Developing for Testability: Best Practices and the Opinion and Practice of OutSystems Professionals

Fernando Reinaldo Ribeiro[1,2], José Carlos Metrôlho[1,2], Joana Salgueiro[2]
[1]R&D Unit in Digital Services, Applications and Content
[2]Polytechnic Institute of Castelo Branco
Castelo Branco, Portugal
e-mail: fribeiro@ipcb.pt, e-mail: metrolho@ipcb.pt, e-mail: joana.salgueiro@ipcbcampus.pt

***Abstract*— Implementing best practices during the software development process can significantly influence the test automation process. This is true in all software applications, regardless of the platform or the programming language used, but it is even more important when the software is developed using low-code development platforms. These platforms are commonly used together with agile methodologies, and they are designed to accelerate software development with a minimum of hand-coding. Generally, when using these platforms and methodologies, the focus is on verbal and informal communication rather than documentation. The focus is on getting high-quality source code, adequate test sets, and greater interaction with the end customer. This highlights the need to use best practices in software development to achieve better quality software and facilitate the test automation process. In this work, we analyse the test automation on low-code development platforms and, more specifically, how the best practices for OutSystems development influence the test automation process. A survey on the opinion and practice of OutSystems platform professionals, 27 respondents, is also analysed and discussed. The goal is to understand how they recognise the influence that best development practices have on the testing automation process and how they apply these best practices in their daily activities.**

Keywords- low-code platforms; OutSystems; software quality, software testing; test automation.

I. INTRODUCTION

More than 3,300 IT professionals in all kinds of industries share their insights in a research report on the state of application development. Findings from this report [1] show that forty-one per cent of respondents said their organisation was already using a low-code platform, and a further 10% said they were about to start using one. They also report that the number of applications respondents had planned for delivery in 2019 was 60% higher than the assessment they had done in the previous year. This growing demand is one of the reasons why most organisations have invested in customer-centric practices in the past year (2018), including agile (60%), design thinking (30%), customer journey mapping (20%), and lean UX (11 %) [1]. These results show the growing interest in adopting agile methodologies and adopting Low-Code Development Platforms (LCDP). Similar conclusions are also presented by the Low-Code Development Platform Market [2], as it is notice that the global LCDP market size is projected to grow at a rate of 28.1% during the 2020-2026 period. These studies show the growing popularity of LCDP and its growing adoption by IT companies. They may help fill the gap between business and IT through abstraction and automation and accelerate the software release time.

Some of the reasons that have been used to justify this growth are the same ones that are often pointed out as advantages of these platforms: they allow to reduce the software delivery time and to update and deliver new features in shorter periods [3]; they allow applications to be built for multiple platforms simultaneously [3]; they integrate many of the same tools' functionalities that developers and teams use to design, code, deploy and manage their applications [4]; developers may still need to do some coding for a specific task, but a significant part of the job can be done through the drag-and-drop interface [5], and many of the data integration features have already been developed and can be easily customised. [3]. Also, LCDP are often associated with agile development (e.g., [6][7]), which have implications for the way tests are managed. This is because agile methodologies are based on reduced use of documentation and more frequent interactions with end-users. However, it is also because, in certain situations, the testing process may derail some of the benefits associated with the low-code development and agile methodologies. Bug fixing and application scalability are made easier in these platforms using high-level abstractions and models, but low-code development is not synonymous with error-free development. LCDP democratise application development to software practitioners with distinct backgrounds. This brings more professionals to IT areas, reskilling some of them from different areas of knowledge and greater employability difficulties, but the lack of specialised knowledge can lead to a higher number of bugs in the developed software. This further highlights the need to test the software developed on these platforms and the importance of studying various test strategies and tools that best suit these platforms. A study [8] of around 5K Stack Overflow forum posts that contain discussions of nine popular LCDP found that most of the questions are related to the development phase, and low-code developers also face challenges with automated testing. Low-code development introduces new concepts and characteristics that led to new challenges and opportunities in the software testing process.

 ISBN: 978-1-61208-894-5

Some of the best practices that should be used during software development are discussed, and a study to understand how OutSystems professionals know and apply these best practices and how they value testing activities in software developed at OutSystems LCDP is presented and discussed. We choose this platform because it is a platform widely used by software development companies in Portugal and because we have a collaboration with that company for several years, under which we have accessible software licenses. Another important fact in the choice is that this platform is one of the leaders in the low-code market [9]. The goal is to investigate the importance of the best practices in low-code development, their impact on the test automation process, and to understand how professionals know and apply these best practices. As a methodology to achieve this goal, the influence of best practices in low-code development in the software testing process was first analysed and then a survey was carried out to understand the professionals' opinion and practice. Section 1 presents a brief overview of the problem under study and presents its motivation and objectives. Section 2 describes some works that addressed test automation on low-code platforms. Section 3 presents background about test automation on LCDP and analyses the best practices for OutSystems development and its influence on test automation. Section 4 presents and discusses the results based on a survey about the opinion and practice of OutSystems platform practitioners about best practices in development in low-code software testing automation. Finally, Section 5 presents some conclusions that were obtained while conducting this study.

II. Related Work

Software testing and test automation are essential topics that deserve the attention of everyone involved in the software development process, regardless of the technologies or the methodologies they used. However, in the specific case of software developed using LCDP, usually following agile methodologies, there is not much documentation and research on this topic.

Some well-known LCDP have made efforts to provide some documentation on this topic and provide tools to support testing activities. The Mendix Application Test Suite [10] is a suite of tools for embedding testing in the application lifecycle. These tools are built-in Mendix, on top of Selenium. In Power Apps, testing can be performed with test studio [11] that is developed specifically to support automated end-to-end UI testing of an application. OutSystems provide the BDD Framework [12] that is an open-source application that provides a set of tools for producing Behaviour Driven Development (BDD) Test Scenarios and can also be used for automated testing. Other studies have looked at various LCDPs to compare their approaches to testing. In [13] five commercial LCDP (Mendix, Power Apps, Lightning, Temenos Quantum, OutSystems) were analysed to identify low-code testing advancements from a business point of view. They analyse the testing facilities embedded in each platform and they identify some challenges when testing low-code software. When using LCDP, automation is possible on all test levels. Component testing is essential for developers to test the software they develop. Moreover, as low-code applications use many integrations to other services using APIs, besides system/ End-to-End tests, automated integration/API tests are also essential. A good testing strategy enables continuous quality assessment and is essential. It is well known that development practices influence the test automation process. It is therefore important that developers know and apply best practices in development to facilitate subsequent testing activities. In this context, some works have analysed the development/tests relationship when LCDP is used. A study of the test automation process on the OutSystems low-code development platform is described in [14]. Their focus is on Unit, Integration / API and System / End-to-End testing levels. Their examples illustrate that the implementation of best practices during the development process can have a significant influence on the test automation process.

Few research works address test automation on LCDP and how development practices influence testing activities. It is necessary to study this relationship and understand the awareness of LCDP professionals regarding the importance of testing the software developed using LCDP.

III. Low-Code Software Testing Automation

Automated testing is essential, and there are many situations where these approaches are more beneficial than manual testing approaches. These advantages are important, especially when it may be helpful to repeat tests already carried out, such as regression tests. Nevertheless, there are other advantages. Manual testing is often complex, or impractical, or can be time-consuming and vulnerable to inaccurate results. Test automation enables continuous quality assessment and may save significant time and effort. In LCDP, test automation is possible on different tests such as unit tests, Integration/API tests, System/E2E tests, etc. However, the specific features of those platforms raise a set of challenges in low-code testing. Some of these challenges are identified in [13], namely:

- The role of citizen developer and its low-level technical knowledge in the testing activities: Test cases are usually derived from the requirements, and it is common to involve partners with low-level technical knowledge in the testing activities, which poses some challenges.
- The importance, and the challenges, in offering high-level test automation: In software developed using LCDP, several situations should be continuously tested (e.g., many integrations to other services, and these integrations should be continuously tested). To facilitate test automation, these tools should allow high-level test automation,

 ISBN: 978-1-61208-894-5

be undemanding technical skills, and require little manual scripting for writing tests.

- Leveraging the cloud for executing tests and for supporting testing of cloud-based applications: LCDP are cloud-based and they support the development of cloud-based applications using cloud resources. Test automation must be adapted to this environment.

Despite the challenges it raises, test automation allows continuous quality assessment, and it is essential in agile and low-code development. To be efficient and beneficial, it is also vital that best practices are used during development. This can help to reduce the work required for test automation and significantly reduce the need to write manual scripting.

A. Testing on the OutSystems Low-Code Development Platform

Low-code development is often associated with error-free development. However, although these platforms provide several features that allow reducing the probability of errors occurring, they can always occur, introducing bugs that may later lead to failures in the software. In the OutSystems LCDP, several features are available that help developers to develop software with fewer bugs and consequently with better quality. The OutSystems platform performs continuous integrity validation that checks the impact of all changes in application layers (data model, business logic or presentation) to ensure that everything is integrated at the time of implementation. When changes are made in the applications data models, API, and architecture, the OutSystems platform automatically updated all existing dependencies. At a more general level, the OutSystems platform performs an impact analysis for multiple applications when creating deployment plans, evaluating the impact of moving new versions of selected applications to the target environment before the deployment is performed. As a result of this process, the number of bugs introduced is generally lower than traditional development technologies, leading to fewer test cycles and issue fixes, reducing the effort associated with development and delivery.

Despite this support provided by the OutSystems platform, there is no guarantee that errors will not occur, and the need for testing remains. Therefore, the life cycle of an OutSystems application includes several stages when testing activities must be performed. The four levels of testing, provided in the International Software Testing Qualifications Board (ISTQB) classification [15], are included:

- Component Tests are used to verify the behaviour of code units. In some cases, code units are not easily accessible to be tested. The developers deliver these tests as part of the activities developed in the sprint performed in the development environment (DEV) and the continuous integration environment (CI). Usually, they are automated tests performed using the BDD Framework [16].
- Integration Tests are tests to verify the integration with external systems. These tests are critical since it is widespread that LCDP make use of external API. These tests must be performed in the DEV environment by the developers or Quality Assurance (QA). These tests can be automated.
- System Tests are usually run through a web or mobile interface. They are performed considering the perspective of the end-user or the system (End-to-End tests). The quality team can automate this type of test if they are UI tests. Usually, they are performed in a quality QA environment.
- The clients perform Acceptance Tests. Usually, they are performed manually in the QA environment.

Also necessary, the Regression Tests. They must be used whenever new features are added. In addition to these tests, other tests are also planned, such as Security Tests and Performance Tests.

B. Best Practices for OutSystems Development and its Influence on Test Automation

Regardless of the development platform or programming language used, applications must be developed to facilitate testing activities to facilitate tests that validate its correctness. This is often called developing for testability. To make this possible, there is a set of good practices, architectural and design decisions, which must be followed. Some of these best practices are applicable when developing applications on the OutSystems platform, but they are also applicable when applications are developed on other LCDP or programming languages. Some of these practices can significantly facilitate test automation at various levels, and their influence on the testing process has already been studied (e.g., [14]). For example:

- Integration Tests (API tests): to facilitate the automation of these tests, it is important to isolate the API consumption in a specific module that exposes the API methods through public actions. Other modules, which need access to the API, will have to do it through this specific module, avoiding implement and run tests on every module that is consuming this specific API.
- System Tests: in this case, test automation usually involves simulating and recording a user's interactions in a browser to complete the functionality under test. To be less hard work, test automation tools, which are being used, should correctly identify the web elements found on the web page. To make it possible, it is necessary that the web elements identifiers (names and ID) are easily found and identified by the test tool. It often implies the use of personalised identifiers in place of the identifiers assigned by the development

 ISBN: 978-1-61208-894-5

platforms. In applications developed in OutSystems, those elements should be appropriately identified in Service Studio by the developer. The developer must customise the elements' identifiers to ensure that all elements have an identifier that would be uniquely identified during the test automation process. This will have a positive effect on the test automation process but, on the other hand, will require more time and more resources and can be a complex task for developers without specialised skills.

These practices, and their effect on the test automation process, are known. Nevertheless, it is important to know how they are applied and the opinion and practices of professionals regarding their use. This is particularly important when referring to LCDP professionals since the allocation of time to facilitate or develop the tests, and the adoption of certain development practices can undermine some of the benefits associated with the use of low-code platforms.

IV. Opinion and Practice of OutSystems Platform practitioners

In this section, we present the survey addressed to IT professionals, with experience in OutSystems development. The goal is to analyse their perception of the importance of software testing in low-code development and the influence of the best development practices in test automation.

A. Survey

The survey was disseminated among professionals from 4 software companies that use the OutSystems LCDP to develop their products and was organised in two parts. The first part was aimed to characterise the respondents regarding their experience in the IT area and, in particular, their experience with LCDP and in the area of software testing and quality. This part had seven questions about: age; years of experience in IT; technologies/software development tools that they use, or have used, in their professional activity; LCDP that they use/have used in their professional activity; activities/roles to which they dedicate more time in their professional activity; if their professional activity involves development, for which platforms they develop; and most common development methodologies in the projects in which they have participated. With this first part of the survey, we were able to characterise the universe of respondents in terms of experience, roles, skills and dominant activity of their respective professional activities. We had the participation of 27 respondents.

The second part was intended only for participants who had some experience in testing activities. The objective was to allow a characterisation of the respondents to perceive testing activities and how functionality description and development practices influence test automation activities. Furthermore, it was also an objective to obtain a characterisation about the tools they use for testing. This part had nine questions, where information was collected about: the importance they give to testing; difficulty in deciding what should be tested; how the way of describing the functionalities (use cases, user stories, etc.) contribute to facilitating the test design; how the way the code is developed contributes to facilitating the test activity (write, implement, and execute the test cases); the way the test cases are written; types of functional tests that are performed more frequently; tools/platforms used to perform the tests; opinion about the BDD Framework (if used by the respondent); for those who use the BDD Framework, opinion on advantages and disadvantages of it. We had the participation of 25 respondents for this part.

From the analysis of the responses to the first part of the survey, we conclude that:

- Most of the participants in the survey (48.1%) are between 26 and 30 years old, and 25.9% are between 36 and 40 years old.
- In terms of years of experience in IT, 88.8% of the respondents have between 3 and 15 years of experience, 40.7% have between 3 and 5 years, 25.9% have between 6 and 10 years, and 22.2% have between 11 and 15 years.
- Regarding the technologies used, most of the respondents answered that they use, or have used, HTML/CSS, JavaScript, C#, Java, and PHP.
- Regarding the LCDP that they use/ have used in their professional activity, all the participants answered that they use, or have used, OutSystems. Two of them pointed out that they have also used two other LCDP.
- As for the feedback on the professional activity to which the participants currently devote more time, we found that almost 59.3% of respondents are developers and 22.2% of respondents are team leaders or managers.
- In terms of target platforms (web, Android, iOS or Multiplatform), we obtained 26 responses, of which 73.1 % of respondents indicated multi-platform and 26.9% for the web.
- Regarding the development methodology that is most common in the participants' projects, only 1 of the participants answered "Lean", while the remaining 26 participants answered Scrum.

In summary, the sample involved an experienced population, from 4 different companies, with development experience in OutSystems, experience in Agile Scrum methodology and in several development technologies, and mainly composed of staff dedicated to both web and multi-platform development tasks.

B. Data analysis and discussion

The second part of the survey was addressed only to professionals with experience in software testing.

- The question of this part was intended to find out how the participants see the testing activity. With a

total of 25 answers, 100% of the participants considered that "Testing is important and should be performed regardless of the development methodology used".

- The second question, to evaluate the difficulty of the participants in evaluating what should be tested and how it should be tested, revealed that 52% of the respondents (13 of 25 feel these difficulties sometimes and still 16%, 4 participants, feel difficulties many times.
- From the 25 respondents, 13 answered that they strongly agree, and 9 that they agree that the way functionalities are described (use cases, user stories, etc.) contribute to facilitating the test design. The other 3 respondents had no opinion.
- To the question, "Does the way the code is developed contribute to facilitating the testing activity (write, implement, and execute the test cases)?", 11 out of 25 participants answered that they agree, eight answered that they strongly agree, five neither agree nor disagree, and only one answered that he disagrees. In other words, 76% (19 out of 25) of the respondents acknowledge that the way they develop their software has implications on the testing activities of that software.
- 33.3% of the participants (8 of 24) answered that they use common sense to write the test cases, and 45.8% (11 of 24) answered that they use recognised design techniques, such as BDD (Given - When - Then) and user stories acceptance criteria.
- To the question "In your testing activities, when you perform functional tests, at what level do you perform the most frequently?", 23 respondents answered, of which 73.9% of the participants (17 out of 23) answered that they perform unit/component tests, 17.4% (4 out of 23) answered that they perform system tests and, finally, 8.7% (2 out of 23) answered that they perform integration tests. These answers seem to be in line with the fact that a significant number of the respondents are currently developers, and therefore unit/component testing is more common.
- 14 respondents answered to the question "If your professional activity includes implementation and execution of tests, and if you use any testing tool, please indicate which you have used". All of them (14) pointed out that they have used BDD Framework, and 1 respondent has also used Tricentis Tosca and Katalon.
- Regarding the experience with the BDD Framework tool, it was asked that "If you use BDD Framework in your testing activity, how do you rate your experience with this tool?". In response, 53.3% of the participants (8 out of 15) answered that they have had or have a positive experience, 33.3% (5) answered that the experience was neither positive nor negative, and finally, 13.3% (2) of the participants answered that they have had or have a very positive experience with BDD Framework.
- Finally, the last question allowed respondents to write an open-ended answer to the following question "In relation to your answer to the previous question; please indicate the most positive aspect (strength) and the most negative aspect (weakness) of the tool you use". All respondents reported having used the BDD Framework tool. In their opinion the strengths of the BDD Framework, in the opinion of the participants are:
  - Ease of use and organisation of tests.
  - Tests are developed oriented to the user story, which enables task-test mapping.

  The weaknesses mentioned were the following:
  - Heavy reliance on the user story.
  - If the user story is not well written, the tests may not be implemented correctly.
  - Requires extra time to implement, which can have a significant impact on the project delivery time.
  - In agile, if the requirements change a lot, the tests developed may become useless, and therefore there is a waste of time.
  - It generates an extra effort in preparation.

  In other words, some limitations to the use of BDD Framework are pointed out by some of the survey respondents, but it is a user-friendly tool. The fact that tests are related to user stories is also a point of disagreement among the participants because some say that it enables task-test mapping while others say that they are dependent on user stories.

A cross-check was also done to analyse the impact of years of experience in the testing activities. That is, to analyse if there are some relationships between the number of years of experience and the knowledge or techniques applied at the testing process. First, the relationship between the number of years of experience and their perception of how the application code is developed to facilitate the testing activity was analysed. In this context, the inclusion of the best practices during the software development is of fundamental importance. As can be seen in Figure 1, only 25% of the participants who have between 11 and 15 years of experience disagree that the way code is developed can facilitate the implementation of tests. All professionals with more than 16 years of experience (despite the low number of respondents) strongly agree that the way code is developed to facilitate the implementation of tests. These results seem to suggest that professionals with more experience are more aware of this issue.

 ISBN: 978-1-61208-894-5

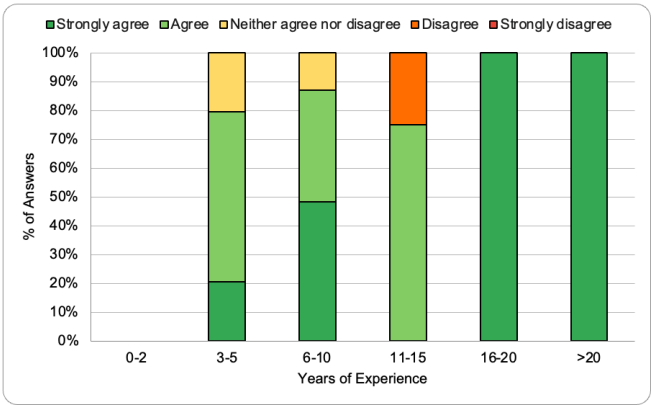

Figure 1. Years of experience vs how the code is developed.

The relationship between years of experience and difficulty in the testing activity was also analysed (see Figure 2). There are slots with more experience (6-10 and 11-15) that express difficulties more often than participants with between 3 and 5 years of experience. Overall, the results to this question seem to indicate that there is no cause-effect relationship between the years of experience and difficulty in the testing activity. The difficulties in testing, manifested by the respondents, were transversal to all professionals.

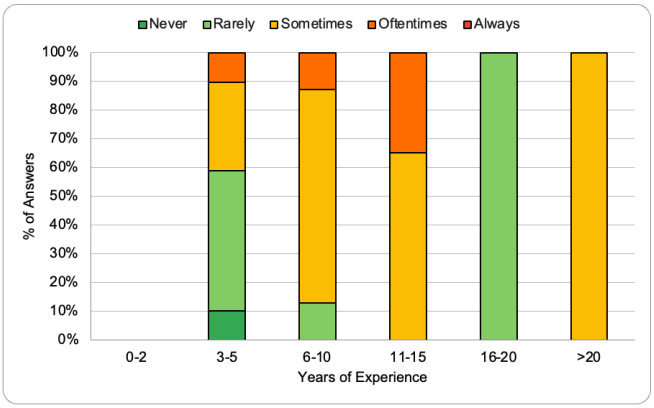

Figure 2. Years of experience vs difficulties in testing activities.

The relationship between years of experience and the way they plan and write test cases was also analysed and presented in Figure 3. In this case, the data is quite similar, and many participants still use only common sense as a way of writing tests regardless of their years of experience. These results reveal that participants do not have training in this area to know and use more test writing techniques to optimise this component of their work.

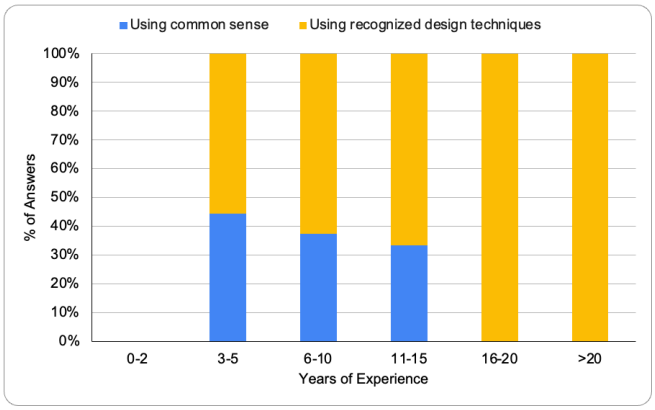

Figure 3. Years of experience vs the way they plan and write test cases.

V. Conclusion and Future Work

Regardless of the development platform or programming language used, applications must be developed to facilitate testing activities to facilitate tests that validate its correctness. To achieve this, a set of good practices, architectural and design decisions, must be followed. These practices, and their effect on the test automation process, are well known. This becomes particularly important when the software is developed using an LCDP since the allocation of time to facilitate or develop the tests, and the adoption of certain development practices can undermine some of the benefits associated with the use of LCDP.

To understand the opinion of IT professionals about the importance of software testing and their perception of the importance of best development practices and their influence on the process of test automation, a survey was conducted. The respondents that work with OutSystems, have some experience with testing activities and use the BDD Framework as a test implementation tool. Although it is the tool most used by the participants and is easy to use, it has some weaknesses in the participants' opinion. All of them recognise the importance of testing regardless of the type of application to be developed, and more than 50% recognise that they often have some difficulty assessing what should be tested and how. They also express the influence that the way functionality is described and how software is implemented have on the process of testing activity.

It results from the analysis made in the study presented in this paper that developing for software testability is recognized as very important also in the case of LCDP. The code abstraction allowed by these platforms does not exclude the need to follow best practices during the development cycle. It is also important that professionals have knowledge of adequate testing techniques and tools that allow more support for testing activities. This stage, due to the importance it assumes for the delivery of high-quality products, requires care so that (as with software development) it is carried out quickly and completely.

 ISBN: 978-1-61208-894-5

REFERENCES

[1] OutSystems, "State of Application Development Report 2019/2020," 2019.

[2] Marqual IT Solutions Pvt. Ltd (KBV Research), "Global Low-Code Development Platform Market By Component By Application By Deployment Type By End User By Region, Industry Analysis and Forecast, 2020 - 2026," Report, 2020. [Online]. Available: https://www.kbvresearch.com/low-code-development-platform-market/ (accessed Aug. 31, 2021).

[3] J. Idle, "Low-Code rapid application development - So, what's it all about?," *Platinum Business Magazine*, pp. 52–53, 2016.

[4] OutSystems, "The Low-Code Development Guide," 2019. https://www.outsystems.com/low-code-platforms/ (accessed Jul. 01, 2021).

[5] C. Boulton, "What is low-code development? A Lego-like approach to building software," *CIO (13284045)*, 2018. https://intellyx.com/2018/03/27/what-is-low-code-development-a-lego-like-approach-to-building-software/ (accessed Jul. 01, 2021).

[6] J. C. Metrôlho, F. R. Ribeiro, and P. Passão, "Teaching Agile Software Engineering Practices Using Scrum and a Low-Code Development Platform – A Case Study," in *The Fifteenth International Conference on Software Engineering Advances*, 2020, no. c, pp. 160–165.

[7] "Mendix Predicts Low-CodeOps Will Deliver Radical New Efficiencies for IT Operations," 2021. https://www.mendix.com/press/mendix-predicts-low-codeops-will-deliver-radical-new-efficiencies-for-it-operations/ (accessed Jul. 19, 2021).

[8] M. A. Al Alamin, S. Malakar, G. Uddin, S. Afroz, T. Bin Haider, and A. Iqbal, "An Empirical Study of Developer Discussions on Low-Code Software Development Challenges," in *Mining Software Repositories Conference*, 2021, p. 12.

[9] J. R. Rymer and R. Koplowitz, "The Forrester Wave™: Low-Code Development Platforms For AD&D Professionals, Q1 2019," 2019.

[10] Mendix, "Test Automation & Quality Assurance." https://www.mendix.com/evaluation-guide/app-lifecycle/test-automation-quality-assurance/ (accessed Aug. 31, 2021).

[11] "Test Studio," 2020. https://docs.microsoft.com/en-us/powerapps/maker/canvas-apps/test-studio (accessed Aug. 31, 2021).

[12] OutSystems R&D, "BDDFramework," 2016. https://www.outsystems.com/forge/component-overview/1201/bddframework (accessed Aug. 31, 2021).

[13] F. Khorram, J.-M. Mottu, and G. Sunyé, "Challenges & Opportunities in Low-Code Testing," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 2020, pp. 1–10, doi: 10.1145/3417990.3420204.

[14] J. Salgueiro, F. Ribeiro, and J. Metrôlho, "Best Practices for OutSystems Development and its Influence on Test Automation," in *9th World Conference on Information Systems and Technologies*, 2021, pp. 85–95.

[15] International Software Testing Qualifications Board, "Certified Tester Foundation Level Syllabus (Version 2018 V3.1)." 2019.

[16] J. Proença, "BDDFramework: overview," 2016. https://www.outsystems.com/forge/component-overview/1201/bddframework (accessed Jul. 02, 2021).

 ISBN: 978-1-61208-894-5