Information and Software Technology 58 (2015) 272–288

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Investigating software testing and maintenance reports: Case study

Paweł Janczarek, Janusz Sosnowski *

Warsaw University of Technology, Faculty of Electronics and Information Technology, Institute of Computer Science, Nowowiejska 15/19, 00-665 Warsaw, Poland

ARTICLE INFO

Article history:
Received 13 December 2013
Received in revised form 27 June 2014
Accepted 30 June 2014
Available online 10 July 2014

Keywords:
Software testing
Software development monitoring
Software measurements
Reliability

ABSTRACT

Context: Although many papers have been published on software development and defect prediction techniques, problem reports in real projects quite often differ from those described in the literature. Hence, there is still a need for deeper exploration of case studies from industry.
Objective: The aim of this study is to present the impact of fine-grained problem reports on improving evaluation of testing and maintenance processes. It is targeted at projects involving several releases and complex schemes of problem handling. This is based on our experience gained while monitoring several commercial projects.
Method: Extracting certain features from detailed problem reports, we derive various measures and present analysis models which characterize and visualize the effectiveness of testing and problem resolution processes. The considered reports describe types of problems (e.g. defects), their locations in project versions and software modules, ways of their resolution, etc. The performed analysis is related to eleven projects developed in the same company. This study is an exploratory research with some explanatory features. Moreover, having identified some drawbacks, we present extensions of problem reports and their analysis which have been verified in another industrial case study project.
Results: Fine-grained (accurate) problem handling reports provide a wider scope of possible measures to assess the relevant development processes. This is helpful in controlling single projects (local perspective) as well as in managing these processes in the whole company (global perspective).
Conclusion: Detailed problem handling reports extend the space and quality of statistical analysis, they provide significant enhancement in evaluation and refinement of software development processes as well as in reliability prediction.

1. Introduction

Typical software lifecycle comprises such processes as: specification of requirements, architectural design, coding, testing, deployment and maintenance. Recently, in many projects these processes are performed in an evolutionary and iterative way [36,40] covering subsequently developed versions, updates and upgrades, etc. Managing these processes, we have to take into account: product (software) quality requirements, delivery deadlines, and user satisfaction. For this purpose appropriate measurements and monitoring schemes are needed [12,15,45,47]. They have been described in many papers on various software issues, e.g. verification and validation [36], testing [9], fault prediction [27,33], software effort estimation [1,2,39], software process improvement factors [8,28], failure proneness aspects [19], reliability models [32], maintainability [17,46].

Measures are required to control and optimize software development and maintenance. Much attention has been paid to software defect prediction, complexity and control metrics [1,13,33,40]. Static prediction techniques correlate defect density with software basic attributes (e.g. size, complexity, object metrics, test coverage). Component based reliability predictions [34] evaluate component reliability in function of the usage profile. Dynamic predictions correlate reliability with distribution of detected defects in the initial state of testing (software reliability growth models – SRGMs [3,29,32]). Control metrics relate to software processes, e.g. functional and structural test coverage, time required to repair defects, software development progress [40]. The significance of such measurements is well known and appropriate standards have been also introduced [2,8,40], however putting this in practice creates many problems. Most papers on software measurements and statistics are of the academic character [16]. Hence, an important issue is to collect and exchange the gained experience from real commercial (industrial) projects – case studies. This paper addresses this issue within the process oriented research and it shows the richness of the use of problem reports (repository)

* Corresponding author. Tel.: +48 22 234 7915.
E-mail addresses: pawel.janczarek@gmail.com (P. Janczarek), jss@ii.pw.edu.pl (J. Sosnowski).

http://dx.doi.org/10.1016/j.infsof.2014.06.015
0950-5849/

in managing software development process. In particular, we provide an in-depth study of handling detected problems.

Usually, case studies are targeted at analysing specific problems, they can be a product (software quality, complexity) or process oriented. Process oriented studies are targeted at studying the effectiveness of software development processes or their phases. Our research belongs to this class, however we concentrate on problem handling within testing, code improvement and deployment (including release phases) followed by initial maintenance. The classical notion of software defects (bugs) is extended to problems, which can be resolved (correction of software defects, code refinement), rejected (e.g. withdrawn due to some misunderstanding or assumption changes), qualified as duplicated (redundant) or processed in different ways. Another important issue is the input data, in our case we mainly deal with the so called problem repository which can be supplemented with test progress repository.

The process oriented case studies published in the literature (compare Section 2) are based on coarse-grained data reports (e.g. aggregated information on detected faults, modifications of requirements, etc.) and they relate to different designs and implementation. They present software measurements and conclusions addressed to different phases of software development and maintenance. We describe our experience on evaluation the effectiveness of processes related to testing, code revisions or refinements and maintenance. The considered projects relate to telecommunication applications and have been developed in a local company using a similar incremental technology. Gained experience is based on monitoring fine-grained problem handling reports for 11 commercial projects (case studies). These reports comprise many attributes characterizing occurrence of problems, descriptions of their manifestations, severity levels, advances in testing or code repairing, performed solutions and related decisions. Included information is more detailed than in other case studies from the literature. An important contribution is introducing problem handling graphs, which facilitate analysis and visualizations. Moreover, we present still another case study with improved problem reports. The presented analysis concentrates on problem tracking and it is supplemented with a discussion on reliability prediction models (SRGMs). Moreover, it can be integrated with other statistical program measures.

As opposed to other approaches usually considered in the literature, development, testing and debugging processes of newer versions are interleaved with the maintenance of the older ones. A significant part of the generated code can be based on reuse-oriented software engineering and software components provided by various vendors (integration and configuration problems). We have introduced two problem analysis perspectives: global (overall view of all handled projects) and local (restricted to a single project). An important conclusion, we draw from the work, is that fine-grained reports (correlated with project development features) are helpful in more accurate process evaluation and identification of possible imperfections and refinements. This approach is useful for project (local observation perspective) and company (global perspective) managements, for this purpose some measures have been introduced. They also can interest system testers or developers and create some feedback in discussions with the management staff.

The remainder of the paper is organized as follows. In the next section, we provide related work. Section 3 presents the context of this study combined with the research methodology. Section 4 describes the considered software lifecycle model and measurement challenges. Section 5 presents a data repository comprising monitoring reports (measurements), which characterize encountered problems and the relevant handling processes. Basing on the collected data from many projects we have derived indicators characterizing software development and maintenance quality (described in Section 6). Section 7 provides a case study of a project with an improved monitoring scheme. Section 8 discusses results, Section 9 summarizes the paper and outlines plans for the future work.

2. Related work

In this section we give a survey on case studies described in the literature, which are targeted at software process assessment and are associated with our research. This is complemented with some remarks on software reliability modelling. We also show differences (in particular coarse-grained analysis) and some gaps of the considered approaches in relevance to our experience with the analysed projects which triggered our research.

In the literature, we can find reports on various case studies targeted at different goals, e.g. requirements engineering, development processes, maintenance or test effectiveness within different contexts [44]. Typically, case studies involve professional programmers and student teams, here we can also include users of the systems. Projects based on student teams provide good controllability to check different design methodologies, verify effectiveness of various ideas (e.g. test methods), however these projects may not be consistent with problems encountered in software development companies. In commercial projects we may face the problem of confidential data, correctness and completeness of provided data or techniques of their collection, so this creates some validity threat. Moreover, interaction with developers, testers and users usually is limited. Hence, we distinguish exploratory (observational) and controlled case studies [37]. The first one is a passive approach with no intervention, in the second case we have the possibility of introducing some changes (resulting from some technical actions) in the analysed processes and verify their effects. Our research can be classified as an exploratory case study with some limited technical interactions related to data collection and reporting. Having analysed many process oriented case studies related to industrial projects, we have found that they are targeted at different goals, they base on different data, relate to various development and reporting models, use different measures and development context, etc.

In [25,26] the authors trace the progress of handling requirements and check its conformance with lean manufacturing concept. Moreover, this was so called controlled experiment developed by two teams and some conclusions have been derived showing possible improvements in organization and developers qualification, etc. Paper [21] concentrates only on review processes and covers such issues as review speed, efficiency (the number of detected defects at peer review per man/hour). It describes an experience of applying software process control technique based on statistical measurements and analysis. This paper confirms the need of using product and process measurements. An interesting experience related to the initial development phase (requirement specifications) is presented in [6]. This is an exploratory case study with detailed explanation of how improvements in requirements engineering practice impact the project quality and management. This study used a web based questionnaire covering opinions of company staff involved in development of a large software project.

In [7] the authors concentrate on using the collected fault (defect) statistics to assess a test process and indicate the improvement potential in software development organization. Two projects are analysed taking into account the following phases: functional testing, system test (integration), tests involving external environment and field tests (the first and the second half of the year). The main analysis goal is evaluating the cost related to fault slippage from earlier phases. This is also coarse-grained analysis with no fault classification, no time analysis.

A detailed case study on maintenance process effectiveness is presented in [31], it is related to telecommunication applications.