineering Workshop, Greenbelt, Maryland, USA, 1996.

Identification of Failure–Prone Modules in Two Software System Releases

Niclas Ohlsson and Claes Wohlin
Dept. of Computer and Information Science
Linköping University, S-581 83 Linköping, Sweden
E–mail: (nicoh, clawo)@ida.liu.se

1 Introduction

This paper presents a case study of fault and failure data from two consecutive releases of a large telecommunication system. In this context it is important to have clear interpretations of errors, faults and failures. Thus, we would like to make the following distinction between them. Errors are made by humans, which may result in faults in the software. The faults may manifest themselves as failures during operation. Thus, faults can be interpreted as defects in the software and failures are the actual malfunction in an operational environment. In this paper we have used *fault–prone modules* to denote the modules that account for the highest number of faults disclosed during testing, while *failure–prone modules* is used to denote the modules accounting for the highest number of faults disclosed during the first office application and in operation. The general objective of the study is to investigate methods of identifying failure–prone software modules. Furthermore, the goal is to use the knowledge acquired to improve the software development process in order to improve software quality in the future.

Some early results using parametric statistics have been reported in (Ohlsson and Alberg, 1996). The models have since been refined and analysed with non–parametric statistics (Ohlsson et al., 1996). Identification of fault–prone modules has also been addressed by other researchers (Khoshgoftaar and Kalaichelvan, 1995) and (Munson and Khoshgoftaar, 1992). Few, if any, studies have exploited the opportunities to identify not only fault–prone modules, but also failure–prone modules which are the main concern of the user. There is also a general lack of studies investigating whether identification of fault–prone modules means that we actually also identify failure–prone modules.

Another important issue is to establish when in the development phase we are able to identify modules which will be failure–prone in the operational phase. This paper investigates three different times for prediction: history (previous release), the design phase and the test phase. One important consideration is to address whether or not fault–prone modules during testing are failure–prone during operation. If fault–prone does not imply failure–prone, then we may have to improve the test methods.

The paper is organized as follows. In Section 2, an overview of the study is presented. Section 3 discusses identification of failure–prone modules based on experience from a previous release, and Section 4 presents results using prediction models based on design measures. In Section 5, results concerning identification of failure–prone modules based on test data are presented. Finally, some conclusions are given in Section 6.

2 Overview of study

This paper is part of a long–term empirical study conducted at Ericsson Telecom AB with the objective of studying how identification of fault and failure–prone modules can be used to achieve cost–effective quality improvement. In release n of the system 130 modules have been analysed and in release n+1 232 modules have been investigated. Fault and failure data have been collected from functional testing, system testing, first office application (i.e. the first 26 weeks and a number of site tests) and operation. It was possible to trace 69 modules developed for release n that were modified in release n+1. Release n+1 is a major system revision. Data is currently being collected for release n+2. The modules are of the size of 1000 to 6000 lines of code each.

Promising results concerning identification of fault–prone modules have been presented elsewhere, i.e. design measures were used to identify fault–prone modules (Ohlsson et al., 1996) and (Ohlsson and Alberg, 1996). The objective here is to study the identification of failure–prone modules based on fault and failure data as well as from design measures. In this paper we have used one failure as threshold for the dependent variable, i.e. modules with one or more failures are classified as failure–prone. The underlying analysis of design measures is based on ordinal analysis, as it allows for changing the threshold with regards to what are viewed as being fault– and failure–prone modules (Ohlsson et al., 1996). Actual threshold–values are not recommendations; thresholds should be determined in individual projects on the basis of, for example, the level of criticality of the system and market requirements. The primary objective of the thresholds as presented in this paper is to illustrate the outcome when applying the methods for identification of failure–prone modules.

The predictability of the different models is viewed in Contigency tables and the kappa coefficients are calculated to measure the agreement in classification of the modules (Siegel and Castellan, 1988). The kappa coefficient is the ratio of the proportion of times that the classifications is correct to the maximum proportion of times that the classifications could be correct. If the classifications completely agree, then kappa=1; whereas if there is no agreement between the classifications, then kappa=0. Kappa will assume -1 if there is a perfect missclassification.

The study is divided into three parts:

1. Identification of failure–prone modules using data from a previous release

   This part is aimed at investigating whether the information from release n concerning fault– and failure–prone modules is a good predictor of failure–prone modules in release n+1. More than 90 percent of the modules in release n had one or more faults. Therefore, it is infeasible to use one fault as a threshold. Thus, when fault–prone modules from release n is used to predict failure–prone modules in release n+1, a threshold of five faults is used for the independent variable as an indication of potential failure–prone modules. When failure–prone modules in release n are used as the independent variable, one failure is used as threshold.

2. Identification of failure–prone modules using design measures

   The initial objective was to build prediction models in release n for identification of failure–prone modules based on design measures, which then should be validated with data from release n+1. Due to variation in quality between the two releases this was not possible. Instead design metrics were only evaluated within release n+1. Only the best design measure is

reported here, as the main objective is to investigate different opportunities to identify failure–prone modules rather than evaluate which measures are the best predictors. To the best of our knowledge there exists no empirical evidence that complexity values higher than a specific threshold would indicate either fault– or failure–prone modules. However, there are results suggesting relative stable distribution in line with the Pareto principle (Ohlsson et al., 1996). Therefore, the threshold is based on the percentage of failure–prone modules in release n+1. That is, 29 percent of the modules in n+1 had one or more failures. Hence, this percentage value is used as a threshold for the design measures.

3. Identification of failure–prone modules from fault–prone modules

   The objective of this part is to investigate whether the fault–prone modules identified in release n and n+1 are good indicators of failure–prone modules in the two releases. This means that fault data from testing is used to predict failure–proneness during operation. The rationale for selecting thresholds is the same as in part 1.

To summarize, the main difference is when prediction can be made. The three parts imply three different points of time in a project, namely: project start (part 1), design phase (part 2), and testing phase (part 3). It is important to remember that the sooner we are able to identify modules which are likely to be failure-prone, the sooner we can take appropriate measures to deal with them. For example, we can allocate the best people, intensify inspections or take other special improvement measures.

3 Failure–prone modules from history

For software systems, it is normal practice that a system is regularly upgraded and released in new versions. This implies that some parts of the system are the same in different releases. This information can be used to apply experience from one release to the next release or following releases. In this empirical study, the hypothesis is that fault– or failure-prone modules in release n are likely candidates for being failure–prone in release n+1. It was possible to trace 69 modules developed for release n that were modified in release n+1. The data from the historical analysis is shown in Table 1. It should be noted that only four modules were failure–prone in release n, see analysis A, while 18 modules were failure–prone in release n+1.

To evaluate the goodness of the predictions, the prediction errors must be considered. This includes two different types of errors: failing to identify failure–prone modules and identification of modules as failure–prone when they are not. These are hereafter referred to as errors of type I and II respectively. It should be noted that a correct identification means actually pin–pointing a certain module correctly.

To evaluate the goodness of the predictions, the prediction errors must be considered. This includes two different types of errors: failing to identify failure–prone modules and identification of modules as failure–prone when they are not. These are hereafter referred to as errors of type I and II respectively. It should be noted that a correct identification means actually pin–pointing a certain module correctly.

Table 3–1.

	Analysis A[a]		**Analysis B[b]**		**Analysis C[c]**	
	Threshold=1		**Threshold=5**		**Threshold=5**	
	Failure(n)		**Fault(n)**		**Fault+ Failure(n)**	
Actual	F	Not F	F	Not F	F	Not F
Failure-prone(n+1) (18 observation)	4	14	14	4	15	3
Not Failure-prone(n+1) (51 observations)	0	51	28	23	28	23
Total observations	4	65	42	27	43	26
Misclassifications of type I and II	78% (14/18)	0% (0/51)	28% (4/18)	55% (28/51)	17% (3/18)	55% (28/51)
Overall misclassifications	20% (14/69)		46% (32/69)		45% (31/69)	

a. Kappa 0.30

b. Kappa 0.16

c. Kappa 0.32

Analysis A in Table 1 illustrates that even though the type I error is as high as 78%, there is no type II error. This means that the modules that are failure–prone in release n are all failure–prone in release n+1. Possible explanations for this are the actual type of failure and late erroneous fault correction in test.

For analyses B and C, we have used five faults as a threshold for the independent variable. It has earlier been suggested (Khoshgoftaar and Kalaichelvan, 1995) that this should be used as threshold for fault–prone modules. The threshold could therefore indicate failure–proneness. Using one fault is not reasonable since this would identify 63 modules as being failure–prone. Even with a threshold of five faults in analysis B as many as 61 percent (42/69) of the modules are identified in release n as failure–prone. However, only 78 percent (14/18) of all the failure–prone modules in release n+1 are identified. Therefore, fault–prone modules in release n are poor predictors of failure–prone modules in n+1. This is also true for analysis C.

Another possible alternative would be to select a threshold based on the percentage of failure–prone modules in release n+1, i.e. assuming that this proportion of fault– and failure–prone modules will be stable over later releases. The number of potential failure–prone modules would be more realistic using 26 percent (18/69) as a threshold. However, only 28 percent of the failure–prone modules would be identified. This also holds for analysis C. Therefore, the two models in analyses B and C are not applicable.

4 Failure–prone modules from design measures

Earlier studies (Ohlsson et al., 1996) have indicated that models built on design metrics are worthwhile when the total number of faults and failures are considered as the dependent variable. Thus, it is reasonable to try this approach for failure–prone modules. In this study, fourteen different design measures are used to build prediction models for release n+1. Spearman's correlation

coefficient (Siegel and Castellan, 1988) was used for a first analysis. All potential variables have low correlation values (below 0.35). There was, however, a rather low correlation among some of the variables, hence it could be possible to improve the model by combining the variables into more complex models. Multiplicative aspects of the potential variables will be investigated in later studies. In this particular case, the best design measure predictor was IS, which is the number of input–signals for a module in the design. The result was later compared with lines of code, which was found to be doing even worse.

It has been suggested that prediction models should first be developed for one release, validated in the succeeding release, and then applied in the third release. However, the quality of the two releases varied widely, and it was therefore not possible to do so in this study. From a modelling point of view, the number of failure–prone modules in release n was too few. Instead, the explanatory ability of design metrics was evaluated by building the best possible model based on data in release n+1. The results shown in Table 2 are based on a threshold of one failure, which corresponds to 29 percent of the modules.

Table 4–1.

	Analysis[a]	
	IS(n+1)	
Actual	F	Not F
Failure-prone(n+1) (67 observation)	28	39
Not Failure-prone(n+1) (165 observations)	39	126
Total observations	67	165
Misclassifications	58% (39/67)	24% (39/165)
Overall misclassifications	34% (78/232)	

a. Kappa 0.18

From Table 2, it can be seen that the explanatory ability is unsatisfactory, i.e. the misclassification is too high, including a large proportion of both type I and II errors. This, in combination with the fact that the quality of the two releases differed, suggests that more complete models should be investigated, for example including verification effort and quality.

5 Failure–prone modules from fault–prone modules

The data from the testing phase can be used for both releases to predict the failure–prone modules. The problem with choosing relevant thresholds, discussed in respect to part 1, is relevant for this part, too. The results of the analyses are shown in Table 3, using a threshold of five faults for the independent variable.

Table 5–1.

Analysis n[a]			Analysis n+1[b]		
	Fault(n)			Fault(n+1)	
Actual	F	Not F	**Actual**	F	Not F
Failure-prone(n) (13 observation)	5	8	Failure-prone(n+1) (67 observation)	47	20
Not Failure-prone(n) (117 observations)	77	40	Not Failure-prone(n+1) (165 observations)	102	63
Total observations	82	48	Total observations	147	83
Misclassifications	62% (8/13)	66% (77/117)	Misclassifications	30% (20/67)	62% (102/165)
Overall misclassifications	65% (85/130)		Overall misclassifications	53% (122/232)	

a. Kappa -0.08

b. Kappa 0.06

The misclassification is also too high in this analysis. This means that modules that are fault–prone during testing are not failure–prone. A possible explanation is that other types of defects are discovered in operation, such as performance problems, that are difficult to test. This explanation is supported by experienced developers from Ericsson. This could also explain the result in part 1. A possible explanation of the fact that failure–prone modules in n are failure–prone in n+1 could be that modules which are critical from a capacity perspective in release n, will remain so in release n+1. The results indicate the need for a better understanding of the types of defects that result in failures and the types of the failures themselves. The results also stress the need to identify factors causing the defects which result in failures. Increased understanding is essential for quality improvement.

6 Conclusions

In this paper we have investigated the opportunity to predict failure–prone modules based on fault and failure data from two succeeding releases, design metrics, as well as test data. The study revealed that failure–prone modules in release n are failure–prone in n+1. Other suggested independent variables are poor predictors of failure–proneness. However, this is not the same as saying that they do not explain any of the variation. It only means that on their own they are poor explanatory factors. Instead, the study suggests that methods that combine these different independent variables are needed.

In this study, we have addressed two consecutive releases of a software system. This is an important aspect as in most cases it is not possible to both build, validate and use a prediction model within one release. It is, thus, important to investigate how to build models in one release, validate the model in the next release and then use the model in the third release. The transferability of a model between a software system's releases is crucial to success in the mission of identifying failure–prone modules prior to the operational phase.

A major problem with predictions is that failures are dynamic, hence it may be difficult to identify failure–prone modules using static measures. This is an issue which has to be further studied. One potential solution would be to take the use of modules into account when predicting failure–

proneness. This would allow for capturing the dynamic aspects of usage in the independent variable.

Another important issue which has been addressed here is the point of time when we are able to identify failure–prone modules. To improve the usefulness of the predictions, they should preferably be done at an early stage. In this study, we have focused on data from the previous release, the design and the test phase. The knowledge from the previous release is important in identifying failure–prone modules, but this is not a feasible approach for new modules. Thus, it is very important to find early indicators of failure–proneness, since this is the only way to enable us to address the problem within the same release.

Models which identify failure–prone modules are important not only in enabling prediction during the operational phase, but also as a planning and control tool during development. Managers may use these models to improve the resource allocation for design, both in terms of effort and experience. Furthermore, knowing which modules are most likely to be failure–prone in operation suggest that the modules will be tested and inspected differently. Therefore more attributes need to be considered and incorporated in the models, for example verification effort and quality, in line with Fenton et al. (Fenton et al., 1995), to explain the variation and to be able to apply the models in subsequent releases.

Future work should not only aim at building these more complete models, but also aim at investigating additative and multiplicative aspects of design measures and measures from different phases, in order to gain more knowledge about how such a component fits into a more complete model. The results in this study also suggest that prediction models that are only based on test data will have limited applicability in real projects aiming at addressing operational issues.

Acknowledgement
The authors would like to thank Ericsson Telecom AB for supporting this empirical study.

References

Fenton, N. E., Neil, M., and Ostrolenk, G. (1995). Metrics and models for predicting software defects. Technical Report CSR/10/02, Centre for Software Reliability, City University, London, UK.

Khoshgoftaar, T. M. and Kalaichelvan, K. S. (1995). Detection of fault-prone programs modules in a very large telecommunication system. In Proceedings of *The Sixth International Symposium on Software Reliability Engineering*, pages 24–33, Toulouse, France.

Munson, J. C. and Khoshgoftaar, T. M. (1992). The detection of fault-prone programs. *IEEE Transactions on Software Engineering*, 18(5):423–433.

Ohlsson, N. and Alberg, H. (1996). Predicting fault-prone software modules in telephone switches. *To appear in IEEE Transactions on Software Engineering*.

Ohlsson, N., Helander, M., and Wohlin, C. (1996). Quality improvement by identification of fault-prone modules using software design metrics. In Proceedings of *Sixth International Conference of Software Quality*, pp. 1-13, Ottawa, Canada.

Siegel, S. and Castellan, N. J. J. (1988). *Nonparametrics Statistics for the Behavioral Sciences*. McGraw-Hill, second edition.