EasyChair Preprint

№ 15500

A Study to Investigate Software Failure Factors in Libyan Organizations

Abdelhakim Rashid, Mohamed Hagal and Naser El-Firjani

November 29, 2024

A Study to Investigate Software Failure Factors in Libyan Organizations

Abdelhakim F. Rashid
School of Engineering and Technology
Libyan International University, Libya
abdelhakim.rashid@limu.edu.ly

Mohamed A. HAGAL
Faculty of Information Technology
University of Benghazi, Libya
Mohamed.hagal@uob.edu.ly

Naser F. M. EL-Firjani
Faculty of Information Technology
University of Benghazi, Libya
naser.elfirjani@uob.edu.ly

***Abstract*—Software has become very important in everyday life. It is *used* in almost every field. Even in local society, the dependency level on software has also increased. Whenever there is a software failure, it might have severe consequences on society and individuals alike. Failure does not only affect the client but also the company that is responsible for the development. This study aims to explore the issues that face users and investigate the factors that cause software to fail. A questionnaire was built to derive the issues that users face and distributed using an online Google form across twelve organizations in Benghazi, Libya. In addition, ten interviews were conducted with twenty developers to identify the factors that cause failure. The most common factors among interviewees were identified, including unrealistic schedules and budgets, a lack of user involvement, and incomplete requirements. Finally some recommendations were provided to help engineers and development companies overcome those issues and reduce the failure.**

Keywords- Software engineering, Software failure factors, Software quality

I. INTRODUCTION

Software is pervasive in modern societies. It is becoming increasingly important in every aspect of daily life, almost in every business domain like education, finance, communication, and more. It is also responsible for safety-critical functionalities such as medical, transportation, and nuclear energy fields. Even in local society, the dependency level on software has also increased since the emergence of COVID-19. Software is one of the most economically challenging and yet one of the most important technologies of this era. It is among the most complicated and error-prone in human history.

Good software should be able to provide the end user with the required functionalities and performance; it should also be maintainable, dependable, and usable [1]. In order to develop good software with the above-mentioned characteristics, engineers and development companies need to follow software engineering methods and techniques. The major challenges that face software engineering are developing reliable software, coping with increasing diversity, and demands for reduced delivery times [1].

Since the beginning of programming and the emergence of software engineering, a vast number of software systems have been developed. Moreover, the capability, size, and complexity of software systems have evolved tremendously over the years, and software usage has expanded in many areas. Many of these systems have failed due to various reasons, causing severe and critical consequences. A study of airworthiness directives indicated that thirteen out of thirty-three issued for the period 1984-1994, or 39%, were directly related to software issues. Also, the medical field faces similar issues, as 79% of the medical devices recalled can be attributed to software defects [2].

In the last thirty years or so, software failures delayed the opening of the hugely expensive Denver Airport for a year and destroyed the National Aeronautics and Space Administration (NASA) Mars mission. In the same period, failures wrecked a European satellite launch, killed four marines in a helicopter crash, shut down the ambulance system in London, which led to thirty deaths, and induced a United States (U.S.) Nautical Army Volunteer Yeomen (Navy) ship to destroy a civilian airline [3].

Aside from the inconvenience and possible safety hazards associated with software failures, there are huge economic consequences as well. A 2002 research study by the National Institute of Standards discovered that software failures cost the American economy $59.5 billion annually. For the fiscal year 2003, approximately, the Department of Defense is estimated to have spent $21 billion on software development. Upwards of 40% of the total, or $8 billion, were spent on software reworking due to quality and reliability issues. In addition to the previously mentioned enormous economic statistics, a delay in the Denver airport's automated luggage system due to software issues costs $1.1 million per day. A single automotive software error led to a recall of 2.2 million automobiles and expenses in excess of $20 million [2].

To better understand the financial and business effects of software failures, Tricentis, which is a software-testing company, studied and analysed 606 software failures from 314 companies. The study showed that in 2018, software failures impacted 3.6 billion people and caused $1.7 trillion in financial losses. In addition, the Consortium for Information and Software Quality (CISQ) published a report in 2020 about the cost of poor software quality. This report revealed that the total cost of failed software projects among U.S. companies is an

estimated $260 billion, while the total cost of operational failures caused by poor quality software is an estimated $1.56 trillion [4].

It could be stated that all of what has been mentioned above clearly demonstrates the possible consequences that may result from software failures, where many of these failure cases have resulted from the incorrect use of software engineering methods.

In general, "failure" is any deviation from the expected results. Software failure is defined by IEEE as "the inability of a system or component to perform its required functions within specified performance requirements" [5]. Failure causes the software to produce an incorrect or unexpected result, or to behave unexpectedly.

Software failure can lead to different degrees of harm to organizations or individuals, which include but are not limited to degradation in performance to end users, resulting in losses to the business, causing damage to the company's reputation, and sometimes resulting in the loss of human lives. Therefore, it is useful to understand the software failure, analyse it and determine what causes it to fail.

The rest of the paper is organized as follows: Section II discusses and reviews related work. Section III describes the phases of the methodology. Section IV presents the results and Section V concludes the paper.

II. RELATED WORK

Several studies have been conducted on software and software failures to identify the reasons or factors that cause the software to fail. Some of these studies focused on failures during the development phase. These studies presented some factors that contributed to the failure during this phase, where these factors can have a significant impact on the operational phase. On the other hand, some research studies have concentrated on software failure during the operational phase and identified the factors and issues that cause failure during this phase, which is extremely important due to the critical consequences that result from the failure and the increasing dependency level of societies on software. Therefore, this section discusses and reviews some previous studies related to the failure.

The research study in [6] defined unsatisfactory project outcomes and stated that projects involve several classes of participants or stakeholders, including customers, developers, users, and maintainers. Each class has different but highly important satisfaction criteria. In addition, the researcher stated that "unsatisfactory outcome" is multi-dimensional. Schedule and budget overruns are unsatisfactory for customers and developers. Products with incorrect functionality, user interface shortfalls, low performance, or poor reliability are unsatisfactory for users. The poor software quality is unsatisfactory for maintainers. Based on a survey of several professional project managers, this study identified the top ten primary risks as sources of software failure. These are personnel shortfalls, unrealistic schedules and budgets, developing the wrong properties and functions, designing the wrong user interface, continuous streaming of requirements changes, gold plating, shortfalls in externally developed components, performance shortfalls, shortfalls in externally performed tasks, and straining computer science capabilities.

The factors identified in [7] were responsible for the failures during the development phase. Based on the related literature, this study extracted that the five major factors that contribute to the failure are related to the project managers, customers, and other stakeholders, technology, process, and the project team. Questionnaires and personal interviews with project managers were used in this study to collect data. The researchers used a Structural Equation Model to analyse the gathered data and discovered that the project's internal factors have a greater effect than external factors. Internal factors, such as the project manager and the project team, have a great effect on the project's failure. In particular, the project manager plays a vital role in deciding which development process and technology to use. On the other hand, in external factors, customers play a very important role in the failure of the project. Customers primarily affect the final project deliverables by "requirements definition" and "change of requirements". In particular, a clear expression of the demands of the project from the customer is vital to the success of the project. This study did not identify the failure factors but only presented a classification of the failure factors.

In [8], the researchers revisited the failure factors linked to certain governments' information and communication technologies (ICT) projects. This study analysed the failure factors in Malaysian government agencies and compared them to the previous studies. Based on literature review and interviews, the result of this research contributed to the identification of twenty-one failure factors that affected the Malaysian government's ICT projects, among them: a lack of user involvement, a lack of a project plan, a lack of project management skills and knowledge, the design and technology used not in line with the current technology, poor final product quality, and low or no compatibility between the new system and the existing systems. In addition to the previous factors, the inadequate cost estimation, no standard methodology in place, the end-user is not involved in the user acceptance process, user requirements are not met, and no systematic project evaluation process. These factors were classified into six main categories related to project management, top management, technology, organization, complexity/size, and the process. This study presented effective results by identifying and categorizing the failure factors.

The work presented in [9] studied the software failures in medical devices in the case of the Therac-25 linear medical accelerator to understand the importance of software quality assurance in preventing and reducing failures. This study reviewed and analysed the occurred six failure incidents of the Therac-25 and identified the factors of software failure based on the available documents and case studies. These factors are lack of proper inspection, lack of testing, lack of education and training, lack of documentation and guidelines, poor user interface, and confusing malfunction errors.

Numerous success and failure factors for software projects were discussed in [10]. The researchers reviewed a set of case studies and software project reports to properly comprehend the

success and failure of projects. This research presented failure factors identified in other studies, among them: instability of requirements, incomplete requirements, poor project planning, high schedule pressure, users not involved, poor working environment, problems in project management, poor project progress tracking, unrealistic project objectives, and problems in risk management.

According to the researchers in [11], software failure occurs when the software deviates from the expected behaviour or cannot perform the task it was developed for. This study examined and reviewed several works on software failure and focused on the factors that cause the software to fail or become inoperable. The analysis revealed that failure occurs due to requirement mismatch or conflict, insufficient budget, discrete allocation of tasks, frequent requirement changes, wrong application of engineering principles, schedule pressure, incomplete requirements, and lack of technical skills. In addition to the previous factors, poor communication, market and competitive pressure, lack of proper planning, software development outsourcing, adoption of new technology into legacy systems, and lack of testing are additional factors. These factors must be carefully put into consideration by the analyst, developer and user to produce a credible and reliable software product. These factors can be classified into four categories: management, technical, user and human factors.

Several success and failure factors for software projects were discussed in [12]. This research critically appraised the previous works of researchers and investigated the factors contributing to a software project's success or failure. A non-probability sampling technique was carried out across different software development corporations in the Republic of Mauritius, and an online survey was used to gather data. In addition, a cross-tabulation was performed to identify the factors that impact the success or failure of software projects. Eventually, a set of guidelines and best practices were proposed. The results obtained from this research consolidate what has been observed in related work while adapting it to the context of Mauritius, where the Information and communication technologies (ICT) industry is one of the pillars of the economy. Some failure factors included a lack of clear goals from top management, unrealistic objectives and expectations, and a bad project schedule.

The work presented in [13] is based on prior research. It also drew from a recent experimental investigation. The MMTE Company's software service provider system (ERP) was the subject of this study. They reviewed the assessment and results of their investigation. Based on the survey and study, failure factors were identified and ranked. According to the obtained results, it was concluded that among all the factors mentioned in order: factors of new technologies, lack of planning and insufficient planning, early identification of risks, incomplete, unclear, and ambiguous needs of the client, changes in the project by the client, and commitment to the client are among the main factors of failure for this software project.

From the available literature, it can be concluded that software failure is frequently caused by poor software quality, where failure can have severe consequences on individuals and societies. The previous studies can demonstrate that modern societies rely more and more on the proper functioning of software systems. Even in local society, the dependency level on software has also increased since the emergence of COVID-19. Furthermore, the critical consequences of software failure and the associated economic costs, along with the associated legal liabilities, have made software failure an area of extreme importance.

III. METHODOLOGY

This study utilizes a mixed research approach, combining quantitative and qualitative research. The data collection methods used in this study are questionnaires and interviews to collect primary data. The questionnaire uses closed-ended questions to provide quantitative data expressed in the form of numbers. Items of the questionnaire were adapted from several previous studies. The interview uses open-ended questions to provide qualitative data expressed in the form of text or words to increase the understanding.

A. Questionnaires

The questionnaire questions were designed based on the sub-characteristics of the following quality characteristics which include functionality, performance, reliability, security and usability. Maintainability and portability characteristics were not included because they reflect the developer's view. These questions were evaluated and tested with ten developers and twenty users to help organize, remove flaws and ensure that the questionnaire is easy to understand, fill in and avoid confusion.

Structured close-ended questions were used with limited yes or no answers (also called dichotomous questions) for this questionnaire to provide quantitative data. Since the questions are not open-ended, there is no measurement scale used in this questionnaire, only yes or no. Close-ended questions point toward specific answers, so the scope for uncertainty is limited. Each question covers or expresses one sub-characteristic. The targeted participants were randomly selected software users. The participants were asked if they faced any quality issues during the usage of the software, which may lead to failure, and determine what type of issues they had based on the mentioned characteristics. This questionnaire was distributed across various organizations in both public and private sectors in Benghazi city, such as oil companies, banks, educational organizations, health organizations, airlines and social security. An online Google form was used instead of a hard copy questionnaire because it saves time, is faster to fill in, reaches more participants, reduces cost, and provides response analysis to draw conclusions and make observations on the quality issues that face users. Email and social media were used to distribute this questionnaire. As shown in table 1, 250 participants (users) responded and filled in the form across 12 different organizations.

TABLE 1: Summary Participants' distribution percentage

Organization name	*Number*	*Percentage*
University of Benghazi	20	8%
Arabian Gulf Oil Company	25	10%
Brega Petroleum Marketing Company	25	10%
National Oil Corporation	18	7.2%
Islamic Bank	25	10%
Bank of Development and Commerce	20	8%
Social Security	23	9.2%
Ibn Sina Clinic	20	8 %
Dar Alshifaa Hospital	18	7.2%
Libyans Airlines	19	7.6%
Berniq Airways	19	7.6%
Great Man-made River Administration	18	7.2%
Total	250	100%

The table shows that 8% of the participants included were from the University of Benghazi, 10% were from Arabian Gulf Oil Company, 10% were from Brega Petroleum Marketing Company, 7.2% were from National Oil Corporation, 10% were from Islamic Bank, and 8% were from the Bank of Development and Commerce. While 9.2% were from Social Security, 8% were from Ibn Sina Clinic, 7.2% were from Dar Alshifaa Hospital, 7.6% were from Libyan Airlines, 7.6% were from Berniq Airways, and 7.2% were from the Great Man-made River. The results will be presented in pie charts with text to explain the chart and reported in a table to summarize the results.

B. Interviews

An interview was used because it is useful as a follow-up to particular questionnaire responses. The targeted participants were software developers. The participants were selected based on having five years or more of experience to gather better responses. Semi-structured interviews with open-ended questions were used in this interview to allow free and in-depth discussion, which provides qualitative data and helps collect detailed information. Ten interviews have been conducted. Seven of these interviews were face-to-face, and the other three were conducted online using Google Meet. Each of these interviews was carried out with two participants (developers) to get more precise data. Responses were recorded as notes, followed by a review session to interpret and analyse the data that emerged from the participants using content analysis. All the interviews combined had twenty developers.

IV. THE RESULTS

In this section, the results of the questionnaires and interviews are presented as follows:

A. Questionnaires results

This section presents the issues identified by users, where 250 participants responded to the questionnaires and answered the questions. As shown in table 2.

TABLE 2: Summary of questionnaire results

Characteristic	Yes	NO
Quality issues	(60%)	(40%)
Completeness	(46.7%)	(53.3%)
Correctness	(80.7%)	(19.3%)
Interoperability	(75.3%)	(24.7%)
Time behavior	(47.3%)	(52.7%)
Resource utilization	(82%)	(18%)
Learnability	(82.7%)	(17.3%)
Operability	(81.3%)	(18.7%)
Understandability	(82%)	(18%)
Maturity	(44%)	(56%)
Availability	(84.7%)	(15.3%)
Fault tolerance	(22.7%)	(77.3%)
Confidentiality	(85.3%)	(14.7%)
Integrity	(81.3%)	(18.7%)

The table showed that users suffered from many issues as a result of poor quality. There were 150 users, representing 60% of all participants, who faced some issues during the usage of the software. Some of these issues have a high percentage of users complaining. Out of the 150 users who had issues, there were 80 (53.3%) users suffered from missing functions, 29 (19.3%) suffered from incorrect results, and 37 (24.7%) complained about poor interoperability, as the software did not operate properly with other software in the same environment. For the performance issues, 79 (52.7%) users suffered from poor response time behaviour, while 27 (18%) complained about poor resource utilisation as the amount of used resources was not reasonable. In terms of usability, 26 (17.3%) users found it difficult to learn how to use the software, 28 (18.7%) users found that learning how to operate and control the software was not easy, and 27 (18%) users found it complicated to understand the usage of the software. For reliability issues, 84 (56%) users complained about immature software and that the software was not stable as errors arose during usage. Also, 23 (15.3%) users suffered from the unavailability of software as it was not always operable, and 34 (22.7%) users found that software was affected by hardware and other software faults in the same environment. In terms of security, 22 (14.7%) users complained about unauthorized access to data, and 28 (18.7%) users found that the software did not prevent unauthorized modification of data. It can be noticed that the issues with the highest percentage are missing functions, poor response time behaviour and immature software. These issues cause the software not to perform as required and deviate from the expected results, which is exactly the definition of software failure.

B. Interviews Results

This section presents the failure factors identified by developers in each of the ten interviews conducted with a total of twenty developers, as follows:

The first conducted interview revealed that software fails because of some factors such as staff shortfalls, unrealistic schedule and budget, development of the wrong functions,

requirements changes, lack of user involvement, incomplete requirements, new technology, and poor planning. In addition to inadequate or no management, poor requirement definition, poor communication, lack of process and standards, unmanaged risks, inability to handle the project's complexity, bad coding practices, stakeholder politics, and commercial pressures.

The developers who participated in the second interview identified that the factors contributing to software failure are: unrealistic schedule and budget, lack of user involvement, poor planning, poor requirement definition, unmanaged risks, commercial pressures, and the project was underestimated. In addition, the delivery decisions were made without adequate information about the project, the user was not involved in the user acceptance test, user requirements were not met, and there was poor or no design.

The result of the third interview revealed the following factors: requirements changes, incomplete requirements, lack of process and standards, inadequate testing, failure resulting from unanticipated use, lack of user training, lack of documentation, poor user interface, requirements conflict, lack of technical skills, and poor performance.

The fourth conducted interview revealed that software fails due to certain factors. These factors include: developing the wrong functions, lack of user involvement, incomplete requirements, poor planning, inadequate or no management, poor requirement definition, lack of process and standards, and inability to handle the project's complexity. They also include: bad coding practices, stakeholder politics, the delivery decision was made without adequate information about the project, the user is not involved in the user acceptance test, user requirements are not met, and poor user interface.

During the fifth interview, the developers who participated in this interview identified that software failed due to staff shortfalls, unrealistic schedule and budget, requirements changes, new technology, and poor requirement definition. The developers also mentioned poor communication, project was underestimated, user was not involved in the user acceptance test, poor or no design, inadequate testing, lack of user training, lack of documentation, lack of technical skills, and poor performance.

The sixth interview revealed only eight factors. As follows: unrealistic schedule and budget, lack of user involvement, inadequate or no management, poor communication, unmanaged risks, bad coding practices, and inadequate testing. While the seventh interview also defined eight factors. As follows: lack of user involvement, incomplete requirements, lack of process and standards, stakeholder politics, commercial pressures, inadequate testing, and requirements conflict.

The failure factors identified by developers who participated in the eighth interview are: unrealistic schedule and budget, developing the wrong functions, lack of user involvement, poor planning, poor requirement definition, poor communication, and commercial pressures. These factors also comprise delivery decision was made without adequate information about the project, poor or no design, inadequate testing, lack of user training, poor user interface, lack of technical skills, and poor performance.

The ninth interview revealed the following factors: unrealistic schedule and budget, development of the wrong functions, incomplete requirements, unmanaged risks, bad coding practices, and stakeholder politics. These factors also include: the user is not involved in the user acceptance test, failure resulting from unanticipated use, lack of documentation, poor user interface, and poor performance.

The last interview revealed that the factors contributing to software failure are: requirements changes, lack of user involvement, inadequate or no management, poor requirement definition, poor communication, commercial pressures, the user is not involved in the user acceptance test, inadequate testing, lack of technical skills, and poor performance.

Thirty factors were identified by the developers who participated in the conducted interviews. For each interview, the number of factors ranges from eight to seventeen. Some of these factors are common among all interviewees. Based on the number of occurrences of each failure factor, these are the most common:

- Unrealistic schedule and budget.
- Developing the wrong functions.
- Requirements changes.
- Lack of user involvement.
- Incomplete requirements.
- Poor planning.
- Inadequate or no management.
- Poor requirement definition.
- Poor communication among stakeholders.
- Lack of process & standards.
- Stakeholder politics.
- Inadequate testing.

V. RECOMMENDATIONS

Based on the previously presented results, it is clear that the software industry in the local market suffers from many issues that contribute to software failure. After studying the operational issues that users face and the identified failure factors, this section provides some recommendations to act as proactive steps to help engineers and development companies overcome those issues and reduce the failure factors. Here are some recommendations:

- User involvement is key. Involve the operation team from the beginning of the project to make users work closely with the development team and eliminate the gap between development and operation (developers and users).
- Before development begins, establish a set of development and documentation standards to be used in each stage of the development life cycle to help increase the quality.
- Select a development methodology that suits the project.

- Build acceptance criteria based on the quality characteristics.
- Early testing, where it is far more effective to detect and fix faults early than in the future when hundreds of lines of code must be identified and corrected. It minimizes the risk of rework and the cost associated with overruns.
- Increase the amount of testing to detect as many faults as possible, thus increasing the reliability of the software.
- It's better to conduct testing activities in pairs to improve the error detection effectiveness and efficiency.
- Set a realistic time frame and budget, where tight schedules and low budgets affect the development process.
- Establish a strong and constant communication strategy among stakeholders.
- Use effective management methodology to control the project.

VI. CONCLUSION

Software is one of the most important and yet one of the most economically challenging technologies of this era. It is among the most complex and error-prone in human history. From the available literature, it is clear that software has become very important in every aspect of daily life, almost in every business domain, like education, finance, medicine, transport, communication and more. Even in local society, the usage rate of software has also increased and become important in many business applications. In addition, software failure can have severe consequences on individuals, businesses and the whole society. Such consequences are degradation in performance for end users, resulting in losses to the business, causing damage to the company's reputation, and sometimes resulting in the loss of human lives. Thus, the study of software failure is essential to help reduce failure in future projects, which ultimately reflects on enhancing the quality of life.

This study has shown that users suffer from different issues when using software, and factors that contribute to software failure in Libyan organizations were identified. In addition, some recommendations were provided to help enhance software development with the overall goal of specifying, designing, implementing, and testing a software system with better quality to reduce the failure rate.

The limitation of this study was that some organizations refused to collaborate, especially in the banking sector. As known, this sector has many issues that impact individuals and the whole society.

For future work, it is recommended to extend this study to include a larger sample and investigate failure in other cities.

REFERENCES

[1] I. Sommerville, Software engineering. "TENTH edition Tenth Edition," 10th ed.

[2] W. W. Schilling, "A Cost Effective Methodology for Quantitative Evaluation of Software Reliability using Static Analysis by The University of Toledo College of Engineering," no. December, 2007.

[3] E. E. Ogheneovo, "Software Dysfunction: Why Do Software Fail?," J. Comput. Commun., vol. 02, no. 06, pp. 25–35, 2014, doi: 10.4236/jcc.2014.26004.

[4] Krasner, "Software Quality in Report," pp. 1–46, 2021.

[5] Ieee, "IEEE Standard Glossary of Software Engineering Terminology," Office, vol. 121990, no. 1, p. 1, 1990, doi: 10.1109/IEEESTD.1990.101064.

[6] B. W. Boehm, "Software risk management: Principles and practices," Softw. Manag. Seventh Ed., no. January, pp. 365–374, 1991, doi: 10.1109/9780470049167.ch11.

[7] X. Lu, H. Liu, and W. Ye, "Analysis failure factors for small & medium software projects based on PLS method," ICIME 2010 - 2010 2nd IEEE Int. Conf. Inf. Manag. Eng., vol. 3, pp. 676–680, 2010, doi: 10.1109/ICIME.2010.5478254.

[8] H. S. A. Nawi, A. A. Rahman, and O. Ibrahim, "Government's ICT project failure factors: A revisit," 2011 Int. Conf. Res. Innov. Inf. Syst. ICRIIS'11, pp. 2–7, 2011, doi: 10.1109/ICRIIS.2011.6125738.

[9] K. Madadipouya, "Importance of software quality assurance to prevent and reduce software failures in medical devices: Therac-25 case study," Works.Bepress.Com, no. November, pp. 0–17, 2018, [Online]. Available: https://works.bepress.com/kasra-madadipouya/1/download/

[10] M. Ibraigheeth and S. A. Fadzli, "Core factors for software projects success," Int. J. Informatics Vis., vol. 3, no. 1, pp. 69–74, 2019, doi: 10.30630/joiv.3.1.217.

[11] D. I. A, N. B. K, and A. I. M, "Software Failures: A Review of Causes and Solutions," J. Sciecnce Technol. Educ., vol. 9, no. 1, pp. 415–423, 2021.

[12] Kotowaroo, M.Y., Sungkur, R.K, "Success and Failure Factors Affecting Software Development Projects from IT Professionals' Perspective," 2022, doi: 10.1007/978-981-19-3590-9_60.

[13] Shokrizadeh1, J, "Identifying and ranking key failure factors in software projects(Case study: software services of MMTE Company)", 2023, Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4783773