ISSN: 2093-5374 / E-ISSN: 2093-5382

Impact of Critical Success Factors on Productivity Gain during Automation Testing

S.M. Bindu Bhargavi[1], V. Suma[2*], and R. Vijaya Arjunan[3*]

[1]Assistant Professor, Information Science and Engineering Department, Dayananda Sagar College of Engineering, Bangalore, Visvesvaraya Technological University, Belagavi, India.
bindu.sm@gmail.com, https://orcid.org/0000-0003-2700-7985

[2*]Professor, Department of Computer Science and Design, Dayananda Sagar College of Engineering, Bangalore, Visvesvaraya Technological University, Belagavi, India.
hod-csd@dayanandasagar.edu, https://orcid.org/0000-0003-1942-6741

[3*]Additional Professor, Department of Computer Science and Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education (MAHE), Manipal, Karnataka, India.
vijay.arjun@manipal.edu, https://orcid.org/0000-0002-1402-6573

Received: March 18, 2024; Revised: June 06, 2024; Accepted: July 15, 2024; Published: September 30, 2024

Abstract

Software testing, a linchpin of quality assurance in software development, encompasses both manual and automation approaches. While manual testing ensures meticulous scrutiny, automation accelerates efficiency, reducing time and costs in dynamic landscape of software development. This research investigates the impact of critical success factors on productivity gain in software testing, with a focus on automation, within CMMI Level 5 companies. Drawing from a dataset comprising thirty real-time projects spanning banking, retail, and industrial applications, study explores dynamics of software development under agile and waterfall models. Interviews with developers and testers reveal insights into key aspects influencing productivity, including skills, experiences, and training.

Emphasizing the need for strategic automation in projects subject to changes and maintenance, the research analyzes over thirty projects, scrutinizing variables such as manually designed test cases, execution time, and subsequent automation of test cases. Strategic selection of test cases for automation emerges as a cost-efficient practice. Results highlight a correlation between manual and automated test cases, indicating productivity gains. Research introduces a productivity gain metric, showcasing a break-even point where automation significantly reduces testing time. Overall, the findings offer a comprehensive understanding of software testing within CMMI Level 5, guiding organizations toward efficient automation practices and improved productivity.

Keywords: Software Test Automation, Manual Testing, Success Factors, Productivity Gain.

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), volume: 15, number: 3 (September), pp. 92-108. DOI: *10.58346/JOWUA.2024.I3.007*

Corresponding author: [2]Professor, Department of Computer Science and Design, Dayananda Sagar College of Engineering, Bangalore, Visvesvaraya Technological University, Belagavi, India; [3*]Additional Professor, Department of Computer Science and Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education (MAHE), Manipal, Karnataka, India.

1 Introduction

Software testing is the process of confirming the functionality and quality of the software product. The testing process is carried out for evaluating the software according to requirements specified to check for any missing requirements, errors, faults and bugs. Testing facilitates early identification and detection of bugs as it is necessary to reduce the financial losses. This is supported by the reports from National Institute of Standard Technology (NIST) which states that the impact of software testing takes around 4.9 hours to fix the bugs identified during coding or unit testing in comparison to the bug identification after the product release or product usage, taking approximately 15.3 hours (Planning, 2002) to fix the bug.

In the current era, most software development processes are based on agile methodology making a transition from the traditional waterfall model. While testing is carried out as the last step of software development in waterfall model, testing is carried out throughout the software development process in the agile methodology of software development. Testing is performed either manually or by automation or both. Testing when carried out manually can be slow, expensive and error prone requiring an extra skill set of the testers. To overcome these drawbacks automated testing has been employed mainly to achieve faster execution of test cases with minimal or no human intervention. According to Global Market Insights, the rate of using test automation over manual testing in the software development market will increase by more than 16% by the end of 2027 (Verified Market Research). Test automation is more suitable to speed up the testing process covering a greater number of test cases and execution of similar test cases multiple times facilitating in faster delivery of products/release of the products.

In the case of incremental software models and agile software models where the software is deployed in several iterations to the user, performing testing on each iteration is a must. Testing when performed manually utilizes a lot of time and resources along with the extra skill set requirements from the testers. A more suitable solution for this is to perform testing through automation. Incremental development in software is carried out such that with the addition of each new feature or configuration change, core functionality of the software should remain the same without any deviations. This may require the need for regression testing to ensure there are minimal or no deviations from the major functionalities of the software (Wiklund et al., 2017). It has been observed from recent studies that an average of 13 hours is being spent on debugging and fixing software bugs. Automation testing offers better and early identification of defects in the software, facilitating fixing the bugs or defects at the early stage of software development in a faster rate, which substantially contributes towards efficient management of cost and other resources of the software (Ferdiansyah et al., 2023).

Automation testing requires specific skillset of the testers and use of suitable tools for the generation of test cases making it expensive when compared to manual testing. Test automation when carried out under the influence of favorable factors can facilitate in achieving better automation gain in terms of cost, time and quality. So, it is important to identify and understand the factors that influence automation testing. The contribution of the identified factors to automation gain must be further evaluated to understand the impact and effectiveness these factors can have on automation testing. The proposed work performs the evaluation of identified factors with the introduction of a metric termed as productivity gain to measure the effectiveness of automation testing. The proposed metric also emphasis the break-even condition on which automation testing can be performed with the automation gain in terms of true Return on Investment (ROI) (Offutt, 2023).

The organization of the paper is as follows: Section 1 provides the role of software testing in software engineering, along with emphasizing on the role of automation testing in software testing. Importance of regression testing in software development and the reasons for why regression is a suitable candidate for automation has been analyzed. Section 2 provides an insight into the applications of automation testing and the dependency of the current real-world applications on automation testing. The section also covers the process carried out for automation testing, widely used tools and corresponding details about the same. Section 3 of the proposed work gives a detailed methodology followed right from the data collection stage to data analysis stage along with the process followed for identification of suitable factors for calculation of productivity gain. Section 4 briefs about the comparison between different software projects considered for productivity gain calculation with necessary results. Section 5 gives an overview of the statistical analysis carried out so far followed by conclusion.

2 Automation Testing

Automation testing is a testing technique performed to ensure better quality software with the use of skilled testers and specialized software tools. Though automation testing is expensive compared to manual testing, it is widely used as automation testing provides faster bug detection, more test coverage, faster results in less time, better documentation of test results along with minimal or no human intervention. Automation testing is performed when there is need for repetitive execution of specific test suites to ensure no deviations from the functionality of the software, when the number of test cases to be executed is large, when optimal test coverage is to be achieved and in complex scenarios.

1) Automation Process

Automation testing is started by understanding and defining the scope of the software under development based on the initial requirements followed by feasibility check. Once feasibility check is cleared – based on the application under development, automation design is generated which includes tool selection and test plan design. This is followed by a generation of test scripts - a test script is the composition of a set of instructions describing the state of the application, actions corresponding to main functionality of the application followed by expected results. Test scripts are designed to be simple, reusable and independently executable as this simplifies the process of debugging easier. Similar test scripts can be coupled together to form a test suite where a collection of test cases can be executed at once with minimal human intervention. Test cases related to features which are subject to frequent changes are not suitable for automation - since every change requires changes to be made in the test script regularly. Test cases designed manually which are unlikely to change are best suitable for generation of automation test scripts - through an application specific testing tool. Test scripts are executed with the generation of test reports. An automation test report is composed of test objectives, a comprehensive defect report, testing technique employed and features of the testing tool used along with details of test coverage. The report also comprises the overall execution time, time taken exclusively to run each test script, number of test cases executed, number of test cases passed, failed and not executed. Further examination of the test reports facilitates the identification of bugs / defects in the system under test.

Automation testing is expensive as it requires special programming and scripting skill set of the testers, exclusive tools for performing automation, and specialized frameworks with respect to underlying application. Automation testing is further not suitable in scenarios which are ad-hoc and require human intervention for better test cases. But automation testing is best suitable in case of regression testing to check for the integrity of the functions performed by the software. It is also advantageous to use automation testing in Continuous Integration, Continuous Deployment workflow to ensure the validity of the code during continuous development of the software. Since automation testing provides timely validation of functionality of the software, it is used in real-time applications like web development, mobile applications, gaming, financial and healthcare applications. Other advantages of automation testing include delivery of quality software with better defect management in a timely manner, optimal code coverage with better Return on Investment.

2) Tool Usage during Automation Testing

1. To deliver quality software, software engineers confirm that the software is tested at a faster rate by appropriate testing tools and testing techniques. Testing is important as it costs around $59.5 billion to the US economy annually according to the National Institute of Standards and Technology (Jones, 2011). Automation tools and frameworks are designed to support various phases of software development, software management and faster execution of the test scripts and documentation of the test results for further analysis. Automation test cases are derived from manual test cases by following a systematic procedure of identifying the test cases which consume lot of time and resources during manual testing, test cases which are repetitive in nature – especially in case of regression testing or in case of complex applications. To support this many automation tools are currently in use which are either open source or commercial in nature. Further, not all the manual test cases are suitable candidates for automation testing. The decision for automation also depends on the underlying application, development framework of the application, testing technique used, automation tools used, and test strategy followed by the application (Automated Software Testing). Automation testing tools can be categorized based on the artifact being tested which is: unit testing, functional testing, code coverage, test management and performance testing (Umar & Zhanfang, 2019; Myrria De Albuquerque et al., 2023; Riola, 2023). Unit testing tools are used to check for the correctness of the units or methods in the code structure by commonly used tools like JUnit, JMockit etc. When testing is performed to check the compliance of the output obtained with respect to the requirements specified then functional testing is performed by tools like Selenium, HP QuickTest Professional and many more. Tools like Cobertura, PITest etc are used for checking the number of lines of code, statements or blocks with respect to code coverage testing. When testing is performed with respect to each phase of testing life cycle, then the usage of Test Management tools is more effective - which are Test Link, QA Complete etc. Testing the software with respect to non-functional requirements of the software is termed as performance testing which uses tools like JMeter, HP LoadRunner etc. Extensive studies have been undertaken on testing tools, their selection and efficient usage as testing tools play a vital role in test automation.

3 Literature Survey with Gap Identification

Automation testing has been a critical part of the software testing industry with its widespread application in software development. Automation testing is expensive compared to manual testing due to the initial investment on automation tool, training for the testers and exclusive usage of automation framework. Automation testing is employed with the main aim of reducing the test effort with a better return on investment (Oleksandr et al., 2024). To reduce time, cost and effort several factors have been identified which may contribute towards increasing the efficiency of testing. In this regard, a detailed literature survey has been undertaken to understand various factors contributing towards better automation testing and their effectiveness in testing. The study of existing system has provided a clear overview based on the identification of factors contributing to better automation, software development frameworks used, code coverage and its impact along with several other factors.

1) Factors Directly Contributing towards Better Automation

A detailed literature survey has been carried out in where the authors have successfully identified the factors that are hindering the process of automation testing to be effective. The dataset considered for the survey included publications and journal databases with major findings on factors which act as a barrier for better automation. The factors varied from behavioral effects, business and planning, skills, test system and system under test. Behavioral effects dealt with the influence of the organization culture, motivation, expectations and behavior impacting the outcome of the project undertaken. Very high expectations from the organization and process deviations are not suitable when test automation is used. The implementation of automation testing would often require the testers to be well versed in testing the software, developing the software along with extensive knowledge of the automation tools. To perform automation, the test system must be able to provide an environment composing suitable hardware, necessary automation tools and simulators and other software tools (Jayasree & Baby, 2019). So, the testers must ensure all the required criteria is met before performing test automation. Another major issue with automation testing is with respect to platform limitations caused by different operating systems with different configurations, varied coding languages and platforms. The factors identified here are purely non - technical but still have an impact on automation testing.

A systematic literature survey has been carried out in for the identification of vital factors influencing test automation in Agile-based software development (Butt et al., 2023). The factors identified range from selection of test tools, skills of the testers, test environment, use of regression testing and many more. This proposed work used Factor Contribution Analysis (FCA) for the determination of the most prospective factors contributing towards successful test automation. The FCA has been employed under five different categories related to software under test, test system, tests being performed, human and organizational influences and cross - cutting approaches (Riola, 2023). The proposed conceptual model facilitated the testers in better decision making and risk management during testing. Though many factors were identified for successful automation, their impact in terms of Return on Investment has not been incorporated in the existing work.

2) Different Software Development Approaches

The authors in (Kumar & Mishra, 2016) have taken several case studies following an incremental development approach into consideration for analyzing the impact of automation on cost, quality and time. The impact of cost is calculated by the summation of manual test effort and total software testing effort with respect to all versions of the software (Yang et al., 2019). Quality of the software can be determined in terms of attributes like functionality, reliability, maintainability and several secondary attributes. The impact of time on test automation can be determined by considering the development time and several other factors. Since the factors considered have a positive impact on the software quality - it is hence effective to use automation testing when compared to manual testing.

The implementation of machine learning techniques for the reduction of cost and errors has been proposed in (Braga et al., 2018) where the generation of test oracles is done in an automated manner. Since manual testing is error-prone and time consuming, developers often tend to use automation testing to ensure delivery of products within time constraints. This is achieved by employing machine learning techniques for the generation of test oracles which can exclusively distinguish between correct and incorrect behavior of the software (Leu & Tjoa, 2014). As an improvement of the current version, the authors in (Hershkovich et al., 2021) have implemented a suitable tool for the generation of test oracles with the use of a fault prediction model. The advantage of this model is its capability to detect more bugs than compared to coverage-oriented approach (Hemmati, 2015). The proposed model named

QUADRANT is capable of test case generation in case of small and simple software components. The model could easily eliminate a huge number of false positive test cases. The advantage of QUADRANT is its capability of identifying bugs with minimal number of test cases generated. This approach used supervised learning algorithms and assigned scores for components to be tested. Once the scores reached the maximum limit, the component was removed from the sorted components list and further generated tests for components by using the test generation tool (Automated testing). Test cases generated by this technique were relatively less in comparison with the large number of bugs identified thereby facilitating in the reduction of the total test effort.

3) Code Coverage and Cost Dependency

The main objective of any testing technique is to achieve maximum code coverage with minimum number of test cases time and cost effective manner. But maximum code coverage does not always guarantee the detection of all the bugs present in the coding software. Authors in (Perera, 2020) have proposed Search based Software Testing techniques with the implementation of defect prediction algorithms for better prediction of bugs. The defect prediction algorithms are implemented based on Genetic Algorithms which make use of search heuristics for better prediction of defects. Further predicting the severity of bugs in very large software applications / systems has been proposed in where effective bug management has been introduced with the classification and prioritization of the identified bug using Machine Learning models (Hamouri et al., 2023). Evaluation of the models has been performed with accuracy ranging from 93% to 95%.

There is a wide range of usage of automation testing as it consumes less time and cost. Though the initial investment is more, with each run automation testing can guarantee a return on investment. The authors in (https://1902software.com/blog/cost-of-not-testing-software) (Kamaraj et al., 2023) have been able to deduce the dependency of cost of testing with respect to number of test cases executed manually and through automation. With the usage of testing tools, in automation testing there is a possibility of low code visibility - resulting in low code coverage and thereby drastic reduction in fault detection. The authors have also emphasized the key software components which can be easily automated: background processes, database entry, template operations, validation messages, checking the correctness of data search and many more.

Wide range of real time applications has been considered to understand the severity of software testing and importance of automation in software testing. In this regard, an application working on Google Android OS has been implemented in (Myrria De Albuquerque et al., 2023), with the use of Compatibility Test Suite tool for testing the application developed. The product was tested both by manual testers and through automation, with over 75% accuracy rate recorded for automation testing while 37% accuracy rate for manual testers. The results obtained have been able to prove the importance of automation testing over manual testing.

4) Minimizing Maintenance and Test Effort

Several approaches have been introduced over the years to ensure better quality software by performing testing and inspection regularly however without considering effort reduction. In this regard, the authors in (Elberzhager et al., 2012) proposed different areas eligible for effort reduction by performing a survey on over 4020 research articles of which final 144 articles were suitable for classification of existing approaches. The five areas which contribute towards better test effort reduction are: use of different test strategies, test automation, quality assurance techniques before testing, techniques used for predicting the defect prone areas and techniques used for reduction of test input. Test automation is the widely used approach for effort reduction (about 50%) where manual testing can be automated, with generation of automated test reports by using test automation tools, thereby reducing overall testing time and manual effort. Predicting techniques are used to predict the defect content and defect proneness - which provides an estimation of expected defects and areas of expecting a defect. The test input reduction approach considered the use of test case selection and prioritization approaches mainly employed in case of regression testing on the optimized test suite to save the testing time and improve the efficiency of testing. Quality assurance techniques have been less effective as they are performed before the testing process through code inspections and reviews, which may assist in the identification of defects which are less costly than compared to defects found at the later stage of testing. Deciding the test strategy is dependent on the type of application built; the type of modular approach followed in the application and hence can have less contribution towards effort reduction. The survey provided the different supporting areas of effort reduction, along with emphasis on integration of quality assurance techniques and prediction techniques for better test effort reduction.

Reduction of test effort in agile IT projects has been proposed by providing guidelines for minimizing maintenance of test automation (Sebastian, 2023). Guidelines like usage of small and minimal code snippets, loose coupling of the architectural components, with small unit interfaces that are suitable for automation testing have been listed in this work. Following these guidelines is necessary mainly at the maintenance phase, especially if regression testing is used. The analysis results here have shown the impact on software maintenance that is caused due to changes in the requirements. Though a series of guidelines have been listed here to minimize the maintenance effort, it is practically not possible to implement the guidelines considering the other factors of software development.

5) Metrics Used

The process of software testing is measurable, and several metrics have been used for the same. Commonly used metrics mean time between arrival of errors, density of errors, failure rate, test execution productivity, defect density and many more. Each of these metrics is independent of each other making it difficult for the evaluation of the software. A common, more homogeneous metric has been proposed in (Khan et al., 2013) mainly for measuring the quality attributes like correctness, reliability, flexibility, interoperability and usability. The new framework provided by the proposed work suggested mean time to failure (MTTF) and failure rate (FR) as the commonly used metrics as it is more suitable at different stages of testing. These metrics are the derivatives which include several other metrics as a combination, thereby facilitating the measurement of many quality attributes. The proposed work does not provide one suitable metric which can be used to measure the effectiveness of automation testing in terms of productivity.

Test scripting techniques have been proposed in (Hanna et al., 2013) with the main aim of achieving a high-quality system thereby reducing the test effort. Generation of the test scripts play a vital role in increasing the efficiency of testing and this can be carried out by linear scripting technique, structured and shared scripting technique, data - driven scripting technique, keyword-driven scripting technique and the proposed scripting technique which considers clerical activity and intellectual activity used during testing. When the automation scripts are generated through any of the scripting techniques, the speed of generating the test suite is faster, with fewer errors and less time consuming compared to manual testing.

With respect to the dataset collected, the software projects being considered are of the product and application type following the waterfall and agile model of software development. The choice of underlying model used is purely based on the aim and scope of the project under development. While the traditional projects follow the waterfall approach, where development happens mainly on the requirements given, which are subject to minimal changes (Royce, 1987), the agile based development has been used for applications with the scope for incremental development (Fowler & Highsmith, 2001). In agile - based systems, multiple processes within a project are run parallel to each other within every development process or sprint. The type of development model selected will in turn have its impact on the resource allocation which can be in the form of human resources, materials, financial resources and time. Since many types of resources are involved, resource allocation is the major problem to be addressed in the case of Agile System Development and to overcome this issue, Multi - Objective Decision - Making approach has been proposed in (Kaur et al., 2023) as the solution. This approach has been suitable for software development projects but is yet to be implemented for software testing and software maintenance projects.

Extensive study of the existing research work has provided a detailed insight into the importance of automation testing when compared to manual testing and the widespread use of automation testing in current software applications (Rafi et al., 2012). Major work has been carried out with the main aim of reducing the test efforts in terms of cost and time by using machine learning techniques in the generation of test oracles or using machine learning techniques for prediction of bugs or achieving maximum code coverage with minimum number of test cases along with the identification of maximum number of bugs. To achieve a true Return on Investment there is an ardent need to consider factors which are secondary but have an impact on increasing the productivity gain of the software projects. A detailed survey has been carried out for the identification of factors which contribute towards increasing the productivity gain (Bindu Bhargavi & Suma, 2022). The secondary factors identified are in terms of building a test team with team members having exclusive training in test automation, effective tool selection with respect to the underlying application, fixing a proper automation strategy, executing with respect to the automation test plan and selecting of the right test cases for automation. These factors play a vital role in achieving better automation thereby facilitating less cost, time and test effort.

4 Research Methodology

To obtain better understanding of the proposed work, sample data related to testing has been collected from IT industries CMMI (Capability Maturity Model Integration) Level 5 companies. These are the companies which have well defined processes that are effectively monitored and measured. Data related to testing has been collected with respect to applications ranging from banking to web applications. Each of these projects has been implemented using different technologies like JAVA,.NET framework, Cobol, Mainframe etc.

1) Data Collection

The application model used for the projects are either waterfall model or agile model with a scope for maintenance. Since each of these projects are real time IT applications - detailed insight into testing has been obtained by interviewing the developers and testers. A developer in a traditional software project plays a pivotal role in software development by identifying the key requirements from specification, designing the software, implementation and installation followed by testing the project.

Projects developed with respect to the agile model, would require the developer to produce high quality code by considering functional and non - functional requirements like readability, maintainability, security and performance. Once the project is released to the end users, maintenance and any further changes would require the project to be tested again before release. Projects with minor builds have been considered with test cases being generated manually and through automation. The projects are tested using regression testing techniques (Regression Testing?) and automation is performed using tools like Selenium and QTP which are licensed and open-source tools.

Since it is cost efficient to automate all the test cases which are generated manually, there is need to strategically select the test cases which are suitable candidates for automation. With every update or any changes in the code, it is necessary to get the conformance on the functionality of the software to ensure that there are no deviations in its functionalities or any errors after implementing the changes. To ensure this a set of test cases pertaining to the core functionality of the software are to be run each time a modification is done to the software. These are the primary set of test cases that are eligible for automation. Also, test cases which do not undergo any changes even after multiple runs but are necessary to check the functional, non - functional requirements are hence eligible for automation. When the manual test cases take long duration for execution, they can still be automated so that the test cases can be run in lights out mode as batch files without much intervention of the testers or requiring exclusive skills of the testers for further execution. When the test cases are executed in lights out mode, testers mainly ensure the execution of test cases associated with the essential functionalities of the project.

Test data collected with respect to over 30 projects are used for analysis and further derivation of productivity gain is done.

The main objective of this paper is to understand the importance of time, cost and effort in software testing. The proposed work also focuses on the importance of productivity gain with respect to software testing, primary and secondary factors which contribute towards achieving a better productivity gain - thereby reducing the time and cost spent on software testing ultimately gaining a better Return on Investment.

5 Results and Discussion

Automation testing is performed by the developers to achieve a wider coverage to detect bugs and errors during an early or later phase of development - thereby reducing the cost of failure, saving time during regression testing, facilitating in improving the resource productivity. Testing data of around 30 projects has been collected for analysis and estimation of productivity gain that can be achieved in testing. Test data related to around 10 projects under product development, 10 projects related to banking application and 10 projects related to retail, industrial application have been considered here. Since the applications under consideration are subjected to changes during development and at later stages, there is an ardent need for automating the test cases to ensure all the features of the software under development are intact and maintainable over the life of the application. The projects considered are minor build projects where the duration of project development ranges from 3 to 6 months. The projects undertaken with respect to product development have been built based on waterfall model with Quick Test Professional (QTP) as the testing tool (licensed). Projects considered under the banking sector have been developed using both waterfall and agile framework mainly to support the real time requirements of the end users. The automation tool used is Selenium which is an open-source tool. The test data contains details about the number of test cases designed manually, time taken for execution of these test cases, number of test cases automated which are derived from manually designed test cases, time taken for running the automation script along with number of times executed. The data collected is analyzed to understand the relation between time taken for manual testing and time taken for automation testing and deriving the gain in productivity using the dependent factors.

1) Data Analysis

From the entire data set collected, data analysis is started by the identification of dependent and independent variables. This is followed by understanding the relationship between the dependent variables and factors contributing towards increasing the productivity gain (Bindu Bhargavi & Suma, 2022). The applications considered here are banking, retail; industrial/product-based applications which are once developed and released to the customers may go through changes and maintenance over the due course if it's usage. This makes it necessary for the developer and tester to be aware of the underlying technology with efficient knowledge about automation tools and associated scripting languages. Quota sampling is performed for better analysis of data, where primary factors contributing towards the calculation of productivity gain are identified. These factors include time taken for manual testing and time taken for automation testing, number of cycles of execution and time taken for preparation of automation scripts.

Designing a test strategy with a suitable test plan is always the best practice to be followed during test automation. The selection of the test tool depends on the underlying knowledge about the application, as it facilitates in designing optimal number of test cases with maximum coverage. Many software testing tasks can be laborious and time-consuming to be done manually. In addition, a manual approach is not always effective in finding certain classes of defects. Test automation offers the possibility to perform these types of testing effectively. Figure 1 shows the number of test cases generated manually v/s the number of test cases generated by the usage of automation tools. The number of test cases generated depends on the type of project, number of developers and testers working on the project, complexity of the project and the type of software development model used. The graph further represents the functional equivalence that exists between the manual test cases and automated test cases, since manual test cases provide a basis on which automation test strategies are designed, here, testing tools are selected followed of the expected outcome. The pattern of manual and automation test cases facilitates in the visualization of distribution of test cases within a test suite, which signifies that on an average the number of manual test cases developed is 5 times the number of automation test cases that is used during automation testing. Since the projects considered here are developed for business purposes, the comparison between the manual and automation test cases is done on a common scale irrespective of the product, banking and retail domains.

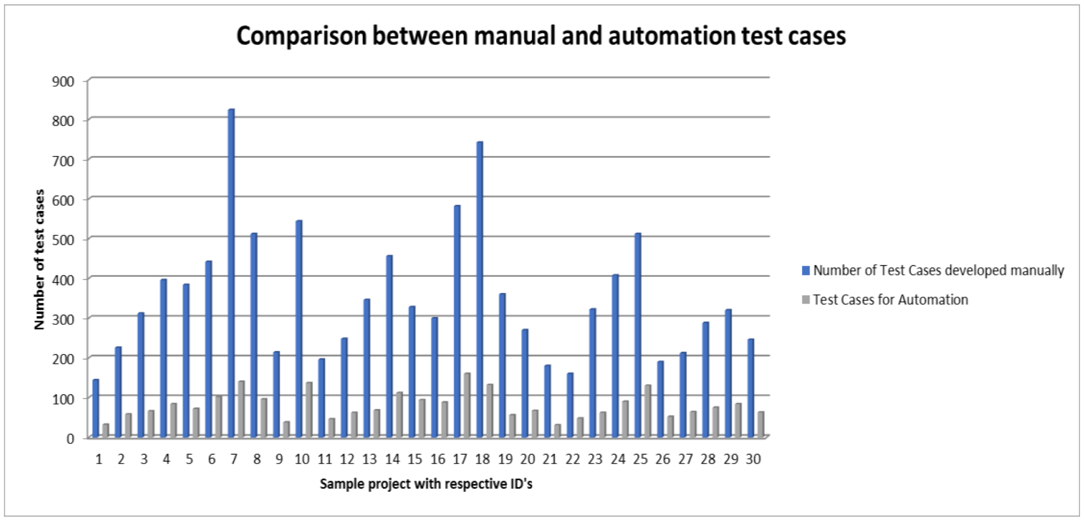

Figure 1: Number of Test Cases Generated Manually v/s Number of Test Cases Generated through Automation: X Axis Denotes the Projects and Y Axis Denotes Number of Test Cases

The generation of automation test cases is done based on a detailed test strategy and suitable test plan, to facilitate better regression testing. A detailed test plan further facilitates the tester and developers towards building batch files for the execution of test cases with minimal or no human intervention. Such batch files can be run in lights-out-mode where the essential and most critical test cases are run with less test cycle time.

2) Identification of Defects

During manual testing, the test cases are generated by considering requirements specification, where most of the boundary conditions or edge cases may not be taken into consideration. Automation testing overcomes this condition by repetitive execution of test cases facilitating the detection of bugs at an early stage of software development. Automation testing when used during regression testing works even more effectively by keeping a check on the deviations caused with respect to existing functionality of the software under development. Since automation testing can be documented, every logged detail would in turn better detection and fixing of the bugs in the software. It is not good practice to automate all the test cases that are generated manually as it is not economical and consumes a lot of time. Of all the test cases available, test cases that are associated with the core functionality of the software are to be automated to check for any deviations from the usual behavior of the software, over the due course of usage and maintenance. Test cases which take longer execution time when manually executed are also subject to automation so that the next run of the same test case does not require any human intervention and can be executed successfully. Direct correlation existing between the critical factors are identified and the relationship between them is analyzed for achieving better test automation measured in terms of the metric termed as productivity gain.

3) Impact of the Critical Factors

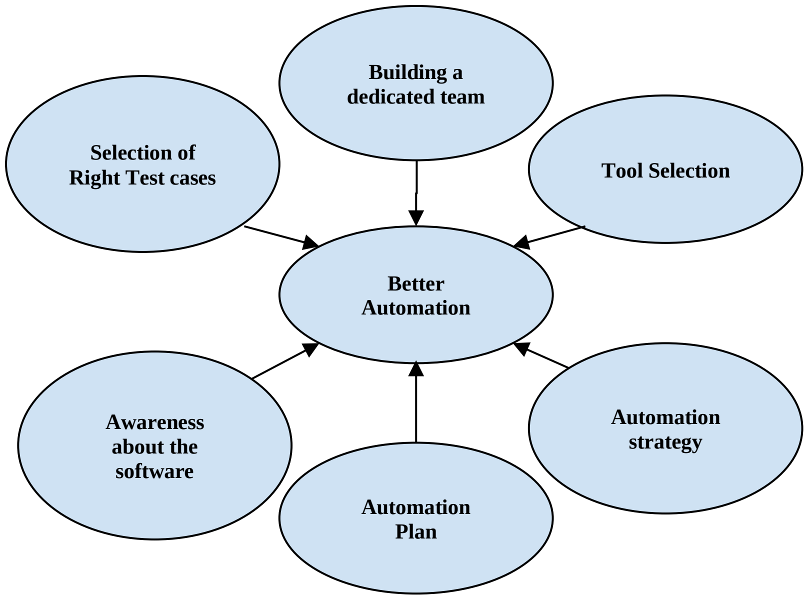
Building a dedicated team
Selection of Right Test cases
Tool Selection
Better Automation
Awareness about the software
Automation strategy
Automation Plan

Figure 2: Critical Success Factors Identified Contributing Towards Better Automation

Automation is termed to be better, when higher test efficiency and better productivity gain are achieved. Testing efficiency is the average number of tests that can be run for an hour of tester time. It is the rate at which specific testing techniques are used for revealing the bugs of the system or the underlying application. Achieving better test effectiveness reduces the testing cost at a later stage along with building customer satisfaction. This can be achieved by devising an automation plan that is most suitable for the underlying application with a team of efficient developers and testers. The skillset of the developers and testers also plays an important role in devising the test plan with an appropriate test strategy corresponding to the different levels of testing applicable for the system/project under test. Replacing manual testing with automation time can save the testing time and test effort represented in terms of productivity gain. Along with improving productivity and efficiency, test automation also enhances team spirit and provides the testers and developers with more time for improving the test plan in a much more efficient manner. Critical Success Factors Identified Contributing Towards Better Automation shown in Figure 2.

The major advantage of automating the test cases is that the process of testing can be easily repeated with any changes to the software/application/project as the time spent on automation creation can promise a better return on investment. Automated software testing can reduce the time to run repetitive tests from weeks to hours. In the projects considered, the costs for manual testing can increase with each new build but not in the case of automated tests. Though the initial cost of automation testing is high, there is a gradual increase in the return on investment. This is a significant time-saving that translates directly into cost savings. In this regard, a new metric to measure the effectiveness of automation testing has been proposed termed as productivity gain: where reduction in the test effort can be achieved by a better productivity gain. Since the identified critical factors have an impact on reducing the test effort, productivity gain can be determined by the equation (1) is given as,

$$\text{Productivity gain} = (t_m * n) - ((t_a * n) - t_s) \qquad (1)$$

Here,
- t_m is time taken to run a manual test script
- t_a is the time taken to run an automation test script.
- t_s is the time taken for writing the automation script
- n is the number of test cycles to be considered

The calculation of productivity gain has been carried out according to the proposed mathematical model.

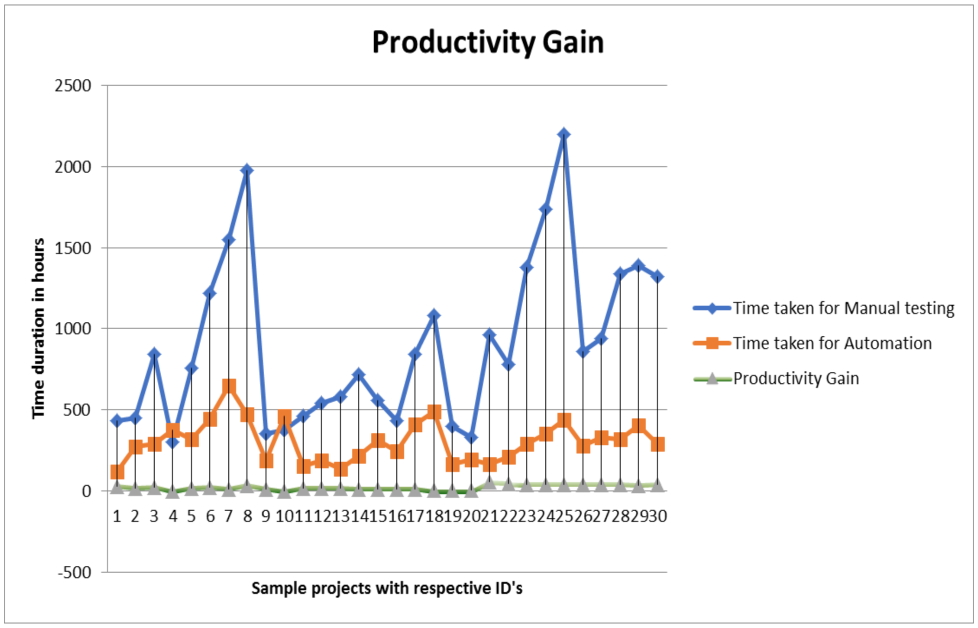

Figure 3: Productivity Gain Measurement v/s Manual Testing v/s Automation Testing

The results obtained when applied on the sample data are as represented in Figure 3. This graph represents the correlation existing between the three primary variables used in productivity gain calculation viz. time taken for manual testing, time taken for automation and time taken for automation of the test scripts with respect to productivity gain.

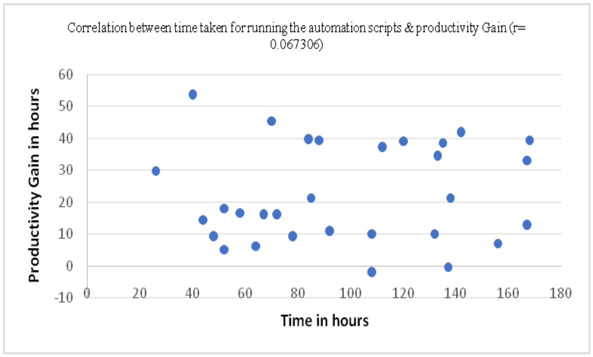

Figure 4: Represents Correlation between Time Taken for Running Automation Scripts & Productivity Gain

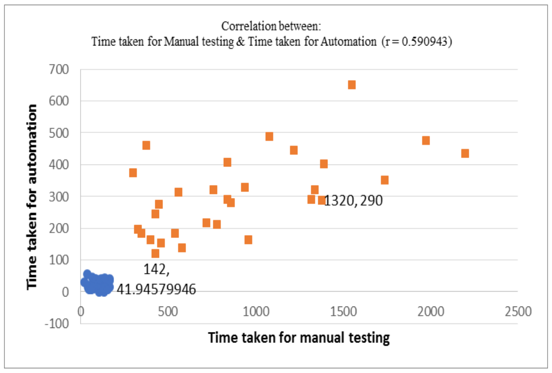

Figure 5: Correlation between: Time Taken for Manual Testing & Time Taken for Automation

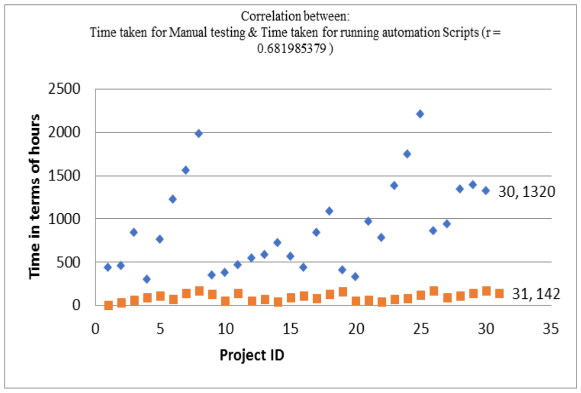

Figure 6: Correlation between: Time Taken for manual testing & Time Taken for Running Automation Scripts

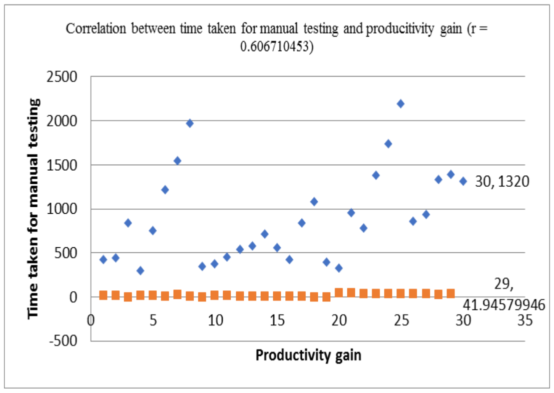

Figure 7: Correlation between Time Taken for Manual Testing and Productivity Gain

The primary analysis of the data set resulted in the identification of these primary variables, which further must be evaluated and verified using correlation analysis to ensure the correctness of the selected primary variables. While automating the test scripts, the time duration taken for scripting is more than compared to execution of the automation scripts. So initially when the automation scripts are run, a breakeven condition is achieved only when test automation is performed beyond the scripting time. Once the breakeven condition is met, any number of automation scripts run will result in a reduction in the time taken for automation resulting in productivity gain. With each release of the project version, scripting time is reduced, increasing the productivity gain. Test automation is termed to be seamless when the test scripts are run as batch files with minimal human intervention, executing the files in lights-out-mode. The results of the correlation analysis performed are as represented in the graphs represented by Figures.

Figure 4 represents the correlation existing between the time taken for running the automation scripts and the productivity gain calculated with the coefficient value r = 0.067306. The value r represents a very weak positive correlation existing between these two variables. Figure 5 represents the correlation existing between the time taken for manual testing and time taken for automation testing with the correlation coefficient value r = 0.590943. The value r represents a moderate positive correlation existing between these two variables. Figure 6 represents the correlation existing between the time taken for running the manual testing and time taken for execution of the automation scripts with correlation coefficient value r = 0.681985379. The value r represents a moderate positive correlation existing between these two variables. Figure 7 represents the correlation existing between the time taken for manual testing and productivity gain with correlation coefficient value r = 0.606710453. The value r represents a moderate positive correlation existing between these two variables.

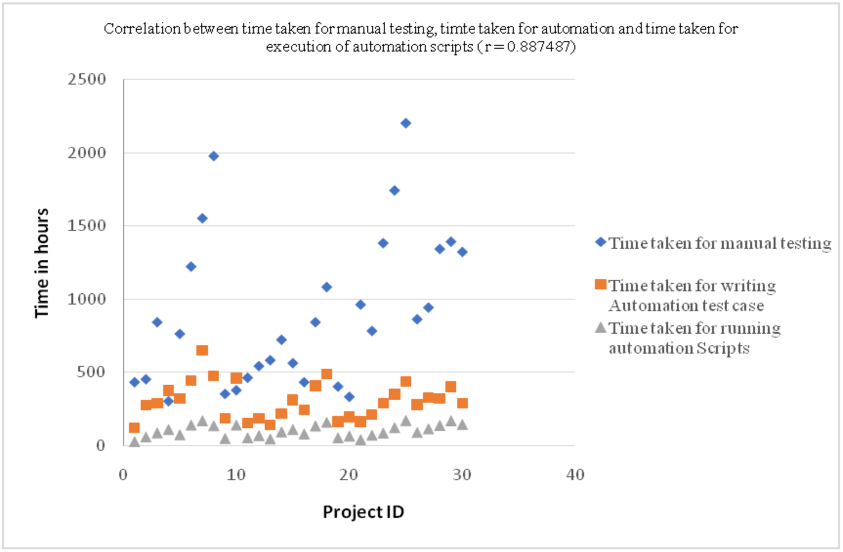

Figure 8: Correlation between Time Taken for Automation and Time Taken for Running the Automation Scripts

Figure 8 represents the correlation existing between the time taken for generation of the automation scripts and time taken for execution of the automation scripts with correlation coefficient value r = 0.887487. The value r represents a very strong positive correlation existing between these two variables. Since correlation existing between the different combinations of variables is in the range of 0.6 to 0.8, the contribution of each of these variables in the calculation of productivity gain metric is hence justified.

6 Conclusion

This research has delved into the realm of software testing, elucidating the nuanced significance of both manual and automated approaches within the context of CMMI Level 5 companies. The findings highlight the indispensable role of software testing as the linchpin of quality assurance in the software development life cycle. By meticulously analyzing over thirty projects spanning diverse domains and technologies, the study has illuminated the multifaceted nature of software testing. The projects' adherence to both agile and waterfall models add layers of complexity, mirroring the real-world challenges faced by modern development teams. The interviews with developers and testers have unraveled crucial insights into the pivotal factors influencing productivity, emphasizing the need for a harmonious blend of skills, experiences, and training. Automation, identified as a strategic necessity, emerges not merely as a time-saving mechanism but as a linchpin for ensuring the integrity, maintainability, and functionality of software throughout its lifecycle. The careful selection of test cases for automation, highlighted in the research, shows a cost-efficient approach that accelerates efficiency and minimizes human intervention. The introduced metric of productivity gain provides a quantitative lens through which the efficacy of automation is measured. The findings unveil a significant break-even point, signaling the juncture where automation substantially reduces testing time, contributing to enhanced productivity. This revelation is not only pivotal for organizations aiming to optimize their testing practices but also serves as a beacon for navigating the evolving landscape of software development. In essence, the present study encapsulates the essence of software testing, elucidating its critical importance in the pursuit of software quality. As the industry continues to evolve, these insights provide a comprehensive understanding of the intricacies involved, guiding organizations towards strategic decisions that optimize both time and costs in the pursuit of a more productive and efficient software development process.

Competing Interests – No Competing Interests.

Funding Information – Not Applicable.

Author contribution - All authors contributed to the study of conception and design.

Material preparation, data collection and analysis were performed by Bindu Bhargavi S M and Dr. Suma V.

e-search Involving Human and /or Animals – Not Applicable.

Informed Consent – Not Applicable.

Note: On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

[1] Automated Software Testing. https://www.test-institute.org/Automated_Software_Testing.php

[2] Bindu Bhargavi, S. M., & Suma, V. (2022). A survey of the software test methods and identification of critical success factors for automation. *SN Computer Science*, *3*(6), 449. https://doi.org/10.1007/s42979-022-01297-5

[3] Braga, R., Neto, P. S., Rabêlo, R., Santiago, J., & Souza, M. (2018). A machine learning approach to generate test oracles. *In Proceedings of the XXXII Brazilian Symposium on Software Engineering*, 142-151.

[4] Butt, S., Khan, S. U. R., Hussain, S., & Wang, W. L. (2023). A conceptual model supporting decision-making for test automation in Agile-based Software Development. *Data & Knowledge Engineering*, *144*, 102111. https://doi.org/10.1016/j.datak.2022.102111.

[5] Elberzhager, F., Rosbach, A., Münch, J., & Eschbach, R. (2012). Reducing test effort: A systematic mapping study on existing approaches. *Information and Software Technology*, *54*(10), 1092-1106.

[6] Ferdiansyah, D., Isnanto, R. R., & Suseno, J. E. (2023). Strategy Indicators for Secure Software Development Lifecycle in Software Startups Based on Information Security Governance. *Journal of Internet Services and Information Security, 13*(4), 104-113.

[7] Fowler, M., & Highsmith, J. (2001). The agile manifesto. *Software Development*, *9*(8), 28-35.

[8] Hamouri, S. K., Shatnawi, R. A., AlZoubi, O., Migdady, A., & Yassein, M. B. (2023). Predicting Bug Severity Using Machine Learning and Ensemble Learning Techniques. *In IEEE 14th International Conference on Information and Communication Systems (ICICS),* 1-6. https://doi.org/10.1109/ICICS60529.2023.10330494.

[9] Hanna, M., El-Haggar, N., & Sami, M. (2013). Reducing Testing Effort using Automation. *International Journal of Computer Applications*, *81*(8), 16-21.

[10] Hemmati, H. (2015). How effective are code coverage criteria?. *In IEEE International Conference on Software Quality, Reliability and Security*, 151-156. https://doi.org/10.1109/QRS.2015.30

[11] Hershkovich, E., Stern, R., Abreu, R., & Elmishali, A. (2021). Prioritized test generation guided by software fault prediction. *In IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 218-225.

[12] https://1902software.com/blog/cost-of-not-testing-software

[13] https://www.functionize.com/automated-testing

[14] https://www.guru99.com/regression-testing.html

[15] Jayasree, V., & Baby, M. D. (2019). Scientometrics: Tools, Techniques and Software for Analysis. *Indian Journal of Information Sources and Services, 9*(2), 116–121.

[16] Jones, C. (2011). Software Quality in 2012: A Survey of the State of the Art. *Nancook Analytics LLC.* http://sqgne.org/presentations/2012-13/Jones-Sep-2012.pdf.

[17] Kamaraj, K., Lanitha, B., Karthic, S., Prakash, P. N., & Mahaveerakannan, R. (2023). A Hybridized Artificial Neural Network for Automated Software Test Oracle. *Computer Systems Science & Engineering*, *45*(2), 1837-1850. https://doi.org/10.32604/csse.2023.029703.

[18] Kaur, J., Singh, O., Anand, A., & Agarwal, M. (2023). A goal programming approach for agile-based software development resource allocation. *Decision Analytics Journal*, *6*, 100146. https://doi.org/10.1016/j.dajour.2022.100146.

[19] Khan, M. S., Khan, N., Khan, M. A., & Javed, M. A. (2013). A New Approach for Reducing the Testing Effort.

[20] Kumar, D., & Mishra, K. K. (2016). The impacts of test automation on software's cost, quality and time to market. *Procedia Computer Science*, *79*, 8-15.

[21] Leu, F.Y., & Tjoa, A.M. (2014). Guest Editorial: Emerging Software Reliability and System Security Technologies. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, *5*(1), 1-3.

[22] Myrria De Albuquerque, A., Barbosa, H., Lancellotta, P., Santos, J., & Sousa, J. (2023). Automating Android Rotation Vector Testing in Google's Compatibility Test Suite Using a Robotic Arm. *In Proceedings of the 8th Brazilian Symposium on Systematic and Automated Software Testing*, 54-63.

[23] Offutt, J. (2023). Test Automation: From Slow & Weak to Fast, Flaky, & Blind to Smart & Effective. *In IEEE Conference on Software Testing, Verification and Validation (ICST)*, 11-11. https://doi.org/10.1109/ICST57152.2023.00009

[24] Oleksandr, K., Viktoriya, G., Nataliia, A., Liliya, F., Oleh, O., Maksym, M. (2024). Enhancing Economic Security through Digital Transformation in Investment Processes: Theoretical Perspectives and Methodological Approaches Integrating Environmental Sustainability. *Natural and Engineering Sciences*, *9*(1), 26-45.

[25] Perera, A. (2020). Using defect prediction to improve the bug detection capability of search-based software testing. *In Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, 1170-1174.

[26] Planning, S. (2002). The economic impacts of inadequate infrastructure for software testing. *National Institute of Standards and Technology*, *1*(2002).

[27] Rafi, D. M., Moses, K. R. K., Petersen, K., & Mäntylä, M. V. (2012). Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. *In IEEE 7th International Workshop on Automation of Software Test (AST)*, 36-42.

[28] Riola, M. (2023). Test automation in video game development: Literature review and Sound testing implementation. Master's Degree Thesis.

[29] Royce, W. W. (1987). Managing the development of large software systems: concepts and techniques. *In Proceedings of the 9th International Conference on Software Engineering*, 328-338.

[30] Sebastian, Ö. (2023). Software Test Automation: A qualitative study on optimizing maintenance in test automation. Bachelor's thesis.

[31] Umar, M. A., & Zhanfang, C. (2019). A study of automated software testing: Automation tools and frameworks. *International Journal of Computer Science Engineering (IJCSE)*, *6*(217-225), 47-48.

[32] Verified Market Research. https://www.netguru.com/services/market-research

[33] Wiklund, K., Eldh, S., Sundmark, D., & Lundqvist, K. (2017). Impediments for software test automation: A systematic literature review. *Software Testing, Verification and Reliability*, *27*(8), e1639. https://doi.org/10.1002/stvr.1639

[34] Yang, C. T., Chen, S. T., Lien, W. H., & Verma, V. K. (2019). Implementation of a Software-Defined Storage Service with Heterogeneous Storage Technologies. *Journal of Internet Services and Information Security*, *9*(3), 74-97.

Authors Biography

S.M. Bindu Bhargavi, has been serving as an Assistant Professor in the Department of Information Science and Engineering, Dayananda Sagar College of Engineering, Bangalore, since July 2015. She has obtained her Master of Technology degree from Sri Venkateshwara College of Engineering affiliated to Visvesvaraya Technological University in 2014. She has published several research articles and book chapters in various International Conferences and Journals. Her research interests include Software Engineering, Software Testing, Machine Learning, Deep Learning, Cloud Computing and Discrete Mathematics.

V. Suma, has been serving as Vice Principal & Head, in the Department of Computer Science and Design, Dayananda Sagar College of Engineering. She has been working in this institution for the past 27 years. She has been recognized as a research supervisor under Visvesvaraya Technological University and many other esteemed universities. She has published several research articles and book chapters with around 1459 citation count. Her area of research includes Software Engineering, Cloud Computing, Machine Learning, Deep Learning, Data Mining, Internet of Things, Computer Vision and Image Processing.

R. Vijaya Arjunan, has been serving as an Additional Professor in the Department of Computer Science and Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, since July 2010. During his tenure at Manipal Institute of Technology, he was on deputation to the School of Engineering and IT, Manipal, Dubai campus, from 2014 to 2017. He obtained his Master of Engineering and Ph.D. in Computer Science and Engineering from Sathyabama Institute of Science, Technology and Sankara University in 2005 and 2013 respectively. He has published around 50+ research articles in various International Conferences and Journals. His research interests include Computer Vision, Image Processing, Machine Learning, Deep Learning, and Data Mining. He is a life member of Broadcast society of India (BES).