Cataloguing Most Severe Causes that lead Software Projects to Fail

Vikas Sitaram Chomal[1]

vikschomal80@gmail.com
1*Assistant Professor*
The Mandvi Education Society
Institute of Computer Studies,
Mandvi, Surat, India

Dr. Jatinderkumar R. Saini[2]

saini_expert@yahoo.com
2*Director (I/C) & Associate Professor*
Narmada College of Computer Application,
Bharuch, Gujarat, India

***Abstract*--**Nowadays software system is a vital constituent of each and every business representation, be it core product manufacturing, banking, healthcare, insurance, aviation, hospitality, social networking, shopping, e-commerce, education or any other sphere. If any business has to be leveraged and shorten then software has to be integrated with the main stream business of any organization. Crafty and development of any software system entails massive capital, a lot of time, intellectual, domain expertise, tools and infrastructure. Despite the fact that the software industry has highly developed quiet a lot in past decade, however percentage of software failure has also enlarged, which led to the loss of capital, time, good-will, loss of information and in some cases severe failures of critical applications also lead to the loss of lives. This paper presents a classification of the grounds that leads software to be unsuccessful. We have surveyed the allied literature and endeavour to present an extensive and ample catalogue of the diverse causes accountable for making software out of order. The paper also presents a scrutiny as well as categorization of the identified causes.

Keywords: *Information Systems Development (ISD), Information Systems Development Projects, Information Technology, Software, Software Engineering, Software Failure, Software Quality, Software System.*

*****

I. INTRODUCTION

Chomal & Saini [18] defines the primary goal of software engineering as; one basic goal of software engineering is to produce the best possible working software along with the best possible supporting documentation.

Software system is any software product or application supporting any business. A software system could be defined as a system of intercommunicating components based on software forming part of a computer system (a combination of hardware and software). It "consists of a number of separate programs, configuration files, which are used to set up these programs, system documentation, which describes the structure of the system, and user documentation, which explains how to use the system". A system failure occurs when the delivered service no longer complies with the specifications, the latter being an agreed description of the system's expected function and/or service. This definition applies to both hardware and software system failures.

According to Charette [11] SOFTWARE IS EVERYWHERE. It's what lets us get cash from an ATM, make a phone call, and drive our cars. A typical cell phone now contains 2 million lines of software code; by 2010 it will likely have 10 times as many. General Motors Corp. estimates that by then its cars will each have 100 million lines of code. The average company spends about 4 to 5 percent of revenue on information technology, with those that are highly IT dependent--such as financial and telecommunications companies--spending more than 10 percent on it.

According to Lyytinen et al [6], the practice of Information Systems Development (ISD) has undergone a radical transformation during the last decade. Advancing technologies have encouraged a migration away from the tradition, life cycle methods to development toward more flexible and dynamic approaches in which reusable components are assembled into working systems in a radically shorter time. Today, there are few technical reasons for companies to experience the delays and backlogs that plagued systems development 20 years ago. However ISD remains a high proposition. Information system projects continue to fail at an alarming rate, and the problem of 'run away' development projects has never been more serious.

Chomal & Saini [16] and Hilburn et al [14] defines software quality as; software quality is one of those terms that everyone uses; but there is probably not a universal, consistent understanding and agreement about its meaning. Software engineers (and academics) apply the term "quality" to both the product being produced and to the process used to produce it. Although these two types of "quality (product quality and process quality) are dependent on each other, they involve different techniques, measures, and implications. Broadly speaking, product quality is related to how well the product satisfies its customer/user requirements. Related to this (but maybe not specified) are the usability, performance, reliability, and the

maintainability of the software. Process quality is concerned with how well the process used to develop the product worked. In this ease we are concerned with elements such as cost estimation and schedule accuracy, productivity, and the effectiveness of various quality control techniques (e.g., inspection rates and yields)

Every association commences a project with intention of deploying it productively to carry out the purpose specified by the client or as required by the commerce, however there are causes that this target of the association is not accomplished due to some imperfection which later results in failures. Software development project failures have become routine. With approximately each day frequency these failures are reported in newspapers, journal articles, or popular books. These failures are defined in terms of cost and schedule over-runs, project cancellations, and lost opportunities for the organizations that embark on the difficult journey of software development. Rarely do these accounts include perspectives from the software developers that worked on these projects.

So this is not an erroneous proclamation to articulate that software failure could happen at any stage of software product development. Software failure term is in general used when the software doesn‘t perform its intended function or crashes after deployment.

This paper discusses on why most projects fail and what the top reasons for project failure are in software development project.

II. LITERATURE REVIEW

Dalall & Chhillar [13] states that, a recent survey of 800 IT managers says that 62% of total software fails, which is true. 49% software suffered budget overruns, 47% had higher than expected maintenance costs and 41% failed to deliver the expected business value and ROI. Few software while designing never thought of considering the requirements which cause threats and failures later in the stage at the time of utilizing the product for example- information security, hacking, virus threats, scaling up to the level of usage, maintainability and performance. Software projects fail for various reasons from all the domains and technologies. So this paper would consider case studies showing threats, risks and failure of software systems supporting nation security, banking and financial analysis application, aviation, medical and social networking applications which are used globally.

According to Linberg [7], one recent study reported that 31.1% of all corporate software development projects are cancelled before they get completed and 52.7% are costing 189% of their original estimates (Standish Group International, 1994). The Standish study defined project failure as either a project that has been cancelled or a project that does not meet its budget, delivery, and business objectives. Conversely, project success, is defined as a project that meets its budget, delivery, and business objectives. With this definition, the average software project success rate in the Standish study was a dreadful 16.2%.

Rajkumar & Alagarsamy [4] in their research represented, The Standish group and [13] CHAOS report is, only 9% of projects in large companies were successful. At 16.2% and 28% respectively, medium and small companies were somewhat more successful. A massive 61.5% of all large company projects were challenged compared to 46.7% for medium companies and 50.4% for small companies.

Nauman et al [1] states that, the study of the failure of Information System projects in developed and developing countries is one of the hot research areas and many authors have done their work to identify the factors that can minimize the failure rate. . The scope of project could not be visualized by all of the stakeholders which influence the system analysts to overlook or not fully understand the requirements of different users. On the other hand the high expectations by the users about the system or project can cause a project to fail. In Information System Development Projects that support the existing business processes, the alignment of business and IT strategy goals is one of the critical success factors. Leading cause of some IS projects failure is the lack of alignment between business and IT departments in the organization. Environmental problems like procurement, management continuity and optimistic estimations of benefits can also cause project failure. Other factors like differences in age, education, system development experience and managerial position, can also have a profound effect on the success or failure of an Information System Development Project. Some analysis suggests that issues like constructivism and the sociology of technology also affect the success of an Information System Development Project.

Sauer [2] has attempted to classify the failure categories. The classification is given in the following Table 1 and this classification provides a suitable framework to help us make initial diagnosis of the type of failure.

Table 1: Classification of Failure Categories

Type	Description
Correspondence failure	Failure to achieve predefined objectives
Process failure	Failure to produce a system in given limits
Interaction failure	Level of use or user satisfaction

	failure
Terminal failure	Project terminated, can't be tolerated more
Expectation failure	Inability to meet the expectations of specific stakeholder

Morisio et al [9] defines a project failure as, project failure is a project that is "one day late, or one dollar over budget, or one requirement short" an unsuccessful one.

According to Ahmad [19], failure of IT projects can be attributed to some generic root causes, the main categories of the project failure taxonomy. It also indicates that the success of IT projects can be attributed to generic root causes, the main categories of the project success taxonomy. This study reveals that any reason for IT projects' success or failure should belong to one of the categories in the corresponding taxonomy independent of the application domain (banking, telecommunication, healthcare, human resources, etc.)

According to Chomal & Saini [15] and Arlat et al [8] System failure occurs when the delivered service no longer complies with the specifications, the latter being an agreed description of the system's expected function and/or service'. This definition applies to both hardware and software system failures. Faults or bugs in hardware or a software component cause errors.

According to CHAOS MANIFESTO 2013 for the year 2004 - 2012, the percentage of success, failure and challenged software are mentioned in Table 2

Table 2: Percentage details from 2004 – 2012

	2004	**2006**	**2008**	**2010**	**2012**
Successful	29%	35%	32%	37%	39%
Failed	18%	19%	24%	21%	18%
Challenged	53%	46%	44%	42%	43%

Nasir et al [10] states that, the term 'Software Crisis' emerged to describe the software industry's inability to provide customers with high quality products within schedule and under budget. Hardware costs were dropping while software costs were rising rapidly. Major computer system projects were sometimes years late, and the resulting software was unreliable, hard to maintain and performed poorly.

Antoroso et al [3] defines Software trust as the degree of confidence that exists that the software will be acceptable for "one's needs". This means that software trust is established only after one has become convinced, presumably based on a meaningful set of information, that the software does not include flaws that will prevent it from meeting its requirements. The implication is that both innocent and malicious introduction of flaws must be explicitly avoided in trusted software.

Chomal & Saini [17] as well as Selby & Basili [12] in their work define several error related concepts as: (1) Error related effort: Error isolation effort – How long it takes to understand where the problem is and what must be changed. Error fix effort – How long it takes to implement a correction for the error. Error correction effort – How long it takes to correct an error, which is the sum of error isolation effort and error fix effort. (2) Error type: Wrong – Implementation require a change. The existing code or logic needs to be revised, the functionality is present but it is not working properly. Extra – Implementation requires a deletion. The error is caused by existing logic that should not be present. Missing – Implementation requires and addition. The error is caused by missing logic or function.(3) Error severity (trouble reports): Level 1 – Program is unusable, it requires immediate attention. Level 2 – Program is unusable, but functionality is severely restricted and there is no work – around. Prompt action is required. Level 3 – Program is usable, but has functionality that is not critical, it can be avoided. Level 4 – Problem is minor, example message or documentation error and is easily avoided. 4) Error severity (inspection): Major – Error could lead to a problem reported in the field on a trouble report. Minor – Anything that is less than "major" severity, example, typographical mistakes and misspelling. 5) Error reporter type (trouble reports): User – Error is reported by field user or found by a developer while using the product. Developer – Error is discovered by a developer during field testing or when looking at the source code or searching for error in a released system. 6) Inspection type: Design inspection – These are inspections held during the high level and low level design phase. Engineering inspection – These are code inspection that are held after the completion of unit testing.

III. FINDING AND ANALYIS

This section presents our recognition of the major concerns found from the review of related literature. We also present a scrutiny of the recognized points which make a software product inoperative. We deem that these are the grounds for gap in the life cycle of software product development which starts from software requirement collection and tends to complete with the final software product developed.

1) Failure rates of larger IT projects emerge to be higher than small IT projects. Large projects have a high risk of failure since complication increases when the degree of project becomes larger.

2) When you do a project and the customer does not contribute in it, the project is destined to be unsuccessful.

Without user input you cannot feel dedicated to the product, to keep away from the reason of project failure, senior management needs to set up a working environment in which the customer can enthusiastically participate in the project and communicate with the team. Then, project prospect will be clear and right priorities will be set up. Therefore senior management need to continuously support the project to make it clear to staff it is a priority.

3) Time required to complete a particular task refers to the time on a task, while duration represents the time spent on it. Estimating schedule based on the time on task is a common mistake in project management.

4) Although monitoring and checking the work improvement regularly is supposed to be a decisive element for a project triumph, many IT project managers fall short to meet this prerequisite. A probable collapse in communication can consequence in poor requirements understanding for all parties, leading to an abstract project evaluation.

5) Numerous projects are elaborated progressively and in these situations project managers rely on rolling wave planning. As a result, the goal of a project may be only partially clear due to a poor requirement gathering in the definition stage of the project. In such case, the scope and schedule developed by project managers cannot possibly be accurate because their objectives are unclear. Defining clear requirements for a project can take time and lots of communication. Further, changes in objective is been regarded as a natural phenomenon in IT projects by many managers. Without awareness of the differences between initial objectives and new requirements, an IT project can be led in a wrong direction.

6) The project collapse is often caused by lack of testing resources. While software developers focus on generating code, they do not deal with testing. Frequently lack of testers and their poor skills and knowledge will make a project unacceptable because acceptance tests to see whether the product meets the business requirements are not run. Poor testing may be caused by poor requirements set, lack of change control, inadequately trained staff, lack of time for performing testing.

7) Former to picking a software package, management should meet with their staff i.e. end users to determine what the software must do, and for whom. Management often fail to evaluate actual and complete needs by interviewing the staff to discover what is needed and what is desired.

8) Prior to opting for a software program, management must be familiar with the technical capabilities of the staff that will be using and supporting the proposed software system. How much of the configuration can be done by staff? How much training will be necessary to prepare staff to use a new software program? Failure to know this in advance leads to insufficient budget and time allotment in the project plan for implementation and for training.

9) Many management begin researching software without having a clear idea of how much they can afford to spend—not just on the project, but on the total cost of the program, including the cost of allocating its own staff to assist in the project. Failure to allocate funds for change management leads unmet expectations. This may cause the program to either be eliminated, or result in costly changes being required late in the project.

10) Project planning means creating work breakdown, and then assign responsibilities to the developers over time. Project planning consists of construction of various tasks, timelines and essential pathways including Gantt charts and PERT charts and different written plans for various situations. Allocation of roles and responsibilities has to be clearly defined, and it becomes crucial while hiring the stall from outside. Proper scheduling is also required before the start of the project. It includes the time scheduling, teams scheduling. Project managers don't know what they have to plan and schedule. They just only tell the programmer what to do and the programmers can come up with a proper solution. The top secret of a winning software development project is to control the quality up and lower the risk.

11) Risk management is a significant issue on the way to software project failure if it's not administered timely and effectively. As nothing can be calculated that what will ensue in future so we have to take the necessary steps in the present to take any uncertain situation in the future. Risk management means dealing with a concern before it becomes a crisis.

12) Projects with unrealistic expectations are also about evenly likely to be poorly managed projects that fail to validate the feasibility of satisfying user expectations.

13) The majority of products have a number of groups of users who might use different subset of features have different frequencies of use, or have different knowledge level. If software developers do not make out the main user classes for product in premature stage then user needs would not be met. After identifying all user classes, make user that each has a voice.

14) Software projects fall short because we do not recognize that good engineering ideology should be applied to software projects.

15) If the organization or company is unfitted with the necessary technology to complete the project, it will be almost impossible to thrive. This also means having a staff that is equipped with the skills to use that technology, which sometimes gets overlooked if the technology is new to the company. Projects not only fail due to outdated technology, but also because they use new technology that not everyone is caught up on. Technology can also affect a project if learning to use it takes longer than planned.

16) Top management proves a smaller amount of loyalty towards software project task.

Based on the scrutiny of above points, we now propose a cataloguing of these acknowledged points into below stated heads.

1) Size
2) User
3) Stakeholders
4) Goals and objectives
5) Participation/Communication
6) Requirements
7) Technological factors
8) Risk
9) Budget
10) Planning
11) Development Process
12) Management

IV. CONCLUSION

This paper deals with basic aspects which can cause the software development project not to succeed. All of these factors are to be considered at the management level and then transferred to the lower management. Projects succeed and sometimes, projects fail. Knowing what factors can lead to project failure is important for the project manager so they know what to look for when managing their projects. The processes and practices complement each other for improvement and both require human effort from everyone in the organization. As future effort in this course, we mean to mechanize a methodology that will lessen, possibly eradicate altogether, the causes of errors identified by us.

REFERENCES

[1] Abou Bakar Nauman , Romana Aziz and A. F. M . Ishaq, "Information Systems Development Failure: A Case Study to Highlight the IS Development Complexities in Simple, Low Risk Projects in Developing Countries", The Second International Conference on Innovations in Information Technology (IIT'05)

[2] Chris Sauer, Why information systems fail: a case study approach, Alfred Waller Ltd., Publishers, Oxfordshire, UK, 1993

[3] Ed Antoroso, John Watson, Thu Nguyen, Pete Lapiska, Jon Weiss, Terry Starr, "Toward an Approach to Measuring Software Trust", AT&T Bell Laboratories GE Aerospace

[4] G.Rajkumar, Dr.K.Alagarsamy, "THE MOST COMMON FACTORS FOR THE FAILURE OF SOFTWARE DEVELOPMENT PROJECT", The International Journal of Computer Science & Applications (TIJCSA) Volume 1, No. 11, January 2013 ISSN – 2278-1080

[5] J.C.Laprie (Ed.). Dependability: Basic Concepts and Terminology. Springer-Verlag, Wein, New York, 1992.

[6] Kalle Lyytinen, Daniel Robey Learning failure in information system development

[7] Kurt R. Linberg, "Software developer perceptions about software project failure: a case study" , The Journal of Systems and Software 49 (1999) 177±192

[8] Laprie, J. Arlat, C. Béounes and K. Kanoun, "Definition and Analysis of Hardware- and- Software Fault- Tolerant Architectures", Computer, 23 (7), pp.39- 51, July 1990.

[9] Maurizio Morisio, Evgenia Egorova, Marco Torchiano , "Why software projects fail? Empirical evidence and relevant metrics" Proceedings of the IWSM - Mensura 2007

[10] Mohd Hairul Nizam Nasir, Shamsul Sahibuddin , "Critical success factors for software projects: A comparative study" , Scientific Research and Essays Vol. 6(10), pp. 2174-2186, 18 May, 2011

[11] Robert N. Charette - September 2005 IEEE Spectrum: Why Software Fails

[12] Richard W. Selby and Victor R. Basili., "Analyzing Error – Prone Systems", IEEE Transactions on Software Engineering Volume 17 Issue 2, Pages 141- 152, (1991)

[13] Sandeep Dalal1, Dr. Rajender Singh Chhillar,"Case Studies of Most Common and Severe Types of Software System Failure", International Journal of Advanced Research in Computer Science and Software Engineering Volume 2, Issue 8, August 2012 ISSN: 2277 128X

[14] Thomas B. Hilburn, Embry-Riddle, "Software Quality: A Curriculum" Published in Proceeding SIGCSE "00 Proceedings of the thirty – first SIGCSE technical symposium on Computer Science Education Pages 167 – 171 ACM New York, NY, USA ©2000 ISBN:1-58113-213-1

[15] Vikas S. Chomal, Dr. Jatinderkumar R. Saini, "Identification and Analysis of Causes for Software Failures", NATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY] Volume: 04 | Issue: 02 | July – December – 2012

[16] Vikas S. Chomal, Dr. Jatinderkumar R. Saini, "Software Quality Improvement by Documentation – Knowledge Management Model", National Journal of System And Information Technology, Volume 6, Number 1, June 2013, ISSN : 0974 – 3308, Page Number: 49 – 68

[17] Vikas S. Chomal, Dr. Jatinderkumar R. Saini, "Finding Trend of Both Frequency and Type of Errors from Software Project Documentation", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 2, Issue 5, September – October 2013 ISSN 2278-6856

[18] Vikas S. Chomal, Dr. Jatinderkumar R. Saini, "Software Template to Improve Quality of Database Design on basis of Error Identification from Software Project Documentation", International Journal of Engineering and Management Research, Volume-4, Issue-1, February-2014, ISSN No.: 2250-0758, Page Number: 168-179

[19] Walid Al - Ahmad , "Knowledge of IT Project Success and Failure Factors: Towards an Integration into the SDLC", International Journal of Information Technology Project Management, 3(4), 56-71, October-December 2012 Available online at http://www.academicjournals.org/SRE DOI: 10.5897/SRE10.1171 ISSN 1992-2248