Results and Experiences from an Empirical Study of Fault Reports in Industrial Projects

Jon Arvid Børretzen and Reidar Conradi

Department of Computer and Information Science,
Norwegian University of Science and Technology (NTNU),
NO-7491 Trondheim, Norway
borretze@idi.ntnu.no, conradi@idi.ntnu.no

Abstract. Faults introduced into systems during development are costly to fix, and especially so for business-critical systems. These systems are developed using common development practices, but have high requirements for dependability. This paper reports on an ongoing investigation of fault reports from Norwegian IT companies, where the aim is to seek a better understanding on faults that have been found during development and how this may affect the quality of the system. Our objective in this paper is to investigate the fault profiles of four business-critical commercial projects to explore if there are differences in the way faults appear in different systems. We have conducted an empirical study by collecting fault reports from several industrial projects, comparing findings from projects where components and reuse have been core strategies with more traditional development projects. Findings show that some specific fault types are generally dominant across reports from all projects, and that some fault types are rated as more severe than others.

1 Introduction

Producing high quality software is an important goal for most software developers. The notion of software quality is not trivial, different stakeholders will have different views on what software quality is. In the Business-Critical Software (BUCS) project [1] we are seeking to develop a set of methods to improve support for analysis, development, operation, and maintenance of business-critical systems. These are systems that we expect and hope will run correctly because of the possibly severe effects of failure, even if the consequences are mainly of an economic nature. In these systems, software quality is important, and the main target for developers will be to make systems that operate correctly all the time [1]. One important issue in developing these kinds of systems is to remove any possible causes for failure, which may lead to wrong operation of the system.

The study presented here investigated fault reports from two software projects using components and reuse strategies, and two projects using a more traditional development process. It compares the fault profiles of the reuse-intensive projects with the other two, in several dimensions; Fault type, fault severity and location of fault.

J. Münch and M. Vierimaa (Eds.): PROFES 2006, LNCS 4034, pp. 389 – 394, 2006.

2 Previous Studies on Software Faults and Fault Implications

Software quality is a notion that encompasses a great number of attributes. When speaking about business-critical systems, the critical quality attribute is often experienced as the dependability of the system. According to Littlewood et al. [2], dependability is a software quality attribute that encompasses several other attributes, the most important are reliability, availability, safety and security.

Faults in the software lessen the software's quality, and by reducing the number of faults introduced during development you can improve the quality of software. *Faults* are potential flaws in a software system, that later may be activated to produce an error. An *error* is the execution of a fault, leading to a failure. A *failure* results in erroneous external behaviour, system state or data state. Remedies known for errors and failures are to limit the consequences of a failure, in order to resume service, but studies have shown that this kind of late protection is more expensive than removing the faults before they are introduced into the code [3]. Faults are also known as *defects* or *bugs*, and a more extensive concept is *anomalies*, which is used in the IEEE 1044 standard [4]. Orthogonal Defect Classification – ODC – is a way of studying defects in software systems [5, 6, 7, 8]. ODC is a scheme to capture the semantics of each software fault quickly.

It has been debated if faults can be tied to reliability in a cause-effect relationship. Some papers like [6, 8] indicate that this is valid, while others like [9] are more critical. Still, reducing the number of faults will make the system less prone to failure, so by removing faults without adding new ones, there is a good case for the system reliability increasing. This is called "reliability-growth models", and is discussed by Hamlet in [9]. Avizienis et al. states [10] that fault prevention aim to provide the ability to deliver a service that can be trusted. Hence, preventing faults and reducing their numbers and severity in a system, the quality of the system can be improved in the area of dependability.

3 Research Design

Research questions. Initially we want to find which types of faults that are most frequent, and if there are some parts of the systems that have more faults than others:

RQ1: *Which types of faults are most typical for the different software parts?*
When we know which types of faults dominate and where these faults appear in the systems, we can choose to concentrate on the most serious ones in order to identify the most important issues to target in improvement work:

RQ2: *Are certain types of faults considered to be more severe than others by the developers?*

Research method. This study is based on data mining, where the data consists of fault reports we have received from four commercial projects. The investigation has mostly been a bottom-up process, because of the initial uncertainty about the available data from potential participants. After establishing a dialogue with the projects, and acquiring the fault reports, our initial research questions and goals were altered accordingly.