A survey on software test automation return on investment, in organizations predominantly from Bengaluru, India

S. Reine De Reanzi and P. Ranjit Jeba Thangaiah

Abstract
Software industry has adopted automated testing widely. The most common method adopted is graphical user interface test automation for the functional scenarios to reduce manual testing and increase the repeatability. Reducing execution time and redundancy to achieve quality "go/no go decisions" provides rational for the executive management to allocate funds to adopt automation and invest in the setup including people, process, and tools to achieve faster time to market. There are a variety of practices engaged by testers, like frameworks, tools, methods, procedures, models, and technologies to achieve automation. Nonetheless, the actual effectiveness in terms of return on investment (ROI) is not known, though there are various formulas to calculate ROI of test automation. The factors that determine the ROI are maturity of test automation, purpose, or intent, picking right tests for automation, knowledge of the tester to derive test coverage, domain expertise, defining right metrics like defects found by automation runs, cost versus reduction of time, labor versus quality of scripts, repeatability, and results. These factors form the base of the questions designed for the survey. The paper presents a survey and analysis to understand the ROI of test automation from industry test professionals from both product and services organizations.

Keywords
Automation return on investment, testing, Bengaluru, survey, quality assurance

Date received: 11 June 2021; accepted: 25 October 2021

Introduction
Executives in the commercial IT industry strive to automate all test cases to save time, get repeatability, identify defects early, increase quality, and cut cost.1 The nature of software testing activity is a support function, and hence, it is treated as cost center. Therefore, the inclination to automate all test suites to minimize the cost is widely common.2 100% automation is rarely achieved in the organization3 because of various impeding factors. In a survey of industry professionals, only a meager 6% think that 100% automation is possible.2

Often the first step to automation by software companies is to embrace functional test automation, which involves the graphical user interface (GUI). This takes considerable investment, and the challenges are plenty.4 There are a lot of research and case studies that describe various approaches to increase benefits and reduce cost. There is still some way to go.5 The important factor where the management underestimates the cost is when it comes to maintenance of the suite.6 The IT services industries provide automation services to the software product organizations. This is one of their mainstream businesses. While the product organizations do own automation teams, it is largely viewed as an activity that can be outsourced. The success depends on achieving repeatability, uncovering defects, and faster execution while keeping their maintenance cost low. This has been the nemesis in the SDLC cycle.

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
There are numerous research works in this area on automation tools, methodologies, comparison of frameworks, return on investment (ROI) computation on using certain tools over other tools, and low code—no code automation. But there is no real answer in terms of ROI on automation for the organization.

The method proposed here is to create a survey questionnaire and gather the answers from the industry professionals and analyze the results, with the current pay scale (from PayScale data as of 2021) and derive the real scenario in the industry. Also, from the data collected, the expectations of the services and the product industry on ROI of automation are illustrated using correlation.

The paper elicits review of literature on automation, its activities, frameworks, tools, ROI, scope of IT industry, and so on. Problem description, research method, sampling distribution mechanism, survey design, survey validity, survey execution and data collection method, response quality, PayScale data for automation engineers, data analysis, response variation, observations, results, and conclusion are discussed.

Review of literature

The systematic review of literature discusses the various aspects, factors and benefits of test automation, and the works of researchers in terms of ROI yield. Efficiency, effectiveness, and precision are the three main factors that determine the success of automation in any organization. Efficiency means optimized test coverage, execution time, write once run anywhere and maintenance. Effectiveness means the right tests that are beneficial to the customer’s usage, and precision means able to repeat the correct results. The review of literature is carried out to present the various researches that are already published. They are classified and referenced according to the various categories that are highlighted in italics. These served as a base for designing the survey questionnaire. There are a lot of generic automation framework for different applications and References 10–14 provide the, research that is published in this area. Web automation framework is an action-related framework and it is UI driven. References 15–18 provide the research that is published in this area. Device automation framework has been gaining popularity with device farms and device automation frameworks to enable mobile device automation frameworks. Choosing the right GUI tool with accessibility, federated, and industry case studies are discussed in References 2, 24–31. Automation frameworks based on GUI Models are expected to ease the automation activity. Dynamic test generation methodologies like codeless automation and auto generation of tests have been in research and being adopted by the industry professionals. Model-based automated test generation is used for high coverage, and keyword-driven GUI tests are user-friendly. UML-based test generation for textual and visual patterns for generating tests is used for functional test coverage. With DevOps, the adoption of continuous integration and development became imperative for automated code deployments. The same is now extended as QAOps for continuous verification for executing automated regression test suites. Unit test cases provide the code coverage and they are even automated and autogenerated unit based on models. Behavior-driven test for graphical user interface tests using Cucumber and Gherkin is widely adopted for mobile GUI automation. ML/AI test framework, the recent researches present ways to create machine language–based automation of tests. Low-code automation is the latest adoption by the industry for building blocks of code that can be easily assembled to quickly build workflows and functional scenarios and use cases. The main advantage is it saves time because the blocks are reusable. The same concept is adopted for testing for non-technical, citizen testers to write automation scripts. Reusable libraries framework and reusable test libraries reduce maintenance effort and maximize test coverage. The services organizations employ a test factory mechanism where it is persona-based automation. An experience report on industry practice is discussed. The product code is refactored every now and then to ensure scale, technology adoption, and optimization. Similarly, there is research for identifying tech debt in test automation and applying software design principles for test automation development. The criteria for when and what to automate are decided by the factors eligibility, priority, cost, coverage, end user usage, quality, maintainability, and time. Certainly, there are various impediments for automation, and the factors contributing to hindrance of a robust test automation suite are discussed in Reference 81. Automation framework challenges typically are usability and metrics like test coverage, test effectiveness, usability, and defects found. It is interesting to note that automation practice is researched by region. Industry practice adopted by professionals from various regions is presented in References 83–85. Again automation practice by industry verticals and its study on different organizations are highlighted in References 86–88. Similarly, automation practice by infrastructure, for example, DevOps-related software test automation, is discussed in Reference 50. Benefits and limitations of automation are measured by cost, quality, time, effort, maintainability, and return on investment (ROI). With the microservices, serving different clients and applications running on different multitenant servers, rises a need for distributed test agent framework to enable cross platform frameworks to function. Automation metrics define the usefulness of the automation and measure the quality of the product with cost and time. Maintenance involves effort and the ease of maintaining the automated suite. Automation complexity like greedy-based test suite reduction methods is mostly still at the research level. Automation test coverage ascertains the quality of tests that yield maximum
benefits. Automation is also used as decision support system to support go and no go decision for production code promotions. Automation effectiveness is measured by the number of defects found and quality of software under test. Automated suite optimization helps in reduction of tests to minimize effort and time and results in overall quality for the software under test. Automation ROI estimation helps to justify the time and material spent on the automation suite with respect to quality.

Problem description

Every software company, with faster to market strategies in place, strives to adopt automation to provide quality by investing in resources like tools, frameworks, infrastructure, consultation, and people. The average testing spends in the last 5 years according to Statista are 28% of the overall IT spend. Also, the estimated market size for test automation by the year 2026 is $60.4 billion.

One of the main focuses of automation is to achieve customer- and user-centric test coverage with repeatability and reduced time, catch defects early, and save cost. Unfortunately, achieving ROI requires knowledge in test coverage, maintenance of suite, and setting automation goals. The current trends in automation are all about tools and number of tests automated.

Continuing with this prevents the focus on the test coverage, test optimization, test effectiveness, etc. Developing more informed KPIs for test automation could help the software organizations to measure the cost and saving to derive the ROI on automation spend. The proposed paper aspires to present the current understanding of industry professionals on test automation and what might be the present ROI for the software organizations from the sample. For collection of data, a specifically designed survey questionnaire is created to capture present usage and metrics and circulated via electronic medium targeted to the current industry professionals who practice/manage test automation. The Conclusion section presents the results with required charts and analysis.

Research method

The research method adopted is the survey research method because it requires to collect data from various software organizations on their current practice, observations, and results. This approach collects the qualitative and quantitative primary data by self-reported measures, on the understanding of test coverage and actual automation coverage from a sample of industry professionals from their respective organizations. The secondary data for cost of tools are referred from the top-used tools website. The secondary data for cost of test automation professional is collected from PayScale website. The experimental data is derived from the primary data to correlate unit test coverage to functional coverage, automation ROI from the perspective of industry test professionals who work for product organizations versus services organizations. The analysis of data from industry practice professionals presents analysis for the questions—“is there an ROI for automation?”, “Is there a difference in expectations—product versus services” because it is a cost for product organizations and revenue for services organization.

Survey design

As stated in the problem statement that IT spends 28% of total IT budget on testing activities with a major push toward test automation. The survey questions are designed in such a way the operations of the test activities are questioned with a focus on concepts like test coverage, test effectiveness, test strategy, knowledge of test professionals, tools used, cost of test automation, and type of industry (Table 1). The actual survey questions are given in the appendix-1 and the actual survey responses are given in appendix-2.

Sampling and distribution mechanism

Scope of IT industry

This section depicts the target organizations and the survey execution (Figure 1). The software companies registered with Ministry of Corporate Affairs (MCA.gov), India, and have presence in Bengaluru, including the international software organizations represent the target population of the survey. The survey was a paid survey and researcher self-funded the research.

Survey execution and data collection methodology

The survey questions are added to a Google Form. The sample selected are the test professionals from the software industry from this region. The form was circulated to a. email list of industry professional, b. shared via WhatsApp to list of phone numbers of industry professionals belong to this region, and c. to Bengaluru-based testing groups on Linkedin. The phone number was an optional field; if the respondent wanted to be paid, they chose to provide the phone number and was paid INR 50, for each successful response. The survey duration was from 23 March 2021 to 29 April 2021.

Survey validity

The survey was sent to 118 industry professionals, and 54 responses are received from the target commercial software organizations from the target region.
Quality of survey questions

The design of survey concentrates on concept versus practice-centric questions to draw the response toward the ROI, as explained in the Survey Design section. Some of the respondents conveyed their appreciation on quality of questions verbally, as well as in the comments section of appendix-2. Few respondents did require help in understanding the concepts to answer the survey. The tone of the questions is maintained neutral to eliminate any bias where most of the questions could be answered descriptive and if they are multiple choice, there was a choice for open-ended answer.

Quality of response

The responses when tabulated are found to be 95% good to do further data analysis. The 5% of data cleanup had to be done in the following areas like a. 15 responses for unit test % were left blank (here, an average is considered as 20%), b. 2 responses for organization name are left blank, c. 9 responses for test coverage are No (This factor is considered as limited knowledge for test coverage among industry test professionals), and d. 2 outlier responses for ROI, namely, 300% and 150%. Hence, out of 54 responses, 49 responses are considered with categorizing them into product organizations and services organization. However, this research also presents the outliers data in comparison with the cleaned-up data, wherever it is required.

Categorization of responses

The responses are primarily categorized into product and services organizations based on the organization name mentioned by the respondents. This helps to understand aspects of testing activities from different point of views from the industry professionals. If they are part of product organizations, then they are the owners of quality for the product. If they are part of services organization and provide engagement to support testing activities of a product organization, they act as support function. The other major contributor to the difference in point of view is it is an expenditure/investment for a product organization, whereas for services, it is this activity that brings revenue. Since the motivation and goals may vary depending on the engagement model they belong to, the responses are categorized into responses from industry professionals working in product organizations and services organizations. The one-way ANOVA results (Figure 2) notice that \( p \)-value is 0.93, \( p > 0.05 \), and hence, the means are equivalent between the groups. Also, F-value < F-crit value. Hence, accept the null hypothesis.

Data analysis, observations, and inferences

Response variation

31 numbers of responses are from respondents working in product organizations, 23 numbers of respondents are from
services, organizations, and 2–3 numbers of respondents are from the same organization often large but different divisions. The (Figure 3) charts show the organization type and the years of experience from the survey responses. For our research purposes, the IT education and IT banking are custom solutions and mostly employ people for their own back-office purposes—hence the classification as product. Interestingly, the product organization has more experienced professionals than the services industry. This may be due to the engagement model of the services sector to be profitable, where the professionals are trained and deployed in a client organization (typically a product organization).112

**Average of factors from respondents**

The averages of all the factors collected from the respondents are stated below (Table 2). The averages are shown with and without outliers for comparison purposes. The standard deviation in the case of number of bugs found, years of experience, and team count are high because of the nature of the mix of organization types. The rest of the data is good with less deviation.

It is observed that the test coverage, bugs found, and ROI yield are low. Besides, the maintenance (fix of script failures and change of scripts due to application changes) is high at 22%–30%. In the coming sections, each factor is analyzed separately, in combination with related factors to further examine them to deduce if the above factors have any impact on the ROI yield.

**Number of defects versus automation team count**

The following Figure 4 shows that irrespective of the team count, the number of defects found is mostly below 10 and concentrated near 5. Notice that few professionals from services have quoted beyond 10.

| Average of factors from respondents | μ with outliers | μ without outliers | σ | Min value with outliers | Max value with outliers | Product | Services |
|-----------------------------------|----------------|-------------------|---|------------------------|------------------------|---------|----------|
| % app failures post deployment    | 28%            | 29%               | 0.06 | 0                      | 90%                   | 30%     | 29%      |
| % of script failures post deployment | 18%           | 19%               | 0.02 | 0                      | 70%                   | 16%     | 22%      |
| % of unit test                    | 33%            | 30%               | 0.3  | 0                      | 90%                   | 29.02%  | 33%      |
| Team count                        | 9              | 10                | 5    | 2                      | 20                    | 10      | 8.7      |
| Experience of respondents         | 11             | 10.7              | 4    | 3                      | 20                    | 11.9    | 9.2      |
| No of bugs found via GUI automation | 4.6           | 4.3               | 3    | 0                      | 20                    | 3.8     | 5        |
| % of test suite automated         | 63%            | 61%               | 0.1  | 10%                    | 100%                  | 66%     | 54%      |
| % of test coverage                | 48%            | 44%               | 0.3  | 0%                     | 100%                  | 51%     | 36%      |
| % ROI of GUI automation           | 52%            | 43%               | 0.2  | 0%                     | 300%                  | 44%     | 43%      |

![Figure 2. Response variation with ANOVA table.](image)

![Figure 3. Response variation with box plot and ANOVA table.](image)

![Table 2. Average of factors from respondents.](table)
Return on investment yield from overall responses

The ROI yield, as quoted by the industry professionals, is plotted alongside the test coverage, and failures due to application changes and script failures. The maintenance is the overhead and contributes to the robustness of the suite. Failures are main factors that contribute to the stability of the automated suite. Test coverage contributes to the confidence of the suite. Hence, they are important in determining the ROI. The below charts compare the factors with and without outliers.

As observed in Figure 5, the data for outliers shows that where the test coverage is high and the failures are low, the ROI is more, especially the outliers Verizon with 300% and other with 150%. Their primary suite is API based and not GUI based and hence is treated as outliers.

As observed in Figure 6, the data without outliers shows that the trend of ROI range is between 26% and 51%. There

Figure 4. Number of defects versus automation team count.

Figure 5. ROI yield from overall responses with outliers.

Figure 6. ROI yield from overall responses without outliers.
are a few interesting observations: a. Where the failures are low and test coverage is high, the ROI is also high and vice versa and b. few industry professionals have quoted otherwise. Let us further dissect the data in the next section to understand if the organization category or any other factor is a contributor to this.

A mathematical computation on correlation is shown in Figure 7 to support the above observations. Note that there is an overall positive correlation between test coverage and ROI yield and negative correlation between ROI yield and failures due to app changes, script failures.

Clarity in understanding of “return on investment on test automation” by respondents

This section aims to see if the understanding of the industry professionals is in the right direction, especially with categories, namely, product and services. Figures 8 and 9 call out the difference in responses from product and services professionals. Figure 8 represents the responses from professionals from product organizations and Figure 9 represents the responses from the services organizations.

The observations from the sample are tabulated in Table 3. The table presents the results of the statistical analysis from the data with various combinations to understand the responses better from the initial aim of the concepts used for the survey design. This table represents the difference between inference of sample collected from industry professionals from product and services organizations.

A mathematical computation on correlation is shown in Figure 11 to support the above observations. For product, there is a positive correlation of ROI with coverage, defects, unit tests, team count, years of exp, % of suite automated, and 95% pass achieved and a negative correlation of ROI with failures on both app changes and script issues. For services, there is a negative correlation of ROI with team count and failures and interestingly failures due to app changes are positively correlated.

From the response it is found that ∼82% mentioned GUI suite is not productive without unit tests (Figure 12) and

---

**Figure 7.** Correlation between ROI and other factors (with and without outliers).

**Figure 8.** Clarity in understanding of ROI on test automation—product organizations.
∼57% mentioned GUI suite is not productive without unit tests (Figure 12). These unit test cases help to generate technical wealth.124

Finally, the responses are sorted to get the responses with high (~70%) ROI. The following Figure 13 highlights the gap in clarity on test automation. In the case of product, low failures, high coverage, more scripts automated, and high pass percentage contribute to high ROI. In the case of services, few responses with high failures, not known coverage, and average number of scripts automated cannot contribute to high ROI, but then the respondents have quoted high ROI.

Table 3. Results from various statistical analyses of the responses.

| Professionals from product organization | Professionals from services organization |
|-----------------------------------------|-----------------------------------------|
| ~64% know the % unit test for the application under test (Figure 8). | ~57% know the % unit test for the application under test (Figure 9). |
| % of suite automated is concentrated from 60% to 80% (Figure 8). | % of suite automated is scattered (Figure 9). |
| ~85% know the test coverage (Figure 8). | 47% know the test coverage (Figure 9). |
| ROI concentration is around 40% (Figure 8). | ROI % is scattered or at ~30% (Figure 9). |
| ~39% achieve >50% and ~14% achieve >80% ROI (Figure 8). | ~52% achieve >50% and ~9% achieve >80% ROI (Figure 9). |
| Failures and ROI are inversely proportional (Figures 10 and 11). | Failures and ROI are directly proportional (Figures 10 and 11). |
| Number of defects found are around 3 and 7 from ~39% to ~58% of responses, respectively. (Figure 10). | Number of defects found are 3, 7, and >10 from ~37%, ~33%, and ~29% of responses, respectively (Figure 10). |
| ~82% mentioned GUI suite is not productive without unit tests (Figure 12). | ~57% mentioned GUI suite is not productive without unit tests (Figure 12). |

Figure 9. Clarity in understanding of ROI on test automation—services organizations.

Figure 10. Failure of scripts versus number of defects found by categories.
Table 1: Correlation - Product

| Years of Exp | Team count | Unittest % | GUI Script % | %% on dep $A_i$ | Bugs | 95% pass? | % on dep Scit test coverage | ROI % |
|--------------|------------|------------|--------------|-----------------|------|-----------|----------------------------|-------|
| 1            | 0.2533155  | 0.15127083 | 0.38376705   | 0.35968802      | 1    | 1         | 0.353012204                 | 1     |

Table 2: Correlation - Services

| Years of Exp | Team count | Unittest % | GUI Script % | %% on dep $A_i$ | Bugs | 95% pass? | % on dep Scit test coverage | ROI % |
|--------------|------------|------------|--------------|-----------------|------|-----------|----------------------------|-------|
| 1            | 0.30797542 | 0.17269118 | 0.30794553   | 0.118599623     | 1    | 1         | 0.16503074                 | 1     |

Figure 11. Correlation of ROI with failures and defects found by categories.

Figure 12. GUI automated suite productivity without unit test cases by categories.

Figure 13. Comparison of high ROI responses (>70%)—product versus services.

Figure 14. Comparison of low ROI responses (<30%)—product versus services.
When the responses are sorted to get the responses with low (<30%) ROI, the following Figure 14 highlights the gap in clarity on test automation from both product and services respondents.

**Tools and licensing cost data**

The paid tools has more adoption in product organizations, though the usage percentage is about 5% and the cost ranges from $2000 to $20,000 per year based on the tools that are used. But predominantly, open source tools are preferred with in-house frameworks depending on code base and other operations based on release cycles, that vary from one team to another.127

Figure 15 shows overall open source is predominantly adopted for automated testing by both product and services, although services professionals use commercial tools like UiPath and QTP/UFT.128

Figure 16 shows the overall tool dispersion against the tool dispersion between the product and services industry.

**PayScale data**

For computing the ROI, the average salary per year for the test professional commonly titled as SDET (software engineer in test) in the region of Bangalore is given in Table 4. * The data from PayScale.com is taken as on 25th May 2021 and $1=₹72.78 on the same day.

**Average investment based on sample versus ROI yield**

Average team size from the sample = 10.

**Assumption**

- 7 members, one lead, one Manager, and one test architect;
- Average Lower Limit for SDET + Senior SDET: (₹8,00,000+₹13,00,000) = ₹21,00,000/2 = ₹10,50,000;
- Average Upper Limit for SDET + Senior SDET: (₹10,00,000+₹15,00,000) = ₹25,00,000/2 = ₹12,50,000;
- No. of average team member (lower limit) = 7 ➞ 7*₹10,50,000 = ₹73,50,000, No. of average team member (upper limit) = 7 ➞ 7*₹12,50,000 = ₹87,50,000, Average Lower Limit for Lead + Test Architect + Manager: (₹15,00,000+₹23,00,000+₹25,00,000) = ₹63,00,000,
- Average Upper Limit for Lead + Test Architect +

---

*Figure 15. Various tools used—all respondents.*

*Figure 16. Tools usage—product versus services.*
Manager: \( \text{(₹18,00,000+₹26,00,000+₹30,00,000)} = \text{₹74,00,000} \), Average Total Team cost with lower limit = ₹1,36,50,000, Average Total Team cost with upper limit = ₹1,61,50,000. Hence, average spend range for a 10-member team is anywhere between ₹1,36,50,000 to ₹1,61,50,000 ⇒ ($187,552 to $ 221,902).

The PayScale data mentioned in this paper are from industry professionals employed by organizations. Hence, the amount mentioned is based on salaries drawn by them. If a services organization is engaged for consulting, the costs vary, and typically, the consulting engagements are higher than the salaries provided to employees in a range of about 20%-40%. So, these costs would go up by 20%-40%. Typically, QA services are rarely tiered and listed on websites. But QA Mentor, a testing services organization, provides some view into a pricing structure. Adding to this, the tools and licenses section of this paper highlights the cost of commercial tools. Hence, if these tools are used, then the cost may be incremented anywhere from $2000 to $20,000.

### Conclusion

The data from the samples presented from the point of view of respondents are from product or services organizations. In the real-world scenario, the services industry mostly deploys their professionals in the product organizations for aiding the test automation in the product organizations as that is the business model on which they operate. From the analysis of this paper, the observations deduce that the product and services show a difference in their respective point of views, with respect to test automation.

In reference to Statista, the average spend on testing from overall IT budget is 28% for the last 5 years, but the analysis of the sample collected from different organizations for this paper highlights the areas (presented in Table 3) that contribute to low ROI on test automation. The common denomination is the difference in understanding the perspectives of automation (product and services organizations) and its goals (way to automate vs automating the right test). The results show us unit tests and the right automation helps to generate technical wealth that aids in faster, cleaner, and quality deliveries with low maintenance.

Dimensions of expertise, project automation, and quality goals are indirectly considered with the experience. Expanding this may provide more insights. For future work, dimensions of expertise, automation goals, metrics, etc. may be added to measure before and after experiments, on the actual testing experience from the organizations that have low ROI yield. Experiments could be to set the automation goals and metrics and align with the right understanding and then measure and optimize the ROI continuously.

### Acknowledgments

We are grateful to all the respondents who have filled in the questionnaire for automation ROI.

### Author Contributions

Survey and analysis.

### Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

### Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

### Availability of data and material

https://docs.google.com/forms/d/1b6aBQOBzLj0XiQc3xvivv5Cptuw8-gMhSNajE6SajDXM/viewanalytics

### ORCID iD

S. Reine De Reanzi © https://orcid.org/0000-0002-0391-4233
References

1. Alégroth E. Visual GUI Testing: Automating High-Level Software Testing in Industrial Practice. Göteborg: Chalmers Univ. of Technology, 2015.

2. Gamido HV and Gamido MV. Comparative review of the features of automated software testing tools. IJEEC 2019; 9(5): 4473. DOI: 10.11591/ijece.v9i5.pp4473-4478.

3. Berner S, Weber R and Keller RK. Observations and lessons learned from automated testing. In: Proceedings of the 27th International Conference on Software Engineering. St. Louis, MO, 2005, p. 571. ICSE. DOI: 10.1145/1062455.1062556.

4. Alégroth E, Feldt R and Ryrholm L. Visual GUI testing in practice: challenges, problems and limitations. Empir Softw Eng 2015; 20(3): 694–744. DOI: 10.1007/s10664-013-9293-5.

5. Liebel G., Alégroth E. and Feldt R. State-of-practice in GUI-based system and acceptance testing: an industrial multiple-case study. In: 2013 39th Euromicro Conference on Software Engineering and Advanced Applications. Santander, Spain, September 2013, pp. 17–24. DOI: 10.1109/SEAA.2013.29.

6. Alégroth E, Feldt R and Kolström P. Maintenance of Automated Test Suites in Industry: An Empirical Study on Visual GUI Testing. "arXiv:1602.01226 [cs], 2016. http://arxiv.org/abs/1602.01226. Accessed 24 March, 2021.

7. www.payscale.com."https://www.payscale.com/research/IN/Job=Lead_Software_Development_Engineer%2C_Test_(SDET)/Salary

8. Dudekula Mohammad Rafi K, Katam Reddy Kiran Moses MV, Petersen K, et al. Benefits and limitations of automated software testing: systematic literature review and practitioner survey. In: 2012 7th International Workshop on Automation of Software Test. Zurich, Switzerland, June 2012, pp. 36–42. DOI: 10.1109/IWAST.2012.6228988.

9. Li Feng F and Sheng Zhuang S. Action-driven Automation Test Framework for Graphical User Interface (GUI) Software Testing. ’’in 2007, Baltimore, MD: IEEE Autotestcon, 2007, pp. 22–27. DOI: 10.1109/AUTEST.2007.4374197.

10. Yu T, Srisa-an W and Rothermel G. An automated framework to support testing for process-based race conditions. Softw Test Verif Reliab 2017; 27(4–5): e1634. DOI: 10.1002/stvr.e1634.

11. McMaster S and Memon AM. An extensible heuristic-based framework for GUI test case maintenance. In: 2009 International Conference on Software Testing, Verification, and Validation Workshops. Denver, CO, 2009, pp. 251–254. DOI: 10.1109/ICSTW.2009.11.

12. Lu Y, Yan D, Nie S, et al. Development of an improved GUI automation test system based on event-flow graph. International Conference on Computer Science and Software Engineering. Wuhan, China, 2008, pp. 712–715. DOI: 10.1109/CSSE.2008.1336.

13. Walker JT. Software Test Automation with Robot Framework. JICA 2020; 175(25): 8–14. DOI: 10.5120/ijca2020920784.
52. Ampatzoglou A, Tsintzira A-A, Arvanitou E-M, et al. Applying the single responsibility principle in industry. In: Proceedings of the Evaluation and Assessment on Software Engineering. Copenhagen Denmark, April 2019, pp. 347–352. DOI: 10.1145/3319008.3320125.

53. Koteska B and Mishev A. Change management and version control of scientific applications. ICSIT 2014; 6(2): 153–161. DOI: 10.5121/icsit.2014.6211.

54. Beller M, Gousios G, Panichella A, et al. Developer testing in the IDE: patterns, beliefs, and behavior. IEEE Trans Softw Eng 2019; 45(3): 261–284. DOI: 10.1109/TSE.2017.2776152.

55. Latorre R. Effects of developer experience on learning and applying unit test-driven development. IEEE Trans Softw Eng 2014; 40(4): 381–395. DOI: 10.1109/TSE.2013.2295827.

56. Badri M, Badri L, Hachemane O, et al. Measuring the effect of clone refactoring on the size of unit test cases in object-oriented software: an empirical study. Innov Syst Softw Eng 2019; 15(2): 117–137. DOI: 10.1007/s11334-019-00334-6.

57. Gorman BL. Unit testing, integration testing, and mocking. In: Practical Entity Framework. Berkeley CA: Apress, 2020, pp. 501–538. DOI: 10.1007/978-1-4842-6044-9_12.

58. Bündner H and Kuchen H. Towards behavior-driven graphical user interface testing. SIGAPP Appl Comput Rev 2019; 19(2): 5–17. DOI: 10.1145/3357385.3357386.

59. Durelli VHS, Durelli RS, Borges SS, et al. Machine learning applied to software testing: a systematic mapping study. IEEE Trans Rel 2019; 68(3): 1189–1212. DOI: 10.1109/TR.2019.2892517.

60. Kim J, Ryu JW, Shin H-J, et al. Machine learning frameworks for automated software testing tools: a study. Int J Contents; 13(1): 38–44. DOI: 10.5392/IJOC.2017.13.1.038.

61. Li JJ, Ulrich A, Bai X, et al. Advances in test automation for software with special focus on artificial intelligence and machine learning. Softw Qual J 2020; 28(1): 245–248. DOI: 10.1007/s11219-019-09472-3.

62. https://techbeacon.com/app-dev-testing/how-ai-changing-test-automation-5-examples.

63. https://www.ee.ryerson.ca/~bagheri/papers/scke11.pdf.

64. https://www.infoq.com/articles/test-automation-ai-ml/.

65. Aggarwal A, Shaiikh S, Hans S, et al. Testing Framework for Black-Box AI Models, http://arxiv.org/abs/2102.06166. Accessed 11 June, 2021.

66. Khorram F, Motto J-M and Sunyé G. Challenges & opportunities in low-code testing. In: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, Virtual Event Canada, October 2020, pp. 1–10. DOI: 10.1145/3417990.3420204.

67. https://devops.com/low-code-automated-software-testing-drives-devops/.

68. Mariani L, Hao D, Subramanyan R, et al. The central role of test automation in software quality assurance. Softw Qual J 2017; 25(3): 797–802. DOI: 10.1007/s11219-017-9383-5.

69. Jha AK, Kim DY and Lee WJ. A framework for testing android apps by reusing test cases. In: IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft). Montreal, QC, May 2019, pp. 20–24. DOI: 10.1109/MOBILESoft.2019.00012.

70. Chen H, Zou B, Bian N, et al. A reusable component-based library for GUI regression testing. In: First International Workshop on Knowledge Discovery and Data Mining (WKDD 2008). Adelaide, Australia, January 2008, pp. 326–329. DOI: 10.1109/WKDD.2008.79.

71. Bartusevics A, Novickis L and Lesovskis A. An approach for development of reusable function library for automation of continuous processes. Proc Comp Sci 2017; 104: 112–119. DOI: 10.1016/j.procs.2017.01.082.

72. Al Dallal J and Sorenson P. Estimating the coverage of the framework application reusable cluster-based test cases. Inf Softw Tech 2008; 50(6): 595–604. DOI: 10.1016/j.infsof.2007.07.006.

73. Tiwari R and Goel N. Reuse. SIGSOFT Softw Eng Notes 2013; 38(2): 1–11. DOI: 10.1145/2439976.2439982.

74. Vieira LS, Barreto CGL, dos Santos EB, et al. Test automation in a test factory. In: Proceedings of the XIV Brazilian Symposium on Information Systems - SBSI’18. Brazil, Caxias do Sul, 2018, pp. 1–8. DOI: 10.1145/3229345.3229356.

75. Tom E, Aurum A and Vidgen R. An exploration of technical debt. J Syst Softw 2013; 86(6): 1498–1516, Jun. 2013. DOI: 10.1016/j.jss.2012.12.052.

76. Wiklund K, Eldh S, Sundmark D, et al. ‘Technical debt in test automation. In: 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation. Montreal, QC, April 2012, pp. 887–892. DOI: 10.1109/ICST.2012.192.

77. Spinola RO, Zaworka N, Vetro A, et al. Understanding automated and human-based technical debt identification approaches—a two-phase study. J Braz Comput Soc 2019; 25(1): 5. DOI: 10.1186/s13173-019-0087-5.

78. Garoussi V and Mäntylä MV. When and what to automate in software testing? A multi-vocal literature review. Inf Softw Tech 2016; 76: 92–117. DOI: 10.1016/j.infsof.2016.04.015.

79. Garoussi V and Pfahl D. When to automate software testing? A decision-support approach based on process simulation. J Softw Evol Proc 2016; 28(4): 272–285. DOI: 10.1002/smr.1758.

80. Sahaf Z, Garoussi V, Pfahl D, et al. When to automate software testing? decision support based on system dynamics: an industrial case study. In: Proceedings of the 2014 International Conference on Software and System Process - ICSSP 2014. Nanjing, China, 2014, pp. 149–158. DOI: 10.1145/2600821.2600832.

81. Wiklund K, Eldh S, Sundmark D, et al. Impediments for software test automation: A systematic literature review. Softw Test Verif Reliab 2017; 27(8): e1639. DOI: 10.1002/stvr.1639.
82. Garousi V, Felderer M, Kuhrmann M, et al. Exploring the industry’s challenges in software testing: an empirical study. J Softw Evol Proc 2020; 32(8). DOI: 10.1002/smr.2251.

83. Bures M. Automated testing in the Czech Republic. In: Proceedings of the 15th International Conference on Computer Systems and Technologies - CompSysTech ’14. Bulgaria, 2014, pp. 294–301. Ruse. DOI: 10.1145/2659532.2659605.

84. Usman Awais M and Hussain A. DevOps in Pakistani Software Industry. In: 2020 14th International Conference on Open Source Systems and Technologies (ICOSST). Lahore, Pakistan, December 2020, pp. 1–5. DOI: 10.1109/ICOSST51357.2020.9333011.

85. Cruzes DS, Melsnes K and Marczek S. Testing in a devops era: perceptions of testers in norwegian organisations. In: Misra S, Gervasi O, Murgante B, et al. (eds). “Testing in a DevOps Era: Perceptions of Testers in Norwegian Organisations,” in Computational Science and Its Applications – ICCSA 2019, vol. 11622. Cham: Springer International Publishing; 2019, pp. 442–455. DOI: 10.1007/978-3-030-24305-0_33.

86. Altinger H, Wotawa F and Schurius M. Testing methods used in the automotive industry: results from a survey. In: Proceedings of the 2014 Workshop on Joining AcadeMiA and Industry Contributions to Test Automation and Model-Based Testing - JAMAICA 2014. San Jose, CA, USA, 2014, pp. 1–6. DOI: 10.1145/2631890.2631891.

87. Tosun A, Dieste O,ucci D, et al. An industry experiment on the effects of test-driven development on external quality and productivity. Empir Softw Eng Dec. 2017; 22(6): 2763–2805. DOI: 10.1007/s10664-016-9490-0.

88. Martin D, Rooksby J, Rouncefield M, et al. ‘Good’ organisational reasons for ‘bad’ software testing: an ethnographic study of testing in a small software company. In: 29th International Conference on Software Engineering (ICSE’07). Minneapolis, MN, May 2007, pp. 602–611. DOI: 10.1109/ICSE.2007.1.

89. Kumar D and Mishra KK. The impacts of test automation on software’s cost, quality and time to market. Proced Comp Sci 2016; 79: 8–15. DOI: 10.1016/j.procs.2016.03.003.

90. Yao Y and Wang X. A distributed, cross-platform automation testing framework for GUI-driven applications. In: Proceedings of 2012 2nd International Conference on Computer Science and Network Technology. Changchun, China, December 2012, pp. 723–726. DOI: 10.1109/ICCSNT.2012.6526035.

91. Murthy P and Ulrich A. Distributed GUI Test Automation. In: 14th IEEE India Council International Conference (INDICON). Roorkee, December 2017, pp. 1–6. DOI: 10.1109/INDICON.2017.8487560.

92. Farias G, Dantas A, Lopes R, et al. Proceedings of the 9th Latin-American Conference on Pattern Languages of Programming - SugarLoafPLoP’12. NatalBrazil: Rio Grande do Norte, 2012, pp. 1–11. DOI: 10.1145/2591028.2600817.

93. Rodrigues A and Dias-Neto A. “Relevance and impact of critical factors of success in software test automation lifestyle: a survey,”. In: Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing - SAST. Maringa, Parana, Brazil, 2016, pp. 1–10. DOI: 10.1145/2993288.2993302.

94. Grechamik M, Xie Q and Fu C. Experimental assessment of manual versus tool-based maintenance of GUI-directed test scripts. In: IEEE International Conference on Software Maintenance. Edmonton, AB, September 2009, pp. 9–18. DOI: 10.1109/ICSM.2009.5306345.

95. Lin C-T, Tang K-W, Wang J-S, et al. Empirically evaluating Greedy-based test suite reduction methods at different levels of test suite complexity. Sci Comp Programming 2017; 150: 1–25. DOI: 10.1016/j.scico.2017.05.004.

96. Mazandaran University of Science and Technology, Iran-Hedayati A, Ebrahizmzadeh M, Ebrahizmzadeh M, et al. Investigating into automated test patterns in erratic tests by considering complex objects. JITCS 2015; 7(3): 54–59, Feb. 2015. DOI: 10.5815/jitcs.2015.03.08.

97. Zhi J and Garousi V. On Adequacy of Assertions in Automated Test Suites: An Empirical Investigation. In: IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops, 2013, pp. 382–391. DOI: 10.1109/ICSTW.2013.49.

98. Hierons RM and Xie T. On automation in software engineering. Softw Test Verif Reliab 2020; 30(6). DOI: 10.1002/stvr.1753.

99. Garousi V and Elberzhager F. Test automation: not just for test execution. IEEE Softw. 2017; 34(2): 90–96. DOI: 10.1109/MS.2017.34.

100. Raulamo-Jurvanen P. Decision support for selecting tools for software test automation. SIGSOFT Softw Eng Notes 2017; 41(6): 1–5. DOI: 10.1145/3011286.3011304.

101. Baral K, Mohod R, Flamm J, et al. Evaluating a test automation decision support tool. In: IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). Xi’an, China, April 2019, pp. 69–76. DOI: 10.1109/ICSTW.2019.00034.

102. Amrit C and Meijberg Y. Effectiveness of test-driven development and continuous integration: a case study. IT Prof 2018; 20(1): 27–35. DOI: 10.1109/MITP.2018.014121554.

103. Jones JA and Harrold MJ. Test-suite reduction and prioritization for modified condition/decision coverage. Proc IEEE Int Conf Softw Maintenance. ICSM 2001; 92–101, Nov. 2001. DOI: 10.1109/ICSM.2001.972715.

104. Khari M, Kumar P, Burgos D, et al. Optimized test suites for automated testing using different optimization techniques. Soft Comput 2018; 22(24): 8341–8352. DOI: 10.1007/s00500-017-2780-7.

105. Haider AA, Nadeem A and Rafiq S. On the Fly Test Suite Optimization with FuzzyOptimizer. In: International Conference on Frontiers of Information Technology, Islamabad, Pakistan, 11 December 2013, pp. 101–106. DOI: 10.1109/FIT.2013.26.
APPENDIX-I

Actual Questionnaire

1. Organization Name
2. (Optional) Phone Number [only if the respondent wanted to be paid for taking the survey]
3. Years of Experience in Test Automation or IT industry? [Short answer text field to provide answers]
4. Automation Team Strength for the application that you are testing? [Short answer text field to provide answers]
5. Do you know the unit test coverage for the application that you are testing? [Radio button: Yes, No, May be]
   a. (Optional) If you have answered yes/may be, for the above question, what is the % of unit test for the application that you are testing? [Short answer text field to provide answers]
6. What % of your functional/black box/GUI tests are automated? [Short answer text field to provide answers]
7. Each time a deployment is made, what % of your GUI automated tests fail due to application changes? [Short answer text field to provide answers]
8. How many application bugs do you find using the GUI automated scripts for each release cycle? [Radio Button: <1, 1 to 5, 5 to 10, >10, >20]
9. Do you achieve 95% pass of the automated GUI tests to give a Go/No go decision for code promotion to production? if not how much % gets passed? [Short answer text field to provide answers]
10. Each time a deployment is made, what % of your GUI automated tests fail due to Script failure issues? [Short answer text field to provide answers]
11. Do you know the test coverage % of the GUI automated test suite [Radio Button: Yes, No, May be]
   a. (Optional) If you have answered yes/may be, for the above question, what is the % of GUI automation test coverage for the application
that you are testing? [Short answer text field to provide answers]

12. What technologies or tools do you use for GUI automation for the application that you are testing? [Long answer Text Field to provide answers]

13. What % of ROI does the GUI suite yield for the application that you are testing? [Short answer text field to provide answers]

14. In your experience, have you seen the GUI tests being productive without the backing of unit tests? why? [Long answer Text Field to provide answers]

15. Do you have any comments, opinion or suggestions? [Long answer Text Field to provide answers]

APPENDIX-2

Actual Survey Response
https://docs.google.com/forms/d/1b6aBQOBzLj0XIQC3xvivv5Cptuv8gMhSNajE6SAjDXM/viewanalytics

Data availability statement
The SURVEY data used to support the findings of this study have been deposited in the IEEE repository ([DOI: 10.21227/9s1c-ma33]).

URL
https://ieee-dataport.org/documents/automation-roi-industry-data-bengaluru-india