IJSGS

ISSN: 2488-9229
FEDERAL UNIVERSITY
GUSAU-NIGERIA

INTERNATIONAL JOURNAL OF SCIENCE FOR GLOBAL SUSTAINABILITY

Exploring the Potential Failure Modes in the Software Development Process

[1]Musa Murtala Abubakar and [2]Bashiru Lawal
[1]Department of Computer Science, Federal University Gusau, P.M.B 1001, Zamfara State, Nigeria
[2]Department of Computer Science, Federal College of Education (Technical) Gusau – Nigeria
Corresponding Author's Email & Phone No.: musamurtala247@gmail.com, +2348103407578

Received on: August, 2020 **Revised and Accepted on:** September, 2020 **Published on:** October, 2020

ABSTRACT

This study was conducted to contribute to the practice of risk management in software development project by identifying other important risk dimension that was overlooked in the literature. The focus is on identification of risky potential failure modes in the Software Development Process. To achieve this objective, an exploratory case study involving 15 purposefully selected software development practitioners from 5 software organizations in Abuja Nigeria was conducted to identify and suggest the likely causal factors of the potential failure modes that may exist in the 5 studied stages of the SDP. Multiple research instruments like focus group discussion, audio/video tapes, post-mortem analysis of artefacts were used in this research to gather the required data. The method used to analyse the collected data collected was based on the Miles and Huberman (1994) method of analyzing qualitative data. The findings of this study revealed that the risky potential failure modes that exist in the studied stages of the SDP relatively vary in nature and possess different causal factors characterized with specific inherent symptoms. The relative information about the risky potential failure modes in the SDP is expected to expand project managers understanding of other important risk dimension that can adversely influence project outcome. The study suggested a further qualitative study on the research findings to assess the risk factors associated with the identified potential failure modes that will inform project managers of their likely effect on project outcome.

Keywords: Exploring, Potential failure modes, software development process risk management, *project outcome*

1.0 INTRODUCTION

1.1 Background of the Problem

The Software development process (SDP) also known as Software Development Lifecycle (SDLC) is a framework that defines various activities at each stage in the SDP. Each stage of the SDP is susceptible to several risk factors (Hijazi, Alqrainy, Muaidi, & Khdour, 2014) that threaten the software project success by causing the projects not delivered on time and in line with specifications.

Project risk management on the hand, can be perfectly viewed as a set of principles and practices aimed at identifying, analyzing and handling risk factors to improve the chances of achieving a successful project outcome and/or avoid project failure (Giuseppe, 2017; Kerzner, 2003). However, the era in which project manager's intuitiveness and application of various testing techniques serve as the primary model for ensuring a positive project outcome, high reliability and quality projects are no longer sufficient to prevent our today's software development projects from failure (KPMG, 2019; Castsoftware, 2015). It is sad to note that managers do not give risk management the deserved attention, even though that attitude has damaging implications on the organization's reputation and survival (GitLab, 2019). Hence, research should be focused more on areas that will motivate and improve the practice of risk management in software projects.

In contribution to effective project risk management, Barki, Rivard, & Talbot, (1993) discover that the risks factors that exist in software development projects are multidimensional and suggest that their assessment should be handled through a multidimensional approach. Barki, et al., propose an alternative approach, in which, every risk should be separately defined and be theoretically as well as empirically analyzed. Keil, Li, Mathiassen & Zheng, (2008), hypothesize that assessing project risk factors in a multiple dimension will lead to a clear direction for research and practical purposes. However, despite the huge benefits that can bederived from dimensioning risk factors in the SDP, not much research has been carried out on this issue.

Over the past decades, research conducted on project risk factors with the aim of improving the practice of risk management in software development projects are reported in literature e.g Schmidt, Lyytinen, Keil, & Cule (2001) and Wallace, Keil, & Rai (2004); There are other studies on risk factors that focus on risk that are generally associated with software development projects. For example, Menezes, Gusmão, & Moura, (2018) developed 148 risk factors associated with software development projects and identified requirement stage of the SDLC as the most vulnerable to risk. Hijazi, Alqrainy, Muaidi, & Khdour, (2014) identified 100 risk factors in all the phases of the SDLC to support software practitioners' risk management experiences.

However, findings from previous research on project risk factors did not specifically describe the causal factor and the method of detecting the associated risk dimension in the project. Also, while the specifications and assessment of risk factors at various dimensions, which most previous research provided, empower managers to probe risk levels, previous research does not provide managers information about risk factors associated with the potential failure modes in the software development process.

The implication of this problem is that the lack of information about the relative importance of risk factors associated with the potential failure modes in the SDP contribute to managers' ignorance of their possible threat. This in turn, misdirects project managers and other practitioners' to make risky decisions that can impact negatively on the software development projects (GitLab, 2019). In order words, if project team members and managers are adequately informed about risk factors associated with important dimension relevant to the SDP, and are provided with alternative action plan and reliable techniques to counter the effect of these important risks, there is a good chance they will develop positive risk management practice.

This study is conducted to explore and identify the potential failure modes in the SDP. The study also identifies the possible causal factors and the method of detecting the potential failure modes in the SDP. This study would be of significance to software project managers, software development practitioners as well as software organizations as a means to improve the practice of risk management in software development projects.

The persistent failure of software development projects over the years has made software development practitioners show concern on how to improve the practice of risk management in software development projects. Researchers suggested that poor managerial attitude to risk management in software development was amongst the important factors responsible for this persistent project failure (Castsoftware, 2015). Despite efforts to improve this situation, the rate of project failure does not seem to be decreasing (GitLab, 2019). The lack of sufficient information on important risk dimension such as potential failure modes in the SDP accounted for the poor managerial attitude to persist.

1.3 Research Questions

The following research question were raised to guide the conduct of this study::

1. What potential failure modes do software development experts consider as important risk, whose occurrence in the software development process will have adverse effect on software project outcome?

2. What factors are considered by software development experts to be the likely causes of the identified potential failure modes

3. What failure symptom(s) can be used to detect likely occurrence of the potential failure mode in the Software Development Process?

1.4 Research Objectives

This study was conducted to explore the risky potential failure modes that can adversely influence the software project outcome. To be more specific, the study was set identify:

The potential failure modes that are considered by software development experts as important risk, whose occurrence in the software development process will have adverse effect on software project outcome

The factors that are considered by software development experts to be the likely causes of the identified potential failure modes

1. The failure symptom(s) that can be used to detect likely occurrence of the potential failure mode in the Software Development Process

2.0 METHOD

In this study, a qualitative method involving an exploratory case study research with software development practitioners was used to answer the research questions. The method approach was deemed suitable for this research because rich data and deep insights from the software development practitioners about the manner of possible failure occurrence in the software development process are require. The population of the study is comprised of the software development practitioners in Abuja - Nigeria. The sample consisted of 15 software development practitioners that have practical experiences in software development and are purposefully selected from 5 software organizations in Abuja - Nigeria. Multiple research instruments like focus group discussion, audio/video tapes, post-mortem analysis of artefacts were used in this research to gather the required data. This is to ensure reliability and validity of data to be used for the research. The method used to analyse the qualitative data collected was based on the Miles and Huberman (1994) method of analyzing qualitative data. Miles and Huberman method is most appropriate because the number of participants and data captured during the discussion are small. Hence, the need to use customized software for analyzing the qualitative data was not required.

3.0 RESULTS AND DISCUSSION

Results and discussion of this study are analysed based on the research questions and presented in tabular form (Table 1 to Table 5). The risky potential failure modes in each of the 5 studied stages of the SDP (Planning, Requirement Specification and Analysis, Design, Coding and Implementation. And Installation and Maintenance) are presented in Table 1 to Table 5 respectively, with each table consisting of three columns: Potential Failure Mode, Likely Causes and Symptoms of Detection. Information presented in the three columns provided answer to the three research questions respectively. In other words, information provided in Potential Failure Modes' column answers the research question 1 while information in the column Likely Causes answer the research question 2 and that of symptoms of detection answer research question 3.

Table 1: Planning Stage of SDP

S/N	Probable Failure Mode	Likely Cause	Detection Method
1	Inappropriate chosen software development framework/methodology/models	• May occur when the selected software development model is not appropriate to the scope, magnitude and the complexity of the defined task • May also arises as a result of inexperience management	• If the selected model does not fit the complexity of requirements for the project at the very beginning
2	Communication gap amongst the development team	• If the documentation and results obtained by the chosen software model does not reach the concerned management, development or maintenance team	• If any member of the development team cannot clearly describe the software scope
3.	Ambiguous or unclear specification	• When the specifications for certain project components are not correctly documented	• If the specifications for a certain component of the software project are not clearly defined
4.	Insufficient software project tools and resources	• When the information and documentation about the selected project standards are not enough to complete the required tasks of the project • When the information and documentation about the selected programming language are inadequate for the software development • If the information and documentation about the selected methods and tools for the project are not enough to complete the required tasks	• When the available resources (i.e. people, tools and technologies) are not enough to complete the project
5.	Unrealistic project goals and scopes	• Can be caused by limitations of an inappropriate chosen framework or programming tools	• When project managers find it difficult to specifically determine and state what the project is supposed to do

6.	Unrealistic budget	• Happens when the estimated cost for the project exceeds the available budget • When standard cost estimation model are not used for project	• If the cost estimation of the projects is not correctively realized
6.	Ambiguous project hierarchy model	• When there is no clearly defined hierarchy model for the system users	• Difficulty in identifying the project hierarchy model
7.	Unrealistic project scheduling	• When the schedule made at the during the development the development process fails to determine the needed time for all activities from the beginning to the end and also it • Can be caused by inaccurate cost estimation	• Project duration seems not achievable
8.	Inexperience project team members	• Happens when there is not enough experienced software engineers to cope with the project and • When there are insufficient abilities of the team members to cope with the different problems arising from the project	• If development team cannot provide solutions to the project problems
9	Poor risk management strategies	• When project managers wholly rely on their intuitive ability to manage risk • When risk management is taken as task parallel to management business • When appropriate risk management techniques are not deployed • When all risks associated with the project are not captured	• Experiencing slippage in project budget, schedule and quality
10.	Deviance of system security integration	• When there are hitches in the system which makes it vulnerable to external attacks	• Developing project experiencing security lapses

Table 2: Requirement Specification and Analysis Stage

S/N	Probable Failure Mode	Likely Cause	Detection Method
1	Inappropriate chosen requirement gathering technique	• When the selected technique is not suitable to complete the concrete requirements	• If the selected technique does not fit the complexity of requirements for the project at the very beginning
2	Ambiguous or unclear requirement	• If the requirement gathered is not clearly defined and not clearly understood by the members of the development team	• Members of the development team cannot clearly define the project requirement
3.	Contradictions and confusion in domain specific terminologies	• This may likely occurred when application developers use domain-specific terminologies that are different and not understandable by most end-users	• Mismatch terminologies
4.	Contradictions and confusion in defining requirements in Natural Language (NL)	• may likely occurred when translating some requirements that are not readily expressed in Natural language	• Mismatch specified requirements
5.	Errors in human factor engineering specifications	Can occur due to error in specifications related to: • manual operations • human-equipment interaction • human-supervision • training	• Problems in human engineering operations
6.	Inaccurate requirement	• This happens when the specifications for a certain component of the software projects are not clearly defined, and	• When some requirements are wrongly specified

		• When the specifications for certain project components are not correctly documented	
7.	Unrealistic requirement	• This is caused when there is no concrete evidence that the stated requirement would be sufficiently attainable to be documented and subsequently implemented	• Specified requirements are far from implementation
8.	Inconsistent requirement	• May happen when there is contradictions amongst any requirement gathered for the project	• When there is confusion in the requirement specification
9.	Non-traceable requirement	• Happens when the requirements are not back traceable from the system design • May also occur if the original source of the requirement captured cannot be traced	• When some requirements are missing
10.	Non-verifiable requirement	• Occurred when the captured requirement did not follow any cost effective processes of verification (processes such as testing, inspection, analysis etc.)	• Requirements specified are not relevant to the system development
11.	Infeasible requirement	• Can occurred when there are insufficient resources available for the requirement implementation • Can also be occurred when the requirements cannot be implemented within the constraints of the project	• If requirement implementation practically impossible

Table 3: Design Stage

S/N	Probable Failure Mode	Likely Cause	Detection Method
1	Ambiguous or unclear Requirement Document	• When the system developers were excluded in the planning and requirement capturing and documentation activities	• If the requirement document is difficult to understand
2	Inappropriate chosen architectural design method	• If adequate consideration of choice of architectural design method that will support the programming language to be used was not done	• Design method cannot implement the programming language adopted for the system development
3.	Errors in human factor engineering during design phase	This may likely occurred when important considerations are not made by developers during design modelling. considerations such as : • Data design model • Architectural design model • Procedural design model • Interface design model • Programming language • Implementation	• Series of problems associated with modelling emerging
4.	Poor implementation of design principle	• may likely occurred when there is tight coupling • low cohesion characterizes the design	• When a problem in one program module affects the other in the same project
5.	Errors in human factor engineering specifications	Can occur due to error in specifications related to:	• Series of human related operations evolving

		• manual operations • human-equipment interaction • human-supervision • training	
6.	Complex and complicated design	Can occur due to lack of required expertise and poor design management skills by the developers which result into: • components in the design not correctly combined, • components not well documented and • components affecting other components within the design in unexpected way	• design components too compacted and not working together

Table 4: Implementation and Coding Stage

S/N	Probable Failure Mode	Likely Cause	Detection Method
1	Ambiguous or unclear Design Document	• When the design document is disorganized • Too complex design document • system developers were excluded in the planning and requirement capturing and documentation activities	• Difficulty in comprehending the Design Document by the programmers
2	Inappropriate chosen programming language	• If adequate measure to reconcile the selected programming language to be used with the architectural design was done earlier enough at the design stage	• Architectural design not in alliance with the selected PL for the system development
3.	Errors in human factor engineering during coding and testing	This may likely occurred when wrong modules or functions and or interfaces are developed during programming and testing such as : • Developing the wrong user functions and properties • Developing the wrong user interface Implementation • High fault rate in newly designed components • Poor documentation of test cases • Too many syntax error	• Problems related with human operations evolving
4.	Difficulties in reviewing codes	• may likely occurred when reviewers or testers cannot understand the code • complex and undocumented codes	• Difficulty in code review
5.	No provision for reusability	• This occurs when the component is to be built for the first time, or • If there is no available expertise or framework to upgrade the old ones in order to be used in the new system • inexperience in code maintainability by programmers	• Difficulty in modularity

6.	Misleading and incorrect documentation about the reused component	• Can occur due to lack of required expertise and poor programming skills by the programmers	• Wrong modularity
7.	Poor programming practice such as: too many repetitive code, redundant functions, codes not well commented, too many syntax errors	• Can occur due to lack of required expertise and poor programming skills by the programmers • Manual development of codes in large project • Coding standard and best practices not followed	• Codes cannot compile due to errors
8.	Lack of coordinated team work amongst programmers	• This may likely occurred when different versions of codes and modules are developed by different programmers in an uncoordinated format	• Different code versions for the same component exist
9.	Technology Change	• This can occur when there is need to integrate new technologies, which have not been used before	• Difficulties in the system adaptation to new technology
10.	Inconsistent and complex code	• Occurs when programmers did not follow best programming practices, or • When programmers are inexperienced or lack the required programming skills	• Lengthy lines of codes with multiple code versions for the same component existing
11.	Ambiguous and difficulty in code comprehension	• Likely occurred when the written code is not understandable later for other developers and even for the program author	• Lengthy lines of codes with multiple code versions for the same component existing
12.	Intuitive and or informal testing processes	• When testing is regarded as a non essential activity • When developers rely on their experience and not indulging in formal testing technique • When developers are not well trained on how to conduct testing	• Customer's satisfaction not met
13.	Poor test case documentation	• This can be caused when unqualified testing team were engaged in the testing activity • When some vital test cases are omitted • When manual method was used for testing	• System and unit tests conducted cannot be traced

Table 5: Installation and Maintenance Stage

S/N	Probable Failure Mode	Likely Cause	Detection Method
1	Experiencing difficulties during installation	The new system may be difficult to install If: • the deployers are not well trained or are inexperienced enough and or do not have the adequate knowledge of the system nature and how it works • If the system is complex and distributed and if the • real environment is challenging	• System cannot install
2	System not installed correctly due to change in environment	• This happens specifically when there is hardware advancement	• Error messages such as Incompatible system

		• Can also be caused by long period and continuous Deployment	component displayed during installation
3.	New set of requirements are emerging	• This may likely occurred during the operation, when it has become inevitable to implement new set of requirements not originally captured in order to meet the current actual user needs, business, environmental and organizational changes	• Error messages such as requirement X not recognized or not specified
4.	Persisting difficulties in using the systems	• May likely occurred when end users continuously experiencing difficulties in adapting to the new system • End users cannot understand the operational directives • When the operational directives are complex and undocumented	• System operators experiencing difficulties in system operation
5.	Expected functionalities are omitted	• This occurs when some other expected important functionalities in the new system are missing • If there was insufficient requirement • Miscommunications amongst the project stakeholders • Inconsistencies in requirement documentation	• Error messages such as certain system unit missing
6.	Misleading and incorrect documentation about the system operation	• Can occur due to lack of required expertise on the side of the developers • Due to poor implementation of designing principles • During documentation, specific descriptions are not made, or are not clearly stated during design	• System operators cannot understand the operation manual
7.	Several faults and errors emerging later during acceptance testing	• Can occur when all faults were not adequately mitigated before system operation • It can also occurred due to lack of required expertise amongst testers • Poor project management and or • Standard testing tools and best practices not followed	• Acceptance testing fails
8.	Adequate testing not conducted	• Can happened due to lack of required expertise amongst testers • Poor project management and or • Standard testing tools and best practices not followed	• Customers' satisfaction not met
9.	Difficulties experienced by software engineers in fixing reported problems	This may likely occurred when: • The problem was non-producible by the engineer, or • Due to lack of required expertise on the side of the engineer, or • If end user description of the problem was not detailed enough	Attempted system faults repairs fail
10.	Difficulties in system maintainability	This can be caused due to: • Application of inadequate programming paradigm	• System cannot be reused

		• Rigidity in the system architecture or • due to constraints forced by end users • or constrains forced by the developers	
11.	Change on one component of the new system having adverse effect on the rest of the system	• This may likely happened when adequate impacts analysis to evaluate the effects of the change in the system was not conducted before operation • Poor modularization	• System upgrade impossible
12.	Quest for system change is inevitable due to: System operation being slow for the current loading and concern about some serious security measures	Can likely happened when: • There is presence of new threat • The new system develop faults • Overloaded with complex user's data • There is requirement change	• System consistently developing faults
13.	Miscommunication between end users and product support team	• This may likely occurred if there is communication gap between the customers, managers and developers	• Unhealthy delay in component integration

4.0 CONCLUSION

Based on the findings of this study, the risky potential failure modes that exist in the studied stages of the SDP relatively vary in nature; have different causal factors and exhibit specific inherent symptoms. The relative information concerning the risky potential failure modes in the SDP will expose project managers to their existence and their likely threat on project outcome. It will also empower managers in conducting an effective project risk management in software project.

5.0 RECOMMENDATIONS

Based on the findings of this study, the following the recommendation:

1. A qualitative research is required to assess the risk factors associated with potential failure modes in the studied stages of the SDP. This will assist managers to be informed of their likely effect on project outcome
2. The study recommends that project managers should be mindful of the list of risky potential failure modes, their likely causal factors and their detection symptoms in their practice of risk management in software projects.

ACKNOWLEDGMENT

We would like to acknowledge the support from Head of Department Department of Computer Science Federal University Gusau in person of Mal. Lawal Jabaka who is our guide in all aspect, and we would also have to acknowledge the immense support from our able mentor in person of Dr. Emmanuel Omokhuale, PhD.MNMS, MNAMP, MMAN Department of Mathematical Science Federal University Gusau in undertaking The research with their full support.

REFERENCES

Barki, H., Rivard, S., & Talbot, J. (1993). Toward an assessment of software development risk. *Journal of Management Information Systems*, 10(2), 203-225. http://dx.doi.org/10.1080/07421222.1993.11518006.

Castsoftware, (2015). Software Quality and Developer Productivity: Together Improve Efficiency. Software Intelligence Pulse. Retrieved October, 2016 from GitLab (2019). 2019 Global Developer

Report: DevSecOps. Retrieved October, 2019 from www.castsoftware.com/blog/software-quality-and-developer-productivity-together-improve-efficiency

https://about.gitlab.com/developer-survey/#pdf

Giuseppe, A. (2017). Comparative research about high failure rate of IT projects and opportunities to improve. PM World Journal, 6(2), www.pmworldjournal.net

Hijazi, H., Alqrainy, S., Muaidi, H.,& Khdour, T. (2014a). Risk Factors in Software Development

Phases. European Scientific Journal, 10(3), 121 - 232

Keil, M., Li, L., Mathiassen, L., & Zheng, G. (2008). The influence of checklists and roles on software practitioner risk perception and decision-making. Journal of Systems and Software, 81(6), 908-919. http://dx.doi.org/10.1016/j. jss.2007.07.035.

Kerzner, H. (2003). Project Management: A Systems Approach to Planning, Scheduling, and Controlling, (8th ed).

KPMG (2019). The Future of Project Management: Global Outlook 2019. Australian

Institute of Project management survey 2019. Retrieved December, 2019 from www.aipm.com.au/resources/reports/future-of-project-management-2019.

Menezes, J., Gusmão, C., & Moura, H. (2009). Risk factors in software development projects: a systematic literature review. Software Qual. Journal, 27(2009), 1149–1174, https://doi.org/10.1007/s11219-018-9427-5

Miles, M. B., & Huberman, A. M. (1994). Qualitative data analysis: An expanded sourcebook (2nd ed.). Thousand Oaks, California: Sage Publications

Schmidt, R., Lyytinen, K., Keil, M., & Cule, P. (2001). Identifying Software Project

Risks: An International Delphi Study, Journal of Management Information Systems, 17(4), 5-36.

Wallace, L., Keil, M., & Rai, A. (2004).

Understanding software project risk: a cluster analysis. Information & Management, 42(1), 115-125