Software Testing Education – A Required Knowledge for Software Systems' Developers and Managers

Afaq Ahmad, Dawood Al-Abri and Fahad Bait-Shiginah

***Abstract*— To develop and manage reliable software, it is essential for developers and managers to acquire knowledge for testing procedures at all stages of software development. Therefore, the knowledge of software testing is equally essential for software developers and managers. However, in teaching software engineering, we often talk about software testing, its importance and consequence damages with their after effects generated crisis. By educating software engineering with embedded software testing components, we can reduce the gap between what kind of software engineers we produce and what type of software engineers the real world needs. Thus, in the current fast growing IT-based world, software testing has become increasingly important and popular in practice for improving reliability and hazard free usage of software. This paper gives emphasize on software testing practice and its importance. It also facilitates the testing for software practicum by providing a compiled list of available free open source software testing tools and programs.**

***Index Terms*— Education, information processing, levels of testing, reliable and hazard free software, software testing, testing tools**

I. INTRODUCTION

EVEN a best design cannot guarantee a perfect product which can assure 100% flawless operation in all situations and conditions. Thus, all products need though testing for the embedded components of firmware, hardware and software. If we analyze the role of software in products we find that software is a set of information which facilitate hardware to process and control information. Although since 1950's computers became extensively used for only for business applications but in today's world it is impossible to run any organization without the aid of computers. This is because of software systems which have provided a new way of processing information that is much faster and more efficient. Therefore, it can be said that software is a most important component of any product because today none of the product is complete without embedding software. Software nonperformance, containing error, creating fault and leading to failure are critical which may turn into catastrophic impacts and can potentially cause human loss and monetary expenses, history is full of such examples.

The authors belong to Department of Electrical & Computer Engineering, College of Engineering, Sultan Qaboos University, P.O. 33, Al-Khodh, Muscat-123, Sultanate of Oman (phone: 968-24141327; fax: 968-24413454; e-mail: afaq@ squ.edu.om)

The importance of software testing can be adjudged by the cases of software failures and their consequences by visualizing the Table 1. The media is full of reports of the of software failures however, we provide some of the cases in Table 1. The table also provides the reporting source references (Ref.) with the aim to let the readers to acquire the details of these cases.

Table 1 Some reported cases of software failures

Reported case	Ref.	Impact
Shutdown of the Hartsfield-Jackson Atlanta International Airport: April 19, 2006	[1]	As a result of false alarm due to a software malfunction, the airport authorities evacuated the security area for two hours while searching for the suspicious device, causing more than 120 flight delays and forcing many travelers to wait outside the airport.
Loss of Communication between the FAA Air Traffic Control Center and Airplanes: September 14, 2004	[2], [3]	A bug in a Microsoft system compounded by human error was ultimately responsible for the three-hour radio breakdown. The controllers lost contact with the planes when the main voice communications system shut down unexpectedly. The outage disrupted about 600 flights (including 150 cancellations), impacting over 30,000 passengers. Two airplane accidents almost occurred and countless lives were in jeopardy.
Crash of Air France Flight 447 May 31 / June 1, 2009	[4]	Partly, due to software error the aircraft was carrying 216 passengers and 12 crew members; all of whom are presumed to be dead. In addition to the loss of the aircraft itself, Air France announced that each victim's family would be paid roughly €17,500 in initial compensation. This accident makes it the deadliest disaster for Air France, surpassing the Air France Flight 4590 in 2000 that killed 109 people.
Crash of Korean Air Flight 801 August 5 1997	[5], [6], [7]	Safety critical software usage and failure resulted air crash of the airplane of 254 people: 2 pilots, 1 flight engineer, 14 flight attendants and 237 passengers. In terms of the loss incurred, of the 254 people on board, 228 were killed, and 23 passengers and 3 flight attendants survived the accident with serious injuries. The airplane was destroyed by impact forces and a post-crash fire. In 2000, a lawsuit was settled in the amount of $70,000,000 US dollars on behalf of 54 families.
Crash of American	[8]	The crash of flight 965 also teaches us that one cannot assume that a

Airlines (AA) Flight 965 December 20 1995		computer automatically ensures safe operation; and it is also important for human users to maintain adequate situational awareness. The crash which occurred just five minutes before its scheduled arrival led to a total of 159 deaths and completely destroyed the airplane (a Boeing 757).
Power-Outage across Northeastern U.S. and Southeastern Canada August 14, 2003	[9]	This incidence forced us to learn that unlike physical systems, software is not prone to wearing out. If a system fails due to a software fault, any identical system will also fail to the same fault. As a result, *"redundancy"* (namely, having a backup system running exactly the same software) is not a solution for such cases. The widespread blackouts also resulted in the shutdown of nuclear power plants in the states of New York and Ohio, and air traffic was slowed as flights into affected airports were halted. More than 50 million people were affected and the total cost was $13 billion.
Patriot Missile - system failure February 25, 1991	[10]	Patriot Missile's system software bug at Dhahran, Saudi Arabia failed to track and intercept an incoming Iraqi missile (Scud). Subsequently Scud hit a U.S. Army barracks, killing 28 soldiers and injuring another 98.
Explosion of Ariane 5 – Flight 501 June 4, 1996,	[11]	The error arose in the active and backup computers at the same time, resulting in the shutdown of both, and subsequent total loss of attitude control. The impact of the incident was that 37 seconds after its lift-off, at an altitude of about 3,500 meters, the launcher veered off its flight path, broke up, and exploded. The rocket had undergone a decade of development, and the expense is estimated at $7 billion. The rocket and cargo were valued at $500 million.
Therac-25 Accidents Between June 1985 and January 1987	[12]	Accidents in safety-critical systems involved in a computerized radiation therapy machine called the Therac-25., six known accidents involved due to massive overdoses by the Therac-25 resulted with deaths and serious injuries.
Emergency-Shutdown of the Hatch Nuclear Power Plant March 7, 2008	[13], [14]	The main cause of this accident was the installation of a software upgrade on a computer, without the knowledge of its impact on the entire system. As a result, automated safety systems at the plant triggered a shutdown. Estimating a loss based on electricity prices, the total cost to purchase electricity from external sources during the 48-hour shutdown period would be approximately $5 million. This does not include other operation-related costs.

II. SOFTWARE QUALITY RISK

The summary of an article based on study sponsored by Software Quality Systems (SQS), which surveyed 309 managers and IT decision-makers in companies across Europe and North America ranging from 1,000 to over 5,000 employees is presented below.

"Companies around the world currently invest more than €50 billion annually in applications testing and quality assurance. The awareness of the commercial added value of flawless, fail-safe corporate applications is increasing, and as a result companies are actively seeking opportunities to improve both software and the organization of testing while meeting budget constraints. The study concludes with 91% opinions of the managers recognize that software testing and quality assurances are the most important IT disciplines within their companies" [15].

The SQS reports for top ten software failures of the world for 2011 and 2012 are listed below. The details can be found in [16], [17].

A. Top software failures of 2011

1. Financial services giant fined $25 million for hiding software glitch that cost investors $217 million. A software error in the investment model used to manage client assets resulted in this international financial services giant being fined.
2. Computer system bugs cause Asian banking facilities' downtime. Computer system problems at one of Japan's largest banks resulted in a nationwide ATM network of more than 5,600 machines going offline for 24 hours, internet banking services being shut down for three days, delays in salary payments worth $1.5 billion into the accounts of 620,000 people and a backlog of more than 1 million unprocessed payments worth around $9 billion.
3. Cash machine bug benefits customers by giving them extra money. An Australian bank began giving out large sums of money from 40 cash machines across one city. Officials at the company said they were operating in stand-by mode, so could not identify the account balances of customers.
4. Leading smart phones suffer an international blackout - Core and back-up switch failures resulted in network services across Europe, the Middle East, Africa and Latin America going down for 3-4 days. The blackouts left millions without email, web browsing or instant messaging services and were reportedly due to server problems at one data centre, in Slough.
5. Bugs in social networking app for tablet just hours after delayed release, this social networking sites' long-awaited tablet app was already receiving reports about minor bugs from clicking through to pages via panel icons to problems posting comments.
6. 22 people wrongly arrested in Australia due to failures in new NZ $54.5 million courts computer system. The computer system linking New South Wales courts and allowing documents to be lodged electronically led to damages claims for unlawful arrest and malicious prosecution, after 3,600 defects in the electronic transfer of data from the courts to the police's database led to the wrongful arrest of 22 individuals.
7. 50,500 cars recalled after airbag-related software glitch. A glitch in the automaker's software design and testing approach, that meant airbags for passengers in the right rear seat during a crash may not be deployed, resulted in the recall of 47,401 vehicles in the US and a further 3,099 in Canada and Mexico.
8. Recall of one million cars addresses fire and rollaway concerns. A Japanese car company was forced to initiate a worldwide recall of over one million vehicles affected by a design flaw allowing residue from window cleaners to accumulate, which can degrade the switch's electrical

contacts and potentially cause a fire over time. This recall followed a global 2.5 million recall by the same company due to design flaws that allowed vehicles to shift out of park and engine stalls.

9. Telecoms glitch affects 47,000 customers' meter readings and costs company NZ $2.7 million. After a software glitch that resulted in customers hitting their data limits early, some 47,000 customers, who were overcharged, were reimbursed by a New Zealand telecoms.

10. Army computer glitches hinder coordinated efforts in insurgent tracking. An army computing system designed to share real-time intelligence with troops on the front line has hindered troops by being unable to perform simple analytical tasks. The $2.7 billion cloud-based computing network system runs slowly when multiple users are on the system at the same time and the system's search tool made finding the reports difficult as the information mapping software was not compatible with the army's existing search software.

B. Top Software Failures of 2012

1. Software glitch costs trading firm $440million in 45 minutes. A trading firm's newly-installed software resulted in a $440 million loss after it rapidly bought and sold over a hundred different stocks in 45 minutes using a flawed software algorithm that bought the shares at market price then sold at the bid price - instantly losing a few cents on each trade. The rapid trades pushed the price of the stocks up, resulting in spectacular losses for the trading firm when it had to sell the overvalued stocks back into the market at a lower price.

2. Leading securities markets' operator of a stock trading business launching its initial public offering on its own trading system was forced to withdraw its IPO after an embarrassing computer glitch caused a serious technical failure on its own exchange. A system problem occurred as soon as the exchange tried to open the ticker symbol of the stock, failing to roll into a continuous trading pattern as it was supposed to, halting the trading on the stock before it had even started trading.

3. Stock Exchange IPO trading of social media giant falls flat. Technology problems affected trading in millions of shares of a popular social media website, after software glitches caused a malfunction in the trading system's design for processing orders and cancellations, meaning orders were processed incorrectly, if at all. As many as 30 million shares worth of trading were affected by the glitch.

4. US elections' vote glitch sees nomination problems. Computer problems drew complaints across the US during the 2012 elections, with numerous problems with voting machine glitches reported by voters. An example was touch screen errors automatically changing the vote from one candidate to another and not allowing voters to reselect or correct the error.

5**.** Airline's software glitch strands travelers for the third time. For the third time in 2012, a computer glitch wreaked havoc on thousands of travelers with a US airline, delaying flights for hours. A glitch in the dispatch system software resulted in hundreds of delayed flights across the US and internationally. The two hour outage held up 636 of the 5,679 scheduled flights and resulted in 10 flights being cancelled altogether.

6. Security staff shortage at international sports event. An internal computer systems problem resulted in miscalculation of the number of security staff required to support an international sports event this summer. This internal staff roster glitch resulted in members of the armed forces being drafted in to act as security staff.

7. Teething problems for new revenue service software system. After upgrading its software and revenue service system, at an estimated cost of $1.3 billion through 2024, to promote e-filing of tax returns, the US revenue service saw delays in handling electronic tax returns, with 85 per cent of refunds delayed by over more than 23 days.

8. Gambler loses winnings to computer virus. A gambler, who was under the impression he'd won just over $1 million, was told by a High Court that, despite his anticipated windfall showing in the online game he had played, he was not a millionaire after all. A software error mistakenly reported his winnings as much higher than they actually were and, due to this contingency being covered in the game's terms and conditions, he could not legally claim his anticipated prize.

9. Utility customers in the dark over late notice and incorrect payment charges. An Australian energy company sent thousands of customers late payment charges for bills they didn't receive due to a computer glitch, while a Germany utility company overcharged 94,000 of its customers due to a computer glitch that incorrectly charged exit fees, costing the energy supplier $2.24 million in settlement payouts.

10. Leap Year bugs disrupt banking and healthcare payment systems. A leading multinational corporation's cloud computing service outage, which affected Governments and consumers, was caused by the additional day in February this year. The same leap year date bug also affected an Australian payment system used by the health industry, resulting in 150,000 customers being prevented from using private health care cards for medical transactions for two days.

The study of the cases listed in the above sections reveals that software failures could easily have been avoided through an effective quality management strategy identifying and resolving potential glitches before they appear through training and knowledge of software testing procedures.

III. SOFTWARE TESTING PROCESS

The importance of testing is obvious, however, yet most of the concerned people do not do it, or do not do it enough. The reason behind this tendency is lack of knowledge about software testing, and its severe and dangerous impact. Also, associated many other embedded reasons are hindering in the way of testing such as cost, time, time to market constraints and non existence of testing solutions. As an example let us talk about loop testing that focuses on the validity of the program loop constructs (i.e. simple loops, concatenated loops, nested loops, unstructured loops), involves checking to ensure loops start and stop when they are supposed to (unstructured loops should be redesigned whenever possible). Consider that there are 10^{14} possible paths. If we execute one test per millisecond, it would take 3,170 years to test all the paths. Thus it can be said that if we try to test too much, the development cost becomes prohibitive from point of view of expenses and time. On the other hand if we test too little, the probability of software failure increases. Figure 1 describes this scenario.

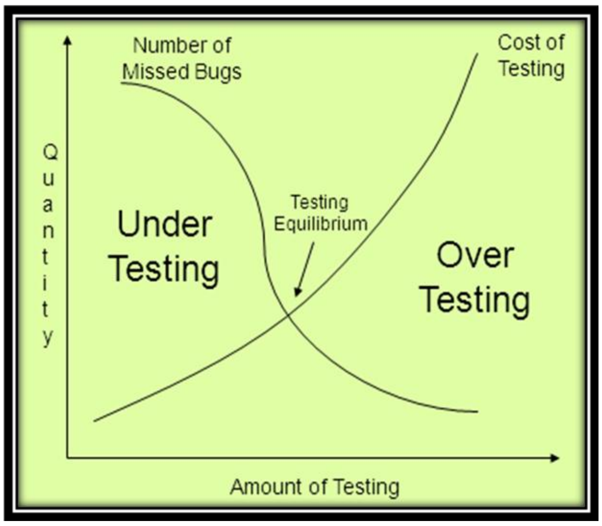


Fig. 1 Testing equilibrium

While it is not possible to remove every error even in a medium scaled software package, but the software engineer aims to remove errors as many as possible. This error removal process is carried out by the software engineer early in the software development cycle. Here it is important to note that testing can only find errors, it cannot prove that a program is bug free at the same time we should remember that test must be planned and designed. Also, we should accept that software testing is an essentially required an important discipline and its knowledge must be acquired as a technical profession.

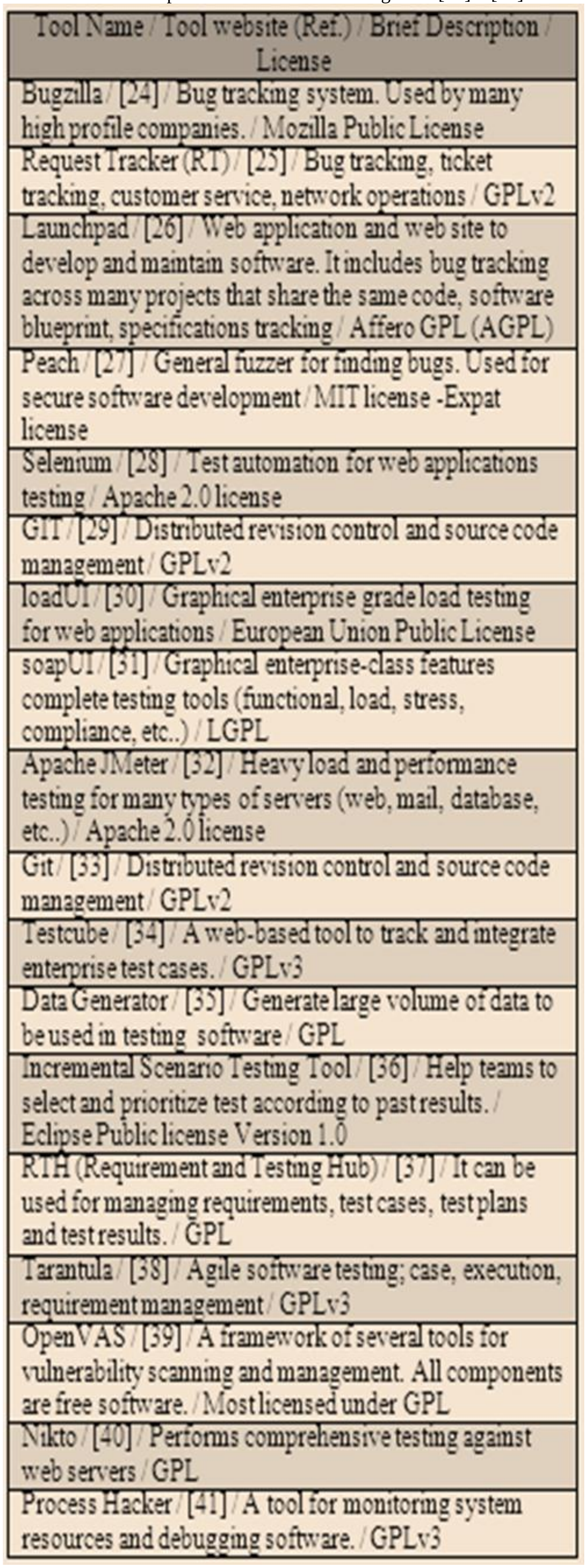
IV. EDUCATE, TRAIN AND EXPERIENCE

It is important for institutions to produce a future generation of qualified software engineers who acquired knowledge, got training and had experience of software testing. We should produce a software engineer who understands of all the key factors that might be relevant for a system, when each is appropriate, and how to evaluate them. These factors include such as risk, safety, performance, reliability, ease of use with clarity, ease of flexibility and modification, hands-on study of real systems and correctness of functioning [18] – [23]. To provide contribution to this end we provide some free and open resources in the field of software testing. The summarized form of the information [24] – [41] is given in Table 2. For more information we provide more resource references [42] – [50] to help the educators.

V. CONCLUSION

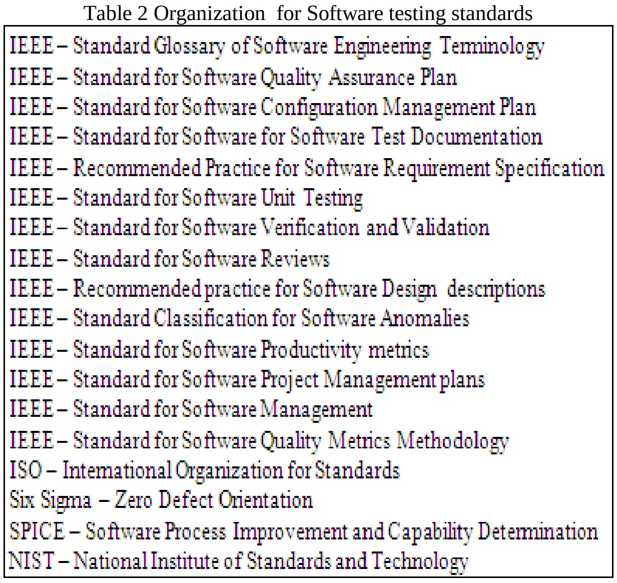
Through this paper we wanted to emphasize the need and importance of knowledge and practice of software testing. In our opinion we believe that software testing education needs to be put right away into the early phase of the learning of introductory programming and computing courses so that it can benefit students throughout their programming projects in later courses as well as their future software development careers. Further, Software engineers should have familiarity with and adoption of industry test standards from organizations. They should keep eye on both internal and external standards. Software engineers must abide by the development and enforcement of the test standards that testers must meet. Institute of Electrical and Electronics Engineers (IEEE) designed an entire set of standards for software and to be followed by the testers. We present a table (Table 3) as a resource for the software testing standards

Table 2 Some open and free software testing tools [24] – [41]



Tool Name / Tool website (Ref.) / Brief Description / License
Bugzilla / [24] / Bug tracking system. Used by many high profile companies. / Mozilla Public License
Request Tracker (RT) / [25] / Bug tracking, ticket tracking, customer service, network operations / GPLv2
Launchpad / [26] / Web application and web site to develop and maintain software. It includes bug tracking across many projects that share the same code, software blueprint, specifications tracking / Affero GPL (AGPL)
Peach / [27] / General fuzzer for finding bugs. Used for secure software development / MIT license -Expat license
Selenium / [28] / Test automation for web applications testing / Apache 2.0 license
GIT / [29] / Distributed revision control and source code management / GPLv2
loadUI / [30] / Graphical enterprise grade load testing for web applications / European Union Public License
soapUI / [31] / Graphical enterprise-class features complete testing tools (functional, load, stress, compliance, etc..) / LGPL
Apache JMeter / [32] / Heavy load and performance testing for many types of servers (web, mail, database, etc..) / Apache 2.0 license
Git / [33] / Distributed revision control and source code management / GPLv2
Testcube / [34] / A web-based tool to track and integrate enterprise test cases. / GPLv3
Data Generator / [35] / Generate large volume of data to be used in testing software / GPL
Incremental Scenario Testing Tool / [36] / Help teams to select and prioritize test according to past results. / Eclipse Public license Version 1.0
RTH (Requirement and Testing Hub) / [37] / It can be used for managing requirements, test cases, test plans and test results. / GPL
Tarantula / [38] / Agile software testing; case, execution, requirement management / GPLv3
OpenVAS / [39] / A framework of several tools for vulnerability scanning and management. All components are free software. / Most licensed under GPL
Nikto / [40] / Performs comprehensive testing against web servers / GPL
Process Hacker / [41] / A tool for monitoring system resources and debugging software. / GPLv3

Table 2 Organization for Software testing standards



IEEE – Standard Glossary of Software Engineering Terminology
IEEE – Standard for Software Quality Assurance Plan
IEEE – Standard for Software Configuration Management Plan
IEEE – Standard for Software for Software Test Documentation
IEEE – Recommended Practice for Software Requirement Specification
IEEE – Standard for Software Unit Testing
IEEE – Standard for Software Verification and Validation
IEEE – Standard for Software Reviews
IEEE – Recommended practice for Software Design descriptions
IEEE – Standard Classification for Software Anomalies
IEEE – Standard for Software Productivity metrics
IEEE – Standard for Software Project Management plans
IEEE – Standard for Software Management
IEEE – Standard for Software Quality Metrics Methodology
ISO – International Organization for Standards
Six Sigma – Zero Defect Orientation
SPICE – Software Process Improvement and Capability Determination
NIST – National Institute of Standards and Technology

ACKNOWLEDGMENT

Authors wish to thank and acknowledgement authorities of Sultan Qaboos University (Sultanate of Oman) for providing environments, facilities and support for carrying out the research work

REFERENCES

[1] CNN, TSA: Computer Glitch Led to Atlanta Airport Scare, April 21, 2006.
[2] IEEE spectrum, vol. 41, no. 11, pp. 16-17, November 2004.
[3] Los Angeles Times, FAA to Probe Radio Failure, September 17, 2004
[4] Air France Flight 447 (http://www.airfrance447.com/about)
[5] S. S. Krause, Aircraft Safety: Accident Investigations, Analyses And Applications, second edition, McGraw-Hill, 2003
[6] National Transportation Safety Board, Abstract on Korean Air Flight 801Conclusions, Probable Cause, and Safety Recommendations," NTSB/AAR-99/02, August 1997
[7] Official Guam Crash Site Information Web Center (http://ns.gov.gu/guam/indexmain.html)
[8] M. Nakao, Crash of American Airlines Boeing, December 1995
[9] Great Northeast Power Blackout of 2003 (http://www.globalsecurity.org/eye/blackout_2003.htm)
[10] E. Schmitt, Army is Blaming Patriot's Computer for Failure to Stop Dhahran Scud, New York Times, May 20, 1991
[11] Ariane 501 Inquiry Board, Ariane 5, Flight 501 Failure, July 1996 (http://en.wikisource.org/wiki/Ariane_501_Inquiry_Board_report)
[12] Nancy, Leveson and Clark S. Turner, An Investigation of the Therac-25 Accidents, IEEE Computer, Vol. 26, No. 7, July 1993, pp. 18-41.
[13] B. Krebs, Cyber Incident Blamed for Nuclear Power Plant Shutdown, Washington Post, June 5, 2008
[14] W. Eric Wong, Vidroha Debroy, and Andrew Restrepo, The Role of Software in Recent Catastrophic Accidents, IEEE Reliability Society 2009 Annual Technology Report.
[15] http://www.testmagazine.co.uk/2012/02/software-testing-as-a-growth-market/
[16] http://www.sqs.com/en/group/about-sqs/press_7273.php
[17] http://uk.prweb.com/releases/2011/12/prweb9029538.htm
[18] Boehm, B., Helping students learn requirements engineering, Proc. Software Engineering Education,1996, pp. 96 -97
[19] Boehm, B. and Port, D., Educating software engineering students to manage risk, Proc. Int. Conference on Software Engineering, 2001.
[20] Ludewig, J., Software Engineering in the year 2000 minus and plus ten, in R. Wilhelm (ed.): Informatics: 10 years back, 10 years ahead, Springer- Verlag, Berlin, Heidelberg, 2001, pp. 102 - 111.
[21] D. Parnas, Software Engineering: An Unconsummated Marriage, CACM, Vol. 40, No. 9 (Sept. 1997), p. 128
[22] D. Parnas, Software engineering programs are not computer science programs, IEEE Software, Vol. 16, No. 6, Nov-Dec 1999, pp. 19-30
[23] Risk Digest, Risks to the Public in Computers and Related Systems Volume 13: Issue 88, Thursday 29 October 1992
[24] http://www.bugzilla.org
[25] http://bestpractical.com/rt/
[26] https://launchpad.net/
[27] http://peachfuzzer.com
[28] http://seleniumhq.org
[29] http://git-scm.com/
[30] http://www.loadui.org/
[31] http://www.soapui.org
[32] http://jmeter.apache.org
[33] http://git-scm.com/
[34] http://www.jatakasource.org/testcube/
[35] http://www.generatedata.com
[36] http://sourceforge.net/projects/istt/
[37] http://sourceforge.net/projects/rth/
[38] http://www.testiatarantula.com/
[39] http://www.openvas.org/
[40] http://www.cirt.net/nikto2
[41] http://processhacker.sourceforge.net/
[42] http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6341619
[43] http://www.indiangnu.org/2011/top-5-open-source-software-testing-qa-tools/
[44] http://www.toolsjournal.com/articles/item/195-10-best-tools-for-test-automation
[45] http://sqa.fyicenter.com/art/Open-Source-Tools-for-Software-Testing-and-QA.html
[46] http://www.testingexperience.com/testingexperience12_12_10.pdf
[47] http://www.opensourcejournal.ro/2009-Volume01/number01/paper012-fullpaper.pdf
[48] https://www.doria.fi/bitstream/handle/10024/63006/nbnfi-fe201005111842.pdf?sequence=3
[49] http://www.rafa-elayyan.ca/publications/24.pdf
[50] http://www.google.com.om/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&ved=0CDIQFjAA&url=http%3A%2F%2Fewh.ieee.org%2Fr10%2Flahore%2FInvited%2520Lectures%2F2009-05-09%2520Saturday%2520 %2520Open_Source_Software_Engineering.ppt&ei=FbAHUb5J9SwhAfVlIDQDQ&usg=AFQjCNEgbngGzo5QZV5OD3yIa_Q75v8pxg