A LITERATURE REVIEW OF CATASTROPHIC SOFTWARE FAILURES WITH EMPHASIS ON GROWING POSITION OF SOFTWARE ENGINEERING RESEARCH

Javaid Iqbal

Department of Computer Science, University of Kashmir,
Srinagar, J&K India
iamjavaid@gmail.com

Abstract

Software pervades all walks of life. Software exist in almost every gadget or machine used. Increasing dependence indicates the amount of confidence that has been reposed in these automations. There are two types of systems which are of particular interest vis-à-vis their reliability. One is safety-critical systems and the other is mission-critical systems. A software failure in safety-critical systems may lead to damage or even loss of life. An example of safety-critical software is software controlling operations of an airplane carrying passengers. On the other hand, a software failure on a mission-critical system may lead to huge financial loss. An example of mission-critical system is software installed on a NASA space mission.

This paper presents a review of the various catastrophic software failures that have resulted in loss of life, financial loss or other inconveniences to public. The literature review clearly indicates need and importance of growing position of software engineering research.

***Keywords*:** Software Reliability; software failure; catastrophic failures; software testing; software engineering.

1. INTRODUCTION

Software testing is the process of executing software to determine whether it matches its specification and executes in its target environment (Whittaker, 2000)**.**The acknowledged limitations of testing are vindicated by following famous statement from Edsger Dijkstra's Turing Lecture (Dijkstra, 1972)"Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence". This means that all software at the end of the day may be released with some faults. This makes some questions crop up on our minds which include questions like Why do we test software, if it is released with faults? How much testing needs to be done before its release? When do we decide to stop testing? When do we release the software? Will the lab-tested software faithfully run in the field environment? Have we covered the randomness and uncertainty of operational environment while testing the software under controlled settings, so on and so forth? Thus it necessitates the importance of attention that needs to be lent to the software development process. In (Jeske, et al., 2005) the authors present a detailed account of some of the formidable obstacles that beset the practice of the theory.

Software reliability stands out as a very important concern for the wide-spread acceptance of such systems. Software Reliability is defined as the probability that software will provide failure-free operation in a fixed environment for a fixed interval of time (Musa, et al., 1987a). The reliability of software applications is measured by use of mathematical formulations called as Software Reliability Growth Models (SRGMs) which basically describe the error-detection and correction processes. The SRGMs find their applications in many practical areas which include safety-critical systems like Shuttle's on-board systems software (Schneidewind, et al., 1992) and weapon systems (Carnes, 1997). According to a paper (Huang, et al., 2010), apart from ANSI/AIAA (American Institute of Aeronautics and Astronautics) use and applications of SRGMs have also been adopted or recommended by a number of leading research institutions and companies which include

AT&T, Bell Canada, North Telecom, Hewlett-Packard (HP), AFOTEC (The Airforce Operational and Evaluation Centre) and JPL (Ehrlich, et al., 1990)(Jones, 1991).

2. Literature Review of some Software failures and Catastrophes

Software failures have been catastrophic. This section shall give an insight into the potential financial loss or even loss of life that may result due to software failures. This is especially true for mission-critical and safety-critical application areas. This means that there is an imperative need to have sound techniques and methodologies for quantifying, measuring or improving software reliability of a system. This is a big motivation for hot-conduct of research in the field of software engineering. Some examples taken from different sources referenced are presented here for record and motivation for need for better software engineering methods.

I. As is documented in (Pham, et al., 2000) it was a software problem that may have prevented the Patriot missile system from tracking the Iraqi Scud missile, resulting in killing of 28 U.S soldiers during the 1991 Gulf War.

II. Mexican Airlines Boeing 727 airliner, on March 31, 1986, crashed into mountains because of a fault in software which could not correctly process position of the mountain(Pham, 2000)(Pham, 2003).

III. Massive Therac-25 radiation therapy machines from March through June of 1986, due to a flawed computer program, over-dozed cancer patients in Marietta (Georgia), Boston (Massachusetts) and Tyler (Texas). This information can be found in (Pham, 2000)(Pham, 2003).

IV. As is documented in (Pham, et al., 2000) the concerns of millennium bug rose high at the verge of millennium. It was feared that the millennium bug i.e. Y2K bug may result in failure of computer systems. The industry and the U.S government agencies remained very concerned about the removal of errors in time.

V. As pointed out in (Pham, 2000), a power outage on September 17,1991 at AT&T switching facility in new York led to disruption in service to 10 million telephone customers for 9 hours. This was due to deletion of 3 bits of code in a software upgrade and failure to test software on the public network before its installation. This information can be found in (Pham, 2003) also.

VI. In London, on October 26, 1992 a computer-aided dispatch system of the Ambulance service broke down just after its installation. This provides service to more than 5000 requests per day to transport patients in emergency situations. On this fateful day many critical patients had serious consequences (Pham, 2003).

VII. As documented in (Bai, et al., 2005) increased use of software in a safety-critical system e.g. an on-board aviation system makes the natures of a failure severe which is vindicated by the fact that the accident of Sweden JAS39 flight plane which took place in 1989 was due to the electrical operator systems failure. Another accident caused by a design error in software which was insufficiently tested led to the explosion of an unmanned Ariane 5 rocket launched on June 4, 1996 by the European Space agency on its maiden flight in less than 40 seconds after its liftoff. This was due to a software failure in the inertial reference system of the flight control system where a few lines of Ada code had three unprotected variables. One of these variables was for rocket launchers horizontal velocity. Overflow occurred in the variable as the horizontal velocity was much greater than the trajectory. For this specific information, kindly see (Pham, 2003) and (Bai, et al., 2005) . This means that we must be certain about the reliability of software used especially in safety-critical and mission critical applications. Using SRGMs for study of reliability levels achieved after testing can help in practice.

VIII. In (Almering, et al., 2007) the authors who are associated with NXP semiconductors and NXP software demonstrate how they put software reliability growth models into practice while they worked on three embedded software development projects (TV2003, TV2004 and TV2005) for developing software for high-end TV sets containing several million lines of code. They have performed failure analysis and reliability prediction on the basis of data sets that stemmed from these three projects under NHPP framework. Their analysis supported management's decisions. This is yet another example of employing SRGMs in practice for reliability analysis of software.

IX. In (Schneidewind, 2008) an introduction to practices of Quantitative Risk Analysis (QRA) at NASA projects has been pointed out. According to this introduction, NASA has been using QRA in its Space Shuttle and other projects since 1986, the year in which Space shuttle Challenger suffered accident (Safie, F.M., undated). Marshall Space Flight Center (MSFC) has been extensively practicing QRA for risk management of flight hardware studies and their reliability predictions. In risk management, life limits on Space Shuttle Main Engine (SSME) program and other MSFC elements using QRA are performed. Trade studies using QRA for evaluation of procedures, inspections and other costs, for their possible elimination are done e.g. whether to eliminate a pre-proof test X-ray inspection on Space Shuttle External Tank welds (Safie, 1994). MSFC used probabilistic structural models to perform reliability prediction on new hardware like X-33 and X34 engines (Safie, 1992). Schneidewind models have been used increasingly since Challengers fate by IBM Federal Systems personnel,the flight software contractor(Schneidewind, 2008); other interesting accounts of use and importance of SRGMs can be found in research papers due to Schneidewind.

X. In (Schneidewind, et al., 1992) the authors present a case study reports (of which, a nutshell is given here, from the same paper) on IBM's approach to the space shuttle's on-board software - IBM Shuttle's Primary Avionics Software Subsystem (PASS). They demonstrate that on the basis of experience gained from a real project team at IBM Federal Services Company in Houston, how reliability models can predict reliability and help develop test strategies. The IBM team evaluated many reliability models depending on the compatibility of their assumptions with PASS. Using the actual failure history of PASS, validation of models was tried for use on this project. Based on the likeness of assumptions of Schneidewind model with those of PASS, Schneidewind model was selected to predict the reliability of shuttles software for the National Aeronautics and Space Administration (NASA). IBM Shuttles Primary Avionics Software Subsystem (PASS) was revised many times in the form of releases to NASA. This nutshell report also underlines the use and importance of SRGMs in practical applications. In fact, according to (Wallace, et al., 2001) increasing dependence of NASA upon mission systems where software component is a major factor meaning that the software systems are critical for the success of NASA missions. The reliability assessment in these systems before NASA accepts them for success of their missions is a huge aspect of software reliability modeling.

XI. As is documented in (Pham, 2003), Hewlett-Packard (H&P) put to use an existing SRGM in their software product development in order to estimate the failure intensity expected for firmware (embedded software in hardware) of two terminals namely HP2393A and HP2394A and to make a decision on the release time of firmware. This helped HP to ensure reliability and reduce development cost by testing modules more efficiently.

XII. As is documented in (Pham, 2003), AT&T software development team put to use an SRGM for its software system T. The SRGM showed unbiased predictions.

XIII. The following is content is taken from the report titled "*Software problems will set back F-35 joint strike fighter another year – report*" is reported in(http://rt.com/usa/f35-jet-software-delay-233/, 2014)- Published time: March 26, 2014 00:55, Accessed at 7:29 P.M IST on http://rt.com/usa/f35-jet-software-delay-233/. This shall further vindicate the challenges in the field of software engineering and the growing position of software reliability engineering in today's world. This report is about the hurdles that face the scheduled delivery of an ambitious US project for development of F-35 Joint Strike Fighter- world's most expensive aircraft- the all-in-one plane, designed for a host of potential missions, is to have similar versions for the US Air Force, Navy, and Marine Corps. According to a new Government Accountability Office (GAO)report, the F-35's mission management system software needs a vast debugging effort to meet the plane's various requirements. "*Challenges in development and testing of mission systems software continued through 2013, due largely to delays in software delivery, limited capability in the software when delivered, and the need to fix problems and retest multiple software versions,*" the GAO auditors wrote. "*The Director of Operational Test and Evaluation (DOT&E) predicts delivery of warfighting capabilities could be delayed by as much as 13 months. Delays of this magnitude will likely limit the warfighting capabilities that are delivered to support the military services' initial operational capabilities—the first of which is scheduled for July 2015—and at this time it is not clear what those specific capabilities will be because testing is still ongoing.*"The GAO said the *plane needs eight million new lines of software code to overcome the current functionary glitches*.

The report added that only 13 percent of the Block 2B segment of software had been tested as of last January. The target for this prime operational component of the plane was 27 percent. Calling the characteristics of the aircraft's Block 2B software"*unacceptable*" Gilmore's (Pentagon's chief weapons tester, Michael Gilmore) report further said "*Initial results with the new increment of Block 2B software indicate deficiencies still exist in fusion, radar, electronic warfare, navigation, electro-optical target system, distributed aperture system, helmet-mounted display system, and datalink,*". This means the scenario would further delay the release of the product. In response to the GAO findings, the F-35 program's head, Maj. Gen. Christopher Bogdan, said in a statement that "*software continues to remain our number one technical risk on the program, and we have instituted disciplined systems engineering processes to address the complexity of writing, testing and integrating software.*"The report, released on 24th of March (Monday), 2014, detailed only the latest problems with what some have dubbed “*the jet that ate the Pentagon*,” plagued with chronic cost overruns and delayed deliveries. The F-35 Joint Strike Fighter program, which started in 2001, is 70 percent over initial cost estimates and years behind schedule.

The importance of development of better software engineering methodologies is quite obvious after presentation of these software glitches that led to loss of life or financial loss. For an industry, the main focus concerns in software development projects are limited budget and time schedule. Overspending on allocated budget could mean financial suicide for the company and missing release time deadline means delayed software into market would again mean loss on profits expected (Pham, et al., 2000). This means avoiding these concerns is the priority of management.

3. Conclusion

In this paper, we have presented a thorough insight into the happenings triggered by software glitches. An insight into the potential financial loss or even loss of life that may result due to software failures is provided. This is especially true for mission-critical and safety-critical applications. This means that there is an imperative need to have sound techniques and methodologies for quantifying, measuring or improving software reliability of a system. This is a big motivation for hot-conduct of research in the field of software engineering.

4. References

[1] Almering V. [et al.] "Using software reliability growth models in practice" [Journal] // Proc. IEEE Software. - 2007. - Vols. 24, (6). - pp. pp. 82–88.

[2] Bai C.G. [et al.] Software failure prediction based on a Markov Bayesian network model [Journal] // The Journal of Systems and Software. - 2005. - pp. 275-282.

[3] Carnes P. Software reliability in weapon systems [Conference]. - : Proceedings of 8th International Symposium On Software Reliability Engineering, 1997. - pp. 114–115.

[4] Dijkstra E.W. The humble programmer [Journal] // Communications of the ACM. - 1972. - Vol. 15(10). - pp. 859–866(ACM TuringLecture).

[5] Ehrlich W. K., Lee K. and Molisani R. H. Applying reliability measurement: A case Study [Journal] // IEEE Transaction on Software. - 1990. - pp. 56–64..

[6] http://rt.com/usa/f35-jet-software-delay-233/ Software problems will set back F-35 joint strike fighter another year – report [Online] // www.rt.com. - Reuters Report, March 26, 2014. - March 26, 2014. - http://rt.com/usa/f35-jet-software-delay-233/.

[7] Huang Chin-Yu and Hung Tsui-Ying Software reliability analysis and assessment using queueing models with multiple change-points [Journal] // Computers and Mathematics with Applications. - 2010. - pp. 2015-2030.

[8] Jeske D.R., and Zhang X. Some successful approaches to software reliability modelling in industry [Journal] // J. Syst. Softw.. - 2005. - pp. 85–99.

[9] Jones W. D. Reliability models for very large software systems in industry [Conference]. - : Proceedings of the Second International Symposium on Software Reliability Engineering, 1991. - pp. 35–42.

[10] Musa J.D., Iannino A. and Okumoto K. Software Reliability Measurement, Prediction, Application [Book]. - : McGraw-Hill, 1987a.

[11] Pham H Software reliability and cost models: perspectives , comparison and practice [Journal] // European Journal of Operational Research. - 2003. - pp. 475-489.

[12] Pham H. Software Reliability [Book]. - New York : Springer-Verlag, 2000.

[13] Pham L. and Pham H. Software Reliability Models with Time-Dependent Hazard Function Based on Baysian Approach [Journal] // IEEE Transactions on Systems, Man and Cybernetics-Part-A: Systems and Humans. - 2000.

[14] Safie F.M A risk assessment methodology for space shuttle external tank welds [Conference] // REliability and Maintainability Symposium. - 1994.

[15] Safie F.M Use of probabilistic design methods for NASA applications [Conference] // ASME Symposium on Reliability technology. - 1992.

[16] Safie, F.M. An Overview of Quantative Risk Assessment of Space Shuttle Propulsion Elements [Report] / Marshall Space Flight Center. - Huntsville, Alabama : [s.n.], undated.

[17] Schneidewind N Comparison of Reliability and Testing Models [Journal] // IEEE Transactions on Reliability. - 2008. - pp. 607-615.

[18] Schneidewind N. F. and Keller T. W. " Application of reliability models to the space shuttle" [Journal] // IEEE Transaction on Software. - 1992. - Vol. 9 (4). - pp. 28–33..

[19] Schneidewind N. F. and Keller T. W. Application of reliability models to the space shuttle [Journal] // IEEE Transaction on Software. - 1992. - Vol. 9 (4). - pp. 28–33..

[20] Wallace D and Coleman C. Application and Improvement of Software Reliability Models [Report] : Technical Report / NASA, Goddard Space Flight Centre(GSFC) ; Software Assurance Technology Center. - 2001.

[21] Whittaker James A. What is software testing? And why is it so hard? [Journal] // Software. : IEEE, 2000. - pp. 70-79.