Defect Classification and Defect Types Revisited

Stefan Wagner
Technische Universität München
Boltzmannstr. 3
85748 Garching b. München, Germany
wagnerst@in.tum.de

ABSTRACT

There have been various attempts to establish common defect classification systems in standards, academia and industry. The resulted classifications are partly similar but also contain variations and none of which is accepted as a basic tool in software projects. This position paper summarises the work on defect classifications so far, proposes a set of challenges and the direction to a solution.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Software quality assurance (SQA)*; D.2.5 [**Software Engineering**]: Testing and Debugging

General Terms

Documentation, Measurement

Keywords

Bugs, Faults, Defects, Defect Types, Defect Classification, Defect Taxonomy

1. INTRODUCTION

There is an obvious drive of humans to classify everything around them in order to cope with the world more easily. This also holds for defects in software systems. Various classifications and typings have been developed over the last decades for different reasons such as improving defect detection or educating developers. Although there are several often cited classifications and even an IEEE standard [7], none of which have become a truly and broadly applied practice. In practice, we often see only a classification of impact or severity. This helps in project management to decide which defects to correct first or at all. However, many of the other anticipated benefits of such classifications cannot be realised. We summarise the state of the art in defect classification and defect types, pose the current challenges in that area and conclude with some ideas for the way ahead.

2. STATE OF THE ART

The large variety of research on the differences of defects and their nature cannot be covered completely here but we concentrate on some prominent examples. We can roughly divide them in three categories: (1) defect taxonomies, (2) root cause analysis, and (3) defect classification. Defect taxonomies are categorisations of faults, mostly in code, that are based on the details of the implementation solution, e.g., wrong type declaration, wrong variable scope, or wrong interrupt handling. A well-known example of this kind is the taxonomy of Beizer [1]. An even more detailed approach is root cause analysis where not only the faults themselves are analysed but also their cause, i.e., the mistakes made by the development team. The goal is to identify these root causes and eliminate them to prevent faults in the future. Root cause analysis has, for example, been used at IBM [9]. In general, root cause analysis is perceived as rather elaborate and the cost/benefit relation is not clear. Therefore, defect classifications aim at reducing the costs but sustain the benefits at the same time. The categorisation uses more coarse-grained *defect types* that typically have multiple dimensions.

An IEEE standard [7] defines several dimensions of defects that should be collected. This starts from the process activity and phase the defect was detected, over the suspected cause, to the so-called type that is similar to a taxonomy. Interestingly, also the source in terms of the document or artefact is proposed as a dimension of the classification. However, applications of this standard classification are not frequently reported.

The mainly used defect classification approaches have been proposed by companies: IBM and HP. The IBM approach is called *Orthogonal Defect Classification (ODC)* [2]. A defect is classified across the dimensions (1) defect type, (2) source, (3) impact, (4) trigger, (5) phase found, and (6) severity. The defect type is here one of eight possibilities that allow an easy and quick classification of defects and are sufficient for analysing trends in the defect detection. Triggers are the defect-detection techniques that detect the defects and hence it is possible to establish a relationship between defect types and triggers. Kan [8] criticises that the association between defect type and project phases is still an open question and that the distribution of defect types depends also on the processes and maturity of the company.

Similar to ODC is the HP approach called *Defect Origins, Types, and Modes* [6]. The name already gives the three dimensions a defect is classified in. The origin is the source of the defect – as in the IEEE standard –, the types are also a coarse-grained categorisation of what is wrong, and the mode can be one of *missing*, *unclear*, or *wrong*. Again the type of artefact – the origin – is documented as opposed to the activity. In contrast to ODC we can analyse the relationships between defects and document types but the defect-detection techniques – the triggers in ODC – are not directly documented.

However, it has been found in different case studies [4, 13, 5] that general defect type classifications are difficult to use in practice and need to be refined or adapted to the specific domain and project environment. In [5, 10] it is shown how specific adaptions and domain-specific defect types can be found and defined. Yet, we are not aware of a larger use of these approaches.

3. CHALLENGES

Having discussed the current state-of-the-art, the question is what does hamper the full-blown adoption of defect classification in practice? What do we have to achieve first?

Different Artefacts. Over the life-cycle of a software system a variety of artefacts are created and for many of them we are actually interested in the types of defects they contain. However, can we use similar defect classifications for all of these artefacts? If not, how can we describe the interconnections of the defects (e.g. defect propagation)?

Dimensions. All the existing classifications available use partly different dimensions. However, it is unclear what dimensions are necessary. What is the basic set that allows the different scenarios of usage of these classification? What can be expected to be documented by the quality engineers? A large amount of empirical studies are necessary to analyse the important factors influencing defects and the effort for collecting them.

Defect Type Distributions. For some classifications, mainly the ODC, there is data published about typical distributions of defects but in general the empirical knowledge is rather sparse [11]. However, for using defect types for goals like optimisation of defect detection [12] far more knowledge is necessary. Is it possible to find reasonable but general distributions of defect types? If not, on what factors does it depend? For example, can we have domain-specific defect type distributions?

Connection to Quality Models. In the end, what we want to achieve with defect prevention and detection is quality. However, the work on defect classifications does only partially relate the classifications to quality models. There are sometimes *security* defects, for example, but this is not consequently related to quality models such as the ISO 9126. Hence, this connections must be clearly established in order to have a clear quality improvement when a certain defect type is prevented or corrected. Actually, modern quality models [3] define rather low-level quality attributes that could be seen as kind of defect types.

4. CONCLUSIONS

The ideal for quality assurance would be first to have means to make a well-founded estimate of the defect types in the important artefacts developed during the software life-cycle. Second, good, general guidelines for which defect-detection techniques are well-suited to detect which defect types are needed. Then we would be able to optimise the quality assurance by using those techniques that best find "our" defects. For this, we need to analyse in more detail the factors that influence defect introduction and reflect those findings in defect classifications. Therefore, we propose to invest more effort in developing applicable but standardised defect classifications in order to aid the collection of empirical knowledge.

5. REFERENCES

[1] B. Beizer. *Software Testing Techniques*. Thomson Learning, 2nd edition, 1990.

[2] R. Chillarege. Orthogonal Defect Classification. In M. R. Lyu, editor, *Handbook of Software Reliability Engineering*, chapter 9. IEEE Computer Society Press and McGraw-Hill, 1996.

[3] F. Deissenboeck, S. Wagner, M. Pizka, S. Teuchert, and J.-F. Girard. An activity-based quality model for maintainability. In *Proc. 23rd International Conference on Software Maintenance (ICSM '07)*. IEEE Computer Society Press, 2007.

[4] J. Durães and H. Madeira. Definition of Software Fault Emulation Operators: A Field Data Study. In *Proc. 2003 International Conference on Dependable Systems and Networks (DSN '03)*, pages 105–114. IEEE Computer Society, 2003.

[5] B. Freimut, C. Denger, and M. Ketterer. An Industrial Case Study of Implementing and Validating Defect Classification for Process Improvement and Quality Management. In *Proc. 11th IEEE International Software Metrics Symposium (METRICS '05)*. IEEE Computer Society, 2005.

[6] R. B. Grady. *Practical Software Metrics for Project Management and Process Improvement*. Prentice-Hall, 1992.

[7] IEEE Std 1044-1993. *IEEE Standard Classification for Software Anomalies*, 1993.

[8] S. H. Kan. *Metrics and Models in Software Quality Engineering*. Addison-Wesley, 2nd edition, 2002.

[9] R. G. Mays, C. L. Jones, G. J. Holloway, and D. P. Studinski. Experiences with Defect Prevention. *IBM Systems Journal*, 29(1):4–32, 1990.

[10] T. Nakamura, L. Hochstein, and V. R. Basili. Identifying domain-specific defect classes using inspections and change history. In *Proc. 2006 ACM/IEEE International Symposium on Empirical Software Engineering (ISESE '06)*, pages 346–355. ACM Press, 2006.

[11] S. Wagner. A Literature Survey of the Quality Economics of Defect-Detection Techniques. In *Proc. 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE '06)*. ACM Press, 2006.

[12] S. Wagner. A Model and Sensitivity Analysis of the Quality Economics of Defect-Detection Techniques. In *Proc. International Symposium on Software Testing and Analysis (ISSTA '06)*, pages 73–83. ACM Press, 2006.

[13] S. Wagner, J. Jürjens, C. Koller, and P. Trischberger. Comparing Bug Finding Tools with Reviews and Tests. In *Proc. 17th International Conference on Testing of Communicating Systems (TestCom'05)*, volume 3502 of *LNCS*, pages 40–55. Springer, 2005.