IJER © Serials Publications
13(5), 2016: 2277-2283
ISSN: 0972-9380

FAULT REPORTING PROCESS OF BUSINESS INFORMATION SYSTEMS

Mohmmed Aref Abdul Rasheed*

Abstract: *Now we are much reliant on software or information systems. The use of software has already been increased tremendously in business organizations and further continuing in this direction. The financial systems relies on increasingly larger and more complex computer and software systems. To sustain with performance and reliability of systems we rely on a plethora of methods, techniques and processes. This paper is based on empirical study performed in few companies that develop business software. The aims is to study the use of fault reporting approaches. Few specific software methods adopted from safety-critical software engineering practices, while others taken from general software engineering. This study brought several important key issues from industry and information system development companies. With consideration of these issues during the development process can result significant reduction of faults.*

Keywords: *Business-Critical Software, Information System, Software system, Fault,*

INTRODUCTION

In our society rapid technological development has lead to software systems being introduced into an increasing number of different business domains. In many areas we are much dependent on software systems, and their potential weaknesses could have grave consequences. In this concerned, software can be categories into: *safety-critical* software (e.g. controlling traffic signals), *business-critical* software (e.g. for banking) and *non-critical* software (e.g. for word processing). For safety-critical applications, the result of a failure could easily be a physical accident or an action leading to physical harm for one or more human beings. In the case of business-critical systems, the consequences of failures are not that grave, in the sense that accidents do not mean real physical damage, but that the negative implications may be of a more financial or trust-threatening nature. Ian Sommerville states that business-criticality signifies the ability of core computer and other support systems of a business to have sufficient QoS to preserve the stability of the business.

* Department of MIS, College of Commerce and Business Administration, Dhofar University, Sultanate of Oman, *E-mail: mohammed_aref@du.edu.om*

Since business-critical applications are not really an established phrase or topic within the software engineering community, it is therefore difficult to point out specific methods and techniques that are being used when business-critical systems are being developed. Instead, the most common methods of software engineering presents with additional comments on how they may be best utilized to aid the development of business-critical systems. On the other hand, software or information systems have social and economic value, by making people more productive, improving their quality of life, and making them able to perform work and actions. Software engineering technologies and practices help developers by improving productivity and quality. The ongoing goal is to improve technologies and practices, seeking to improve the productivity of practitioners and the quality of applications for the users.

Ian Sommerville states that business-criticality signifies the ability of core computer and other support systems of a business to preserve the stability of the business. Thus business-critical systems are those whose failure could threaten the stability of a business. In software systems the phase change from fault to failure can be expressed as: faults are potential flaws or incorrect versus explicitly stated requirements in a software system, that later may be activated to produce an error. An error is the execution of a passive fault, and my lead to a failure. The most common result of failed software development projects are projects that overrun their schedule and budget, but more serious consequences may also be the result of poorly executed software projects. Software defects can cause property damage. Poor software security allows hackers to steal identities, and defective control systems can damage the physical systems the software is controlling. The result is lost time, money, and damaged reputation.

Much efforts are required to reduce the probability of software failures, but this cannot be removed. The need for post-release fault-fixing. Faults in the software are detrimental to the software's quality, to a greater or lesser extent dependent on the nature and severity of the fault. Therefore, one way to improve the quality of developed software is to reduce the number of faults introduced into the system during initial development.

Thus business-critical systems are those whose failure could threaten the stability of a business. The identification of fault in business information system can improve software technologies, including processes used for developing business-critical software. In order to do this, empirical studies of projects have been performed in this paper. The main objective of this exercise is to:

- Obtain a better understanding of the problems encountered industry during development, operation and maintenance of business-critical software.
- Study the effects of introducing safety-critical methods and techniques into the development of business-critical software, to reduce the number of system failures (increased reliability).

- Provide adapted and annotated methods and processes for development of business-critical software.

CHALLENGES

The general challenges in software engineering are: poor requirements, rising complexity, ongoing change and failure to pinpoint causes. In this paper the concentration is on issues dealing with reduction of product risk, improving requirement specifications, coping with complexity, and helping with pinpointing causes for failures. This adds up to a main challenge for business-critical software development without increasing costs too much compared to the received benefit from these methods. As far as empirical software engineering research is concerned, an important challenge is that of following up research and technology change proposals with continued observation and measurement in the field when practitioners put theory into practice. Much of the research being performed concerns the software development process up to a point, but does not follow the eventual implementation of these results further.

LITERATURE REVIEW

Catal 2009 reviewed of software fault prediction studies with a specific focus on metrics, methods, and datasets. The usage percentage of public datasets increased significantly and the usage percentage of machine learning algorithms increased slightly since 2005. In addition, method-level metrics are still the most dominant metrics in fault prediction research area and machine learning algorithms are still the most popular methods for fault prediction. Researchers working on software fault prediction area should continue to do so. An investigation took place by Jiang & Bojan, 2007 that whether metrics available early in the development lifecycle can be used to identify fault-prone software modules. predictive models was build using the metrics that characterize textual requirements and compared the performance of requirements-based models against the performance of code-based models and models that combine requirement and code metrics. Using change management data from a very large, long-lived software system, Todd, Alan, 2000 explore code based fault finding method in the software.

METHODOLOGY

The principal goal of this paper is through empirical studies to understand and improve the software technologies and processes used for developing business-critical software. This entails both quantitative and qualitative studies, and in some cases a combination. Several been considered when performing such studies, and particularly deciding on the:

- Metrics used in the investigations.
- Process of retrieving information (observation, surveys).

To establish a basis for the most commonly used methods and most common problems encountered in companies that develop business-critical software, several semi structured interviews were carried out with representatives from cooperating companies. These companies were chosen both for relevance to 'business-critical' issues, but also in some ways out of convenience of location and availability.

DATA ACQUISITION

The goal of this research is to explore quality issues, with focus on fault reporting and management, as well as the use of safety analysis techniques for this type of software development. In this study overall focused on the following questions to be answered:

- What is the role of fault reporting in existing industrial software development?
- How can we improve on existing fault reporting processes?
- What are the most common and severe fault types, and how can we reduce them in number and severity?
- How can we use safety analysis techniques together with fault report analysis to improve the development process?

To obtain answers to these research questions, a common metrics been used, including attributes like structural fault location, functional fault location and fault repair effort. As received the actual data from industry, the scope reduced to somewhat. This was due to lack of complete information in the data material, and great variation between organizations on what data they stored. The main metrics was identified for fault report studies were:

- The number of detected faults is an indirect metric, attained simply by counting the number of faults of a certain type or for a certain system part etc.
- The metrics that are used directly from the data in the fault reports are the reported type, severity, priority, and release version of the fault.

Structured interviews: Preliminary interviews about business critical software and state-of-practice in Indian IT industry.

INTERVIEWS WITH COMPANY REPRESENTATIVES

Before the interviews, a list of topics were analyzed and decided, on which the later interviews/talks with the company representatives were based.

a) How the use of well known software development methods may improve business-critical system development?

b) Do company know much about safety-critical methods at all? If so, how do they view the possibility of using such safety methods to improve business-critical system development?

c) What are the most common reliability/safety-related problems in business-critical system development? – Identify the most important factors leading to failures or accidents.

d) What are the most important hindrances for achieving high quality products when developing business-critical software?

e) How does industry handle these problems now?

f) What are the most important problems encountered during the operation of business-critical software?

g) How can we remove or reduce these problems by changing the way business-critical systems are developed, operated and maintained?

The main validity concerns in this study would be the relatively low number of respondents and that the interviews were carried out.

ANALYSIS

The selected interviewees was based on actively involvement and who also had hands-on experience with fault management in the organization. The interviews were conducted as open-ended interviews, with the same questions asked to each interviewee. Each question in the interview guide was related to one or more local research questions, and the different responses for each question were compared to extract answers related to the research questions. In line with using the constant comparison method, coded with each answer into groups. The codes were post-formed, i.e. constructed as a part of the coding process, since the interviews were open-ended. Additionally, we received feedback about the topic at hand through discussions. The research questions have resulted in the following contributions:

- Describing how to utilize safety criticality techniques to improve the development process for business-critical software.
- Identification of typical shortcomings in fault reporting.
- Improved model of fault origins and types for business-critical software.

RESULTS

In order to learn more about the way business-critical software projects are being executed, companies been sought out the and conducted short interviews with representatives from these companies. Four interviews were conducted in four different companies. The companies were picked partly for being representative IT industry, partly because of our suspicions of their relevance to the business-critical topic, and also partly because of convenience with respect to geographic location and general availability. The companies were represented by persons in different positions in the company structure, from directors to project managers and developers. Each interview was noted appropriately. The main results to be extracted from the interview sessions were the following:

a) The industry defines the term 'business-critical' as something that is related to their economy, their reputation and their position in the market.
b) Some variant of regional unified process, is common among companies who actually employ some specified process.
c) Business-critical software development is a very common activity among software development companies.
d) A typical problem in development of business-critical software is communication, both within the company and towards the customer.
e) The companies generally do not consider the technical risk aspects of a project in detail, perhaps mainly due to a lack of an instrument for this.

DISCUSSION AND CONCLUSION

An overview of studies performed concerning fault reports been presented, and shown the type of information that exists and is lacking from such reports. What could be learnt from the studies of the fault reports is that the data is in some cases under-reported, and in most cases under analyzed.

By including some of the information that the organization already has, more focused analyses could be made possible. For instance, specific information about fault location and fault correction effort is generally not reported even though this information is easy to register. One possibility is to introduce a standard for fault reporting, where the most important and useful fault information is mandatory. A reasonable approach to improving fault reporting and using fault reports as a support for process improvement is to start by being pragmatic. At first, can be used the readily available data that has already been collected, and in time change the amount and type of data that is collected through development and testing to tune the process.

Also, learnt that the effort spent by external researchers to produce useful results based on the available data is quite small compared to the collective effort spent by developers recording the data. There many issues concerned to privacy, security and to undisclosed the faults. Organization administrative systems doesn't permit to disclose their inside to outside. That is the reason name of organization did not mentioned in this paper.

Finally, there are two main points required to convey as a result of the studies:

- It is important to be able to approach the subject of fault data analysis with a bottom-up approach, at least in early phases of such research and analysis initiatives. The data is readily available, the work that has to be performed is designing and performing a study of the data.
- Much of the recorded fault data is of poor quality. This is most likely because of the lack of interest in use of the data.

- Lack of willingness to provide and availability of the data.

References

Basili, V.R., Calidiera, G., Rombach, H.D.: Goal Question Metric Paradigm.In: Marciniak, J.J. (ed.): Encyclopaedia of Software Engineering, pp. 528-532, Wiley, New York, 1994.

Catal, Cagatay, and Banu Diri. "A systematic review of software fault prediction studies." Expert systems with applications 36.4 (2009): 7346-7354.

Deepinder Kaur, Arashdeep Kaur, Sunil Gulati, Mehak Aggarwal, "A clustering algorithm for software fault prediction", Computer and Communication Technology (ICCCT) 2010 International Conference on, pp. 603-607, 2010.

Dubelaar, Chris, Amrik Sohal, and Vedrana Savic. "Benefits, impediments and critical success factors in B2C E-business adoption." Technovation25.11 (2005): 1251-1262.

Graves, Todd L., *et al.* "Predicting fault incidence using software change history." IEEE Transactions on software engineering 26.7 (2000): 653-661.

IEEE Standard Classification for Software Anomalies," in IEEE Std 1044-2009 (Revision of IEEE Std 1044-1993) , vol., no., pp.1-23, Jan. 7 2010, doi: 10.1109/IEEESTD. 2010. 5399061.

Jiang, Yue, Bojan Cukic, and Tim Menzies. "Fault prediction using early lifecycle data." The 18th IEEE International Symposium on Software Reliability (ISSRE'07). IEEE, 2007.

Sommerville, Ian. "Integrated requirements engineering: A tutorial." IEEE software 22.1 (2005): 16-23. *http://www.cs.uleth.ca/~benkoczi/3720/papers/sommerville_tutorial.pdf*

Todd L. Graves, Alan F. Karr, J.S. Marron, and Harvey Siy, "Predicting Fault Incidence Using Software Change History" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 26(7), 653-661, JULY 2000.

Zelkowitz, M.V., Wallace, D.R.: Experimental models for validating technology. IEEE Computer, (31)5, pp. 23-31, May 1998.