SELINUS UNIVERSITY
OF SCIENCES AND LITERATURE

IMPROVING SOFTWARE VERIFICATION AND VALIDATION STATE-OF-PRACTICES: A CASE OF TANZANIA

By

MAGORI ALPHONCE

A DISSERTATION

Presented to the Department of

COMPUTER SCIENCE

Program at Selinus University

In Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Science

ITALY, 2023

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

BIBLIOGRAPHY

REFERENCES

Abbas, N. (2018). *Designing Self-Adaptive Software Systems with Reuse* (Issue 318).

Abdullah, A., Khan, M. H., & Srivastava, R. (2015). Flexibility: A Key Factor to Testability. *International Journal of Software Engineering & Applications*, *6*(1), 89–99. https://doi.org/10.5121/ijsea.2015.6108

Anand, A., & Uddin, A. (2019). Importance of Software Testing in the Process of Software Development. *IJSRD-International Journal for Scientific Research & Development*|, *6*(February), 2321–0613. www.ijsrd.com

Andersson, C., & Runeson, P. (2014). *Verification and validation in industry — A qualitative survey on the state of practice Verification and Validation in Industry - A Qualitative Survey on the State of Practice*. *February 2002*. https://doi.org/10.1109/ISESE.2002.1166923

Anwar, N., & Kar, S. (2019). Review Paper on Various Software Testing Techniques & Strategies. *Global Journal of Computer Science and Technology*, *19*(2), 43–49. https://doi.org/10.34257/gjcstcvol19is2pg43

Babbar, H. (2017). Software Testing : Techniques and Test Cases. *International Journal of Research in Computer Applications and Robotics*, *5*(3), 44–53.

Bäckström, K. (2022). *Industrial Surveys on Software Testing Practices : A Literature Review*.

Beyer, D. (2022a). Advances in Automatic Software Testing: Test-Comp 2022. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 13241 LNCS*. Springer International Publishing. https://doi.org/10.1007/978-3-030-99429-7_18

Beyer, D. (2022b). Progress on Software Verification: SV-COMP 2022. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *13244 LNCS*, 375–402. https://doi.org/10.1007/978-3-030-99527-0_20

Bjarnason, E., Runeson, P., Borg, M., Unterkalmsteiner, M., Engström, E., Regnell, B., Sabaliauskaite, G., Loconsole, A., Gorschek, T., & Feldt, R. (2014). Challenges and practices in aligning requirements with verification and validation: a case study of six companies. *Empirical Software Engineering*, *19*(6), 1809–1855.

https://doi.org/10.1007/s10664-013-9263-y

Bondarev, S. E., Chudinov, M. A., & Prokhorov, A. S. (2019). The analysis of existing methods of software verification. *Proceedings of the 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, ElConRus 2019*, *1*, 191–193. https://doi.org/10.1109/EIConRus.2019.8657169

Bryant, M. (2006). Conducting observational research. *Research Methods in Health Promotion*, 107–128.

Campbell, S., Greenwood, M., Prior, S., Shearer, T., Walkem, K., Young, S., Bywaters, D., & Walker, K. (2020). Purposive sampling: complex or simple? Research case examples. *Journal of Research in Nursing*, *25*(8), 652–661. https://doi.org/10.1177/1744987120927206

Carlos, T. M., & Ibrahim, M. N. (2021). *Practices in software testing in Cameroon challenges and perspectives*. *November 2020*, 1–17. https://doi.org/10.1002/isd2.12165

Centers for Disease Control and Prevention. (2018). Data Collection Methods for Program Evaluation: Observation. *Centers for Disease Control and Prevention*, *16*, 2. https://www.cdc.gov/healthyyouth/evaluation/pdf/brief16.pdf

Ciesielska, M., & Jemielniak, D. (2017). Qualitative methodologies in organization studies. *Qualitative Methodologies in Organization Studies*, *2*(December), 1–264. https://doi.org/10.1007/978-3-319-65442-3

Dawadi, S., & Giri, R. A. (2021). *Mixed-Methods Research : A Discussion on its Types , Challenges , and Criticisms*. 25–36.

Dias-Neto, A. C., Matalonga, S., Solari, M., Robiolo, G., & Travassos, G. H. (2017). Toward the characterization of software testing practices in South America: looking at Brazil and Uruguay. *Software Quality Journal*, *25*(4), 1145–1183. https://doi.org/10.1007/s11219-016-9329-3

Edvardsson, J. (2006). *Techniques for Automatic Generation of Tests from Programs and Specifications*. *1034*.

Feldt, R., Marculescu, B., Schulte, J., Torkar, R., Preissing, P., & Hult, E. (n.d.). *Published with permission from: Optimizing Verification and Validation Activities for Software in the Space Industry Optimizing Verification and Validation Activities for Software in the Space Industry*. http://www.bth.se/fou/

Fiechter, A. (2020). *Assessing the Impact of Readability in Software Engineering*. *June*. https://www.inf.usi.ch/lanza/Downloads/MSc/Fiec2020a.pdf

Garousi, V., Felderer, M., & Kuhrmann, M. (2020). *Exploring the industry ' s challenges in software testing : An empirical study*. *February*. https://doi.org/10.1002/smr.2251

Garousi, V., & Varma, T. (2010). A replicated survey of software testing practices in the Canadian province of Alberta: What has changed from 2004 to 2009? *Journal of Systems and Software*, *83*(11), 2251–2262. https://doi.org/10.1016/j.jss.2010.07.012

Gren, L., & Antinyan, V. (2017). On the relation between unit testing and code quality. *Proceedings - 43rd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2017*, 52–56. https://doi.org/10.1109/SEAA.2017.36

Gupta, N. K. (2020). Researching and Writing the Dissertation. In *Prepare, Succeed, Advance*. https://doi.org/10.2307/j.ctv1131gjp.9

Gupta, Y. (1989). Software Quality Assurance. *International Journal of Quality & Reliability Management*, *6*(4), 56–67. https://doi.org/10.1108/02656718910134412

Hunter, R. B., Thayer, R. H., & Paulk, M. C. (2011). Software process improvement. *Software Process Improvement*, *October*, 1–611. https://doi.org/10.1109/9781118156667

Hynninen, T., Kasurinen, J., Knutas, A., & Taipale, O. (2018). Software testing: Survey of the industry practices. *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings*, 1449–1454. https://doi.org/10.23919/MIPRO.2018.8400261

Islamia, J. M. (2017). Research design Research design. *Research in Social Science: Interdisciplinary Perspectives*, *September*, 68–84. file:///E:/Documents/dosen/buku Metodologi/[John_W._Creswell]_Research_Design_Qualitative,_Q(Bookos.org).pdf

Isong, B., & Ekabua, O. (2015). *S TATE - OF - THE -A RT IN E MPIRICAL V ALIDATION OF S OFTWARE M ETRICS FOR F AULT P RONENESS P REDICTION : SYSTEMATIC REVIEW*. *6*(6), 1–18. https://doi.org/10.5121/ijcses.2015.6601

Jamil, M. A., Arif, M., Abubakar, N. S. A., & Ahmad, A. (2017). Software testing techniques: A literature review. *Proceedings - 6th International Conference on Information and Communication Technology for the Muslim World, ICT4M 2016*, *November*, 177–182. https://doi.org/10.1109/ICT4M.2016.40

Kassab, M., Defranco, J. F., & Laplante, P. A. (2017). *Software Testing The State of the Practice*. *August 2019*. https://doi.org/10.1109/MS.2017.3571582

Kassab, M., DeFranco, J., & Laplante, P. (2016). Software Testing Practices in Industry: The State of the Practice. *IEEE Software*, *March 2019*, 1–1. https://doi.org/10.1109/ms.2016.87

Kawulich, B. B. (2005). Participant observation as a data collection method. *Forum Qualitative Sozialforschung*, *6*(2).

Kiger, M. E., & Varpio, L. (2020). Thematic analysis of qualitative data : AMEE Guide. *Medical Teacher*, *0*(0), 1–9. https://doi.org/10.1080/0142159X.2020.1755030

Koshti, S. D. (2013). *Research methodology*.

Latif, B., & Rana, T. (2020). A preliminary survey on software testing practices in Khyber PakhtunKhwa region of Pakistan. *Turkish Journal of Electrical Engineering and Computer Sciences*, *28*(1), 575–589. https://doi.org/10.3906/elk-1903-6

Lee, J., Kang, S., & Lee, D. (2012). Survey on software testing practices. *IET Software*, *6*(3), 275–282. https://doi.org/10.1049/iet-sen.2011.0066

Lu, Y., & Abeysekera, I. (2020). Research methodology and methods. *Social and Environmental Disclosure by Chinese Firms*, *May*, 86–117. https://doi.org/10.4324/9781315797434-11

M. Altaie, A., Gh. Alsarraj, R., & H. Al-Bayati, A. (2020). Verification and Validation of a Software: a Review of the Literature. *Iraqi Journal for Computers and Informatics*, *46*(1), 40–47. https://doi.org/10.25195/ijci.v46i1.249

M., S., Shamsur, M., Z., A., & Hasibul, M. (2018). A Survey of Software Quality Assurance and Testing Practices and Challenges in Bangladesh. *International Journal of Computer Applications*, *180*(39), 1–8. https://doi.org/10.5120/ijca2018917063

Malviya, A. (2019). Software Testing: Concepts and Issues. *SSRN Electronic Journal*, *June*. https://doi.org/10.2139/ssrn.3351067

Mendoza, I., Kalinowski, M., Souza, U., & Felderer, M. (2019). Relating Verification and Validation Methods to Software Product Quality Characteristics: Results of an Expert Survey. *Lecture Notes in Business Information Processing*, *338*(January), 33–44. https://doi.org/10.1007/978-3-030-05767-1_3

MOHAJAN, H. K. (2018). Qualitative Research Methodology in Social Sciences and Related Subjects. *Journal of Economic Development, Environment and People*, *7*(1), 23.

https://doi.org/10.26458/jedep.v7i1.571

Mousaei, M. (2020). *Review on Role of Quality Assurance in Waterfall and Agile Software Development*. *5*(2), 90–97.

Neri de Souza, F., Neri, D. C. D. de S. B., & Costa, A. P. (2016). Asking questions in the qualitative research context. *Qualitative Report*, *21*(13), 6–18. https://doi.org/10.46743/2160-3715/2016.2607

Patel, M., & Patel, N. (2019). Exploring Research Methodology: Review Article. *International Journal of Research and Review Keywords: Research, Methodology, Research Methodology*, *6*(March), 48–55. www.ijrrjournal.com

Peddireddy, S. K. R., & Nidamanuri, S. R. (2021). Requirements Validation Techniques and Factors Influencing them. *Master of Science in Software Engineering*, *February*. www.bth.se

Persson, A. G. M. (2019). Research methodology. In *Foreign Direct Investment in Large-Scale Agriculture in Africa*. https://doi.org/10.4324/9780429020018-4

Polamreddy, R. R., & Irtaza, S. A. (2012). *Software Testing : A Comparative Study Model Based Testing VS Test Case Based Testing*. *March*.

Poudel, I. D. (2018). *Aligning Requirements with Verification & Validation for Software Engineering Process Improvement*.

Quantitative Research Methods. (n.d.).

Quesada-López, C., Hernandez-Agüero, E., & Jenkins, M. (2019). Characterization of software testing practices: A replicated survey in Costa Rica. *Journal of Software Engineering Research and Development*, *7*, 6. https://doi.org/10.5753/jserd.2019.472

Rahim, M. S., Hasan, M. H., Chowdhury, A. E., & Das, S. (2017). Software engineering practices and challenges in Bangladesh: A preliminary survey. *Journal of Telecommunication, Electronic and Computer Engineering*, *9*(3-3 Special Issue), 163–169.

Rajabli, N., Flammini, F., & Member, S. (2021). *Software Verification and Validation of Safe Autonomous Cars : A Systematic Literature Review*. 4797–4819.

Raulamo-Jurvanen, P. (2020). *Evaluating and selecting software test automation tools : synthesizing empirical evidence from practitioners*.

Raulamo-Jurvanen, P., Hosio, S., & Mäntylä, M. V. (2019). Practitioner evaluations on software testing tools. *ACM International Conference Proceeding Series*, 57–66. https://doi.org/10.1145/3319008.3319018

Regulwar, G. B., & Gulhane, V. S. (2010). Software Testing Practices. *International Journal of Computer Applications*, *1*(2), 1–7. https://doi.org/10.5120/68-165

Ridder, H. G. (2017). The theory contribution of case study research designs. *Business Research*, *10*(2), 281–305. https://doi.org/10.1007/s40685-017-0045-z

Rodriguez, M., Piattini, M., & Ebert, C. (2019). Software Verification and Validation Technologies and Tools. *IEEE Software*, *36*(2), 13–24. https://doi.org/10.1109/MS.2018.2883354

Seth, F. P., Taipale, O., & Smolander, K. (2014). *Organizational and Customer related Challenges of Software Testing : An Empirical Study in 11 Software Companies*. *May*. https://doi.org/10.1109/RCIS.2014.6861031

Seuring, S., Yawar, S. A., Land, A., Khalid, R. U., & Sauer, P. C. (2021). The application of theory in literature reviews – illustrated with examples from supply chain management. *International Journal of Operations and Production Management*, *41*(1), 1–20. https://doi.org/10.1108/IJOPM-04-2020-0247

Sp, L. S., & Sp, J. N. (2007). *Model Driven Software Verification and Validation Model Driven Software Verification and Validation*.

Strazdi, L., & Arnicane, V. (2018). *What Software Test Approaches , Methods , and Techniques are Actually Used in Software Industry ?*

Tao, C., Gao, J., & Wang, T. (2019). Testing and Quality Validation for AI Software-Perspectives, Issues, and Practices. *IEEE Access*, *7*, 120164–120175. https://doi.org/10.1109/ACCESS.2019.2937107

Unterkalmsteiner, M. (2015). Coordinating Requirements Engineering and Software Testing. In *2015:08*.

Upadhyay, P. (2012). The Role of Verification and Validation in System Development Life Cycle. *IOSR Journal of Computer Engineering*, *5*(1), 17–20. https://doi.org/10.9790/0661-0511720

Vasanthapriyan, S. (2018). A study of software testing practices in Sri Lankan Software Companies. *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 339–344. https://doi.org/10.1109/QRS-C.2018.00066

Vukovic, V., Djurkovic, J., Sakal, M., & Rakovic, L. (2020). An empirical investigation of software testing methods and techniques in the province of Vojvodina. *Tehnicki Vjesnik*, *27*(3), 687–696. https://doi.org/10.17559/TV-20180713101347

Weber, C. V., Curtis, B., & Chrissis, M. B. (1993). Capability Maturity Model, Version 1.1. *IEEE Software*, *10*(4), 18–27. https://doi.org/10.1109/52.219617

Yadav, B., & Sharma, A. (2017). Gender roles analysis of ornamental enterprises in Maharashtra State, India. *Asian Fisheries Science*, *30*(Special Issue), 333–342. https://doi.org/10.33997/j.afs.2017.30.S1.020

Yadav, P., & Kumari, P. (2015). Review paper on software testing. *International Journal of Engineering Research & Technology (IJERT)*, *1*(12), 588–592.

Yang, H. S., Zheng, L., & Huang, Y. (2012). Critical success factors for MES implementation in China. *IEEE International Conference on Industrial Engineering and Engineering Management*, *9*, 558–562. https://doi.org/10.1109/IEEM.2012.6837801

Zevalkink, J. (2021). Observation method. *Mentalizing in Child Therapy*, *May*, 100–113. https://doi.org/10.4324/9781003167242-6

DECLARATION

I hereby declare that the research work titled "Improving Software Verification and Validation State-of-Practices -case of Tanzania," submitted in fulfillment of the requirements for the award of a Doctor of Philosophy (Ph.D.), is my own work, except where indicated by referencing, and that the work presented in it has not been submitted in support of another degree.

I declare that this research is an original report of my research, has been written by me, and has not been submitted for any previous degree. The experimental and survey work is my own work; the collaborative contributions have been indicated clearly and acknowledged. All supporting literature and resources have been adequately referenced. I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification.

Magori Alphonce

Signature

JANUARY, 2023

CERTIFICATE

This is to certify that the work contained in the thesis entitled "Improving Software Verification and Validation State-of-Practices: A Case of Tanzania," submitted by **Magori Alphonce,** is for the award of the degree of Doctor of Philosophy (Ph.D.) to the Selinus University of Science and Literature. The research work was carried out by him under my direct supervision and guidance. I considered that the thesis has reached the standards and is fulfilling the requirements of the rules and regulations relating to the nature of the degree. The thesis content has not been submitted for the award of any other degree at this or any other university.

Main Advisor ’s Name Date Signature

Supervisor

ACKNOWLEDGMENT

This research is the result of my effort to improve the software verification and validation practices in Tanzanian software development organizations. I've learned a lot about the differences between academic and industrial software verification and validation practices. The more I learn about the industrial and academic focuses, the more I am aware that most software engineers are focusing on developing software with high quality within a deadline and budget, while academic research focuses on theoretical knowledge.

I am truly thankful to the Tanzanian development organizations for supporting me during data collection and giving me the opportunity to do a PhD in Tanzania. I am thankful to all the participants in this research: the managers, software testing experts, developers, and quality assurance team; without them, this thesis could not be completed. I would like to express my deepest gratitude and appreciation to my supervisor.

For his patience, support, and excellent guidance throughout my PhD studies. I cannot express in words how much you have contributed to this study by kindly sharing your knowledge and experience with me. Thank you for everything you have done for me. In addition to this, my heartfelt thanks go to the whole family of Selinus University for allowing me to pursue my study on "Improving Software Verification and Validation State-of-Practice—A Case of Tanzania."

I am grateful to my advisor for his assistance throughout this research. My profound thanks go to my mother, who motivated, encouraged, gave advice, and followed me. My sincere appreciation and thanks go to my father, Alphonce Magori, and all my family members for their patience and sharing of love, care, and financial assistance during my research.

ABSTRACT

In the software industry, quality is an important aspect that every developer should consider. Software development firms are evaluated based on the effectiveness of their verification and validation processes as one of the controls for delivering higher-quality, cost-effective, and timely software to end users.

Software verification and validation activities are applied to all phases of software processes. The aim is to improve the software processes and have higher-quality, cheaper software delivered faster to end users.

This research aims to improve software verification and validation state of practices in the software development process as well as to propose solutions to verification and validation challenges.

The study was conducted at eight software development organizations in Tanzania. comparing existing verification and validation activities in software development organizations, identifying emerging issues with verification and validation processes in software development processes, analyzing the quality controls required for verification and validation in software development processes, and recommending the best solutions and upgraded means of enhancing quality, reducing cost, and saving time for software development phases

The study employed a comparison method for the verification and validation processes in the software development industry. The desk work for the literature review, survey, and experiment used to gather data

The major finding of this study is that verification and validation techniques help provide higher-quality software that is deliverable in the required time period to the customer or end-user.

CHAPTER ONE
INTRODUCTION

1.1 Introduction

Before delivering the final products to end users, software products must be verified and validated. Doing software verification and validation helps determine if software programs meet the client's needs (Mendoza, Souza, et al., 2019). Many software organizations today are focusing on delivering higher-quality software products to users in a timely manner based on market demand (Raulamo-Jurvanen, 2020). Improving the software verification and validation with the aim of increasing the software quality Software process improvement is the key to understanding the software processes used at the moment and changing them with the aim of improving the quality of the software products (Bjarnason et al., 2014).

In this study, the focus is on improving the existing software verification and validation practices.

The study identified existing software verification and validation challenges and proposed a solution to those verification and validation challenges. There are many traditional verification and validation techniques that exist, and many organizations and individuals have applied them today (Vukovic et al., 2020). But some have difficulties helping them improve the quality of the software products and deliver the software within a short timeframe (Torres-Carrion et al., 2018). This study emphasized the importance of improving verification and practice in order to detect software product defects during the stages of software development as a result of a software project failure (Lee et al., 2012; Upadhyay, 2012). The research presented in this thesis focuses on verification and validation techniques that are able to remove a number of bugs or defects and lead to higher-quality software products in a short time. Many countries use software through different applications; in order to make sure that the overall system is working as required (Chandrasekar et al., 2014), it is important to improve the use of verification and validation techniques. The most important factor to consider is how we can bridge the gap between industry research and industry practices for verification and validation, which exist and are used by many organizations and individuals today (Gren & Antinyan, 2017) (Scatalon et al., 2019). Improving the existing verification and validation tools and methods can help software development organizations improve the quality of their software products and deliver them within a short timeframe (Anand & Uddin,

2019). This thesis is based on the field of computer science and highlights verification and validation techniques that already exist and have been discussed in the various literatures that help to improve software processes and software products. There are many traditional verification and validation techniques that are already existing, like software reviews, software inspections, formal methods, and software, but some have some strengths and some have weaknesses once used or applied, like taking a lot of time for discovering bugs or removing This thesis proposes to use verification and validation techniques to improve software processes and deliver higher software quality to the end user in a short period of time. Here we are following the software process improvement, verification, and validation techniques, as well as the software qualities with their characteristics in the background section. Our research methodology employed the survey method, an experiment case study, and a literature survey. The results of this whole work are based on a survey, an experiment case study, and a literature survey. The results and findings derived from the research objectives and research questions of the study

According to several studies on software verification and validation methods, the status and problems of software verification and validation research, and reviews on verification and validation methodology and techniques, as well as testing tools, current software verification and validation processes are far from adequate (Gren & Antinyan, 2017) (Bäckström, 2022). They argue that advanced tools and seamless integration between development and verification and validation are still needed and that there are still gaps between verification and validation research and industry practices (Mohammadi et al., 2013; Quesada-López et al., 2019; Lee et al., 2012).

Empirical studies have shown that in many software development organizations, most of their project time is spent on verification and validation activities (Upadhyay, 2012; Carlos & Ibrahim, 2021; Poudel, 2018; Belay, 2020). Software development organizations in their software development projects spend as much as half of the projected schedule on verification and validation activities (Mendoza et al., 2019) (Nadu & Nadu, 2019). Past studies have shown that different phases of software development have their own verification and validation processes (Naqvi et al., 2020), so if you combine all phases from the designing stage until the product is in the market, nearly half of the time used in the development of the product is used for verification and validation activities (Anwar & Kar, 2019). (Henningsson,

2005; Anasuodei et al., 2021; M. Al Atitaie et al., 2020). Various software teams utilize a wide range of verification and validation procedures, and there is no evidence in the literature about the use and importance of such practices in various industries (Quesada-López et al., 2019). (Dias-Neto et al., 2017; Naqvi et al., 2020). Therefore, there is a gap between academic knowledge and the software verification and validation practices used in software development organizations (Garousi, Felderer, & Kuhrmann, 2020; Quesada-López et al., 2019; Garousi & Zhi, 2013).

1.2 Background of the Study

Software is developed to solve users' needs. In today's competitive world of changing science and technology, software developers and engineers need to deliver quality software consistently and more quickly (Latif & Rana, 2020) (Vukovic et al., 2020). They also need to adhere to the quality and standards required, and that can be achieved through the application of development techniques and tools and the use of verification and validation procedures throughout the development process (Anand & Uddin, 2019). The main objectives of verification and validation in software development are to check if the developed software meets the business needs and specifications (Bondarev et al., 2019) (Latif & Rana, 2020) (Raulamo-Jurvanen et al., 2019). Verification and validation is the combination of analysis and testing activities across the full life cycle and complements the efforts of other quality assurance and control-engineering functions (Fernández-Sanz et al., 2009; Kassab et al., 2016).

Software verification and validation is an important phase of software engineering to ensure the development of high-quality software (Upadhyay, 2012) (Belay, 2020) (Fiechter, 2020). Even if the body of knowledge and the research literature in software verification and validation are vast, there is still a high industry need for more improvements in the effectiveness and efficiency of software verification and validation activities (Al Neaimi, 2012) (Poudel, 2018). While industry and academia are working on their verification and validation activities in a mostly disjoint manner (Quesada-López et al., 2019) (Anand & Uddin, 2019) (Garousi, Felderer, & Kuhrmann, 2020) , it is often not clear what major challenges the industry is experiencing that need more research effort from the academic community (Garousi, Felderer, Kuhrmann, et al., 2020). Furthermore, understanding the specific challenges of the industry in software verification is an important issue in expanding

the contributions and impact of verification and validation research in general (Feldt et al., 2010) (Marttinen et al., 2020) Anwar and Kar (2019)

Software development organizations use a number of software verification and validation methodologies and tools to improve the quality of their products (Latif & Rana, 2020)(Gren & Antinyan, 2017) (ISTQB, 2018) (Henningsson, 2005) (Strazdi & Arnicane, 2018).

To ensure effective software verification and validation in the software development process, many software verification and validation approaches, techniques, and processes are utilized, supported by automated software tools (Beyer, 2022a) (Belay, 2020) (Latif & Rana, 2020) (Garousi, Felderer, Kuhrmann, et al., 2020). Many software verification and validation methods should be designed to be applicable at various levels of system testing, utilizing multiple methodologies and processes for improved software testing (Beyer, 2022a) (Garousi et al., 2020) (Quesada-López et al., 2019). Software verification and validation is a required process for software development organizations veriffy and validate the system under d and afterevelopment (Rahim et al., 2017) (Ullah Khan et al., 2015) (Vukovic et al., 2020). Many leading software development organizations are faced with several complex difficulties relating to technical advances, which include technologies developed by them for the development of developed systems (Latif & Rana, 2020) (Anasuodei et al., 2021) (Garousi et al., 2020) (Kassab et al., 2017). Despite being defined in many different and imprecise ways, quality is without a doubt the most desired component of any software by its stakeholders (Beyer, 2022a) (Upadhyay, 2012) (Chandrasekar et al., 2014) (Mendoza, Souza, et al., 2019) (Fiechter, 2020).

Knowledge of software verification and validation can be applied to various testing activities and purposes (Dias-Neto et al., 2017; Hynninen et al., 2018). Software testing should be conducted throughout the development process because software development is an error-prone task (Strazdi & Arnicane, 2018) (Mousaei, 2020) (M. Altaie et al., 2020) (Mendoza et al., 2019). This ensures that quality software products are produced (Tao et al., 2019). As a result, software testers must collaborate with all other software experts involved in the development process (Fiechter, 2020) (Kassab et al., 2016) (Garousi & Varma, 2010). Importantly, software testers should not only be familiar with a wide range of software testing procedures, but they should also be aware of software development methodologies (Beyer, 2022a) (Seth et al., 2014) (Regulwar & Gulhane, 2010) (Feldt et al., n.d.).

1.2.1 The Evolution of Software Verification and Validation

The evolution of software verification and validation dates back to the 1950s, when it was known as "software testing" or testing in general. Software developers of that time recognized that software testing during software development was an essential and necessary part of the software development process (Ullah, 2019). For example, incomplete or missed software testing has led to disasters such as the crash of an Airbus A400M in 2015 and NASA's Mars Climate Orbiter loss in 1999 (which caused damage of $125 million). Through its evolution, the software industry has understood the need for more process-oriented testing in a phased manner (Ullah, 2019) (Fries, 2012).

So, the software verification and validation we have today didn't evolve in a single day; it took time and sweat to get it where it is today. Testing gurus like Hetzel and Dave Gelprin divide testing into five significant eras:

Debugging-oriented era: This phase was during the early 1950s, when there was no distinction between testing and debugging. The focus was on fixing bugs.

Developers used to write code and, when faced with an error, would analyze and debug the issues. There was no concept of testing or testers.

Demonstration-oriented era: From 1957 to 1978, the distinction between debugging and testing was made, and testing was carried out as a separate activity. During this era, the major goal of software testing was to make sure that software requirements were satisfied.

Destruction-oriented era**:** From 1979 to 1982, the focus was on breaking the code and finding the errors in it. It was Glenford J. Myers who initially introduced the separation of debugging from testing in 1979, although his attention was on breakage testing. It illustrated the software engineering community's desire to separate fundamental development activities, such as debugging, from verification (Ullah, 2019).

Evaluation-oriented era: From 1983 to 1987, the focus was on evaluating and measuring the quality of software. Testing increased the confidence index in how well the software worked. Testers tested until they reached an acceptable point where the number of bugs detected was reduced. This was mainly applicable to large software (Ullah, 2019; M. Altaie et al., 2020).

Prevention-oriented era: 1988–2000 saw a new approach, with tests focusing on demonstrating that software met its specifications, detecting faults, and preventing defects. The code was divided into testable and non-testable sections. Testable code had fewer bugs than code that was hard to test. Identifying testing techniques was critical in this era. The last decade of the 20th century also saw exploratory testing, where a tester explored and deeply understood the software in an attempt to find more bugs. (Latif & Rana, 2020)

The early 2000s saw the rise of new concepts of testing like test-driven development (TDD) and behavioral-driven development (BDD). And in 2004, we witnessed a major revolution in testing with the advent of automation testing tools like Selenium. Likewise, API testing using tools like SOAP UI marked another turning point in the history of testing. Finally, the current era is moving towards testing using artificial intelligence (AI) tools and cross-browser testing using tools like SauceLabs, Browserstack, etc. (Ullah, 2019).

So, many software verification and validation processes happen today because of testing. Online shops deploy millions of lines of code because of the testing in place. Facebook and Instagram developers push code to the live site without any downtime because of the testing mechanism they've set up to ensure zero failures.

In general, the objectives of verification and validation in software development are to ensure that the product satisfies the users' needs. Thus, every aspect of the product's requirements and specifications must be the target of some software verification and validation activity (Anwar & Kar, 2019; Vukovic et al., 2020).

1.2 .2 Quality Software

The quality of software is very important for the industry these days. By "software quality," we mean that the "specified software products or systems" meet the specifications (Mousaei, 2020). The time that software started to be developed, the finishing time, the project lead time, and the timelines of the software project identified a number of attributes to consider when it comes to product-related quality (Gren & Antinyan, 2017). These include efficiency, reliability, usability, and maintainability of the software products. Therefore, the two attributes should be balanced (Mousaei, 2020). These include efficiency, reliability, usability, and maintainability of the software products. Therefore, the two attributes should be balanced. By doing so, higher-quality software can be obtained. It is true that for many software

companies or industries developing software systems, the focus is on delivering higher-quality software to the end user or customer within a short time (Beyer, 2022b). It is not recommended to wait to deliver the software products to the end-user customers until the demands of that software product are met or the demand for that software product is out of the market's range. The cost and timeliness of software projects are major factors in the success of quality software products (Rodriguez et al., 2019). Requirements or specifications. This avoids the consequences or problems of a system failure, such as in socio-technical systems and safety-critical systems (Beyer, 2022).

1.2.3 Software Products Metrics

Software development companies and software developers need to estimate the size, time, cost, and standardization of the software products that they need to develop (Mendoza, Souza, et al., 2019). And follow them in order to make it easy for them to get a clear picture of what products are going to be developed for the specific software projects they need to develop. And follow them in order to make it easy for them to get a clear picture of what products are going to be developed at the specific software project (Bondarev et al., 2019). Based on the software product metrics, it is easy to estimate the size of the software project that needed to be developed. Here, software developers can know how many lines of code are needed during the software coding process, and it is easy to estimate and follow. What is the number of documents that we need to use in the specific software project? Not only that, but by using and following the software product metrics, we can get a number of advantages for software process improvement (Sp & Sp, 2007). For example, we can be able to understand what and how many components are needed for developing the software program or system because there is the possibility of using different components from different vendors if the product metrics are well taken and followed during the software process improvement. (Abdullah et al., 2015)

Therefore, following these metrics also provides for the standardization of the components involved in developing the software system as specified in the original software projects. Also, using the software product metrics helps the software development company or organization estimate the size of software products, which include lines of code (Anasuodei et al., 2021). Also, it is very easy to understand how many test cases will be derived from a specific program. Knowing all the derived test cases that are involved in the software

program, it is easy to test and know how many test cases passed successfully and how many test cases failed. This metric, when applied during software development, makes it easy to establish confidence in the system (Hynninen et al., 2018). Because it is easy to understand what products are used, these also help a lot for the software process improvement because the software developer can be able to understand how many lines of code are involved in the specific program and reuse some of the lines of code for developing the new program, as far as the concept of reusing components is concerned. Research presented in this thesis also shows that following these metrics makes it easy to discover the software products that are as needed, review software development documents involving software development processes (Mousaei, 2020).

1.2.4 Software Process Metrics

The software development process contains different phases; these include identifying the problem to be solved, the software analysis phase, the software design phase, and the software implementation phase (Bondarev et al., 2019). Software process metrics provide a measure of all the phases of software development, from requirement gathering to implementation (Kassab et al., 2016). The applicability of software process metrics is to understand what levels of components are needed and the resources that will be involved in the software development. These include stakeholder capability and efficiency, management goals, and expectations from each phase of the software development process, including the software analysis phase, the software design phase, and the software implementation phase. Therefore, software process metrics provide a measure of all the phases of software development, from requirement gathering to implementation. Through measuring all the phases, it is easy to control and follow each process (Abdullah et al., 2015). The applicability of software process metrics makes it easy to understand what levels of components are needed and the resources that will be involved in the software development. These include stakeholder capability and efficiency, management goals, and expectations from each software development process (Rajabli et al., 2021).

This study shows that there is a need to improve the software process to produce higher-quality software and that it is easy for the software development organization or developers to control their software processes once all of the processes included in the software process are measured. And adhering to these metrics will aid in avoiding the consequences of project

failure. Furthermore, using verification and validation techniques to improve software processes has a number of advantages, including the ability to easily trace the problem by reviewing each metric involved in the software development process (Rahim et al., 2017) (Bäckström, 2022).

These metrics increase the software productivity of the software products through software process improvement approaches (Poudel, 2018). It is easy to have higher-quality software, and the bugs and defects can be discovered easily when injected into or removed from the software system. This can be done through the use of good measurement metrics and the control of software process improvement (Beyer, 2022a).

Figure 1.1: Usages Process Related Quality Attribute

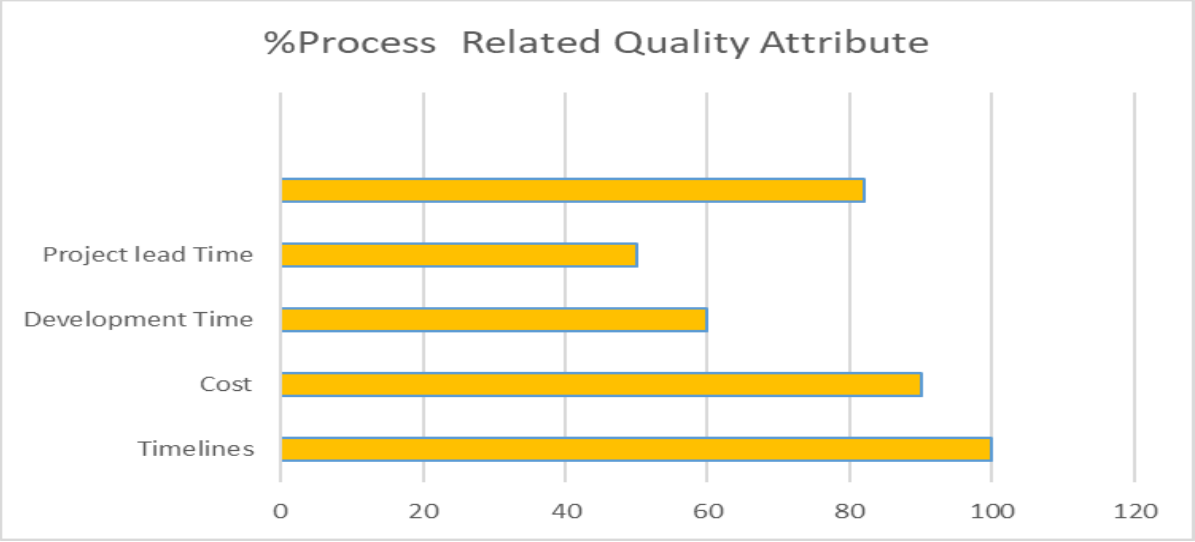
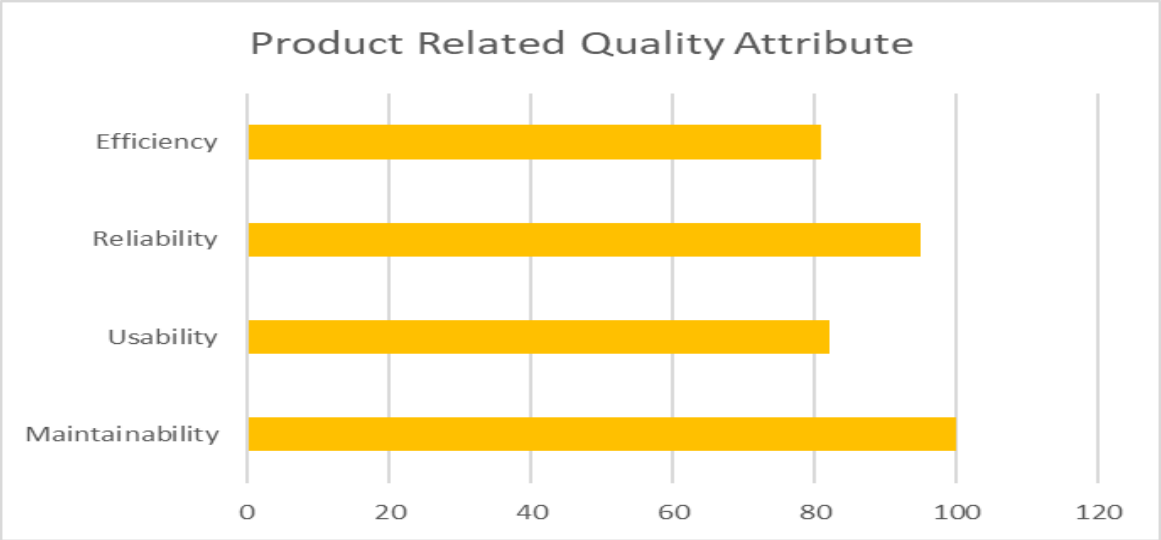


Figure 1.2: Usages Product Related Quality Attribute



1.2.5 Software Verification & Validation

Table 1.1: Methods of Validation

SN	Validation Method	Action performed by	Details
1	Unit Testing	Developers	A single program, module, or unit of code is tested. This is usually performed by the software's creator to ensure that it works as intended.
2	Integrated testing	Involves Software developers with the help of a third-party testing team	The evaluation of related programs, modules, or code components. Confirms that the system's various components interact in accordance with the system's design.
3	System Testing	Involves Team of Independent Testers	An entire computer system is put through its paces. Functional and structural testing, such as stress testing, are examples of this type of testing. Validate the system's specifications.
4	User acceptance Testing	Involves Independent testing team with user support	The process of ensuring that a computer system or elements of a computer system will work in the system regardless of the system requirements

1.2.6 Methods of Verification

Table 1.2: Methods 0f Verification

SN	Verification Method	Details
1	Self-Review	Self-review is highly flexible with respect to time and defect finding, as one need not take an appointment for doing it. Defect found in self-review can help in self-improvement. Understanding-related defects may not be found in self-rev
2	Peer Review	Peer reviews are conducted frequently in SDLC at various stages of development.
3	Online Review	Author and reviewer meet together and review the work product jointly
4	Offline Review	Author informs reviewer that product is ready and reviewer may review product as per his time availability
5	Walkthrough	Walkthrough is a semi-formal type of review as involves larger team along with the author reviewing a work product.
6	Inspection	It is a formal review where external people involved as „inspector". Defects are recorded but solutions are not given by „subject matter experts". This helps the organization to initiate own action plan for fixing the defects.

1.2.7 Software Process Improvement

Software process improvement, focusing on improving the software processes with the intent of increasing the quality of the software products (Peddireddy & Nidamanuri, 2021). There are many software process improvement approaches (Mousaei, 2020). That means focusing on increasing the productivity of the software products within the organization, from the

individual level to the organizational level. Personal software processes and team software processes are the approaches that focus on increasing productivity for individual work or team work so that performance can be increased. That means focusing on increasing the productivity of the software products within the organization, from the individual level to the organizational level. Personal software processes and team software processes are the approaches that focus on increasing productivity for individual work or team work so that performance can be increased (Gren & Antinyan, 2017). These are frameworks for increasing performance at each stage of development (Hunter et al., 2011). Many approaches to software process improvement have been proposed by the Software Engineering Institute. The Capability Maturity Model is one of the process improvement frameworks suggested by the Software Engineering Institute for software products (Weber et al., 1993). There are many software process improvement approaches (Strazdi & Arnicane, 2018). That means focusing on increasing the productivity of the software products within the organization, from the individual level to the organizational level. Personal software processes and team software processes are the approaches that focus on increasing productivity for individual work or team work so that performance can be increased (Raulamo-Jurvanen et al., 2019). That means focusing on increasing the productivity of the software products within the organization, from the individual level to the organizational level. Personal software processes and team software processes are the approaches that focus on increasing productivity for individual work or team work so that performance can be increased. These are frameworks for increasing performance at each stage of development (Rahim et al., 2017). Many approaches to software process improvement have been proposed by the Software Engineering Institute. The Capability Maturity Model is one of the process improvement frameworks suggested by the Software Engineering Institute. This is the model that is used to determine the process maturity of the software development organization. Also, the [personal capability maturity model] is the model for improving the level of individual within the organization; this means that people working within the software engineering area are required to improve their competence and have new knowledge according to changes in software engineering technology (Poudel, 2018).

Software process improvement approaches are iterative and continuous. This includes and uses "plan-do-check-act (PDCA) processes (Hynninen et al., 2018). Software process improvement is an iterative approach to software development. Through this approach, there

is no time limit on the application for software process improvement (Bäckström, 2022). It is possible to restart with the planning stage of the cycle according to what is required to be done in order to meet the goal of the software project. This development cycle is focused on meeting the specified requirements. Also, the time spent on each stage of the cycle depends on what activities have to be done for each stage. The software development cycle has no time constraints and is focused on problem solving. If it happens that the problem is not solved after the completion of all stages of the cycle, the cycle will restart.

1.2.8 Plan-Do-Check-Act (PDCA)

The cycle consists of four steps, as explained in more detail below:

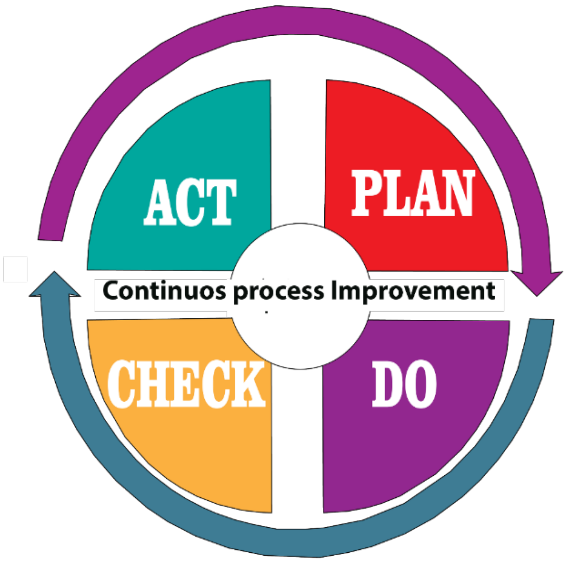


Figure 1.3: The Cycle of Plan-Do-Check-Act

1.1.6.2 Planning: This is typically used to suggest and identify processes that should be used to improve the software process; additionally, this step is used to define the problems that should be solved and find the source of the problems.

1.1.6.3 Doing: This section includes the implementation of the processes and solutions suggested in the planning section. This is an important part, where the implementation of the suggested solution should be done carefully for a better solution.

1.1.6.4 Checking: This part is used to check if the solutions have been implemented correctly or not. These include steps such as reviewing previous measures, which can be done ahead of

time. The aim of reviewing is to see if the specification is related to the solution implemented or not.

1.2.9 Acting: During the acting process, we focus on following up on all implemented solutions, such as the suggested solutions above.

The Plan-Do-Check-Act (PDCA) cycle that helps to understand inputs and outputs is used during the development of software products. This shows if the solutions obtained are working as required, efficiently, and reliably for the suggested systems. Focusing on the improvement of the software processes, we are focusing on selecting appropriate processes and proper methods that should be used for improving the quality of the software products (Peddireddy & Nidamanuri, 2021). This includes comprehending the software processes in use at the time and altering them to improve performance.

The goal of software process improvement in the software industry is to help them improve the performance and quality of their products in response to customer demands. Also improve individuals' skills and performance in software engineering areas.

1.3 Related Studies

Bäckström (2022) conducted a survey on software testing practices in Finland. The finding was that over two-thirds of surveyed populations utilize test levels but that results can vary substantially depending on the surveyed community, with the exception of unit testing. In decreasing order, functional testing, regression testing, performance testing, and usability testing are the most commonly used test types, with security testing being used less frequently.

Carlos & Ibrahim, (2021) conducted a study to provide a summary of Cameroonian practices for software testing. According to the results' interpretation, software testing is still a comparatively small part of software development in Cameroon, even though it occasionally occurs concurrently with development activities. For beta testing, many pass on the price of testing to the clients. This suggests conducting further research to determine, with supporting experimental evidence, the contexts in which the use of best practices and test automation results in a lower.

Hynninen et al. (2018) conducted research on software testing practices and concluded that; first, the use of automation in testing has increased. Automation has become more popular at all levels of testing in recent years. Second, the use of formal software process models and capability maturity models appears to have decreased, while testing tools have increased in use and effectiveness. This change is reflected in the organizational considerations around testing tools: the tools no longer limit the organizational unit as much as they did in 2009, but configuration issues and a lack of platform support have become more common in exchange.

Regulwar & Gulhane, (2010) conducted a study on software testing practices, it was discovered the majority of testing procedures and methods haven't changed all that much in 20 years. In addition to using the right procedures, effective testing calls for a tester's creativity and experience. Testing entails more than just fixing bugs. Testing serves other purposes besides identifying and fixing flaws. The measurement of reliability, validation, and verification are also done using it. Testing is costly. Saving money and time can be accomplished through automation.

Poudel, (2018**)** conducted a study on Aligning Requirements with software testing for Software Engineering Process Improvement The findings were that the greatest problems were incomplete and ambiguous requirements; lack of knowledge about the specific system; communication gaps; and unclear requirements that could cause more problems for the project. The conclusion of the findings was that improving collaboration between the teams, clear communication, and interaction are important solutions to most software development problems. A requirements walkthrough and inspection are needed.

Strazdi & Arnicane (2018) conducted a survey of what Software Test Approaches, Methods, and Techniques are Actually Used in Software Industry? In the IT industry and they came to the conclusion that functional testing is the most commonly used testing method in Latvia's IT industry, with only 52.63 percent of respondents claiming they use non-functional testing frequently. Non-functional testing has some advantages, such as determining the system's overall performance and determining if it performs as expected under normal and expected settings.

Garousi, Felderer, & Kuhrmann (2020) concluded that in software testing, industrial and academic focus areas are disjointed. While academicians are more interested in theoretically

difficult challenges, test engineers in practice are looking for ways to increase testing efficacy and efficiency.

Latif & Rana, (2020) conducted a preliminary survey on software testing techniques in Pakistan. A survey was performed in February of 2018. Despite the fact that the IT sector in Pakistan is still in its infancy, 70 firms answered and the results were collated. In conclusion, the have learned that Pakistan IT enterprises are tiny and inexperienced in applying a standard testing technique.

Quesada-López et al.(2019) conducted as survey on the characterization of software testing practices in Costa Rican Replication. The finding there was a gap between the state of the art and the state of the practice in software testing. The data supports the idea that organizations mostly employ ad hoc criteria to decide when to end testing.

In 2014, a study of unit testing practices was conducted in Sweden. According to the survey's author, Andersson & Runeson, (2014)**,** participants agreed on the scope of unit testing, but they disagreed on whether the test environment is an isolated harness or a partial software system. Furthermore, both technically and strategically, unit testing is a developer issue. Unit testing methodologies and practices appear to be unaffected by test management or quality management. Although unit tests are structural or white-box in nature, developers rarely assess their completion in terms of structural coverage. The majority of the businesses polled wanted to automate their unit tests, but they were having problems distributing best practices across their organizations.

Seth et al., (2014) conducted research on the organizational and customer-related challenges of software testing. The study finds that the development of software quality is an information-intensive process that is influenced by organizational structures and information flow within firms. The project manager acts as a mediator between the development teams and the clients. Their choices could enhance or degrade software quality and productivity.

Kassab et al., (2017) conducted a survey on software testing practices in the industry**.** The findings show that systematic test case design and data definition are standard techniques. Systematic test case design involves using a system for identifying test cases in order to reduce the number of test cases required without sacrificing test efficacy

Vukovic et al. (2020) conducted a survey, and the result showed that most of the techniques in the ISO/IEC/IEEE 29119 testing standard are used to a high level.

Raulamo-Jurvanen, Hosio, and Mäntylä (2019) conducted a study survey on Practitioner Evaluations of Software Testing Tools, and the finding was, there is a need for practical and efficient techniques to conduct tool evaluations that offer software practitioners reliable empirical evidence. More research is required to obtain a better understanding of the situation and to establish more definitive, tool-specific evidence.

Hynninen et al., (2018) conducted research on software testing practices and concluded that; first, the use of automation in testing has increased. Automation has become more popular at all levels of testing in recent years. Second, the use of formal software process models and capability maturity models appears to have decreased, while testing tools have increased in use and effectiveness.

M. et al., (2018) conducted the survey in Bangladesh According to the findings of the survey; many software development companies lack a dedicated testing team. This finding could point to a lack of maturity or financial resources to support a dedicated testing team (M. et al., 2018). They discovered that there are more developers than testers. In addition, they discovered that the tester-to-developer ratio is usually 1:2. However, in some cases, the ratio is 1:10, which is clearly stressful for a tester and for an organization's overall quality goal

Rahim et al., (2017) conducted the survey in Bangladesh According to the findings of the survey; many software development companies lack a dedicated testing team. This finding could point to a lack of maturity or financial resources to support a dedicated testing team. They discovered that there are more developers than testers.

Nadu & Nadu (2019**)** conducted a study in the field of software testing, and the finding was that software testing is usually less formal because of the tough practice and methodologies of testing. Software testing is a collaborative project in which each individual must play their part in producing bug-free software. Though testers aim at producing 100% bug-free software, there will be defects found during the maintenance. Efforts should be made to remove bugs and produce quality software within the specified time and cost.

Isong & Ekabua, (2015) conducted a study on the State-Of-The-Art in Empirical Validation of Software Metrics for Fault Proneness the finding was that the object-oriented paradigm has gained widespread popularity, coupled with software dependability. It is important that high software quality should not be compromised. Object-oriented design metrics should always be used to assess software quality during software development.

Lee, Kang, and Lee (2012) conducted a survey with a wide range of firms and experts from 1000 companies working in software testing to identify existing practices and potential for software testing methodologies and tool improvement. The results of the poll showed five key conclusions about current software testing methodologies and tools, as well as areas for improvement: Low utilization of software testing techniques and tools, challenges caused by a lack of software testing methods and tools, limited use of testing tools, There is a desire for interoperability support between software development and testing techniques and tools, as well as instruction on how to evaluate software testing methods and tools and characterize their capabilities.

There are many verification and validation processes that already exist, and software developers and engineers apply them to the software development process (Upadhyay, 2012) (M. Altaie et al., 2020). There is still a problem when there is a change in requirements (Poudel, 2018) (Al Neaimi, 2012) (Quesada-López et al., 2019) (Dias-Neto et al., 2017) (Vasanthapriyan, 2018). It seems verification and validation processes are not flexible enough to adapt when there are changes in software requirements (Bjarnason et al., 2014).

1.4 Statement of the Problem

Regardless of the use of software testing, verification, and validation activities in the software development process, there are still difficulties in helping software development organizations deliver quality software that is affordable and timely to end users. Meanwhile, verification and validation methods and tools are not flexible enough when there are changes in software system requirements. This study fills these gaps by improving the existing verification and validation state of practices in software development organizations by identifying emerging issues in the verification and validation activities in software development processes, developing and recommending the best solutions, and upgrading means of enhancing software quality, reducing costs, and saving time.

1.5 Significance of the Study

Software development processes have become complex and challenging to develop and maintain because of scalability issues. Software verification and validation is a demanding task, and the challenges of verification and validation of large-scale software cannot be overemphasized due to the large test suite size. This study conducted a cost-effective software verification and validation study to improve the state of the practice and identify cost-effective software verification and validation techniques that produce the same quality product in the same amount of time at a lower cost. The study provides alternative software verification and validation techniques and approaches to software quality management that help software development companies or organizations develop higher-quality software delivered on time that meets the requirements and specifications of their customers or end users. The study also documents the challenges and possible mitigation measures faced by software development companies and organizations in terms of software verification and validation activities during the software development process.

1.6 Study Objectives

1.6.1 Main Objective

The main objective of this study is to improve software verification and validation state of the practices in software development companies

Specifically, the following objectives have been sought to achieve:

i. Investigate the status of existing software verification and validation practices in software development companies.
ii. Identify challenges concerning existing software verification and validation practices in software development companies.
iii. Propose solutions for existing software verification and validation practices challenges in software development companies.
iv. Evaluate the Proposed solutions for existing software verification and validation challenges in software development companies.

1.6.2 Research Questions

The following research question used to design to meet the study objects

i. How do software development companies perform verification and validation?

ii. What are the challenges concerned existing software verification and validation practices in software development companies?

iii. It is possible to propose solutions for the existing software verification and validation practices challenges identified in (ii)? If yes, what possible solutions are there?

iv. How do you compare the efficiency and effectiveness of proposed solutions for the existing software verification and validation practice challenges?

1.7 Scope of the Study

This study focus on improving software verification and validation practices. This study is limited to research on the verification and validation state of practices and their activities in software development companies in Tanzania. The study conducted to eight software development organizations in Tanzania engaged with software development. The studies identified activities conducted during software development and describe verification and validation testing tools and techniques. This study is limited to research on the software verification and validation state of practices and their activities in software development organizations.

CHAPTER TWO
LITERATURE REVIEW

2.1 Definition of Key Terms and Concepts

Different scholars and academicians have defined software verification and validation in various ways. For example**,** M. Altaie et al. (2020) have defined verification as a set of activities or procedures that insure correctness at each stage or phase in the software development process. Or it is the set of activities which compare software products related to the life cycle of the software or system against needed characteristics (Mousaei, 2020). While validation can be defined as the set of activities guaranteeing that the system has the capacity to accomplish its intended goal and use (meeting the requirements of stakeholders or customers) in the intended operational environment (Upadhyay 2012), software validation has been the process of evaluating software at the end of its development to ensure that it has been free from failures as well as complying with its requirements (Schumann & Goseva-Popstojanova, 2019) (Ullah Khan et al., 2015).

According to Upadhyay (2012), verification is about building the software product right ("Are we building the system right?") and its conformation to the specification, while validation is about building the right software product ("Are we building the right software product?"), and the software product should do what the user really requires. Verification should check whether the program meets its specifications as written in the requirements document. This may involve checking that it meets its functional and non-functional requirements. Validation ensures that the product meets the customer's expectations. This goes beyond checking that it meets its specifications; as we have seen, system specifications don’t always accurately reflect the real needs of users.

According to Markosian et al. (2011), verification and validation is a systematic program of review and testing activities performed throughout the development life cycle for digital system hardware and software. Verification is the process of evaluating a system or component during development. Validation is the process of evaluating a system or component at the end of the development process under conditions representative of its intended use.

2.2 Software Verification and Validation

The difference between the two terms is mainly based on the role of specifications. Validation is the process of checking whether the specification captures the customer's requirements, while verification is the process of checking that the software meets the specification (Anwar & Kar, 2019). Verification includes all the activities associated with producing high-quality software; these include testing, inspection, design analysis, specification analysis, and so on (Markosian et al., 2011; Upadhyay, 2012). It is a relatively objective process, in that if the various products and documents are expressed precisely enough, no subjective judgments should be needed in order to verify software (Vukovic et al., 2020).

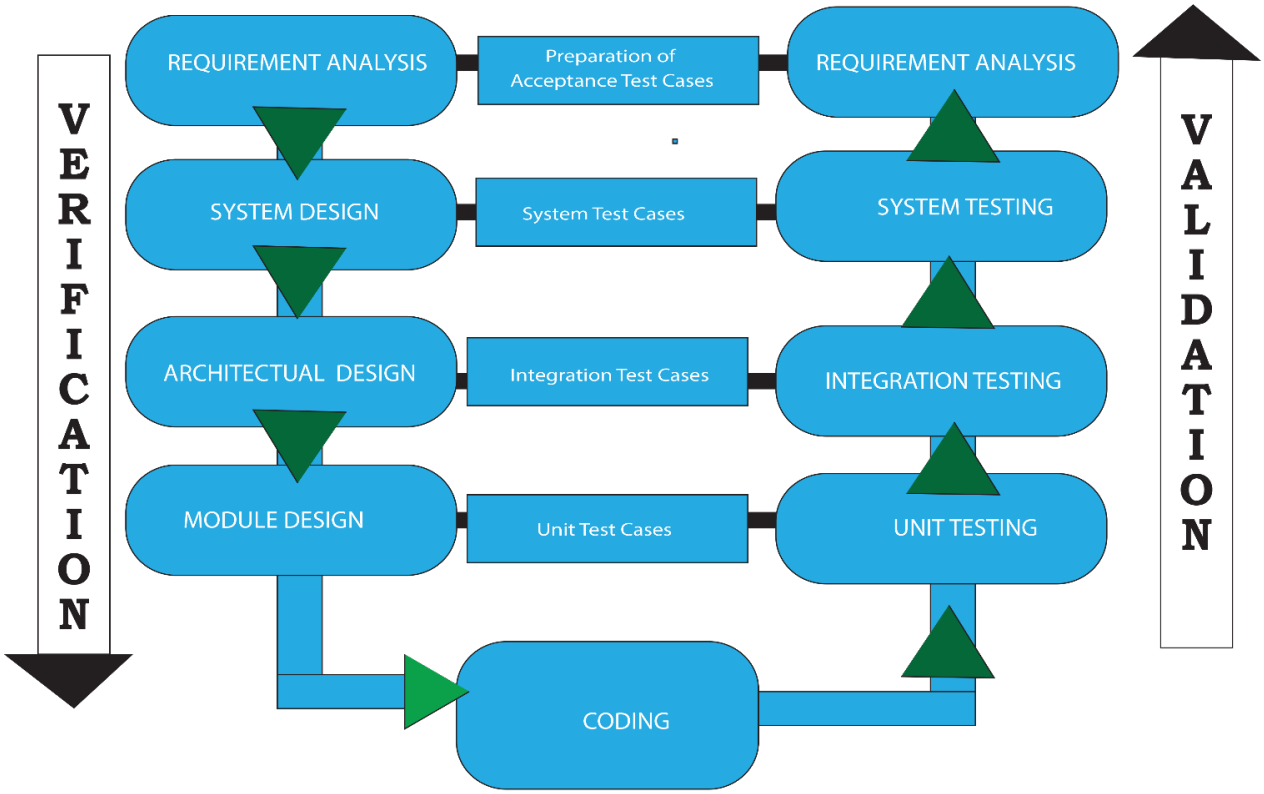
In contrast, validation is an extremely subjective process. It involves making subjective assessments of how well the proposed system or software addresses a real-world need (Anand & Uddin, 2019). Validation includes activities such as requirements modeling, prototyping, and user evaluation. In a traditional phased software lifecycle, verification is often taken to mean checking that the products of each stage or phase satisfy the requirements of the previous stage or phase.

Validation is relegated to just the start and end of the project: requirements analysis and acceptance testing. This view is common in many software development engineering textbooks and is misguided (Rajabli et al., 2021). It assumes that the customer's requirements can be captured completely at the start of a project and that those requirements will not change while the software is being developed. In practice, the requirements change throughout a project, partly in reaction to the project itself: the development of new software makes new things possible. Therefore, both validation and verification are needed throughout the lifecycle.

2.3 V-Model in Software Development Verification and Validation Activities

The V-Model used for software verification and validation activities is as follows: Business requirement analysis, system design, architectural design (high level design), module design (low level design), and coding phases are verification phases, while unit testing, integration testing, system testing, and acceptance testing are validation phases.

Figure 2.1: Describes the activities and procedures for software verification and validation, as well as how testing can be integrated into each phase of the software development process.



2.4 Time for Software Development

Empirical studies have shown that in many software development organizations, most of a project's time is spent on the verification and validation activities. (Andersson, 2003) (Beyer, 2022) discovered that software development organizations in their software development (Unterkalmsteiner, 2015)

Projects spent up to half of their projected time on verification and validation activities. Past studies have shown that different phases of the software development have their own verification and validation processes (M. Al Atitaie et al., 2020; Jan et al., 2016), so if you combine all phases from the designing stage until the product is in the market, nearly half of the time used in the development of the product is used for verification and validation activities.

2.5 Quality of the Product

Conformance to explicitly stated and agreed functional and non-functional requirements and specifications may be referred to as "quality" for the software product offered to prospective customers or end users (Limaye, 2009) (Babbar, 2017). Software quality interacts with each phase of every software development process (P. Yadav & Kumari, 2015). Planning should occur in the initial phase of a software development project and address the methods and techniques to be used in each phase (Malviya, 2019) (Abbas, 2018). A description of each product should be defined in order to provide a basis for objectively identifying satisfactory completion of the phase (Upadhyay, 2012; Brown, 1987).

The software quality management processes must address how well software products will, or do, satisfy customer requirements, provide value to the customers, and provide the software quality needed to meet software requirements and specifications (Jamil et al., 2017). Some of the specific Software Quality Management processes are defined in the standard (IEEE 12207.0-96), which includes the following aspects: Quality Assurance, Verification Process, Validation, Review, and Audit Process (Y. Gupta, 1989) (P. Yadav & Kumari, 2015)

Software quality assurance processes provide assurance that the software products and processes in the software life cycle conform to their specified requirements by planning, enacting, and performing a set of activities to provide adequate confidence that quality is being built into the software (Seth et al., 2014; Peddireddy & Nidamanuri, 2021). Software quality assurance seeks to maintain product quality throughout the development and maintenance of the product through the execution of a variety of activities at each stage of product development, which can result in the early identification of problems, an almost inevitable feature of any complex activity (P. Yadav & Kumari, 2015) (Edvardsson, 2006). Verification and validation address software product quality directly and use testing techniques that can locate defects so that they can be addressed (Mendoza, Souza, et al., 2019; Peddireddy & Nidamanuri, 2021). It does, however, evaluate intermediate products and, in this capacity, intermediate steps in the software life cycle processes.

2.6 Empirical Review

2.6.1 Empirical Studies Conducted in Various Countries and Scales.

Several studies have been conducted to identify the software practices and challenges of software verification and validation in several countries. However, no such study on software verification and validation practices in Tanzania has ever been conducted; some of the studies and their findings include Bäckström (2022), which investigated software testing practices in Finland, and Latif and Rana (2020), which investigated software testing techniques in Pakistan. Spanish researchers (Fernández-Sanz et al., 2005; Fernández-Sanz et al., 2009), Swedish researchers (Runeson, 2006; Grindal et al., 2006; Engström and Runeson, 2010), Korean researchers (Park et al., 2008; Yli-Huumo et al., 2014), Sri Lankan researchers (Vasanthapriyan, 2018), Geras etBhui From 2004 to 2017, a set of replications surveying testing practices in Canada was conducted (Geras et al., 2004; Garousi and Varma, 2010; Garousi and Zhi, 2013; Garousi et al., 2017); and from 2006 to 2018, some studies of software testing practices in South America were conducted (Dias-Neto et al., 2006; De Greca et al., 2015; Dias-Neto et al., 2017According to the findings of the software testing practices and challenges, industrial and academic focus areas are disjointed.While academicians are more interested in theoretically difficult challenges, test engineers in practice are looking for ways to increase testing efficacy and efficiency (Garousi et al., 2020). There is a gap between the state of the art and the state of the practice of software testing (Quesada-López et al., 2019). The data supports the idea that organizations mostly employ ad hoc criteria to decide when to end software testing (Quesada-López et al., 2019). Software testing is still a comparatively small part of software development, even though it occasionally occurs concurrently with development activities. There is a need for practical and efficient techniques to conduct tool evaluations that offer software practitioners reliable empirical evidence (Raulamo-Jurvanen et al., 2019), and many software development companies lack a dedicated software testing team. There are more developers than testers (Rahim et al., 2017; M. et al., 2018). This finding could point to a lack of maturity or financial resources to support a dedicated testing team. More research is required to obtain a better understanding of the situation and establish more definitive, tool-specific evidence. For beta testing, many pass on the cost of the test to the clients. This suggests conducting further research to determine, with supporting experimental evidence, the contexts in which the use of best practices and test automation results in a lower cost (Carlos & Ibrahim, 2021).

Software verification and validation are usually less formal because of the tough practices and methodologies of software testing. Software testing is a collaborative project in which each individual must play their part in producing bug-free software (Polamreddy & Irtaza, 2012). Though testers aim to produce 100% bug-free software, there will be defects found during maintenance. Efforts should be made to remove bugs and produce quality software within the specified time and cost (Nadu & Nadu, 2019).

Over two-thirds of the people polled use test levels. In decreasing order, functional testing, regression testing, performance testing, and usability testing are the most commonly used test types, with security testing being used less frequently (Bäckström, 2022). Unit testing methodologies and practices appear to be unaffected by test management or quality management. Although unit tests are structural or "white box" in nature, developers rarely assess their completion in terms of structural coverage. The majority of the businesses polled wanted to automate their unit tests, but they were having problems distributing best practices across their organizations (Andersson & Runeson, 2014). The use of automation in testing has increased. Automation has become more popular at all levels of testing in recent years. Second, the use of formal software process models and capability maturity models appears to have decreased, while testing tools have increased in use and effectiveness. This change is reflected in the organizational considerations around testing tools: the tools no longer limit the organizational unit as much as they did in 2009, but configuration issues and a lack of platform support have become more common in exchange (Hynninen et al., 2018). The majority of testing procedures and methods haven't changed all that much in 20 years. In addition to using the right procedures, effective testing calls for a tester's creativity and experience. Testing entails more than just fixing bugs. Testing serves other purposes besides identifying and fixing flaws. The measurement of reliability, validation, and verification are also done using it. Testing is costly. Saving money and time can be accomplished through automation (Regulwar & Gulhane, 2010). The greatest problems are incomplete and ambiguous requirements, a lack of knowledge about the specific system, communication gaps, and unclear requirements that could cause more problems for the project. Improving collaboration between the teams, clear communication, and interaction are important solutions to most software development problems. A requirements walkthrough and inspection are needed (Poudel, 2018).

Functional testing is the most important type of testing in organizations (Strazdi & Arnicane, 2018). This is reasonable, considering that without functionality, all other non-functional parts of a system become meaningless. Functional testing is followed by user acceptance testing, which is becoming increasingly important due to the growing importance of users (ISTQB, 2018). Agile methodologies are becoming increasingly popular, emphasizing the necessity for suitable testing processes and techniques as well as testing competency certification. (1) In over 80% of cases, in-house test teams are in charge of software testing. The fact that the test team does not report to development in the majority of cases (84%) confirms the adoption of role segregation. (2). It is becoming more common to use testing tools for defect tracking, test execution, test automation, test management, performance testing, and test design. With a 72 percent usage rate, test automation has become fairly common in the market. 40% of According to the responders, the percentage of automated test cases in use is far above 20%. (3) Test tool and automation consulting is another popular external service, with test automation being the area with the largest room for improvement (ISTQB, 2016). The development of software quality is an information-intensive process that is influenced by organizational structures and information flow within firms. The project manager acts as a mediator between the development teams and the clients. Their choices could enhance or degrade software quality and productivity (Seth et al., 2014). The object-oriented paradigm has gained widespread popularity, coupled with software dependability. It is important that high software quality not be compromised. Object-oriented design metrics should always be used to assess software quality during software development (Isong & Ekabua, 2015). Systematic test case design and data definition are standard techniques. Systematic test case design involves using a system for identifying test cases in order to reduce the number of test cases required without sacrificing test efficacy (Kassab et al., 2017). There is low utilization of software testing techniques and tools, challenges caused by a lack of software testing methods and tools, limited use of testing tools, a desire for interoperability support between software development and testing techniques and tools, as well as instruction on how to evaluate software testing methods and tools and characterize their capabilities (Lee et al., 2012). IT enterprises are tiny and inexperienced in applying a standard testing technique (Latif & Rana, 2020).

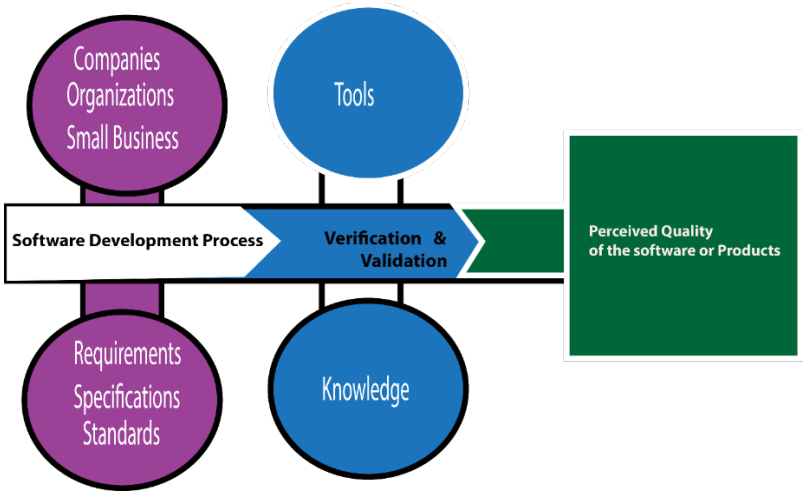
2.7 Theoretical Framework

Software verification and validation is the process of not only finding the errors, but also revealing at what level the quality has been achieved (Rajabli et al., 2021). Hence, software verification and validation provide information about defects and problems in software or products and simultaneously evaluate the achieved quality. Much effort is devoted to the improvement of areas such as analysis and requirements, design, and code reviews. However, in the spirit of continuous improvement in software quality, there is not much effort to improve its testing techniques to reduce customer-found defects.

2.8 Conceptual Frame Work

In this study, the conceptual frame work has been developed to describe the relationship between various phases of software development project. In the first part of the conceptual framework we have companies, organizations and small scale business enterprise which are engaged with software development. We also have requirements, specification and standard that exist in the market. The second section describe software development process where verification and validation is done in every stage of the software development process. In the third part we have testing tools, technique, testing knowledge skills and experience which will lead to quality of the software or product. Figure below illustrate the conceptual frame of the study.

Figure 2.2: Conceptual Framework



CHAPTER THREE
METHOD AND MATERIALS

3.1 Study Area

This study was conducted in four regions in Tanzania, namely Dar es Salaam, Mwanza, Arusha, and Dodoma. These areas have been purposefully chosen, as they are where most software development organizations are located.

3. 2 Research Methodology

Empirical research in the area of computer science has evolved over the last decades (Raulamo-Jurvanen, 2020). Empirical studies in computer science have become a key approach for researchers who want to understand, evaluate, and develop methods in the field (Persson, 2019). An empirical study is basically a systematic observation that lets the researcher gain quantitative or qualitative evidence concerning the object under study (Seuring et al., 2021) (Kiger & Varpio, 2020). Thus, the research allows for the confirmation of theories and hypotheses based on measurable observations rather than belief (Casteel & Bridier, 2021).

Even though the field has evolved, empirical computer science research has been criticized for being immature. However, recent guidelines for evaluating situations, methods, and techniques are proposed in several directions (Neri de Souza et al., 2016; Person, 2019). The research methodology is based on the same principles that can be used in other areas, like social, medical, and psychological research, but when criticized, it is compared to these more mature research fields. The software engineering field is relatively young when compared to these research fields (Eungoo & Hwang, 2021).

3.3 Methodological Approach

The research presented in this thesis employed two approaches: fixed and flexible research designs (Vaismoradi & Snelgrove, 2019). In fixed-design research, consider doing a large amount of pre-specification about what to do and how to do it before getting into the main part of the research study (Snyder, 2019). The intention of the approach is that we need to know what to do and collect all the data before starting to analyze it (Patel & Patel, 2019). Considering the use of fixed design on quantitative data and statistical analysis in contrast to flexible design, which is also referred to as qualitative design, the flexible design often results in qualitative data, which is typically non-numerical, and much less pre-specification is used;

the design evolves as the research proceeds (Persson, 2019), and the data collection and analysis are intertwined (Ridder, 2017).

3.4 Research Design

The research was carried out in eight Tanzanian software development organizations using a multiple-case study design. According to Ababacar, Sy Diop, and Liu (2020), a research design is a specific procedure used in carrying out a research process. The multiple case study design used in this study was chosen because it facilitated the development of deeper insights and better exploration of multiple units of analysis (Ridder, 2017; Breink, 2018) through the empirical study of software verification and validation methods and tools. The researchers employed multiple case study designs to combine information from multiple units of analysis with multiple data sources to gain in-depth information on the improvement of the verification and validation practices for industry.

3.5 Surveys

A survey is considered a strategy with a fixed design (Patel & Patel, 2019), but it can also have a flexible design (Islamia, 2017) (Ponto, 2015). The main features of surveys are the collection of data from a relatively large number of individuals and the selection of representative samples of individuals (Lu & Abeysekera, 2020). Surveys are very common in other areas, like computer science and other fields. The main features of surveys are the collection of data from a relatively large number of individuals and the selection of representative samples of individuals (Koshti, 2013). Surveys are very common in other areas, like computer science and other fields (Bryant, 2006). It is not as easy to control variables that influence the studied field in a survey as they are in other investigation methods. Interviews and questionnaires are important tools for gathering data in a survey (Lu & Abeysekera, 2020). The results are analyzed to be generalized according to the specified sample size of the population. Considering that any survey results obtained in one software development organization are difficult to find in other software development organizations

The survey method described by Ponto (2015) was used as the research method in this study. The objective of a survey method is to collect information from people about their feelings and beliefs. Furthermore, when information should come directly from the people, a survey is most appropriate (Snyder, 2019; Mathiyazhagan, T., 2010). B. Kitchenham et al. (2009) divide comparable survey studies into exploratory studies, from which explanations and

estimates can be drawn, and confirmatory studies, from which strong conclusions can be drawn.

3.6 Literature Survey

The main literature sources are research databases with connected search engines used to collect secondary data about the research presented in this thesis. A literature study aims to map out the current publications relevant to software verification and validation practices with the help of keywords and prior knowledge. A literature review concentrated on peer-reviewed sources to maintain relevance and quality as the input for additional information processing (Torres-Carrion et al., 2018). A literature survey was conducted in accordance with Seuring et al. (2020) and Torres-Carrion et al. (2018) standards in order to establish a theoretical basis for software verification and validation practices.

3.7 Data Collection and Analysis

Thematic analysis was used to analyze the data obtained through interviews and questionnaires (Nowell et al., 2017). As a result, themes generated from the collected data will be presented and discussed in order to improve the state of practice in software verification. Thus, the process involved familiarizing with the data, generating initial codes, searching for themes, reviewing the themes, defining and naming themes, and producing a report (Vaismoradi & Snelgrove, 2019; Marttinen et al., 2020) (Kiger & Varpio, 2020).

Without good analysis and interpretation of the collected data and information for all activities of this prospective study, the essence of the data will not be revealed, nor will it be possible to communicate. However, there might not be a clear point in time when the data collection ends and analysis begins (Ciesielska & Jemielniak, 2017). With a flexible design, the data collection and analysis are overlapping, which may also result in a higher quality of both the collected material and the analysis (Centers for Disease Control and Prevention, 2018). When not focusing on confirming predefined solutions and initial interpretations, the overlapping may give new insights, but alternative explanations will not be revealed, nor will it be possible to communicate (Zevalkink, 2021). However, there might not be a clear point in time when the data collection ends and analysis begins. With a flexible design, the data collection and analysis are overlapping, which may also result in a higher quality of both the collected material and the analysis (Kawulich, 2005). When not focusing on confirming predefined solutions and initial interpretations, the overlapping may give new insights and

alternative explanations (Islamia, 2017). In a study with a fixed design, however, the analysis occurs after all of the data has been collected (Zevalkink, 2021). The analysis of quantitative data can range from being simply organized to being exposed to some complex statistical analysis. However, qualitative data should also be systematically analyzed.

3.7.1 Questionnaire

This study used questionnaires to collect qualitative data about concerns, ideas, and assessments. Utilizing questionnaires helped quantify certain qualitative data. Questions were permitted on questionnaires. Abawi (2017) explains that questionnaires are sets of questions that are sent to the specified respondents, who answer them and return the filled questionnaires to the researcher. Over 100 people were able to provide feedback.

3.7.2 Results Observations:

Observations for data collection were employed and used in the exploratory method to observe what was going on in a certain situation and watch the actions and behaviors of people as they responded (Ciesielska & Jemielniak, 2017). What has been observed is then described, analyzed, and interpreted. Much research in life science involves direct or indirect observations of humans, but for example, experiments in computer science represent a kind of controlled and well-accountable observation (Zevalkink, 2021).

The observation method employed in this thesis is mainly of the type of participatory observation (Ciesielska & Jemielniak, 2017), where the observer participates in the group under study.

3.7.3 Interviews:

The study employed the interviewing method to obtain more qualitative data from the participants. Semi-structured interviews are the research method used in this study (European Commission, Eurostat, 2017). Although the interviewee will receive the questions in advance, there will also be time for discussion, clarification, and follow-up questions.

Considering that the advantage of data collection through interviews is the flexibility of the data and participants, the study's researchers have the option of following up on the ideas provided, and using the interview method, it is simple to interpret feelings (*Quantitative Research Methods*, n.d.). All answers given in a questionnaire must be interpreted on their own, while in an interview, attendant questions can be given and the answers thereby catch

subtler information. On the other hand, interviews are rather time-consuming (Kiger & Varpio, 2020). There are several necessary activities that should be conducted: the preparation, the execution, and the processing of the data. It is also a subjective technique, with a risk of bias both from the interviewer's and the interviewee's point of view.

It was observed that the more standardized the interview is, the easier it is to process the data. The fully structured interview is similar to a questionnaire. The semi-structured interview has predetermined questions (Ridder, 2017), but the interviewer can change the order as well as the wording of the questions, and explanations can be given (Mohajan, 2018) (Dawadi & Giri, 2021). The unstructured interview frequently has a topic about which the interviewer asks open-ended questions.

The study employed the interviewing method to obtain more qualitative data from the participants. (European Commission, Eurostat, 2017). Although the interviewee received the questions in advance, there was also time for discussion, clarification, and follow-up questions.

In order to address the study objective, the following theme and sub-themes were used and covered during interviews:

Interview Theme: Improving Software Verification and Validation Practices in Software Development Organizations

Among the sub-themes are:

i. Software verification and validation methodologies and techniques: methodology, barriers, benefits, and expected improvement areas
ii. Verification and validation tools: tools used, tool objectives, and automated tool issues
iii. Verification and validation standards: internal standards as well as implementation issues
iv. Challenges of software verification and validation in software development organizations
v. Software verification and validation processes and metrics

3.7.4 Experiment

Experimentation within software verification and validation practices is conducted. Experimentation addresses qualitative research questions (Neri de Souza et al., 2016). Experimentation was not used in its full implementation. However, experimentation will be within the scope of the study.

In this thesis, experiments presented compare software design review and software code review as verification and validation techniques for software process improvement. And for improving the quality of the software products, the main intention of using the experiment method was to establish a means of obtaining results on the applicability of software verification and validation practices in software development organizations.

3.8 Sampling Techniques

Purposive sampling was used to recruit managers, team leads, development team members, and software quality assurance team members for the study. Purposive sampling (also known as judgmental, selective, or subjective sampling) is a sampling approach in which samples are selected based on pre-determined criteria (B. Yadav & Sharma, 2017) (N. K. Gupta, 2020). Respondents were managers, team leads, members of the development team, and members of the software quality assurance team from eight software development organizations in Tanzania.

3.9 Study Population and Sample Size

The term "population" refers to the entire set of cases from which the sample size is drawn; thus, the sample size refers to a subset of the population (Casteel & Bridier, 2021; Campbell et al., 2020; Yang et al., 2012). For this study, the population of practitioners applying verification and validation practices from software development organizations was sampled; a total of 100 respondents were selected. A sample size of 100 respondents from software development organizations was selected because it falls within an acceptable range as per the requirements of scientific research, whereby a sample size is not required to be below 30 respondents. According to Rusu Mocănaşu (2020), sample sizes larger than 30 and less than 500 are appropriate for most research, and in multivariate research, the sample size should be several times (preferably 10 times or more) as large as the number of variables in the study.

3.10 Data Validity and Reliability Test

According to Noble and Smith (2015), reliability and validity are concepts used to evaluate the quality of research since they indicate how well a method, technique, or test measures something. Reliability is about the consistency of a measure, and validity is about the accuracy of a measure (Noble & Smith, 2015). The study ensured optimum validity of the tool used in the study by giving the initial pool of interview instruments to three academic experts in the areas of computer science, software development, and software engineering to ensure that the tool has both face and content validity that are sufficient for further analysis to come up with the study's objectives. Moreover, reliability was assured by collecting only information related to the selected software development organizations.

3.11 Research Ethics Considerations

According to Eungoo & Hwang (2021), research ethics refers to the appropriateness of researcher behavior in light of the rights of those who become the subject of the study. The study was considered ethical throughout the period of the study by ensuring that all information provided by respondents remained confidential and that none of the respondents was forced to participate in the study (Imenda, 2014). Hence, participation was voluntary. Furthermore, during the study, no respondent was required to provide his or her name in the questioner.Moreover, ethical standards will be adhered to by observing all background information provided by other studies appropriately.

This study's final thesis was submitted to the Selinus University of Science and Literature Ethical Committee Board for ethical clearance. Permission to conduct the study was sought from the respective software development organizations in Tanzania in particular, following a thorough explanation of the aim and benefit of the study.

CHAPTER FOUR
RESULTS AND ANALYSIS

4.1 Introduction

Findings and results from a research question on how software development organizations in Tanzania perform their verification and validation activities were summarized. This research presents a survey of the state of practice in eight Tanzanian development organizations. The survey was conducted to improve verification and validation practices in the software development organization. The data were collected through interviews in the software development departments at the participating organizations and thereafter assessed and analyzed.

In the survey, it was concluded that software development organizations invested in documented verification and validation activities. These organizations relied more on experienced employees than on documentation.

The development among the surveyed organizations was either incremental or Increments were used among more process-focused organizations, while daily builds were more frequently utilized in less process-focused organizations. Organizations can begin testing early, during the first developed increments with limited functionality, by using incremental development or daily builds. Thus, the cycle time for a release will be reduced since it allows testing to run in parallel with development.

Among the surveyed organizations, no specific approach to process improvement could be identified. The approach taken depended on the persons involved, their backgrounds, and their experiences. Test automation and test management were regarded as areas for improvement by several organizations. Handling the legacy parts of the product and related documentation presented a common problem in improvement efforts for product evolution. The test automation was carried out using scripts for products with a functional focus and recorded data for products without a functional focus.

4.2 Findings and Results for the Surveyed Organizations

RQ1. How do software organizations perform verification and validation?

RQ: how does your company improve the competency level of your software verification and validation?

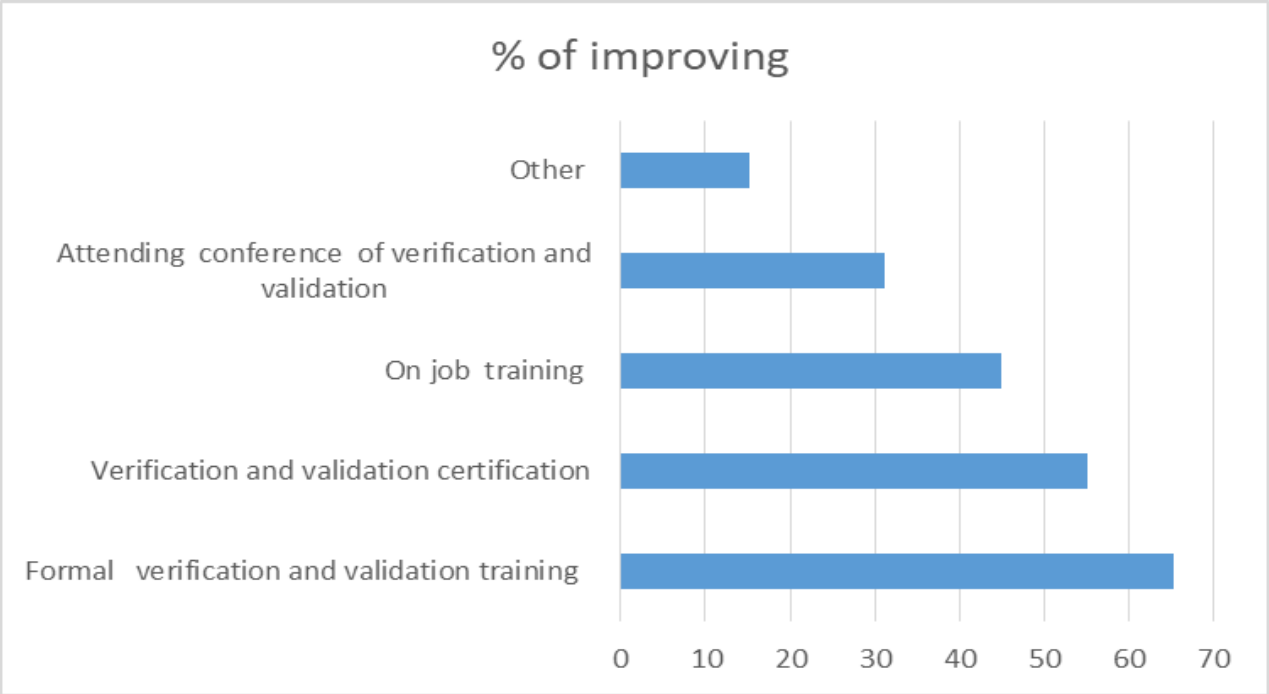
Results concluded that Formal training (65.2%) and certification (55.2%) rated highly as approaches to improving the competency of testers after on the job

Training (47.9%).

Table 4.1: Company improve the competency level of your software verification and validation

Improving level of software verification and validation	% of improving
Formal verification and validation training	65.2
Verification and validation certification	55.2
On job training	45
Attending conference of verification and validation	31.2
Other	15.2

Figure 4.1: company improve the competency level of your software verification and validation



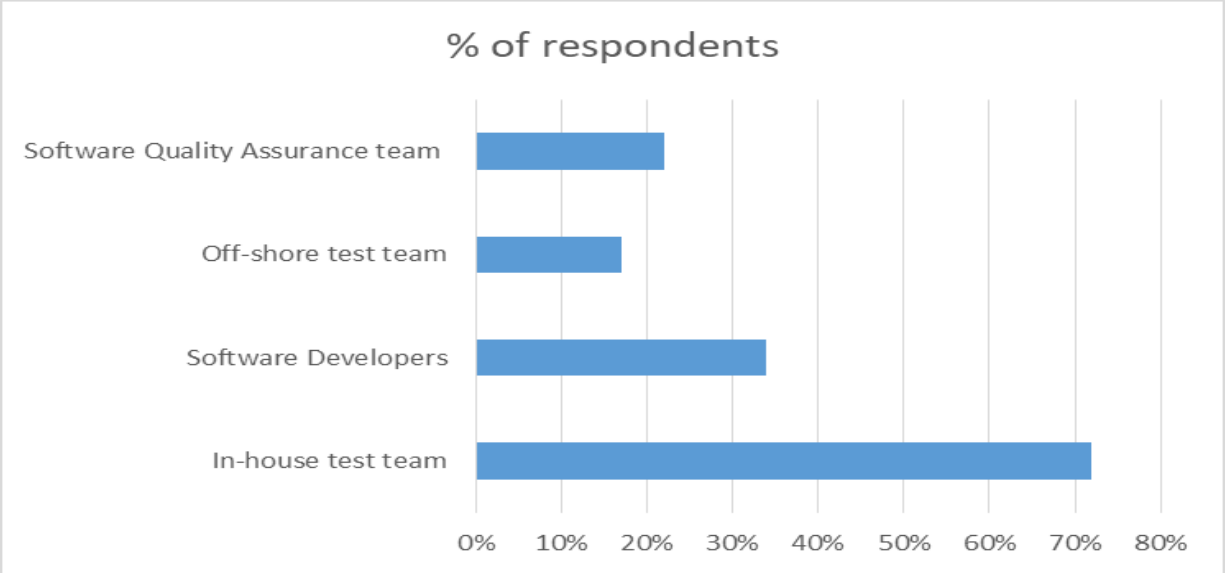
RQ: In your company, who is in charge of verification and validation?

The results from the surveyed software development organization indicated that respondents indicated that among their organization's employees, a majority are assigning their testing to an in-house test team (72%), and 34% of all respondents are using software developers.

Table 4.2: Personnel in charge of software verification and validation activities in software development

S/N	Software testing responsible Personel	% respondents
1	In-house test team	72
2	Software Developers	34
3	Off-shore test team	17
4	Software Quality Assurance team	22

Figure 4.2: Summarize Personnel in charge of software verification and validation activities in software development organizations in Tanzania



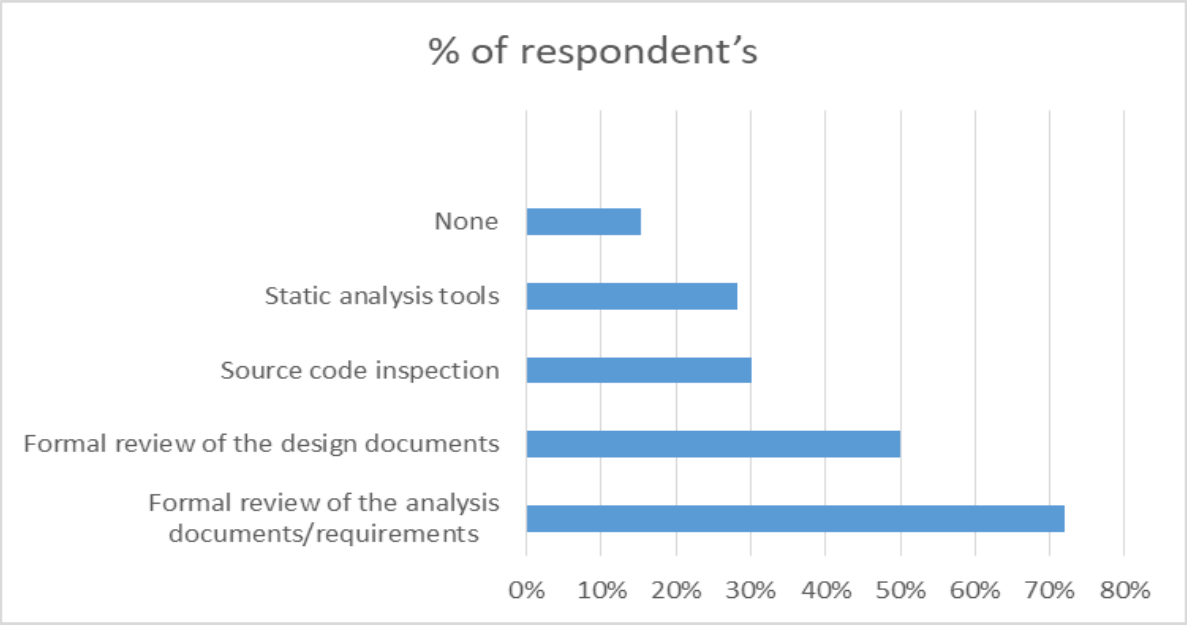
RQ: Which verification and validation methods are used? What activities do you use to find defects before test executions?

Results from respondents from eight Tanzanian software development organizations indicated that document analysis and requirements review are the most common activities for early defect detection.

Table 4.3: Verification and Validation methods and activities used to find defects before test executions

S/N	Activities	%Percentages
1	Formal review of the analysis documents/requirements	72
2	Formal review of the design documents	50
3	Source code inspection	30.2
4	Static analysis tools	28.2
5	None	15.2

Figure 4.3: Analysis Verification and validation methods and activities used to find defects before test executions



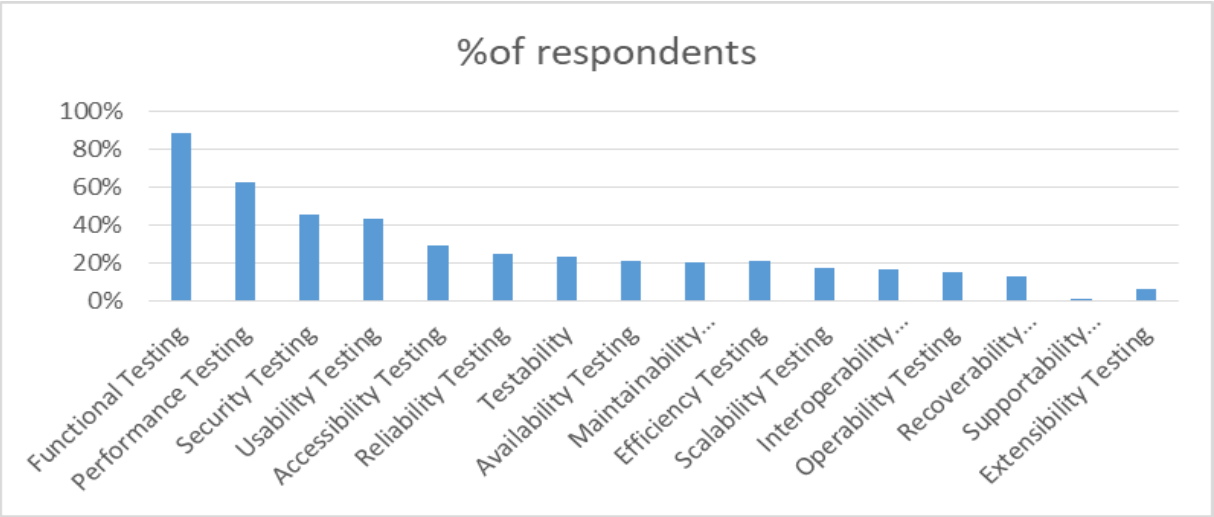
RQ: What types of testing are used in organizations?

According to the findings, respondents from eight Tanzanian software development organizations believe that functional testing (88%) is the most important type of testing, followed by user acceptance testing (70.0%).

Table 4.4: The types of testing used in Tanzanian software development organizations are listed here.

S/N	Testing Types	% of Respondents
1	Functional Testing	88
2	Performance Testing	62.7
3	Security Testing	45.6
4	Usability Testing	43.2
5	Accessibility Testing	29.2
6	Reliability Testing	24.4
7	Testability	23.5
8	Availability Testing	20.7
9	Maintainability Testing	20.2
10	Efficiency Testing	20.8
11	Scalability Testing	17.5
12	Interoperability Testing	16.4
13	Operability Testing	14.8
14	Portability Testing	13.1
15	Recoverability Testing	11.5
16	Supportability Testing	0.7
17	Extensibility Testing	6.2

Figure 4.4: In the software development organizations, identify the most important type of testing.



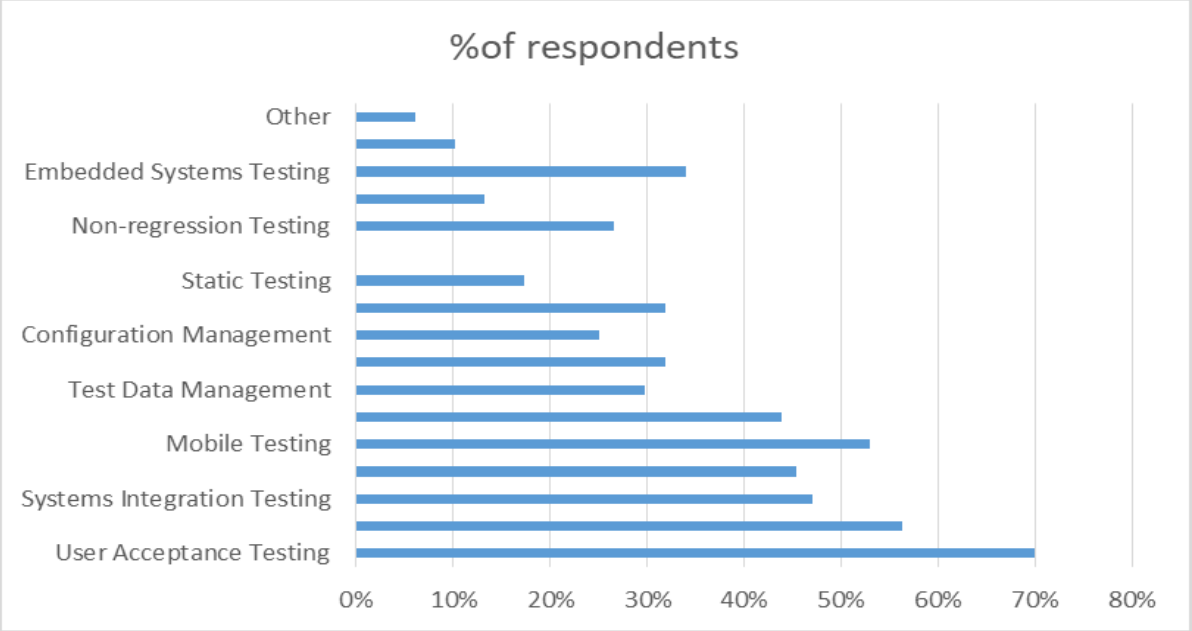
RQ. What verification and validation topics are important for your company?

The results of the surveyed software development organization indicate that respondents from eight software development organizations in Tanzania concluded that functional testing (88%) is the most important type of testing, followed by user acceptance testing (70.0%).

Table 4.5: Details of verification and validation topics used and implemented in software development organizations

Testing topics	% Respondent organizations
User Acceptance Testing	70
Exploratory Testing	56.3
Systems Integration Testing	47.1
Web Based Aplication Testing	45.3
Mobile Testing	53
Release Management	43.8
Test Data Management	29.7
Test Environments Management	31.9
Configuration Management	25
Test Metrics and Test Effort Estimation	27.5
Static Testing	29
Testing Systems of Systems	19.1
Non-regression Testing	26.5
Cloud Testing	17.3
Business Intelligence / Big Data Testing	15.6
Embedded Systems Testing	13.2
Internet of Things Testing	10.2
Other	6.1

Figure 4.5: Analysis verification and validation topics used and implemented in software development organizations



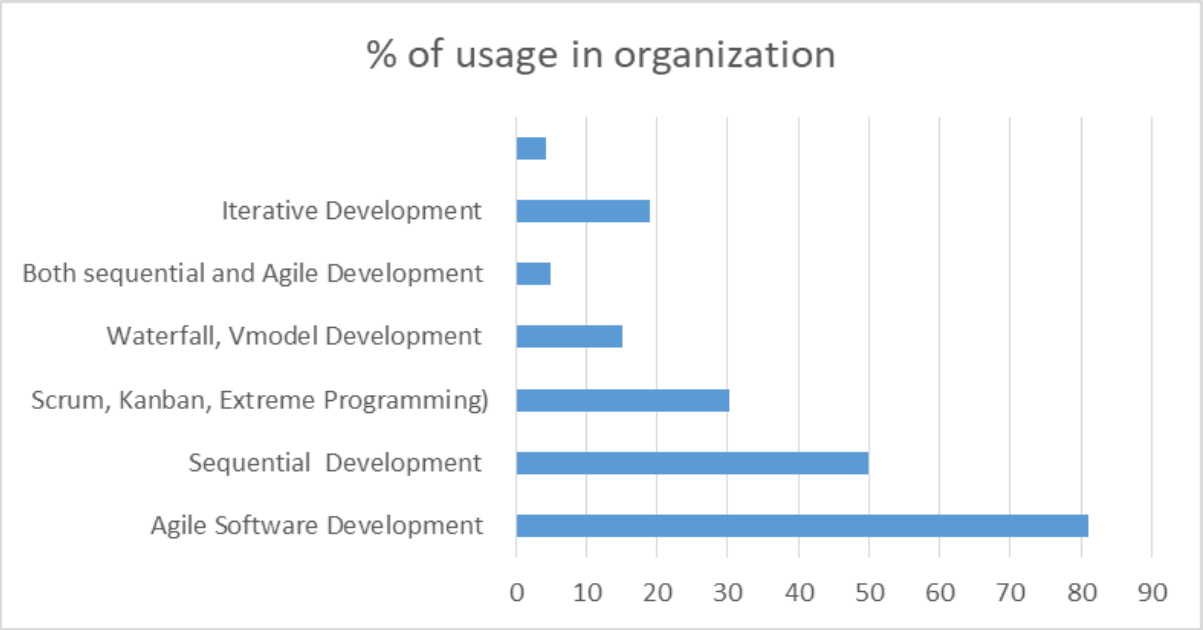
RQ: Which software development lifecycle (SDLC) model does your organization use?

According to the findings, 81% of organizations now use agile models. Agile and followed by the sequential methods in 50% of the respondents' organizations.

Table 4.6: Details of Software development lifecycle (SDLC) models analysis

SN	Software development lifecycle	% of usage in organization
1	Agile Software Development	81
2	Sequential Development	50
3	Scrum, Kanban, ExtremeProgramming)	30.2
4	Waterfall, Vmodel Development	15
5	Both sequential and Agile Development	5
6	Iterative Development	19
7	None	4.2

Figure 4.6: Software Development Lifecycle (SDLC) Models Analysis



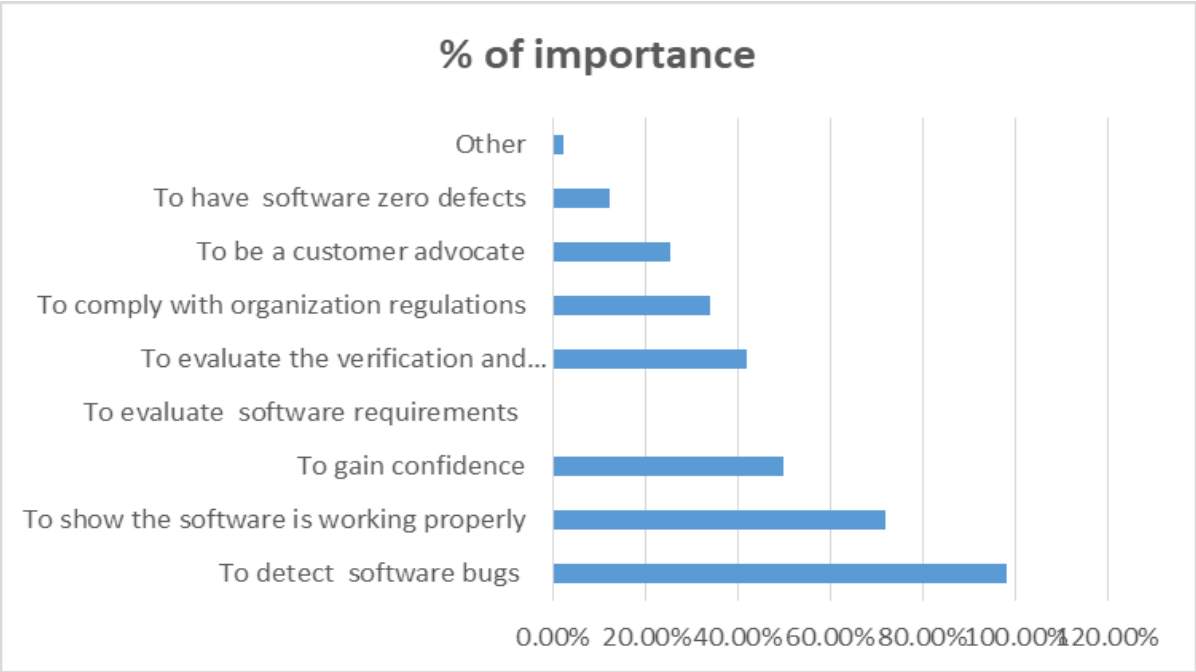
RQ: What are the main objectives of your verification and validation activities?

Results from respondents' surveys indicate that the most common verification and validation activity is "to detect bugs" (98.2%), followed by "to show the software is working properly" (77.2%).

Table 4.7: Verification and validation activities performed in organization

S/N	Activities	% of importance
1	To detect software bugs	98.2
2	To show the software is working properly	72
3	To gain confidence	50
4	To evaluate software requirements	48 5
5	To evaluate the verification and validation user experience	42
6	To comply with organization regulations	34
7	To be a customer advocate	25.2
8	To have software zero defects	12.3
9	Other	2.2

Figure 4.7: Analyzing objectives of verification and validation activities of surveyed organization



RQ: What are the main improvements? Areas of your verification and validation activities

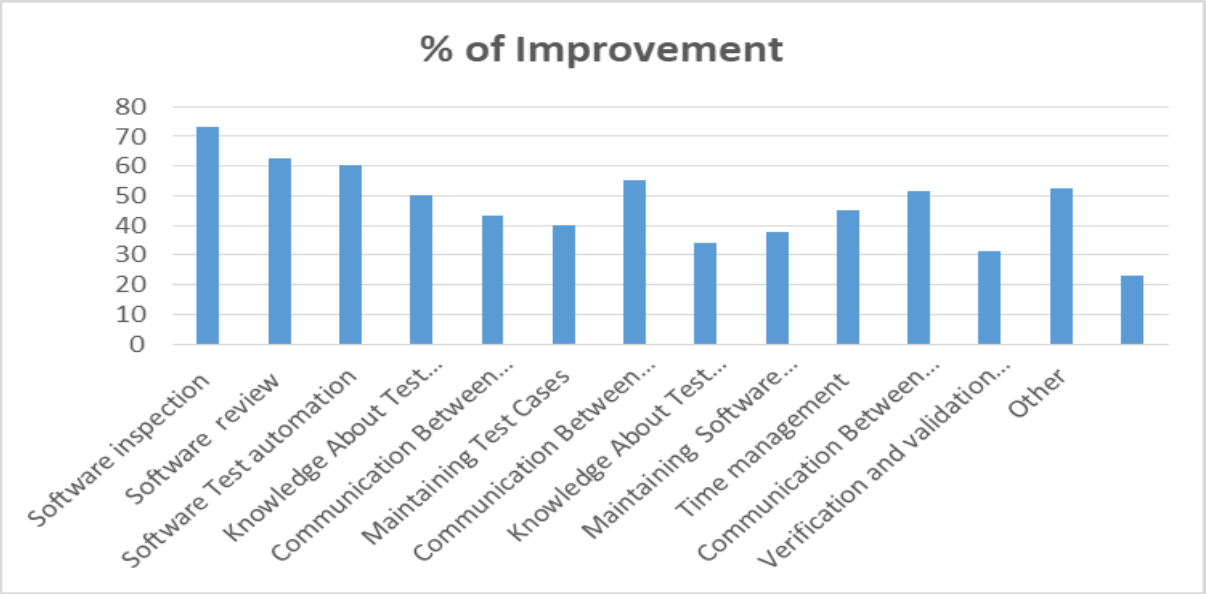
The results from the surveyed organization indicated that the three most important areas for improvement in verification and validation activities are software inspection (73.2%), software review (62.4%), and test automation (60.4%).

Table 4.8: Provide details of the main areas of improvement for verification and validation activities.

S/N	Activities	% of Improvement
1	Software inspection	73.2
2	Software review	62.4
3	Software Test automation	60.4
4	Knowledge About Test Processes	**50.1**
5	Communication Between Development and Testing	**43.2**
6	Maintaining Test Cases	**40.1**
7	Communication Between Business Analysis and Testing	**55.2**
8	Knowledge About Test Design Technique	**34.2**
9	Maintaining Software Test Scripts	**37.6**
10	Time management	**45.2**
11	Communication Between Project Management verification and validation	**51.7**
12	Software Test Data Preparation	**54.5**

13	Having Well Testing Trained Personnel	**31.2**
14	Verification and validation Budget	**52.6**
15	Other	**23.2**

Figure 4.8: **Indicates areas for improvement in software verification and validation activities for surveyed software development organizations.**



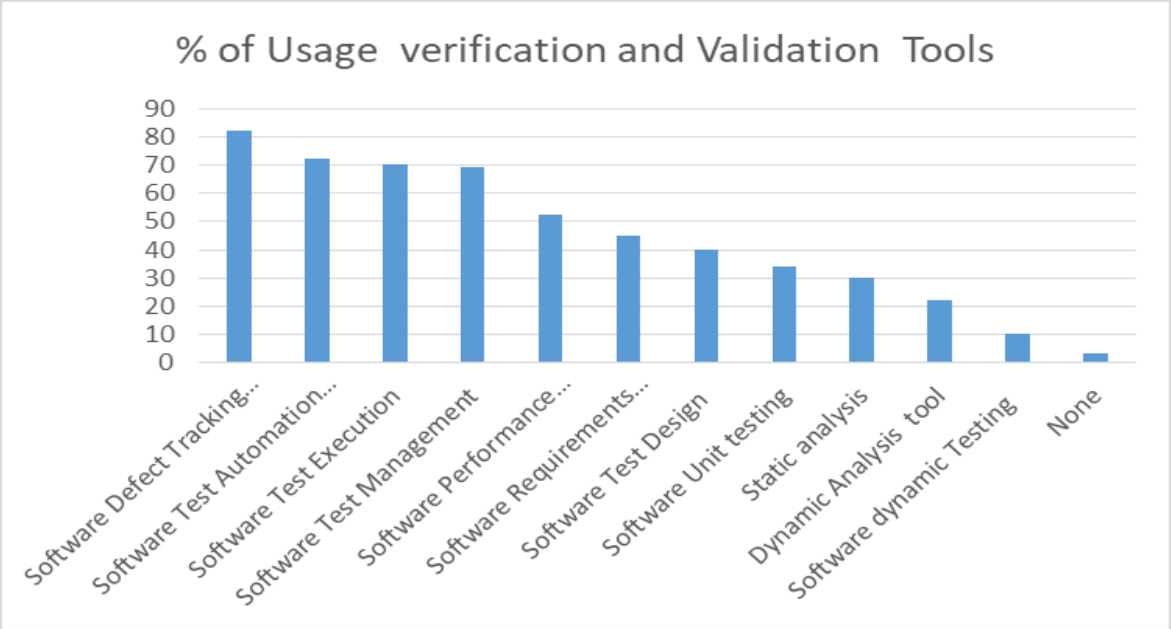
RQ: Which tools do you use it in your organization during verification and validation activities?

The results from the surveyed organizations indicated that the most commonly used tools among test teams are software defect tracking (82.2%), software test automation (72.2%), and software test execution (70.1%).

Table 4.9: Tools do you use it in your organization during verification and validation activities

S/N	Verification and Validation tools	% of Usage verification and Validation tools
1	Software Defect Tracking Tool	82.2
2	Software Test Automation Tool	72.2
3	Software Test Execution	70.1
4	Software Test Management	69.2
5	Software Performance Testing	52.2
6	Software Requirements Traceability	45.2
7	Software Test Design	40.1
8	Software Unit testing	34.2
9	Static analysis	30.1
10	Dynamic Analysis tool	22.2
11	Software dynamic Testing	10.2
	None	3.2

Figure 4.9: Analyze tools organizations use during verification and validation activities



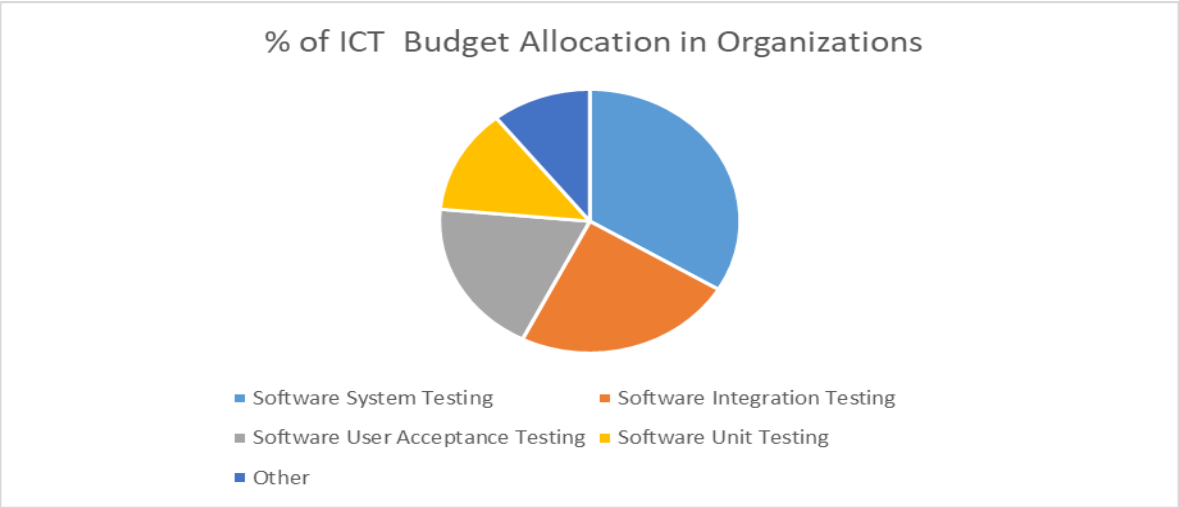
RQ: Which test levels or types (S) receive the majority of your ICT budget?

According to the results of a survey of software development organizations, system testing (71.2%) consumed the majority of the testing budget, followed by integration testing (50.2%), user acceptance testing (40.3%), and unit acceptance testing (27.2%).

Table 4.10: Budgeted Test Levels in the Software Development Organization

S/N	Testing Levels/Types	% of ICT Budget Allocation in Organizations
1	Software System Testing	71.2
2	Software Integration Testing	50.2
3	Software User Acceptance Testing	40.3
4	Software Unit Testing	27.2
5	Other	22.6

Figure 4.10: Analyze Software Test Level Budget



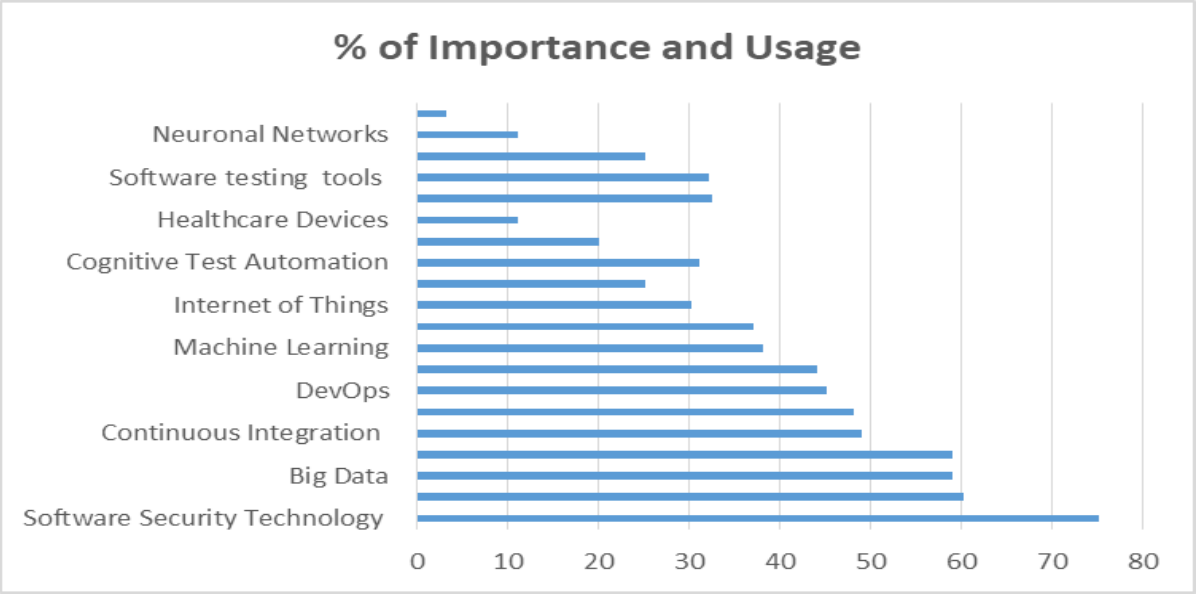
RQ: Which new technologies will be important to the software verification and validation organization in the following two years?

The results of the study indicated that, in the next two years, the most important subject will be software security testing (75.2% for the software verification and validation organizations).

Figure 4.11: New technologies that will important to the software verification and validation organization in the following two years

S/N	Intended Technology	% of Importance and Usage
1	Software Security Technology	75.2
2	Software Artificial Intelligence	60.2
3	Big Data	59.1
4	Systems Cloud	59.0
5	Continuous Integration	49.1
6	Continuous Testing	48.2
7	DevOps	45.2
8	Performance	44.2
9	Machine Learning	38.2
10	Database Management	37.2
11	Internet of Things	30.2
12	Usability	25.2
13	Cognitive Test Automation	31.1
14	Scalability	20.1
15	Healthcare Devices	11.2
16	Software inspection	32.5
17	Software testing tools	32.2
18	Requirement Engineering	25.2
19	Neuronal Networks	11.2
20	Other Technologies	3.2

Figure 4.12: Showcase Technology for Software Development Companies



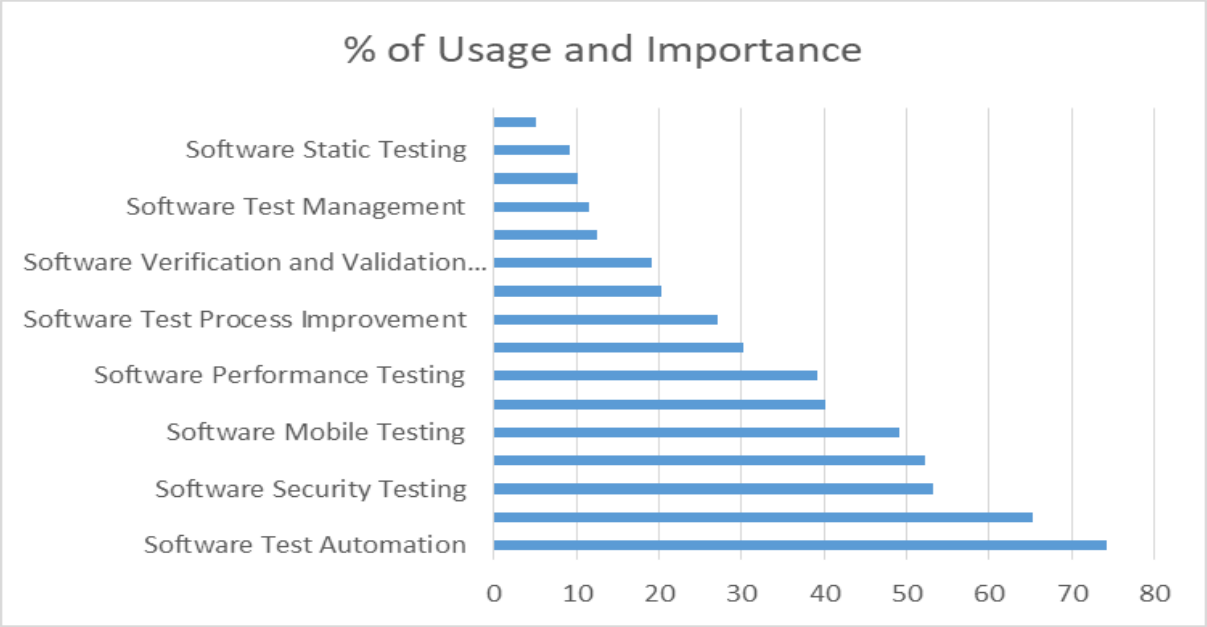
RQ: What will be the most trending topic for the software verification and validation profession in the near future?

The results of the survey indicate that software test automation is the most trending topic for the software testing profession in the near future and is also highlighted as the main improvement area in software verification and validation processes.

Figure 4.13: Indicate what will be the most trending topic for the software verification and validation profession in the near future?

S/N	Verification and validation Topics	% of Usage and Importance
1	Software Test Automation	74.2
2	Software Agile Testing	65.2
3	Software Security Testing	53.2
4	Software Cloud Testing	52.2
5	Software Mobile Testing	49.2
6	Software Continuous Testing	40.1
7	Software Performance Testing	39.2
8	Software Virtualization	30.2
9	Software Test Process Improvement	27.2
10	Software test Data Management	20.2
11	Software Verification and Validation Techniques	19.2
12	Software Usability Testing	12.5
13	Software Test Management	11.6
14	Software Model-based Testing	10.1
15	Software Static Testing	9.2
	Others	5.2

Figure 4. 14: Shows what will be the most trending topic for the software verification and validation profession in the near future.



RQ2 what are the challenges concerning the existing software verification and validation practices in Tanzania

Table 4.11: Software verification and validation challenges

S/N	Item	Challenge
1	Software Requirements Specification	Finding was concluded there is Unstable requirements
2	Verification and validation Testing environment	More observed Defects in testing environment and tools
3	Software Integration testing	Software testing team are Limited focus on integration testing of software components
4	Software Reviews and inspection	Inadequate internal software reviews and inspection
5	Software Unit testing	More focus on white box testing than black box testing
6	Software Independent V&V	Test cases are not reviewed by independent software engineers and software testers
7	Test management and test automation	Test management and test automation are the most challenging test activity types.

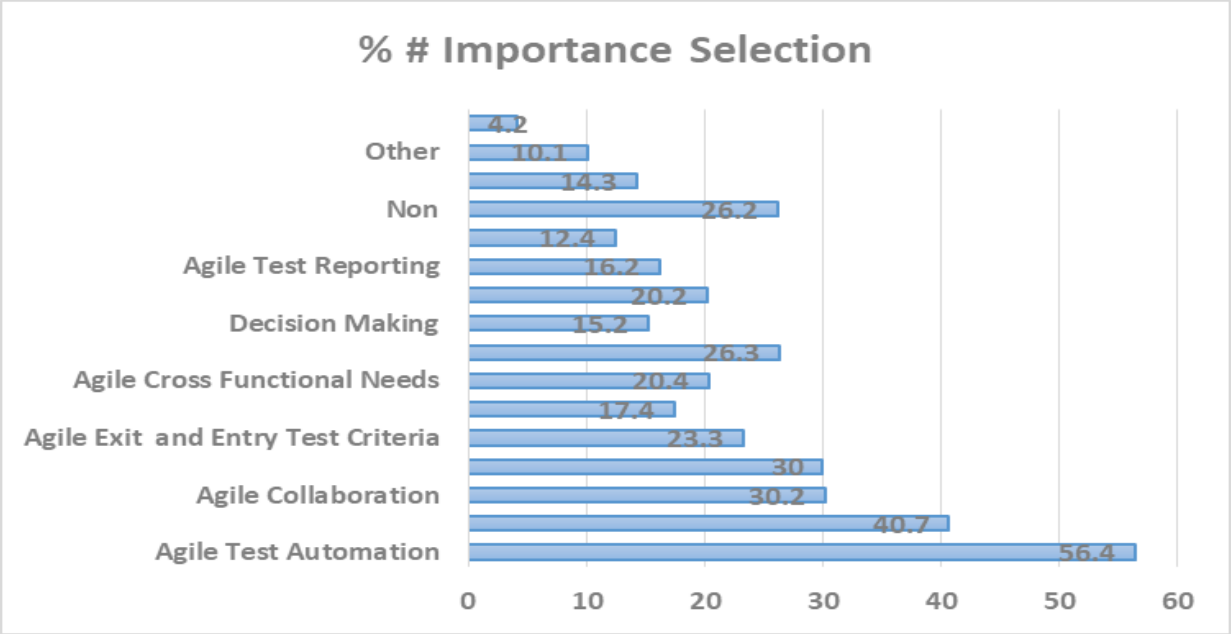
RQ: What are your most difficult verification and validation challenges in agile projects?

Results indicated that the top three testing challenges in agile projects are Agile Test Automation (56.4%), Agile Documentation Challenge (40.7%), and Collaboration (30.2%). Considering the results of the survey, the organization will be able to meet the software agile development challenges in light of these challenges.

Table 4.12: Verification and validation challenges in agile projects

S/N	Challenges in Agile Software Project	% of Importance's
	Agile Test Automation	56.4
	Agile Documentation	40.7
	Agile Collaboration	30.2
	Agile Test Effort Estimation	30
	Agile Exit and Entry Test Criteria	23.3
	Risk Awareness	17.4
	Agile Cross Functional Needs	20.4
	Software Quality Ownership	26.3
	Decision Making	15.2
	Software Traceability	20.2
	Agile Test Reporting	16.2
	Agile Legacy Defects	12.4
	Not Applicable	26.2
	Regulatory / Compliance Issues	14.3
	Other	4.2

Figure 4.15: Verification and validation challenges in agile projects



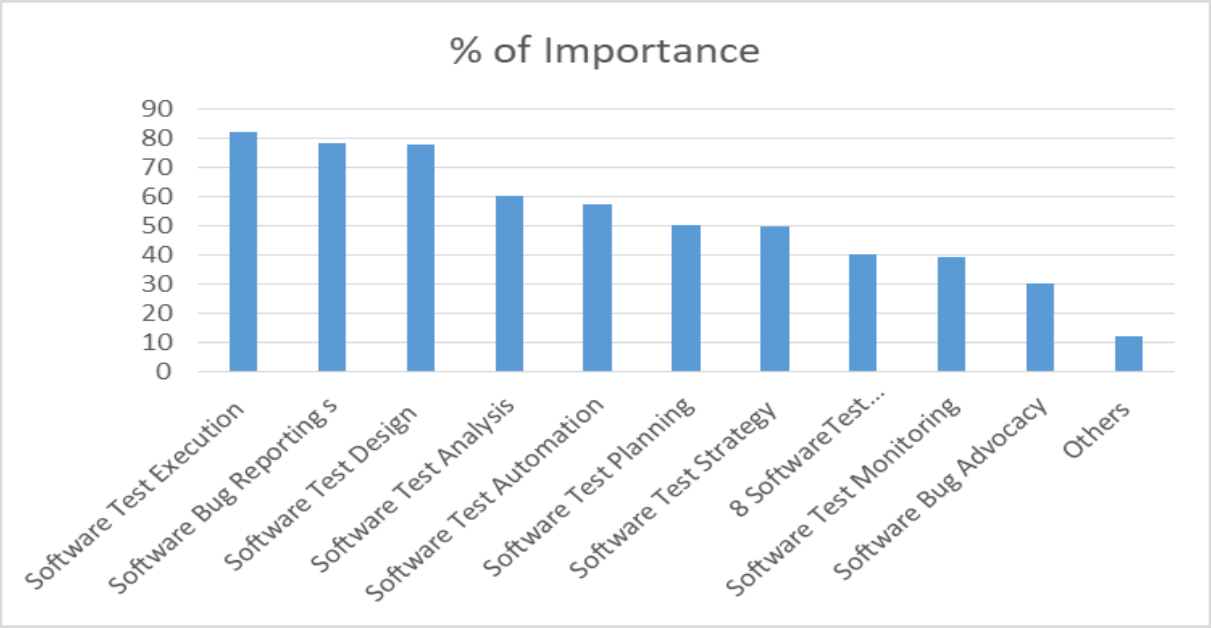
RQ: What expectations and skills do you have for the verification and validation team?

This result shows that a good software tester and verification and validation expert should have a good understanding of the verification and validation process, including software test execution (82.1%), software bug reporting (78.2%) and Software Test design (77.8%) are most important verification and validation skills

Table 4.13: Details of verification and validation skills

S/N	Verification and Validation Skills	% of Importance
1	Software Test Execution	82.1
2	Software Bug Reporting s	78.2
3	Software Test Design	77.8
4	Software Test Analysis	60.2
5	Software Test Automation	57.2
6	Software Test Planning	50.2
7	Software Test Strategy	49.8
8	Software Test Implementation	40.2
9	Software Test Monitoring	39.2
10	Software Bug Advocacy	30.2
11	Others	12.1

Figure 4.16: Analyze Details of verification and validation skills



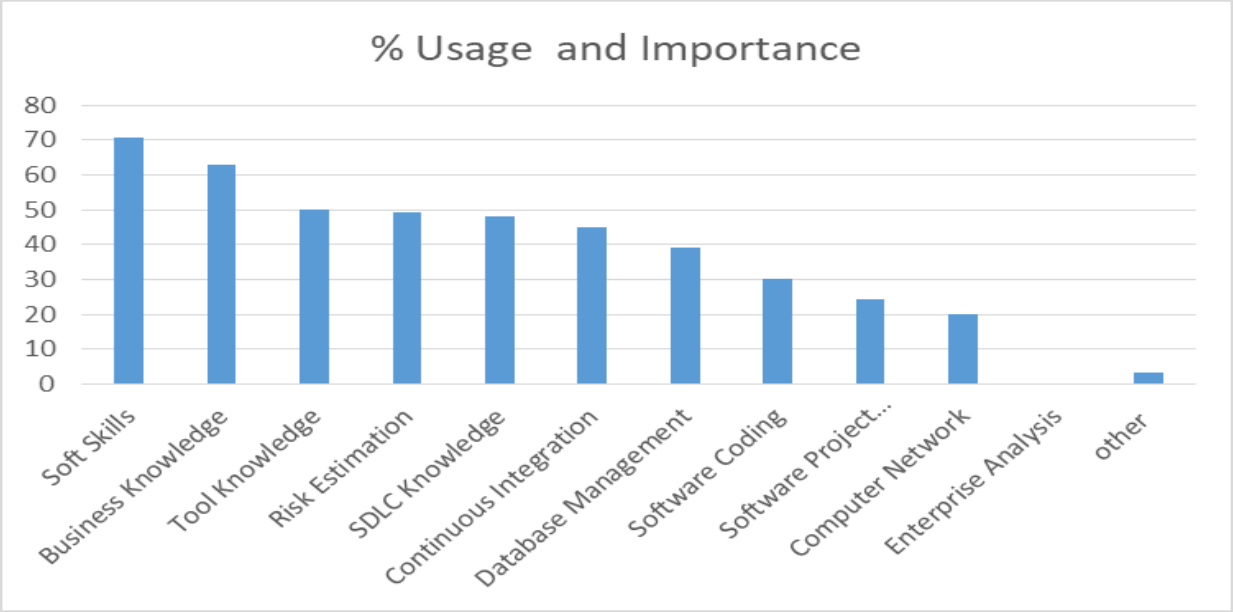
RQ: Which of the following non-software verification and validation skills are most expected from an agile tester in your organization?

Soft skills (70.8%) and business/domain knowledge (62.9%) are non-software verification and validation skills that are most expected from an agile tester, according to the results of the survey.

Table 4.14: Details of Non-software verification and validation skills are most expected from an agile tester

S/N	Non-software verification and validation skills are most expected from an agile tester	% Usage and Importance
1	Soft Skills	70.8
2	Business Knowledge	62.9
3	Tool Knowledge	50.1
4	Risk Estimation	49.2
5	SDLC Knowledge	48.2
6	Continuous Integration	45.2
7	Database Management	39.2
8	Software Coding	30.1
9	Software Project Management	24.2
10	Computer Network	20.2
11	Enterprise Analysis	18.2
13	Others	3.2

Figure 4.17: Non-software verification and validation skills are most expected from an agile tester



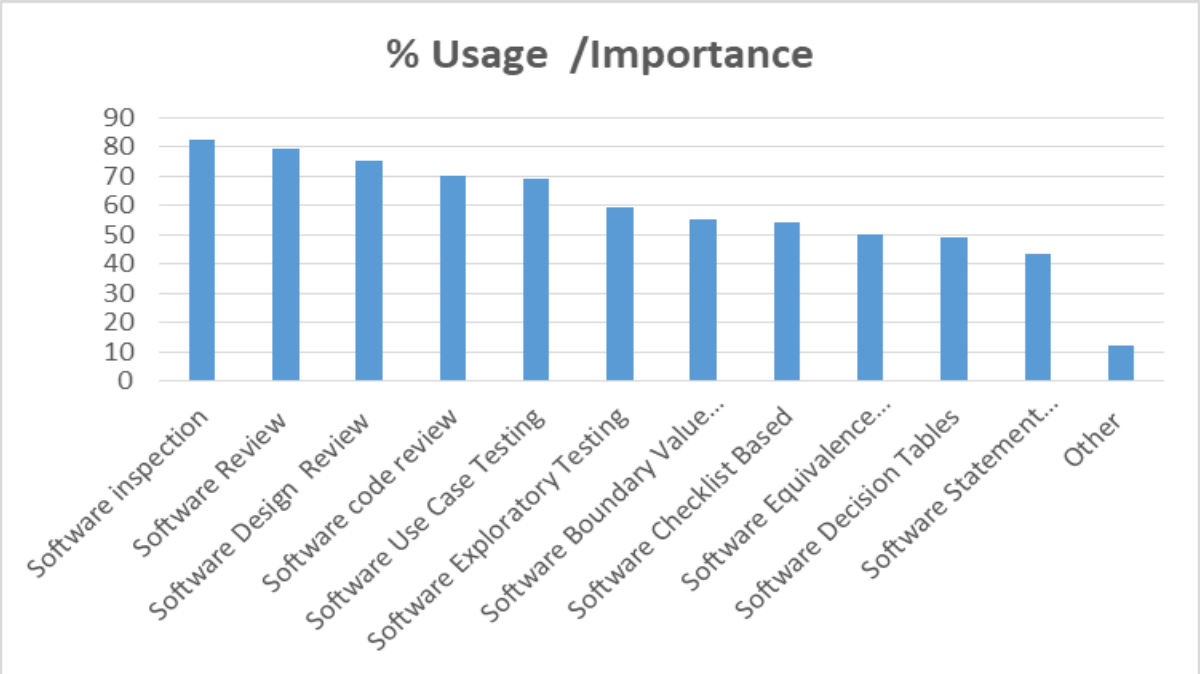
RQ: Which verification and validation techniques are used by your verification and validation team?

The top five verification and validation techniques selected by the survey respondents are software inspection (82.2%), software review (79.2%), and software design review Review (75.2%), software code review (70.2%), and software use case testing (69.2%).

Table 4.15: Verification and validation techniques are used verification and validation team

S/N	Verification and validation techniques are used by verification and validation team	% Usage /Importance
1	Software inspection	82.2
2	Software Review	79.2
3	Software Design Review	75.2
4	Software code review	70.2
5	Software Use Case Testing	69.2
6	Software Exploratory Testing	59.4
7	Software Boundary Value Analysis	55.2
8	Software Checklist Based	54.2
9	Software Equivalence Partitioning	50.2
10	Software Decision Tables	49.2
11	Software Statement Coverage	43.2
12	Other	12.3

Figure 4.18: Verification and validation techniques are used by verification and validation team



4.2 Experiments Results

4.2.1 In defect analysis, how do review and inspections compare to testing?

The combination of different verification and validation activities is a means for achieving a high-quality software product that is developed with low fault injection and exposed to effective fault detection techniques.

The focus of this research question is on the combination of reviews, inspections, and testing as fault detection activities. In the controlled experiment, the techniques were evaluated in terms of their fault detection capabilities. Related work combining the methods has focused on fault detection on code artifacts, while the work in this thesis emphasizes the importance of also investigating and comparing the activities on a higher abstraction level.

In the study on improving software review, inspection, and testing, the efficiency and effectiveness of the techniques for the detection of design faults were evaluated. The general results from this study show that the values for efficiency and effectiveness are higher for the review and inspection technique and that the testing technique tends to require more time for learning. Despite the fact that rework was not considered, the study included defect analysis, reviews, and inspections, which were more efficient and effective.

4.2.2 Comparison Between The Software Design Review and Software Code Review Techniques

Experiment's research compares software design review and software code review as verification and validation techniques for software process improvement. And for improving the quality of the software products, results show that both were used in the early stages of software system development. Software design review is used to review all the requirements and design and improve them with the intent of obtaining higher-quality software products and developing the software system within a short time. Some of the surveyed organizations used software code review to investigate if software code was written without error by considering the designs that had been made before. The conclusion of the research indicates that software design review and software code review are mutually exclusive. When the design is done correctly, it is easy to map the design into coding. We noticed that the better the software design, the better the code produced. According to the findings obtained from surveyed software development organizations in Tanzania, software products are of higher quality. In the early stages of software development, the progress of the software can also be seen.

4.2.3 Comparison Between Software Code Review, Software Design Review, and Software Testing Techniques

To boost confidence in the software program, we discovered that software code review and software design review techniques could prevent the introduction of bugs or defects from the

start of the software development process rather than waiting until the end. Both are applied with the intent of detecting and removing a large number of bugs or defects in the early phases of the software development product.

As a result, in order to reduce the time spent on software testing, it is recommended that serious measures be taken to improve the verification and validation techniques used during the early stages of software development. It was noticed that the number of defects discovered during software testing is dependent on the fact that software design review and software code review have been done correctly, so there is a chance of avoiding software testing. But if the verification and validation processes are not applied correctly at the early phases of the software development processes, there is the possibility of detecting a larger number of bugs in the software.

4.2.4 Comparison between Software Inspection and Software Testing Techniques

Software inspection is applicable as a software review technique. During the experiment, I noticed that, using software inspection methods, all the documents were reviewed to check for defects and removed. For the purposes of the experiment, software inspection techniques involved all the activities of the software process and all the stakeholders of the specified software system.

Tracking the progress of the project status is simple once software inspection for verification and validation is applied. Furthermore, it was concluded that it is easy to avoid a large number of defects during the early stages of software development. Software testing techniques depend on the quality of the software inspection. The general conclusion is that the better the software inspection, the better the software testing.

When all software documents are reviewed at the beginning of the software development process, a large number of defects are avoided. Doing the software testing is not enough to establish confidence in using the software system, but doing the software testing is necessary to show the presence of bugs that are involved in that system.

In order to avoid spending a significant amount of time finding and fixing a large number of bugs during software testing, it is advised in this research paper that the use of software inspection techniques must be taken seriously in the early phases of the software development

process. in order to show the presence of bugs without knowing how to prevent them. But both software testing and software inspection are used as verification and validation techniques to help software developing organizations detect, prevent, and avoid the bugs or defects of the software.

4.2.5 Summary of Experiment Results

i. The results of the experiment show that many more software defects were introduced during coding than design defects.

ii. The experiment's findings revealed that many defects were removed during testing, compilation, and code review, but few defects were removed during design review.

iii. The results show that defects were introduced and removed at each stage. And the number of observations for each finding and result presented in this thesis has been specified.

Figure 4.19: Software defects analysis

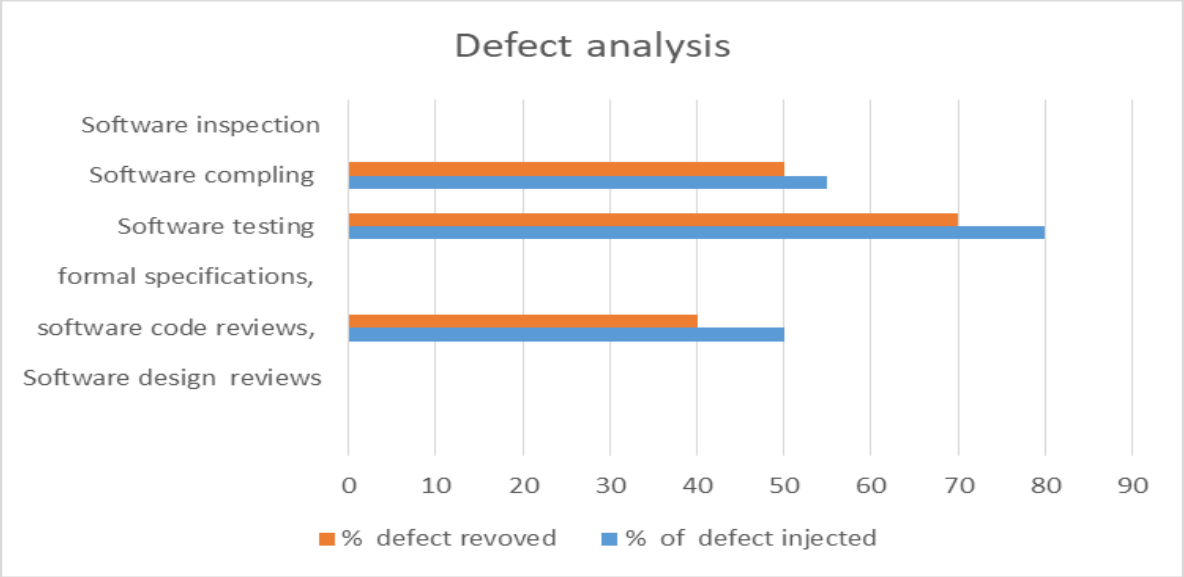


Figure 4.20: The experiment results show that more software defects were introduced during coding than design defects.

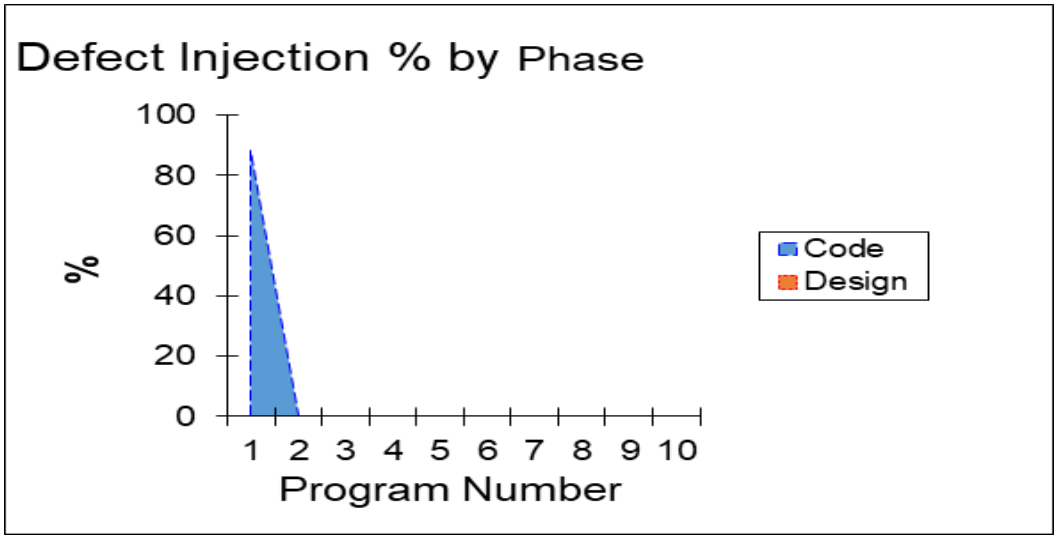
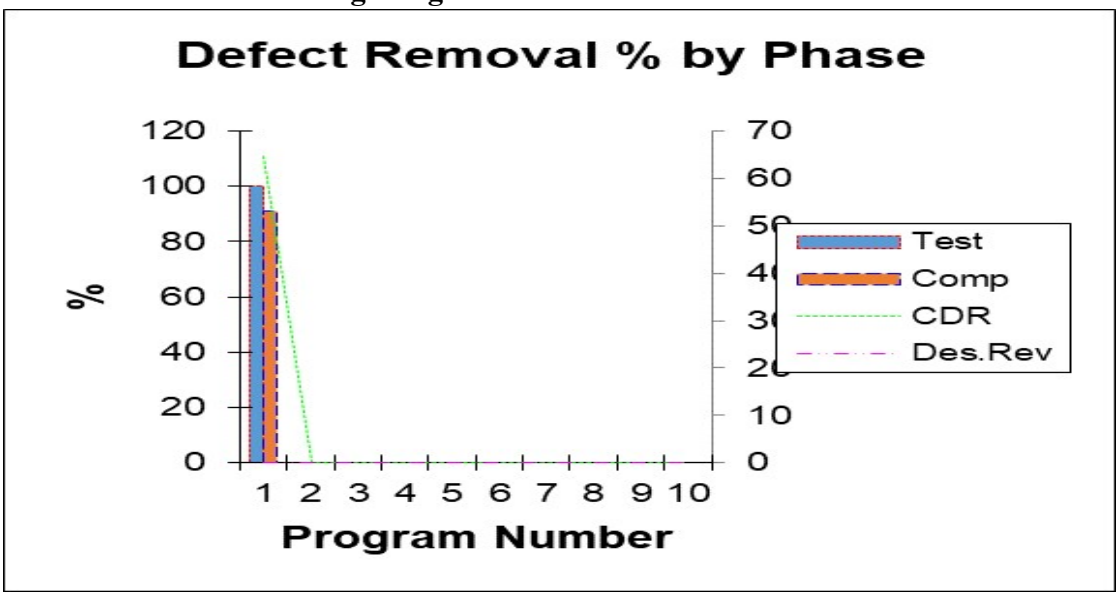


Figure 4.21: The experiment's findings revealed that many defects were removed during testing, compilation, and code review, but few defects were removed during design review.



4.3 Findings of the Literature Review

The main literature sources are research databases with connected search engines used to collect secondary data for the research presented in this thesis. A literature study aims to map out the current publications relevant to software verification and validation practices with the help of keywords and prior knowledge.

Findings from the literature review indicate that the use of automation in testing has increased. Automation has become more popular at all levels of testing in recent years (Feldt et al., 2010). (Jan et al., 2016). Second, the use of formal software process models and

capability maturity models appears to have decreased, while testing tools have increased in use and effectiveness (M. Al Atitaie et al., 2020). It was discovered that there are more developers than testers. Many software development companies lack a dedicated testing team. This finding could point to a lack of maturity or financial resources to support a dedicated testing team. software development organization Low utilization of software verification and validation techniques and tools, challenges caused by a lack of software testing methods and tools, limited use of testing tools Software testing is still a comparatively small part of software development in Tanzania, even though it occasionally occurs concurrently with development activities. It was discovered that the majority of procedures and methods haven't changed all that much in 20 years. In addition to using the right procedures, effective testing calls for a tester's creativity and experience. Testing entails more than just fixing bugs. Testing serves other purposes besides identifying and fixing flaws. Unit testing methodologies and practices appear to be unaffected by test management or quality management. Although unit tests are structural, or "white box," in nature, developers rarely assess their completion in terms of structural coverage. The use of formal software process models and capability maturity models appears to have decreased. Functional testing is the most important type of testing in organizations. The use of automation in testing has increased. Automation has become more popular at all levels of testing in recent years. Second, the use of formal software process models and capability maturity models appears to have decreased. Testing is costly. Saving money and time can be accomplished through automation. Agile methodologies are becoming increasingly popular, emphasizing the necessity for suitable testing processes and techniques. The development of software quality is an information-intensive process that is influenced by organizational structures and information flow within firms.

4.4 Threats to Validity

The key validity threats to these conclusions are related to each research question.

General threats to the external validity of a survey concern whether the sample of the study represents an appropriate population (Yadav & Sharma, 2017; Vaismoradi & Snelgrove, 2019). The sample chosen has diversity in several aspects, though it is still a result of convenience sampling due to the organizations' geographical location in Tanzania (Zach, 2006). Threats to internal validity might also affect the outcome of the survey. One threat

concerns the respondents. They may have different views of reality depending on their role in the surveyed organizations. In the reported survey, most organizations had more than one respondent. The respondents ranged from test managers, developers, testers, and quality assurance team members to project managers (Casteel & Bridier, 2021). As a qualitative study, there is potential for bias from the researchers as well as the respondents (Islamia, 2017). This threat was countered by triangulation: having multiple sources for the data, peer debriefing, and member checking (having the material received from the respondents returned to them for review).

General threats in experimental studies, like the experiment that evaluates defect analysis, often concern external validity, or whether the results are generalizable to other settings. These might be reduced by choosing an appropriate design and considering the experimental environment and its subjects and objects. This study discovered a significant threat to construct validity (Brink, 2018).

CHAPTER FIVE
DISCUSSION

5.1 Discussion of Findings

Through data analysis, Research presented in this thesis concluded that using verification and validation techniques early in the software development process has a significant impact on improving software development procedures. Greater software product quality can be easily attained in the early stages of software development. The findings of this study also show that when verification and validation techniques are used early in the software development process, it is easy to track the progress of software projects as quickly as possible. Apart from that, this study finds that software testing is also one of the methodologies for software verification and validation. It is easy to track the progress of software projects as quickly as possible through the use of software inspection. Software testing is prevalent for software verification and validation. Software testing is insufficient to establish user confidence in the software system because it focuses on detecting problems. This is not a vital aspect of enhancing the software process. Software reviews, software code reviews, formal specifications, and software design reviews have a significant impact on assisting software development organizations in obtaining higher-quality software. Many errors discovered during the development phase of software development are particularly costly to fix when compared to those discovered during the early stages of software development. According to the results of the literature review, software testing is also one of the methodologies for software verification and validation. However, doing software testing is insufficient to establish user confidence in the software system because software testing focuses on detecting problems, which is not a vital aspect of enhancing the software process.

Respondents from eight software development organizations in Tanzania concluded that functional testing (88%) is the most important type of testing, followed by user acceptance testing (70. 0%).Formal training (65.2%) and certification (55.2%) rated highly as approaches to improving the competency of testers after on-the-job training (47.9%). Respondents from eight Tanzanian software development organizations indicated that document analysis and requirements review are the most common activities for early defect detection. Results indicated that respondents indicated that among their organizations, a majority are assigning their testing to an in-house test team (72%), and 34% of all respondents are using software developers. The results of the survey indicate that software test automation is the most

trending topic for the software testing profession in the near future and is also highlighted as the main improvement area in software verification and validation processes. According to the findings, 81% of organizations now use agile models. Agile and followed by the sequential methods in 50% of the respondents' organizations. Results from respondents' surveys indicate that the most common verification and validation activity is "to detect bugs" (98.2%), followed by "to show the software is working properly" (77.2%). The results of the survey indicate that software test automation is the most trending topic for the software testing profession in the near future and is also highlighted as the main improvement area in software verification and validation processes. The results of the study indicated that, in the next two years, the most important subject will be software security testing (75.2% for the software verification and validation organizations). According to the results of a survey of software development organizations, system testing (71.2%) consumed the majority of the testing budget, followed by integration testing (50.2%), user acceptance testing (40.3%), and unit acceptance testing (27.2%). The results from the surveyed organizations indicated that the most commonly used tools among test teams are software defect tracking (82.2%), software test automation (72.2%), and software test execution (70.1%). Results indicated that the top three testing challenges in Agile projects are Agile Test Automation (56.4%), Agile Documentation Challenge (40.7%), and Collaboration (30.2%). Considering the results of the survey, the organization will be able to meet the software agile development challenges in light of these challenges. The results from the surveyed organization indicated that the three most important areas for improvement in verification and validation activities are software inspection (73.2%), software review (62.4%), and test automation (60.4%). According to the findings, Tanzanian software development organizations face verification and validation challenges. Most Tanzanian software development organizations have unstable requirements for software specification documents. During verification and validation testing, more defects were discovered in the testing environment and tools than I had anticipated. The software testing team has a limited focus on integration testing of software components. According to the findings, most software development organizations in Tanzania have insufficient internal software review and inspection. We came to the conclusion that many software organizations in Tanzania prioritize white-box testing over black-box testing. We observed that test cases are not reviewed by independent software engineers and software testers. Results show that practitioners consider test management, test automation, and other test activities to be the

most challenging test activity types. Other test activity types, such as test case design, test execution, evaluation, and result reporting, have been seen as less challenging. Regarding test management, some of the main challenges raised by practitioners are related to assessing the effectiveness and efficiency of testing. Practitioners need guidance for the selection of suitable testing approaches for a given context. Tanzanian software development organizations reuse document templates for ICT software development projects but struggle to reuse source code. The software development organization in Tanzania allows for reusability, but other requirements in the standards make reusability difficult to achieve. The main issue with Tanzanian software development organizations is that they have a high demand for detailed documentation but do not perform software quality assurance or verification and validation tasks. The top five verification and validation techniques selected by the survey respondents are software inspection (82.2%), software review (79.2%), software design review (75.2%), software code review (70.2%), and software use case testing (69.2%). Soft skills (70.8%) and business/domain knowledge (62.9%) are the non-software verification and validation skills that are most expected from an agile tester, according to the results of the survey**.** This result shows that a good software tester and verification and validation expert should have a good understanding of the verification and validation process, including software test execution (82.1%), software bug reporting (78.2%), and software test design (77.8%), which are the most important verification and validation skills.

This research presented in the thesis is not based on full implementation but on experimentation based on experiment of the software verification and validation activities used to support the results presented. Conclusions have been drawn from the results obtained. We conducted experiments with participants from eight software development organizations and analyzed software defects, where each process was checked and some verification and validation activities were performed, and we compared results due to their applicability and efficiency in assisting the use of software verification and validation techniques in order to produce higher-quality software products in a short time. After verification and validation activities were performed. The experimentation in this thesis is based on the development of various software programs with the goal of improving the quality of each program. The results of experiment concluded that that defects were introduced and removed at each stage. And the number of observations for each finding and result presented in this thesis has been specified.

CHAPTER SIX
CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

Software development organizations can produce higher-quality software only if the number of defects introduced through each phase of the software development process is prevented as soon as possible during each phase of the process.

Verification and validation activities must begin in the early stages of the software development process. Instead of applying software testing at the end of the software development process, this will help prevent a greater number of defects in a shorter period of time. Once the verification and validation process is applied as soon as the software development starts, it will reduce the amount of time spent during the software testing phase, and sometimes it is not easy to detect and avoid all defects or bugs using software testing.

In order to establish confidence in using the software system and to make users trust software products, it is important to check if the system developed will meet the needs of the user and the requirements specified (Raulamo-Jurvanen et al., 2019). Doing that will avoid the consequence of the software project failing to perform an operation as required (Ullah Khan et al., 2015).

Employing verification and validation techniques early on will greatly assist software organizations in meeting their objectives and improving the software process.

It is not possible to produce higher-quality software system products without the support of verification and validation techniques. This means that verification and validation techniques are focused on defect detection, preventing, and avoiding defects within the software system.

Software testing is applied by many software organizations, but it is not enough to prove that the software system contains no defects. Because the purpose of software testing is to show the presence of bugs, the research presented in this thesis suggests that, in order to avoid a greater number of bugs in software products, verification and validation processes should be implemented as soon as possible.

Research presented in this thesis explains the current application of verification and validation techniques to software process improvement. Considering this situation, we have discussed the different existing validation and verification techniques from different literature sources and have been using the comparison approach for some existing verification and validation techniques like software inspection, software code review, software testing, and software design review. We conducted experiments to derive the findings of this study, which was done as a project study using the person-software process approach, and from the results presented in the form of graphs, a lot of information can be analyzed from the graphs of the results obtained.

The major conclusion is that there is a need to use verification and validation techniques in order to improve the software process and obtain quality software products. In this research paper, we propose using verification and validation to avoid software project failure and to assist software development organizations in managing and tracking the progress of the software products that they must develop. And as well, the early determination of the software project's failure or success can be visualized as soon as the software development starts.

6.2 Study Recommendations

This thesis recommends employing software verification and validation at the early stage of software development so that higher-quality software can be delivered to the customer within a short time. The domain of this thesis, presented in this paper, is helping software engineering organizations improve the software verification and validation process and produce quality software. Through the use of the software verification and validation processes. Software development organizations improve the software verification and validation process and produce quality software. Through the use of the software verification and validation processes. We can improve the software process, make sure that higher-quality software products are delivered to the customer within a reasonable amount of time, and focus on the market's demands. The end user must also trust the software system that they are going to use. The early involvement of customers in the software development process gives them confidence in the system they are going to use, and it is easy to trust the system delivered by the developers.

6.2.1 Future Work

In the future, based on the research presented in this thesis, we are going to focus on the verification and validation techniques that will help the software process improve when the functionality of the software system changes (Bjarnason et al., 2014). A survey was conducted in eight Tanzanian software development organizations to investigate the current state of verification and validation practices, identify the existing challenges of software verification and validation activities in software development organizations, and conduct a case study on software application development and testing. It was observed that it is important for the verification and validation processes to be flexible due to changes in the software requirements during software development.

In the future, this research study will address the adaptability of verification and validation in order to limit the failure of the software operation when any changes to the software requirements occur. It is true that many software verification and validation processes are applied in software processes, but some are not flexible (Bäckström, 2022). Therefore, future research will focus on how the verification and validation processes will be able to adapt to changes in software functionality or requirements (Carlos & Ibrahim, 2021).

Furthermore, future research presented in this thesis will focus on how the addition of some processes will help the existing verification and validation techniques be flexible and adjustable when the functionality of the software changes during development. This adaptation of the use of verification and validation activities to the software improvement that this research will go on to address in the future will depend on the functionality of the system and what the software product is required to be. Also, this will focus on the general risks that can occur during the use of software verification and validation techniques during software development (Okezie et al., 2019).

APPENDEX

Study questioners and interview questions were developed.

RQ. How do software organizations perform verification and validation?

RQ how does your company improve the competency level of your software verification and validation?

Formal verification and validation training ( )

Verification and validation certification

On job training

Attending conference of verification and validation

Other

RQ: In your company, who is in charge of verification and validation?

Software testing responsible Personnel

In-house test team

Software Developers ( )

Off-shore test team

Software Quality Assurance team

RQ**:** Which verification and validation methods are used? What activities do you use to find defects before test executions?

Formal review of the analysis documents/requirements

Formal review of the design documents ( )

Source code inspection

Static analysis tools

None

RQ: What types of testing are used in organizations?

Functional Testing

Performance Testing

Security Testing ( )

Usability Testing

Accessibility Testing

Reliability Testing.
Testability Testing
Availability Testing
Maintainability Testing
Efficiency Testing
Scalability Testing
Interoperability Testing
Operability Testing
Portability Testing
Recoverability Testing
Supportability Testing
Extensibility Testing

RQ What verification and validation topics are important for your company?
User Acceptance Testing
Exploratory Testing
Systems Integration Testing
Web Based Application Testing
Mobile Testing.
Release Management ( )
Test Data Management
Test Environments Management
Configuration Management
Test Metrics and Test Effort Estimation
Static Testing
Testing Systems of Systems
Non-regression Testing
Cloud Testing
Business Intelligence / Big Data Testing
Embedded Systems Testing
Internet of Things Testing
Other

RQ which software development lifecycle (SDLC) model does your organization use?

Agile Software Development.

Sequential Development

Scrum, Kanban, Extreme Programming) ( )

Waterfall, Vmodel Development

Both sequential and Agile Development

Iterative Development

None

RQ: What are the main objectives of your verification and validation activities?

To detect software bugs

To show the software is working properly

To gain confidence

To evaluate software requirements ( )

To evaluate the verification and validation user experience

To comply with organization regulations

To be a customer advocate

To have software zero defects

Other

RQ: What are the main improvements? Areas of your verification and validation activities

Software inspection

Software review

Software Test automation

Knowledge About Test Processes ( )

Communication Between Development and Testing

Maintaining Test Cases

Communication Between Business Analysis and Testing

Knowledge About Test Design Technique

Maintaining Software Test Scripts

Time management

Communication Between Project Management verification and validation
Software Test Data Preparation
Having Well Testing Trained Personnel ( )
Verification and validation Budget
Other

RQ: Which tools do you use it in your organization during verification and validation activities?
Software Defect Tracking Tool
Software Test Automation Tool ( )
Software Test Execution.
Software Test Management
Software Performance Testing
Software Requirements Traceability
Software Test Design
Software Unit testing
Static analysis
Dynamic Analysis tool
Software dynamic Testing
None

RQ: Which test levels or types (S) receive the majority of your ICT budget?
Software System Testing
Software Integration Testing ( )
Software User Acceptance Testing
Software Unit Testing
Other

RQ: Which new technologies will be important to the software verification and validation organization in the following two years?
Software Security Technology ( )
Software Artificial Intelligence

Big Data
Systems Cloud
Continuous Integration
Continuous Testing
DevOps
Performance
Machine Learning ( )
Database Management
Internet of Things
Usability
Cognitive Test Automation
Scalability
Healthcare Devices
Software inspection
Software testing tools
Requirement Engineering
Neuronal Networks
Other Technologies

RQ**:** What will be the most trending topic for the software verification and validation profession in the near future?
Software Test Automation
Software Agile Testing
Software Security Testing ( )
Software Cloud Testing
Software Mobile Testing
Software Continuous Testing
Software Performance Testing
Software Virtualization
Software Test Process Improvement
Software test Data Management
Software Verification and Validation Techniques

Software Usability Testing
Software Test Management
Software Model-based Testing
Software Static Testing.
Others

RQ what are the challenges concerning the existing software verification and validation practices in Tanzania

RQ: What are your most difficult verification and validation challenges in agile projects?
Agile Test Automation
Agile Documentation
Agile Collaboration ( )
Agile Test Effort Estimation
Agile Exit and Entry Test Criteria
Risk Awareness.
Agile Cross Functional Needs
Software Quality Ownership
Decision Making
Software Traceability
Agile Test Reporting
Agile Legacy Defects
Not Applicable
Regulatory / Compliance Issues
Other

RQ: What expectations and skills do you have for the verification and validation team?
Software Test Execution
Software Bug Reporting's
Software Test Design
Software Test Analysis ( )
Software Test Automation
Software Test Planning

Software Test Strategy
Software Test Implementation
Software Test Monitoring
Software Bug Advocacy
Others

RQ: Which of the following non-software verification and validation skills are most expected from an agile tester in your organization?

Soft Skills
Business Knowledge
Tool Knowledge
Risk Estimation
SDLC Knowledge ( )
Continuous Integration
Database Management
Software Coding
Software Project Management.
Computer Network
Enterprise Analysis
Others

RQ: Which verification and validation techniques are used by your verification and validation team?
Software inspection
Software Review
Software Design Review
Software code review. ( )
Software Use Case Testing.
Software Exploratory Testing
Software Boundary Value Analysis
Software Checklist Based.

Software Equivalence Partitioning
Software Decision Tables
Software Statement Coverage.
Other

RQ How do you compare the software design review and software code review techniques? for Defect injection and removal during program development

RQ How do you compare software code review, software design review, and software testing techniques in terms of defects injected and removed?

RQ How do you compare software inspection and software testing techniques in terms of software quality?