Adopting Test Automation on Agile Development Projects: A Grounded Theory Study of Indian Software Organizations

Sulabh Tyagi[1](✉), Ritu Sibal[1], and Bharti Suri[2]

[1] Netaji Subhash Institute of Technology, Delhi University, New Delhi, India
sulabhtyagi2k@yahoo.co.in, ritusib@hotmail.com
[2] Guru Gobind Singh Indraprastha University, New Delhi, India
bhartisuri@gmail.com

Abstract. The role of test automation in Agile Software Development projects is of paramount importance. It is absolutely necessary to automate tests on agile projects as the number of test cases will continue to grow with each successive sprint. Through a Grounded Theory study involving 38 agile practitioners from 18 different software organizations in India, we identified five key challenges faced by agile practitioners and different strategies to overcome those challenges while practicing test automation. Understanding these challenges and strategies would help agile teams in streamlining their test automation practices.

Keywords: Test automation · Test driven development · Agile software development · Grounded theory

1 Introduction

The widespread use and popularity of agile methodologies are primarily due to their ability to produce quality software in less time with limited manpower. Most of the software industries are using scrum and XP methodologies of agile software development. Testing is an integral part of development in agile projects rather than a distinct Software Development Life Cycle (SDLC) phase [1].

Software test automation refers to the activities and efforts that intend to automate engineering tasks in a software test process using well-defined strategies and systematic solution [2]. According to [3] test automation is one of the most effective solution for projects which have strict deadlines as it speeds up the test execution and increases the test coverage.

Automation on a scrum project is not optional, for a team to sprint effectively and deliver value quickly, it needs to rely heavily on test automation [4]. Crispin and Gregory [5] argued that test automation is the key factor for successful agile software development and the core of agile testing. In a study by Puleio's [6] test automation was seen as a key factor in agile testing to keep development and testing in synchronization. It is evident from the above studies that test automation is a crucial ingredient of agile software development projects. Further, a study from Collins [7] reported that test automation works very well if the agile teams find the right way to implement test automation in their projects and presented some strategies to minimize the risk during test automation implementation.

The objective of this study is to create an understanding on different challenges faced by agile practitioners while adopting test automation on agile projects and to present some possible strategies to overcome those challenges. To provide more empirical insight in this area, a grounded theory study has been conducted that involved 38 agile practitioners from 18 different software organizations in India. We hope our research will help in understanding the issues while adopting test automation on agile projects and streamlining it through proper strategies.

The rest of the paper is structured as follows: in the next section a brief overview of the Grounded Theory is presented; the third section describes the findings of this study; the fourth section discusses these findings; the fifth section presents limitations of this study and the last section concludes the paper.

2 Research Method

2.1 Grounded Theory

Grounded Theory (GT) is a systematic research method where prominence is on the generation of theory that derived from systematic and rigorous analysis of data [8, 9]. The emphasis in GT is on new theory generation which means rather than beginning with a pre-conceived theory in mind, the theory evolves during the research process itself and thus the product of continuous interplay between data collection and analysis of that data [10].

Which version of ground theory. Glaser GT states that researchers should start with the general ‘area of interest’ and beginning a GT study with specific research questions can lead to pre-conceived ideas or hypothesis of the research phenomena [11]. Other two versions of GT are Straussin GT [12] and Charmaz’s constructivist GT [13]. This study employed the Glaserian version as our objective was to find out the issues from the real life experience of the agile practitioners related to our general area of interest i.e. Agile Project Management rather than imposing our own pre-conceived ideas and concerns that could influence this study and also due to plenty of resources available on Glaserian GT [8]. GT has been chosen as our research method for many reasons. Firstly, agile software development focuses on people and interactions, and GT, allows us to study social interactions and the behavior of people. Secondly, GT is most suited to areas of research that have not been explored in detail, and according to our knowledge, the research studies on test automation practices in agile software development is also scarce. Thirdly, GT focuses on theory generation rather than extending or verifying existing theories [14]. Finally, GT is being liberally used to study the agile teams [11, 15–18]. Following Glaser’s guidelines, the study started with a general area of interest – Agile project management – rather than beginning with a specific research problem. Problems and its key concerns will emerge in the initial stages of data analysis and it did [19].

2.2 Data Collection

Data collection in GT is guided through theoretical sampling whereby researchers iteratively collect and analyze their data to decide what data to collect next and where to find the data [20]. A GT study requires the theoretical sampling to be continued until theoretical saturation is reached that is when no more new concepts or categories emerge from the data, and further data collection would be a waste of time [21].

Recruiting Participants. This study involved 38 agile practitioners from 18 different software organizations in India with size varied from 50 to 200,000 employees located in Bengaluru, Mumbai, Pune, Noida and Gurgaon. The project duration varied from 6 to 36 months and team size varied from 7 to 20 people on different projects with wide range of domains like software consultancy, e-commerce platforms. Due to ethical considerations and to keep our participants identity confidential, we used codes P1 to P38 to identify our participants. Table 1 shows the participants and project details of this research study. We contacted members of Agile Software Community of India [22] and also took part in Agile India 2016 International Conference [23] on Agile that provided us the platform to collaborate with many agile practitioners across India and abroad. Many practitioners agreed to be a part of our research and participated in this study.

Interviews. Face-to-face semi-structured interviews were conducted with agile practitioners using open-ended questions over a period of eighteen months. Normally, each interview lasted for about one hour and was scheduled at the mutually agreed location. The interviews were audio recorded with the consent from the participants on ensuring full confidentiality, so that we could concentrate on the conversation. Ten participants were interviewed from four different software organizations in first phase of our study. Interview began with warm up questions regarding participants experience, their roles, nature of duties and different agile project management practices in their respective projects. Each participant had a 3–4 or more years of hands-on experience on either scrum, XP or both. Initial sample of participants comprised Scrum Masters, Developers, Product Owners (PO's) and Testers. Then we progressed to our second phase of interviews and expanded our sample participants to Senior Management people (Chief Technical Officer, Vice-President), Agile coaches and Devops to gain the well rounded perspective from participants, also the set of questions were gradually modified as per Glaser [20] to achieve theoretical saturation of our core category - Adopting Test Automation. After completion of each interview, it was transcribed and analyzed line by line to identify key points, codes, concepts and categories. Data collection and its analysis were performed iteratively. Constant comparison of interview transcripts helped us in guiding future interviews, and then we continuously fed back the analysis of interviews and observations from our study into the emerging results. All the data was personally collected and analyzed by the primary author so that consistency can be maintained in the application of GT.

Observations. We also performed passive observations in two projects denoted as Sigma and Delta in two different Indian software organizations denoted as X and Y. X

Table 1. Summary of participants and project details. (Agile Position: Agile Coach (AC), Chief Technical Officer (CTO), Developer (DEV), Devops (DO), Product Owner (PO), Scrum Master (SM), Senior Agile Coach (SAC), Senior Developer (SD), Senior Quality Analyst (SQA), Senior Tester (ST), Test Analyst (TA), Tester (TES), Vice-President (VP)

Participant (Code)	Agile Position/ Experience (yrs)	Project distribution location	Agile method	Domain	Team size	Project duration (Mos)	Sprint duration (Wks)
P1, P2	TES/3, SM/10	India-UK	Scrum	Finance	10–12, 16–18	12, 24	2
P3, P4	ST/4, PO/5	India-USA	Scrum & XP	Network Mgmt. Services	10	10 to 12	2–3
P5, P10	SM/6, ST/5	India-South East Asia	Scrum & XP	Insurance	12–14, 12	8–10, 15–16	3–4
P6	TES/4	India-Europe	Scrum & XP	Mobile Retail	18	18–20	3–4
P7, P8	TES/3, SD/5	India-USA	Scrum & XP	E-Commerce	14	12–14	1–2
P9	SD/4	India-Australia	Scrum & XP	Banking & Finance	20	24	3–4
P11, P12	AC/12, CTO/16	India-USA - Australia	Scrum & XP	Software Consultancy & Services	14–15, 18–20	12–14, 15–16	2–4
P13, P14	AC X 2/8, 10	India-New Zealand	Scrum & XP	IT & Agile Training	7–8	36	2–3
P15	DEV/3	India-UK	Scrum & XP	Telecom	12–13	42	3–4
P16, P17	TA/5, VP/12	India-UK	Scrum & XP	Insurance	9–10	12	2–3
P18, P19	SM/7, AC/8	India-Western Europe	Scrum	Health Care	18–19	24	3–4
P20, P21	TES/3.5, DO/4	India-USA	Scrum & XP	Energy Metering Solutions	10–12	36	3–4
P22, P23	TES/4, DEV/4.5	India-Canada	Scrum & XP	Finance	9–10, 12	24, 18–20	3–4
P24, P25	ST/5.5, TA/5	India-Australia	Scrum & XP	E-Commerce	10, 9–10	12, 12–14	1–2, 2–3
P26	SQA/4	India-South East Asia-Australia	Scrum	Information Security	8–9	18	3–4
P27	TES/3.5	India-Western Europe	Scrum	Web Portal	12–13	10–12	1–2
P28	SM/8	India-Western Europe	Scrum & XP	IT & Agile Training	10–12	12–14	2–3
P29, P30	SM/10, VP/12	India-USA	Scrum & XP	IT Infrastructure	12–14	16–18	2–3
P31	SAC/12	India-Europe	Scrum & XP	IT & Agile Training	8–9	24	3
P32	SM/6	India-Europe	Scrum & XP	Agile Training	10–11	12	3–4
P33	PO/3	India-Europe	Scrum & XP	Finance	12–13	15–16	2–3
P34	ST/4.5	India-UAE	Scrum	Banking & Finance	15–18	24	2–3
P35	TES/4	India-USA	Scrum	Telecom	10–11	10–12	3–4
P36, P37	SM/7, DEV/4	India-USA	Scrum & XP	E-Commerce	12–14, 18–19	6–8, 18	2–3 1–2
P38	PO/4.5	India-UK	Scrum & XP	Telecom	20	12–14	2–3

is into smart metering and energy management solutions with presence in over 30 countries and Y is into e-commerce business with presence in over 4 countries.

Observation period in Sigma and Delta was 8 and 6 months respectively. Sigma was practicing agile mainly blend of scrum and XP from past 3 years but Delta was relatively new to agile and practicing scrum from past 1 year. We observed daily stand ups, sprint retrospectives, sprint review meetings, end sprint demos, pair programming practices, daily smoke and regression tests and we had taken field notes along the way about our observations and transcribed them for analysis. Moreover, we compared the codes emerged during observations with the codes from the interviews that helped us in achieving triangulation. The interview data was further strengthened by our observations from these two projects.

2.3 Data Analysis

Coding. Following Glaser's two successive stages of substantive coding: open and selective coding, we began our data analysis with open coding. It helps us in directing our research by identifying a core category and serves as the initial step of the theoretical analysis in GT [14]. Then, selective coding was performed to identify the categories that were related to the core and to ascertain theoretical saturation.

Constant Comparative Method. Here, codes are compared with other codes to produce concepts, codes are compared further with concepts to produce new concepts and finally concepts are compared with other concepts to produce categories [14].

Memoing. Memos are written notes to log reflections between data, codes and their relationships as they occur in researchers mind [20]. In our case, we wrote memos as soon as we had some ideas about emerging codes and their relationships.

Phase 1: Identifying the core category. We commenced phase 1 of our interviews on our general area of interest "Agile Project Management" and performed open coding on data that generated initial codes, which guided us on further data collection as per theoretical sampling process of classic GT [20]. We continued collecting and analyzing our data iteratively that gradually led us to our core category i.e. "Adopting Test Automation" on agile projects.

Open Coding. In open coding interview transcripts are being analyzed in detail and key points are identified from each interview transcript [24]. In the next step, key points are collated and particular code is assigned to each key point [25]. Code is a phrase used to summarize the key point in 2 to 3 words. Using the constant comparative method, the codes from each interview were compared constantly with the codes from the same as well as from other interviews and also with data based on our observations and written memos. The constant comparison and grouping of similar codes lead to the second level of abstraction, called concepts. Further, this method is repeated on concepts to produce the third level of abstraction, called categories.

Open coding was ended on identifying our core category *"Adopting test automation"*. Two potential near core categories were also emerged like "Quality work delivery" and "Manage changing requirements", but we selected "Adopting test automation" as our category as it is related to most other categories in a meaningful way. An example of open coding process is shown in Table 2 that depicts the emergence of our core category from the combined analysis of interviews and observations.

Table 2. Example of Open Coding Process

Open coding	Interview Quotation – P5, Scrum Master	Observation (Org.: Y, Project: Delta)
Statement/Field note	"Most important question…whether or not your project is truly time driven, whether or not you are delivering high quality product, time is speed for us and we can achieve that [speed and quality] by embracing automation."	Acceptance testing was practiced manually till sprint 3, consuming lot of time and effort. UI changes were frequent due to constant new product launches, decision to automate acceptance tests, acceptance tests automation started
Key point	Need for timely delivery of quality products, Achieving speed, Quality through automation	Manual acceptance consumes time and effort, Frequent UI changes, Automating acceptance tests
Code	Timely delivery, Quality products, Embracing automation	Time and effort loss, Constant UI changes, Acceptance tests automation
Concept	Achieving quality and speed by embracing automation	Achieving speed by embracing automation
Category	Adopting test automation	

Phase 2: Refining the core category. As per theoretical sampling process, selecting new interviewees and sites for data collection should come from the results of the coding process [14]. We progressed into phase 2 and continued our data collection process.

Table 3. Example of selective coding process

Selective coding	Interview Quotation – P6, Tester	Observation (Org.: X, Project: Sigma)
Statement/Field note	"Our project…lot of business logic, we handle lot of features additions & changes…has accumulation effect on our tests too…which makes them grow in numbers with every sprint and it is really difficult to maintain [test scripts]"	Frequent change requests received from the customers, constant addition, modification of page elements, effect on test scripts size, making test script maintenance difficult for the team
Key point	Adding new features, Test scripts continue to grow, Difficulty in test script maintenance	Frequent change requests, Constant changes in test scripts, Difficulty in test script maintenance
Code	Grow in test scripts, Difficulty in maintaining test scripts	Constant test script changes, Difficulty in maintaining test scripts
Concept	Difficulty in test script maintenance	Difficulty in test script maintenance
Category	Test script maintenance	

Selective Coding. Here, only those interview transcripts were coded that were related to our core category i.e. "Adopting Test Automation". Constant comparative method was used on interview transcripts and observations to find out codes, concepts and finally the categories related to our core. Table 3 shows an example of selective coding process.

The other concepts and categories emerged in a similar manner which sheds light on the problems faced by agile teams while adopting test automation. Observations gathered from the two projects were also analyzed and compared to the concepts derived from the interviews. It was found that our observations supported the data provided in the interviews, thereby strengthening our interview data. During our data analysis one more set of concepts emerged that formed the strategies used by agile teams in order to overcome those challenges as described in the present study. Figure 1.a shows different levels of data abstraction using GT and Fig. 1.b explains the emergence of category choosing the right tool from underlying concepts.

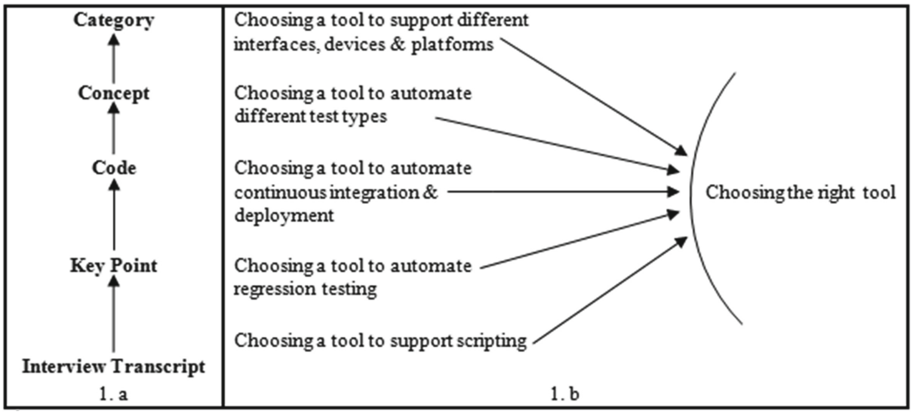
Category
Concept
Code
Key Point
Interview Transcript
1. a
Choosing a tool to support different interfaces, devices & platforms
Choosing a tool to automate different test types
Choosing a tool to automate continuous integration & deployment
Choosing a tool to automate regression testing
Choosing a tool to support scripting
Choosing the right tool
1. b

Fig. 1. **a.** Different levels of data abstraction in GT. **b.** Emergence of category *choosing the right tool* from concepts

Determining Theoretical Saturation. The selective coding continues until the researcher has sufficiently integrated the core category and its connections to other relevant categories [20]. On reaching a stage where further data collection and its analysis were leading us to the same categories with no new data, we found out that our categories have reached saturation. Then we started sorting the theoretical memos conceptually and this process is called sorting that forms the theoretical outline of our study.

The last step in GT is generating a theory also know as Theoretical Coding. It involves the conceptualization of how different categories and their associated properties relate to each other as hypothesis so that can be integrated into a theory [19, 26]. We followed Glaser's guidelines and performed theoretical coding at the later stages of analysis [14].

Table 4 shows different concepts and categories that form the challenges and corresponding concepts that form the adopted strategies while practicing test automation on agile projects. Also, the number within the parenthesis indicates the number of interviewees who referred these challenges/strategies. As the codes, concepts, and categories emerge directly from the data, which is collected from the real world, the resulting theory is grounded within the context of the data [17].

Table 4. Strategies adopted on different agile projects

Challenges	Strategies
Choosing the right tool (26)	• Know your test automation requirements, Know your tool (14) • Cost Benefit Analysis (CBA) (11)
Managing test environment (15)	• Upfront planning for managing test environment (11) • Virtualization (10)
Test script maintenance (18)	• Automation testing framework (12) • Page Object Model (POM) (8)
Mindset toward automation (17)	• Engender automation awareness (12) • ROI evaluation (11)
Effective communication (16)	• One team approach (10)

In the following section, we present the research findings from our study. Selected interview quotations are provided under each category to better explain it in the present context. Our results are grounded further by key points, codes, and concepts from the interviews as well as the observations from two agile projects. It is difficult to describe here in detail due to space reasons.

3 Results: Adopting Test Automation on Agile Projects

In this section, we present our grounded theory: Strategies used by agile practitioners while adopting test automation in their projects. We have selected quotations from our study to explain the challenges faced by agile teams and strategies opted by them.

3.1 Challenge 1: Choosing the Right Tool

Test automation is very important right from the start of any agile project. It is essential to know the project requirements, which tests needs to be automated and what tools are needed. Agile practitioners admitted that while transitioning to scrum and XP, they were still using traditional record and playback tool but results were highly unsatisfactory.

Other associated concerns include choosing a tool for automating continuous integration and deployment, automating acceptance and regression tests and a tool for effective test management.

> *"Output of sprint N has to combine with sprint N + 1, daily defect fixes that continuously check in to the code, this whole process is continuous integration (CI), it also takes lot of time, and only by automating our CI process we could survive our project deadlines."* – P10, Senior Tester

Choice of test automation tool particularly in agile projects is a very crucial decision as if you would end up choosing a wrong tool with the partial or incomplete evaluation; it may lead to loss of efforts spent in each sprint, loss of licensing fees as well as loss of automation opportunities. In order to prevent these losses, some strategies were used to overcome the problem of choosing the right tool. Two adopted strategies are explained below:

Strategy 1: Know your Test Automation Requirements, Know your Tool. One should be scrupulous while choosing a test automation tool in agile projects. Agile teams should understand their project needs and then decide on test automation tool, it is imperative to first know the exact automation requirements of the projects like test types (unit, acceptance, regression, etc.) needs to be performed, coding languages to be used on the project and suitability of choosing between licensed and open source tools; it is good to choose a tool based upon the compatibility with the application under test (AUT).

> *"A lot of licensed and open source tools are available...You must know that what you want to do with that [Tool] and for what [purpose] as requirements may vary depending on project size, cost and allocated time."*– P16, Test Analyst

Strategy 2: Cost Benefit Analysis (CBA). Cost of the tool is also one of the important deciding factors in most agile projects. Licensed tools have certain benefits over open source tools like good user support, sufficient training material and ease of use but that comes with the cost.

> *"...would be using that [tool], whether it's a licensed or open source it depends on CBA (Cost to benefit analysis) of that tool w.r.t our project."* – P32, Scrum Master

It is always better to know what test types needs to be automated, tools utility with project needs, its ability to integrate with other project and defect management tools.

3.2 Challenge 2: Managing Test Environment

The ultimate aim of any agile project is to deliver quality product and test automation plays an important role in adding that quality to the product in such short sprint durations. Keeping test environment as close as possible to production environment ensures the quality of the test automation. Agile teams were facing difficulties while creating multiple test environments for every different configuration, platforms and workflows.

> *"Why it is worth to have Test Automation in agile projects because it helps you in achieving your quality objectives, test environment should be a replica of your live [production] environment...if you practice this then the code that go into upper [production] environment would meet quality criteria."* – P13, Agile Coach

Strategy 3: Upfront Planning for Managing Test Environment. Testing whether it is automation or manual is only been successful when performed in the proper test environment. In agile, it is very common to have multiple test environments, multiple configurations for the single business application so upfront planning for managing test environment is very important.

"… important to have upfront plan for managing your test environments… by maintaining spreadsheets containing all our test environment related information like different configurations, different test devices and test data used by those devices, any database related information and continuously update it." – P29, Scrum Master

Strategy 4: Virtualization. It serves an important strategy in managing issues related to test environment management. Virtual machine setup provides that additional space to both developers and testers to test their application under test (AUT). It was used to reduce the overhead caused by different OS and hardware configurations.

"…by using virtual machines test environments can be created according to the requirement and the scope of the test… Above all it is scalable and has on demand access which reduces our burden of managing test environment."– P25, Test Analyst

Participants were using a document to gather different test environment requirements to plan for managing their existing environment or building a new. VMware workstations were also used for managing test environments related issues.

3.3 Challenge 3: Test Script Maintenance

For every new addition or modification in feature, test script needs to be modified and maintained for the entire duration of the projects with multiple sprints and this was a challenge for them.

"…The scale of regression testing grows with each sprint and so does the test scripts, so how you would add more test cases to the existing regression test suite? How you maintain those scripts?" – P34, Senior Tester

Maintainability of code was a big issue, many participants worked on web based applications where test script was created by identifying web page elements and their associated properties, so if any page element whether it is a dropdown box or submit button had changed then they needed to track and modify that script.

Strategy 5: Automation Testing Framework. Majority of our participants admitted that having a good automation testing framework solved their test script maintenance problem to the larger extent. Automation testing framework is an engine that runs your automation test scripts with the help of some tool like Selenium or Unified Functional Tester (UFT) to test your application under test. Most commonly used frameworks were: Data driven framework – modular functions are stored in external files and called by test scripts; Keyword driven framework – keyword is assigned to every user action (like button click), stored in a spreadsheet and called by test scripts; Hybrid framework – combination of data and keyword driven frameworks; and Behaviour driven framework – creating examples to describe the user behavior while using the application under test.

Strategy 6: Page Object Model (POM). Another technique used by many agile practitioners to make test script maintenance easier was Page Object Model (POM) approach. Here, each web page element (button, text box) is modeled as an object within the test code and represents as one class.

3.4 Challenge 4: Mindset Toward Automation

Whenever any project is transitioning to agile then it is important to have support from the management so that every team member proactively put up his concern and ask for any assistance that is needed to overcome any constraint regarding implementing test automation. They need to understand that test automation is a long term investment and should support the team by providing enough budget and time.

> *"Transition to agile…need support from your senior management particularly when you embrace test automation in agile…have realistic expectations from the team and…accept initial failures and invest in terms of tools or trainings…only this kind of thinking can encourage use of test automation in any agile project."* – P20, Tester

Strategy 7: Engender Automation Awareness. Agile teams need a shift in their thinking while adopting test automation. They should know the merits and demerits of having test automation in their projects and how to use it [test automation] effectively.

> *"When you wrap test automation around agile…not easy to adapt as your team won't have that thinking that agile demands…to create automation awareness in your team…try to create it by providing coaching, workshops or short trainings on test automation in agile environment."* – P13, Agile Coach

Strategy 8: ROI Evaluation. Senior management should provide the required infrastructure and environment necessary to conduct effective test automation practices. Eleven of our participants used ROI (Return on Investment) evaluation to get their support. ROI calculation is based on evaluating the benefits of test automation with respect to its implementation costs in terms of tool cost, manpower cost, time needed to build required infrastructure for automation.

3.5 Challenge 5: Effective Communication

Many participants admitted that lack of communication in their teams often results in poor automation planning, late feedbacks and wrong automation effort estimates. Test automation is teamwork and should be taken care of by both developers and testers.

> *"…have to consider a lot many things…plan automation, what features to automate in each sprint, when to start automation and one thing is crucial…conversation element - PO talking to developers, testers talking to developers and creating a wonderful coordination with effective communication."* – P38, Product Owner

While implementing test automation, it is very important for developers and testers to collaborate with each other, testers should help developers in designing unit test cases and developers should help testers in automating acceptance tests. The more they communicate more effective test automation would become.

Strategy 9: One Team Approach. One team approach was the key crusader in building effective communication between testers, developers and PO's as mentioned by ten agile practitioners. Many agile teams were giving much emphasis to have proactive communication with each other including both verbal and written communication so that every team member developed this feeling that they are working together as one single team not as separate entities.

"*When you automate…expected to not only report defects but also to communicate [defects] effectively to the development team and track it till closure. When you have that [proactive communication] surrounding your team that keeps everyone in one loop then results are more than satisfactory.*" – P32, Scrum Master.

If there is any defect then it should be properly determined whether it is because of script or actually a test case has failed and it can only be possible when testers proactively talk to developers and also send a mail to team's group mail id for better information flow.

4 Discussion and Related Work

Agile projects have daily rounds of unit tests, integration tests, acceptance tests and continuous deployment. The serious effect of not having perfect test automation in place forms the rationale behind our study.

The choice of the right tool from a plethora of available tools is a decisive step towards successful test automation. This is confirmed by studies of Oliveira [27] and Collins [28]. If one tool is not working well for the project, in the next iteration, agile teams should try something new [28]. Yoder [29] discussed the importance of selecting automation tools and when automated tests should be run under "Automate First" pattern.

The implications of managing test environment and test script maintenance revealed by our findings are also supported by a number of studies. Deak [30] highlights a number of negative factors that influence testing like insufficient number of test environments and weak infrastructure. Karhu [31] contributes test environment, test maintenance and implementation time as key concerns about test automation infrastructure. Fewester et al.'s study [32] mentioned negative impact on test automation cost due to improperly managed test script maintenance cost. Bach [33] advocates the benefits of test automation over maintenance cost of constantly changing test scripts suite.

For successful test automation, management should be open to test automation practices and their financial benefits in spite of time constraints. Late testing mindset need to be changed to early testing mindset in agile environment [34] and management support is also desired in terms of having realistic expectations from the test automation [35].

According to [34] efficient communication and interaction between testers and developers improved both testing and development, eventually improving information flow and efficiency in process. Graham [36] suggested active participation of testers in requirement reviews along with developers for performing test planning in parallel. Yoder [29] also reported whole team approach as one of the pattern for agile quality mindset.

5 Limitations

The inherent limitation with grounded theory research study is that the research findings are grounded in the specific contexts that are explored in the research. Data triangulation was used for reducing researcher bias, as we gathered the data from two sources, namely, interviews and observations that may yield more reliable data than using a single data source. The context in this research was governed by our choice of research destinations and the availability and accessibility of agile practitioners to participate in this study. We do not claim that our findings are universally applicable to all the agile projects practicing test automation, however, they accurately characterize the contexts studied.

6 Conclusion

A Grounded Theory study has been conducted over a period of eighteen months that involved 38 agile practitioners from 18 software development organizations in India. This study investigated the test automation adoption from the specific perspective of agile practitioners through their real life project experiences using GT. Unlike most of the participant organizations, some of them were recently transitioned to agile software development methods. However, all of them were striving to build good test automation infrastructure for their projects. During the study, we discovered the various challenges and strategies adopted thereof by agile teams while establishing good test automation practices in their projects. Main contribution of this paper is towards understanding the key challenges while adopting test automation in agile projects and providing some widely used strategies to overcome those challenges. This study can be utilized by agile software development teams to have a plan of action and streamline the test automation to get maximum benefits. We acknowledge this fact that all challenges and strategies adopted by software development organizations practicing test automation in agile projects may not have emerged in this study. This may also serve as the foundation for conducting future studies in the same area.

Acknowledgments. Our big thanks to all agile practitioners for participating in this study. This research is supported by our institute's TRF academic grant. Thanks to Prof. Yogesh Singh for his immense support and guidance.

References

1. Sayed, I.N.: The case of agile testing. White Paper, cognizant 20-20 insights (2016). https://www.cognizant.com/InsightsWhitepaper. Last accessed 08 Jan 2016
2. Gao, J., Tsao, J., Wu, Y.: Testing and Quality Assurance for Component-Based Software. Artech House, Boston (2003)
3. Dustin, E., Rashka, J., Paul, J.: Automated Software Testing: Introduction, Management, and Performance. Addison-Wesley, Boston (1999)
4. Cohn, M.: Succeeding with Agile: Software Development Using Scrum, 1st edn. pp. 314–316. Addison-Wesley Professional, Boston (2009)

5. Gregory, J., Lisa, C.: More Agile Testing. Addison-Wesley, Upper Saddle River (2015)
6. Puleio, M.: How not to do Agile testing. In: Proceedings of the Conference on AGILE 2006 (AGILE 2006), pp. 305–314. IEEE Computer Society, Washington, DC (2006). doi:http://dx.doi.org/10.1109/AGILE.2006.34
7. Collins, E., Lucena Jr., F.: Strategies for agile software testing automation: an industrial experience. In: Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops (COMPSACW 2012), pp. 440–445. IEEE Computer Society, Washington, DC (2012)
8. Glaser, B.: Grounded theory institute: methodology of Barney G Glaser (2010). http://groundedtheory.org/. Last accessed 28 Nov 2015
9. Hoda, R., Noble, J., Marshall, S.: Agile undercover: when customers don't collaborate. In: XP 2010, Norway, pp. 73–87 (2010)
10. Goulding, C.: Grounded Theory: A Practical Guide for Management, Business and Market Researchers. Springer, Berlin (2002)
11. Dorairaj, S., Noble, J., Malik, P.: Understanding team dynamics in distributed agile software development. In: Wohlin, C. (ed.) XP 2012. LNBIP, vol. 111, pp. 47–61. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30350-0_4
12. Corbin, J., Strauss, A.: Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory, 4th edn. Sage, London (2015)
13. Charmaz, K.: Constructing Grounded Theory, 2nd edn. Sage (2014)
14. Glaser, B.: Basics of Grounded Theory Analysis: Emergence vs. Forcing. Sociology Press, Mill Valley (1992)
15. Dorairaj, S., Noble, J., Malik, P.: Understanding lack of trust in distributed agile teams: a grounded theory study. In: 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012), pp. 81–90. IET (2012)
16. Hoda, R., Noble, J., Marshall, S.: Organizing self-organizing teams. In: ICSE 2010, pp. 285–294. ACM, South Africa (2010)
17. Martin, A., Biddle, R., Noble, J.: The XP customer team: a grounded theory. In: Proceedings of the AGILE Conference, pp. 57–64 (2009)
18. Whitworth, E., Biddle, R.: The social nature of Agile teams. In: Agile 2007, pp. 26–36. IEEE Computer Society, USA (2007)
19. Glaser, B.: Doing Grounded Theory: Issues and Discussions. Sociology Press, Mill Valley (1998)
20. Glaser, B.: Theoretical Sensitivity: Advances in Methodology of Grounded Theory. Sociology Press, Mill Valley (1978)
21. Glaser, B.G., Strauss, A.L.: The Discovery of Grounded Theory: Strategies for Qualitative Research. Sociology Press, Aldine (1967)
22. Agile Software Community of India. http://www.agileindia.org/. Last accessed 12 June 2016
23. Agile India 2016. http://www.2016.agileindia.org/. Last accessed 10 Feb 2016
24. Urquhart, C., Lehmann, H., Myers, M.D.: Putting the 'theory' back into grounded theory: guidelines for grounded theory studies in information systems. Inf. Syst. J. **20**(4), 357–381 (2010)
25. Georgieva, S., Allan, G.: Best practices in project management through a grounded theory lens. Electron. J. Bus. Res. Methods **6**(1), 43–52 (2008)
26. Glaser, B.: The Grounded Theory Perspective III: Theoretical Coding. Sociology Press, Mill Valley (2005)
27. Oliveira, J.C., Gouveia, C., Filho, R.Q.: A way of improving test automation cost-effectiveness. In: CAST. EUA, Indianapolis (2006)

28. Collins, E., Lucena Jr., F.: Software test automation practices in agile development environment: an industry experience report. In: Proceedings of the 7th International Workshop on Automation of Software Test (AST 2012), pp. 57–63. IEEE Press, Piscataway (2012)
29. Yoder, J.W., Wirfs-Brock, R., Washizaki, H.: QA to AQ part six: being agile at quality "Enabling and Infusing Quality". In: HILLSIDE Proceedings of 23rd Conference on Pattern Languages of Programs, October 2016
30. Deak, A.: A comparative study of testers' motivation in traditional and agile software development. In: Product – Focused Software Process Improvement, pp. 1–16 (2014)
31. Karhu, K., Repo, T., Taipale, O., Smolander, K.: Empirical observations on software testing automation. In: Proceedings of the 2nd International Conference on Software Testing, Verification, and Validation (ICST 2009), Denver, Colo, USA, pp. 201–209 (2009)
32. Fewster, M.: Common Mistakes in Test Automation, Grove Consultants (2001). https://www.stickyminds.com/sites/default/files/presentation/file/2013/01TAU_M5.pdf. Last accessed 02 Feb 2016
33. Bach, J.: Test automation snake oil. Windows Tech. J., 40–44 (1996)
34. Taipale, O., Smolander, K.: Improving software testing by observing practice. In: Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering (ISESE 2006), pp. 262–271. ACM, New York (2006). doi:http://dx.doi.org/10.1145/1159733.1159773
35. Kettunen, V., Kasurinen, J., Taipale, O., Smolander, K.: A study on agility and testing processes in software organizations. In: Proceedings of the 19th International Symposium on Software Testing and Analysis, pp. 231–240 (2010)
36. Graham, D.: Requirements: requirements and testing: seven missing-link myths. IEEE Softw. **19**(5), 15–17 (2002). doi:10.1109/MS.2002.1032845