IJEMR - April 2012-Vol 2 Issue 4 - Online - ISSN 2249–2585 - Print - ISSN 2249-8672

Business Critical Systems and Quality Improvement

* Mr. P.Jitendra Srinivas Kumar ** Dr. Nageswara Rao K,

* Associate Professor, ANR College, Gudivada
** Professor & Head [Comp. Science & Engg.],
P.V.P. Siddhartha Institute of Technology, Vijayawada

Abstract

Today's society is highly dependent on software systems. The number of areas where functioning software is at the core of operation is growing steadily. Both financial systems and e-business systems relies on increasingly larger and more complex computer and software systems. To increase e.g. the reliability and performance of such systems we rely on a plethora of methods, techniques and processes specifically aimed at improving the development, operation and maintenance of such software. The Business Critical Systems generally seek to develop and evaluate methods to improve the support for development, operation and maintenance of Business Critical System and systems. Improving software processes relies on the ability to analyze previous projects and derive concrete improvement proposals.

***This paper** is a part of the research work on the BCS basic research and development project (Business Critical System). The BCS project was funded by two small software companies, based at Hyderabad, Andhra Pradesh, India, as a basic R&D project in IT in the year 2004.*

***Key Words:** BCS, Business safe systems, software faults, Fault reporting*

Introduction

This paper starts with some important research issues like:

How do software faults affect the reliability and safety of business-critical systems?
What are the common fault types in business-critical systems?
How can we use system safety methods in business-critical application development?

The main goal is to attain a better understanding of business-critical systems, as well as to adapt and improve relevant methods and processes. Through data mining of historical software project data and other studies of relevant projects, we will gather information to be evaluated with the goal of improving business-critical systems development. The main goal was to study the effects of revised development methods for Business Critical System, in order to improve important quality aspects of these systems.

The number of areas where functioning software is at the core of operation is growing steadily. Both financial systems and e-business systems relies on increasingly larger and more complex computer and software systems. To increase e.g. the reliability and performance of such systems we rely on a plethora of methods, techniques and processes specifically aimed at improving the development, operation and maintenance of such software.

Business-Critical Systems is seeking to develop and evaluate methods to improve the support for development, operation and maintenance of Business Critical System and systems. Improving software processes relies on the ability to analyze previous projects and derive concrete improvement proposals.

The work specifically aims to assess the use of fault reporting approaches, and describe how improvement in this area can benefit process and product quality. Some specific software methods will be adopted from safety-critical software engineering practices, while others will be taken from general software engineering.

Actually the main research questions in this work are:

What is the role of fault reporting in existing industrial software development?
How can we improve 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 failure report analysis to improve the development process?

But in this paper the background and research context for the thesis is presented. The paper also introduces the research design, the research questions and the contributions towards the main thesis.

Criticality definitions

Business Critical System systems have a lot in common with safety-critical systems, but there are also quite telling differences. A simplistic way to distinguish them is to put them into classes according to the effects that software anomalies (faults or hazards) may have on the environment. The classes are safety-critical, mission-critical, performance-critical, business-critical, and non-critical.

Safety-critical: A safety-critical system could be a computer, electronic or electromechanical system where a hazardous event may cause injury or even death to human beings, or physical harm to other objects that interact with the system. Examples are aircraft control systems and nuclear power-station control systems, where an accident in most cases will lead to economic losses as well as injury and other physical damage. Common tools to design safety-critical systems are redundancy and formal methods, and a spectrum of specialized technologies exist for safety-critical systems (Hazop, Fault-tree analysis etc). The IEC 61508 standard is intended to be a basic functional safety standard applicable to all kinds of industry, and is also used to define the safety standards of some safety-critical systems [IEC 61508].

Mission-critical: The term mission-critical system reflects military usage and is used to describe activities, processing etc., that are deemed vital to the organization's business success and, possibly, its very existence. Some major software systems are described as mission-critical if such a system, product or service experiences a failure or is otherwise unavailable to the organization, it will have a significant negative impact upon the organization. Such systems typically include support for accounts/billing, customer balances, computer-controlled machinery and production lines, just-in-time ordering, and delivery scheduling. Examples of related technologies are Enterprise Resource Planning tools, such as SAP [SAP].

Performance-critical: The SEI defines performance-criticality as the ability of software-intensive systems to perform successfully under adverse circumstances, e.g., under heavy or unexpected load or in the presence of subsystem failures. One trivial example of this is the performance of the SMS telecom services during New Years Eve. Some services like this can have critical functions, and yet, the behavior of systems under such circumstances is often less than acceptable [SEI].

Business-critical: The difference between a business-critical and a regular commercial software system is really defined by the business. There is no established general definition telling us which software applications are critical to an operation. In a retail business, a Customer Relationship Management (CRM) system may be the most important. On the other hand, it may be the manufacturing or supplier management software that is the most important. We need to consider the impact of relevant services from software on the business operations, and determine how much value each brings to the business and the impacts of such software parts being unavailable. The impact can be lost revenue, corrupted data or lost user time, as well as indirect and more elusive losses in customer reputation, goodwill, slipped deadlines, and increased levels of stress among employees and customers.

Non-critical: Although important enough, some types of software will simply not be classified as critical. Word processors, spreadsheets and graphical design software are examples of such software. Of course it is expected that such tools are reasonably fault free and stable, but should they fail, the damage will usually be limited, typically a person-day of effort in the worst case scenario.

Research design:

As stated in the BCS project proposal, "The principal goal is through empirical studies to understand and improve the software technologies and processes used for developing Business Critical System" [BCS02]. This entails both quantitative and qualitative studies, and in some cases a combination. Several aspects have to be considered when performing such studies, and particularly:

- Deciding on the metrics used in the investigations.
- Deciding on the process of retrieving information (data mining, observation, surveys)

Members of the BCS project have conducted interviews, experiments, data analysis, surveys, and case studies. The methods employed in this part of the BCS project are structured interviews, historical data mining and analysis, and case studies.

Relevance to the Organizations

The technological development in our society has lead to software systems being introduced into an increasing number of different business domains. In many of these areas we become more or less dependent on these systems, and their potential weaknesses could have grave consequences. In this respect, we can coarsely divide software products into three categories: *safety-critical* software (e.g. controlling traffic signals), *Business Critical System* (e.g. for banking) and *non-critical* software (e.g. for word processing).

Evidently, the definition of business-critical versus the other two categories may be difficult to state precisely, and would in many cases depend on the particular viewpoint of the business and users. To clarify the distinction between business-critical and safety critical, we can consider what consequences operation *failure* (observable and erroneous behavior of the system compared to the requirements) will have in the two different cases. 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 [Sommerville04]. Thus business-critical systems are those whose failure could threaten the stability of a business.

The *overall goal* for the BCS project is to better understand and thus sensibly improve software technologies, including processes used for developing Business Critical System.

Research Base:

The very important term used in this paper is *business-safe*, which means that a system fulfils the criteria for business-safety in a business-critical system.

That a system is business-safe does not mean that the system is fault-free, i.e. cannot possibly fail. What this means is that the system will have a low probability of entering a state where it will cause serious losses. In this respect, the system characteristic is close to the term "safe". This term is, however, wider, since it is concerned with all activities that can cause damage to people, equipment, the environment or severe economic losses. Just as with general safety, business-safety is not a characteristic of the system alone – it is a characteristic of the system's interactions with its usage environment.
BCS are considering two groups of stakeholders and wants to help them both:

***The customers and their users.** They need a method that enables them to:

Understand the dangers that can occur when they start to use the system as part of their business.
Write or state requirements to the developers so that they can take care of the risks incurred when operating the system.

***The developers.** They need help to implement the system so that:

It can be made business-safe.
They can support their claims with analysis and documentation.
It is possible to change the systems in such a way that when the operating environment or profile changes, the systems are still business-safe.

BCS aims to help the developers to build a business-safe system without large increases in development costs or schedule. This is achieved by the following contributions from BCS:

BC1: A set of methods for analyzing business-safety concerns. These methods are adapted to the software development process in general and – for the first version – especially to the Rational Unified Process (RUP)

BC2: A systematic approach for analyzing, understanding, and protecting against business-safety related events

BC3: A method for testing that the customers' business-safety concerns are adequately taken care of in the implementation

Why should development organizations do something that costs extra, i.e. is this a smart business proposition? We definitively mean that the answer is "Yes", and for the following reasons:

The only solution most companies have to offer to customers with business safety concerns today is that the developers will be more careful and test more this is not a good enough solution.

By building a business-safe system, the developers will help the customer to achieve an efficient operation of their business and thus build an image of a company that has their customers' interest in focus. Applying new methods to increase the products' business-safety must thus be viewed as an investment.

The return on the investment will come as more business from large, important customers.

BCS will not invent entirely new methods. What we will do, is to take commonly used methods, especially from the area of systems safety such as Hazard Analysis and FMEA, and adapt them to more mainstream software development. This is done by extending the methods, making them: More practical to use in a software development environment and suitable to fit into the ways developers work in a software project environment – concerning both process and related software tools and methods.

Relationship of business-critical and other types of criticality:

Criticality category: Examples

Safety-critical Nuclear reactor control system.

Performance-critical Electronic toll collection in traffic, must process and transfer information quickly enough to keep up with traffic.

Mission-critical Software handling financial transactions between banks.

Functional and non-functional aspects of such applications are considered.

Business Critical System handling financial transactions between banks. As mission-critical, but wider consequences are also considered.

Non-critical Computer games, word processor application.

Techniques and methods used to develop safety-critical systems:

There are a number of methods and techniques that are commonly employed when making safety-critical systems. Some of them will be presented here and related to business-critical computing. According to [Leveson95] and [Rausand91], the most common ones are the following:

1) PHA (Preliminary Hazard analysis): Preliminary Hazard Analysis (PHA) is used in the early project life cycle stages to identify critical system functions and broad system hazards, so as to enable hazard elimination, reduction or control further on in the project. The identified hazards are assessed and prioritized, and safety design criteria and requirements are identified. A PHA is started early in the concept exploration phase so that safety considerations are included in tradeoff studies and design alternatives. This process is iterative, with the PHA being updated as more information about the design is obtained and as changes are being made. The results serve as a baseline for later analysis and are used in developing system safety requirements and in the preparation of performance and design specifications. Since PHA starts at the concept formation stage of a project, little detail is available, and the assessments of hazard and risk levels are therefore qualitative. A PHA should be performed by a small group with good knowledge about the system specifications.

2) HAZOP (Hazard and Operability Analysis): This is a method to identify possible safety-related or operational problems that can occur during the use and maintenance of a system. Both Preliminary Hazard Analysis and Hazard and Operability Analysis (HAZOP) are performed to identify hazards and potential problems that the stakeholders see at the conceptual stage, and that could be created by system usage. A HAZOP study is a systematic analysis of how deviations from the intended design specifications in a system can arise, and whether these deviations can result in hazards. Both analysis methods build on information that is available at an early stage of the project. This information can be used to reduce the severity or build safeguards against the effects of the identified hazards. HAZOP is a creative team method, using a set of guidewords to trigger creative thinking among the stakeholders and the cross-functional team in RUP. The guidewords are applied to all parts and aspects of the system concept plan and early design documents, to find and eliminate possible deviations from design intentions. An example of a guideword is MORE. This will mean an increase of some quantity in the system. For example, by using the “MORE” guideword on “a customer client application”, you would have “MORE customer client applications”, which could spark ideas like “How will the system react if the servers get swamped with customer client requests?” and “How will we deal with many different client application versions making requests to the servers?” A HAZOP study is conducted by a team consisting of four to eight persons with a detailed knowledge of the system to be analyzed. The main difference between HazOp and PHA is that PHA is a lighter method that needs less effort and available information than the HAZOP method. Since HAZOP is a more thorough and systematic analysis method, the results will be more specific. If there is enough information available for a HAZOP study, and the development team can spare the effort, a HAZOP study will most likely produce more precise and suitable results for a safety requirement specification.

3) FMEA (Failure Modes and Effects Analysis): The method of Failure Modes and Effects Analysis, alternatively the variant Failure Modes, Effects and Criticality Analysis (FMECA), is used to study the potential effects of fault occurrences in a system. Failure Modes and Effects Analysis is a method for analyzing potential reliability problems early in the development cycle. Here, it is easier to overcome such issues, thereby enhancing the reliability through design. FMEA is used to identify potential failure modes, determine their effect on the operation of the system, and identify actions to mitigate such failures. A crucial step is anticipating what might go wrong with a product. While anticipating every failure mode is not possible, the development team should formulate a extensive list of potential failure modes. Early and consistent use of FMEAs in the design process can help the engineer to design out failures and produce more reliable and safe products. FMEAs can also be used to capture historical information for use in future product improvement.

4) FTA (Fault Tree Analysis): A Fault Tree Analysis diagram is a logical diagram which illustrates the connection between an unwanted event and the causes of this event. The causes can include environment factors, human error, strange combinations of "innocent" events, normal events and outright component failures. The two main results are: 1) The fault tree diagram which shows the logical structure of failure effects. 2) The cut-sets, which show the sets of events which can cause the top event – system failure. If we can assign probability values or failure rates to each basic event, we can also get quantitative predictions for Mean Time to Failure (MTTF) and failure rate for the system.

5) ETA (Event-tree analysis): An event-tree is a graphical representation of a sequence of related events. Each branching point in the tree is a point in time where we can get one of two or more possible consequences. The event-tree can be described with or without branching probabilities. In economical analyses it is customary to assign a benefit or cost to each possible alternative – or branch. An event tree can help our understanding and documentation of one or more sequences of events in a system or part of a system.

6) CCA (Cause-Consequence Analysis): Cause-consequence analysis (CCA) is a two-part system safety analytical technique that combines **Fault Tree Analysis** and **Event Tree Analysis**. Fault Tree Analysis considers the "causes" and Event Tree Analysis considers the "consequences", and hence both deductive and inductive analysis is used. The purpose of CCA is to identify chains of events that can result in unwanted consequences. With the probabilities of the various events in a CCA diagram, the probabilities of the various consequences can be calculated, thus establishing the risk level of the system. A CCA starts with a *critical event* and determines the causes of the event (using top-down or backward search) and the consequences it might create (using forward search). The cause-consequence diagram can show both temporal dependencies and causal relationships among events. The notation builds on the FTA and ETA notations, and extends these with timing, condition and decision alternatives. The result is a diagram (along with elaborated documentation), showing both a logical structure of the cause of a critical event and a graphical representation of the effect the critical event can have on the system. CCA enables probability assessments of success/failure outcomes at staged increments of system examination. Also, the CCA method helps in creating a link between the FTA and ETA methods. CCA shows the sequence of events explicitly, which makes CCA diagrams especially useful in studying start-up, shutdown and other sequential control issues. Other advantages are that multiple outcomes are analyzed from each critical event, and different levels of success/failure are distinguishable, as CCA may be used for quantitative assessment. In addition to these techniques, we included the Safety Case method tool for use alongside the other safety criticality analysis methods.

7) Safety Case: The Safety Case method seeks to minimize safety risks *and* commercial risks by constructing a demonstrable safety case. Bishop and
Bloomfield [Adelard98, Bishop98] define a safety case as: "A *documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment*". The safety case method is a vehicle for managing safety claims, containing a reasoned argument that a system is or will be safe. It is manifested as a collection of data, metadata and logical arguments. The Safety Case documents will answer questions like "*How will we argue that this system can be trusted/ is safe?*" The Safety Case shows how safety requirements are decomposed and addressed, and will provide an appropriate answer to the above questions. The layered structure of the Safety Case allows lifetime evolution and helps to establish the safety requirements at different detail levels.

Summing up the Paper:

The goal of the BCS project is not to help developers to finish their development on schedule and budget. We are not particularly interested in the delivered functionality or how to identify or avoid process and project risk. This is not because we think that these properties are not important – it is just that we have defined them out of the BCS project.

The goal of the BCS project is to help developers, users and other stakeholders to develop software whose later use is less prone to critical problems, i.e. has sufficient reliability and safety. In a business environment this means that the system seldom behaves in such a way that it causes the customer or his users to lose money, important information, or both. We will use the term *business-critical* for this characteristic.

References:

1) Børretzen, J.A., Conradi, R.: Results and Experiences from an Empirical Study of Fault Reports in Industrial Projects. Proceedings of the 7th International Conference on Product Focused Software Process Improvement (PROFES'2006), pp. 389-394,
Amsterdam, 12-14 June 2006

2) Børretzen, J.A., Dyre-Hansen, J.: Investigating the Software Fault Profile of Industrial Projects to Determine Process Improvement Areas: An Empirical Study. Proceedings of the European Systems & Software Process Improvement and Innovation Conference 2007 (EuroSPI'07), pp. 212-223, Potsdam, Germany, 26-28 September 2007

3) Conradi, R., Marjara, A.S., Skåtevik, B.: An Empirical Study of Inspection and Testing Data at Ericsson. Proceedings of the International Conference on Product Focused Software Process Improvement (PROFES'99), p. 263-284, Oulu, Finland, 22- 24 June 1999

4) R. Chillarege; I.S. Bhandari; J.K. Chaar; M.J. Halliday; D.S. Moebus; B.K. Ray; M.-Y. Wong, "Orthogonal defect classification-a concept for in-process measurements", IEEE Transactions on Software Engineering, Volume 18, Issue 11, Nov. 1992 Page(s):943 - 956

5) Grady, R.: Practical Software Metrics for Project Management and Process Improvement, Prentice Hall, 1992

6) The Gentoo linux project, available from: http://www.gentoo.org/ IEEE 1044] IEEE Standard Classification for Software Anomalies, IEEE Std 1044- 1993, December 2, 1993

7) Mohagheghi, P., Conradi, R.: Exploring Industrial Data Repositories: Where Software Development Approaches Meet. In Proceedings of the 8th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'04), pp. 61-77, Oslo, Norway, 15 June 2004

8) P. Mohagheghi, P., Conradi, P., Børretzen, J.A.: Revisiting the Problem of Using Problem Reports for Quality Assessment. Proceedings of the 4th Workshop on Software Quality, held at ICSE'06, pp. 45-50, Shanghai, 21 May 2006

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