American Academic Scientific Research Journal for Engineering, Technology, and Sciences

ISSN (Print) 2313-4410, ISSN (Online) 2313-4402

http://asrjetsjournal.org/

Codeless Test Automation for Development QA

Ganesh Gatla[a*], Kanchan Gatla[b], Balaji Vishwanath Gatla[c]

[a]480 Meadowhill Drive, Alpharetta, GA, USA

[b]25004 striped maple drive, Aldie, VA, USA

[c]1215 Yarrow Street, Stallings, NC, USA

[a]Email: ganeshgatla73@gmail.com, [b]Email: Kanachangatla12@gmail.com

[c]Email: Balajiv.gatla@gmail.com

Abstract

Codeless test and testing automation promise to make the software testing process and, by extension, the software development Quality Assurance (QA) process easier and more streamlined. It naturally resonates with agile development/testing methodology, which requires repeated and in-development testing, and conventional testing practices and test automation are too time-consuming and resource-intensive (dedicated testers) for most development teams. Codeless testing tools can be used for different testing phases (unit to acceptance testing), and even though certain challenges still remain, they are rapidly being resolved with the development of more sophisticated tools. Technologies like Artificial Intelligence (AI) and good practices like cross-functional tools and shared test libraries can make codeless testing a standard part of development QA.

Keywords: Codeless test automation; testing; QA.

1. Introduction

Testing is a crucial part of software development QA. In the waterfall method, testing is the last step after production is complete, while in agile, Test-Driven Development (TDD) has radically changed development QA. Since testing is integrated into the development, automating it to fast-track the development process was a natural first progression. However, there are significant challenges associated with writing test code for continuous development QA, including “smell” [1] and bugs [2] in the test code.

One solution to this prevalent development QA problem, especially in the age of agile development, arrived in the form of codeless test automation. Test automation has been around longer than codeless testing, with supporting literature going (at least) as far back as 1999. Earlier research identified its potential and looked into its prevalence, possible challenges, and its ability to augment manual testing [3]. We can even find testing automation frameworks [4] in early literature, indicating it’s a strong and relatively mature trend.

----

* Corresponding author.

Codeless testing can be traced back to 2004 [5], when Selenium was first introduced. The earliest literature connected to Selenium dates back to 2006 [6] (2005, if you count one mention of Selenium). Selenium has since been referenced in several other studies as well. Despite its popularity, Selenium is limited to codeless test automation for web applications. The term “codeless testing” (and its variations) do not appear in research literature until a 2011 paper [7] that mentions it in the context of LISA, which is now rebranded to CA DevTest. It’s an automated testing solution, so it can be used for codeless test automation. Codeless testing is mostly mentioned in the context of LISA, even in later literature.

The development of QA has also evolved over the years. The literature on development QA, including courses, stretches back at least three decades. Most of the early literature is about the frameworks and practices associated with manual testing. Some of the earliest papers on automating the software development QA dates back to 1985. The latest papers offer insights into how development QA has matured, especially in the context of automation, even though there might still be limitations to what can be achieved [8] as well as challenges [9]. Current development QA automation trends include their overlap with Artificial Intelligence (AI) and Machine Learning (ML), which may trigger advances in codeless test automation.

From preliminary research, it can be concluded that - Despite the obvious benefits it offers, including faster development cycles and expanding the domain of software testing and QA beyond test developers, codeless test automation has yet to become a norm within the development of QA. This paper aims to explore the reasons behind this phenomenon by referencing the existing literature on the topic of automated testing, codeless testing, and development QA practices and frameworks to identify the friction points. In addition to identifying the friction points, we will also explore the availability of tools like Selenium that may be employed directly or repurposed for development QA that covers more than just web applications.

2. Codeless Test Automation

Test automation can be defined as the process of automating software testing by generating, maintaining, or modifying test data, executing pre-defined testing scenarios, and compiling the results (and generating insights). This can be achieved by writing test code separate from the source code of the software/application being developed, using automation tools, or a combination of the two. Testing automation may take different forms (with different levels of efficiency) based on the type and scope of the testing being performed. Unit testing automation (which pertains to individual components/units of a software package) frameworks were already established and were being refined in 2006 [10](possibly earlier). Large teams moved to automated unit testing in Microsoft [11]. Nowadays, unit testing is automated, mainly using python scripts, but it’s being explored for other languages as well [12].

Integration testing focuses on how different pieces fit together and can be automated using frameworks like Java Spring [13]. Automated system testing is conducted using code, conventional tools, and even GUI-based tools [14]. Automated acceptance testing frameworks and tools started emerging from the early days of agile [15], and tools like JAutomate [16] have been available in the market for several years now.

Codeless test automation is not necessarily scriptless, as testers still need to define the parameters and conditions of testing via scripts. However, it's far more sophisticated than the early iterations of record-playback testing, which identified patterns in manual tests and played them back for similar test-case scenarios. Modern automation tools leverage AI and ML. Codeless test automation is also experiencing a significant overlap with Low-Code Development, and many codeless test automation tools can be integrated with Low-Code Development Platforms [17]. This arrangement comes with certain challenges, including the absence of a low-code or no-code testing framework. If such frameworks are developed and deployed, it can result in a set of good practices.

Codeless testing is the answer to many of the problems associated with test automation, including high startup costs, but they have their own set of challenges, including:

- It hinders innovation. A lot of new code is produced just to automate the testing of a new app/software, and with volume comes new innovative ideas, stretching the eco-system and developer community's collective understanding of automation possibilities. With codeless testing, the growth is limited to the improvement of the testing tools and frameworks instead of spilling out into raw coding and automation.
- Codeless testing is also not as scalable and flexible as dedicated testing automation can be. Even comprehensive testing frameworks have their limitations (beyond desktop usability), especially within a purely codeless environment. Since many of these automation tools allow for scripting test scenarios which give testers more flexibility and scalability options but limit the testing to programmers. A true codeless test automation framework should (ideally) allow non-programmers to test different digital products, software, web and mobile apps, and APIs as comprehensively as programmers can using scripts.

New codeless testing tools can overcome these problems with solutions like shared testing libraries (to compile most test case scenarios), better integrations, cross-scripting capabilities, and product-agnostic testing practices.

3. Development QA

Development QA, whether we remain isolated to desktop applications or include web and app development as well, is more comprehensive than testing the underlying code and the final product for functionality. The framework, practices, tools, and methodologies broadened in the agile development environment. A comprehensive overview of the literature pertaining to QA in agile development [18] in the early days identified patterns in how QA was interpreted under agile. Two crucial differences between QA and testing are scope and stakeholders. Just like testing, QA can also be case/product/industry specific and is highly influenced by how "quality" is defined [19]. For certain industries, more emphasis might be on the usability and aesthetics of a mobile app, while others might be more interested in the security and cross-device functionality of a web/desktop application.

For the purpose of this paper, we can define the software development QA (including web, desktop, and mobile) scope spread out over the entirety of the project. The best QA practices (if applied) make it a proactive process where quality is controlled and assured at every step of the development process, and it starts at the conceptual stages. Testing is more of a reactive activity [20], especially from a unit testing perspective. Although from a later-stage testing perspective, i.e., when functionality is being tested, earlier tests/testing iterations can be considered proactive as they reduce the overall load on the final tester.

Still, testing is a crucial part of the development of QA, and by understanding the crucial elements of the QA framework, it might become easier to understand the potential impact and influence of codeless test automation on modern QA practices and the process as a whole. The problem is that different organizations, developers, and QA engineers have different frameworks for QA. The core elements like following the best practices and aligning development with organizational goals persist in all development QAs, whether the development team is following agile or waterfall methodology or a combination of the two.

4. Codeless Test Automation for Development QA

The development QA process (regardless of the platform) starts with defining the ideal product, devising a development approach, predicting possible complications and compliance hurdles, and choosing the right tools for the job (among other steps). That would be true for both the waterfall and agile approaches to development, though it's likely to be more prevalent in the latter. It’s in the early stages of the development process that the QA engineer and developers can identify whether adopting codeless test automation is a realistic possibility for the said projects, considering its parameters, timeline, resources, testing requirements, and the availability of relevant tools. From a QA perspective, the decision for the adoption of codeless test automation during (or after if the waterfall development method is chosen) the development may be influenced by the following factors:

- How much of the testing load can available codeless testing frameworks take on? If it’s not significant enough, then breaking down the testing approach for codeless or conventional test automation might be an inefficient approach.
- Market availability of codeless testing tools and platforms capable of catering to specific industry needs. This comparison and research, especially if the QA is not familiar with codeless testing, may itself be a time-intensive activity.
- Do the codeless testing tools in consideration capable of testing code compliance and security? This is a major concern in industries like finances, public works, healthcare, etc.
- Will the project be able to realize the full potential of codeless testing, including but not limited to opening the testing up to business teams and other stakeholders outside the development and dedicated testing domain?

The codeless test automation strategy can also be tool-centric and tool-defined. Based on which characteristics of codeless automation tools are most important for the development/testing team (script reusability, collaboration, cross-platform testing, etc.), the team may choose a tool [21] and create a testing strategy or devise a testing approach around it.

The limitations of codeless test automation tools are difficult to distinguish between limitations that are associated with their automation capabilities or the codeless feature. Test automation, especially the record-and-playback version of it, has been around far longer than codeless testing, so we can assume modern tools have overcome many of these limitations. Still, many of the top codeless testing tools still vary greatly in comparison categories [22], like the ease of learning, data-driven testing capabilities, cost (training and subscription/purchase), etc.

AI-powered codeless test automation tools like AccelQ allow users to write testing scenarios in the English language, making it far more accessible and useful for business teams. If we consider it one end of the codeless testing spectrum, the other end might be low-code test automation tools, which require very little coding. However, there is a linguistic/labeling problem here as the tools that are marketed as codeless test automation tools are also often categorized as low-code test automation tools. Assuming the primary reason a company is looking into codeless test automation tools and is developing a testing strategy to support it, script-less test automation/testing tools are another avenue worth exploring from a development QA perspective.

Script-less testing, even in its most basic form, is a superior subset of/alternative to codeless testing, as it takes another step in making the process of testing and, by extension, development QA. That step is script writing (which requires scripting knowledge) for test cases and script maintenance. Script-less testing tools like Testar allow you to test GUIs seamlessly and only save sequences/events that flagged problems in the system under test (SUT) [23]. Another premise of script-less testing is the testing tool mimicking the behavior of a human user (sequences and their variations) to identify potential flaws. This may require Action Selection Rules (ASR) that are manual or with another layer of automation built into them, i.e., an evolutionary algorithm automatically creating ASRs [24].

Automated testing and script-less testing may actually have the potential to not just simplify and streamline development QA and save testing time; it may also redefine parts of development QA and stretch it beyond the current frameworks.

5. Result

From the maturity of codeless test automation tools and advances in script-less testing, it's easy to deduce that it has become a prevalent part of development QA, especially for mobile and web applications. However, most testing tools require scripting knowledge and ability, keeping it mostly in the realm of developers (if not exclusively limiting it to testers). Even script-less testing is limited to certain testing areas and is more relevant to the finalized version of the products that users are likely to interact with. That's because testing code functionalities and integrations would require in-depth knowledge of programming.

Even with the current limitations of the tool and codeless testing paradigm as a whole, it’s easy to see its potential to redefine and limit the role of dedicated development QA teams, testers, etc. With agile and DevOps becoming more prevalent and testing-driven development becoming the norm, automated testing may become a standard part of the development process. However, it may require upskilling and reskilling of the existing workforce, especially if a company decides to limit itself to one or a few proprietary test automation tools. The ability of these tools to test code snippets before they become fully functional and integrable features may also influence how extensive a part of the development QA they become.

6. Discussion

Codeless testing tools are a great asset for testers, making their job easier and testing much faster and far more streamlined (with fewer human errors). It's on its way to becoming an asset for development teams and, with the right tools and testing approach, may help them test code frequently within the development process. However, dedicated studies conducted by (or on) the development teams using codeless test automation tools and following a test-driven approach to development may reveal how effective this practice is. If the entire testing cycle, from unity to functionality and security testing, is reiterated at the end of the development lifecycle, then testing (codeless or conventionally automated) may prove redundant. However, if the flaws revealed at the final QA tests require comprehensive and costly code repair and modifications, then testing during development (with the help of codeless test tools) may prove useful. The question remains that should it become a standard practice of evaluating on a case-by-case basis?

What should be the approach regarding the sharing of test cases? For proprietary tools developed and maintained by a sizable team, it might be easier to maintain a repository of test cases created by their clients as long as they don't reveal anything about the proprietary code and functionality of the digital products of their clients. Such a repository (especially if maintained separately for different industries) can be a powerful asset for a codeless automation tool, enhancing its ability to save testing time and equipping each user/tester with testing data and approaches beyond their individual/team capabilities. However, maintaining such a repository would come with its own challenges and it’s an avenue that should be evaluated and studied from a development and testing perspective.

Codeless test automation and its interaction and augmentation with artificial intelligence is still a nascent concept and needs to be studied for a lot of potential questions. If tools like chat GPT can produce testing scripts, should a company still invest in a costly script-less test automation tool? Can AI-powered testing tools make existing codeless testing tools redundant by automating the testing process out of the hands of developers/testers?

The limitations of codeless testing and script-less testing should also be evaluated, especially with reference to final product quality.

7. Conclusion

Codeless test automation is already becoming a significant part of the web application mobile application testing and, by extension, QA and, to an extent, desktop application/software QA. Best practices, extensive and cross-platform test libraries, new testing paradigms, industry-compliant test automation tools, and orchestration of multiple codeless testing tools will further help codeless test automation become a more standard part of the development of QA. AI-based testing tools may overcome the current automation limitations, and dedicated evolving algorithms (powered by contextually sentient AI tools) may push codeless testing deeper into the development process, effectively reducing the scope of conventional development QA.

References

[1] V. Garousia and B. Küçük, "Smells in software test code: A survey of knowledge in industry and academia," *Journal of Systems and Software,* vol. 138, pp. 52-81, 2018.

[2] A. Vahabzadeh, A. M. Fard and A. Mesbah, "An empirical study of bugs in test code," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Bremen, Germany, 2015.

[3] M. Fewster and D. Graham, Software Test Automation, ACM Press New York, 1999.

[4] C. Rankin, "The Software Testing Automation Framework," *IBM Systems Journal*, vol. 41, no. 1, pp. 126-139, 2002.

[5] D. P. Nguyen and S. Maag, "Codeless web testing using Selenium and machine learning," in *ICSOFT 2020: 15th International Conference on Software Technologies*, Online, France, 2020.

[6] A. Holmes and M. Kellogg, "Automating Functional Tests Using Selenium," in *AGILE 2006 (AGILE'06)*, Minneapolis, MN, USA, 2006.

[7] X. Bai, M. Li, B. Chen, W.-T. Tsai and J. Gao, "Cloud testing tools," in *Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE)*, Irvine, CA, USA, 2011.

[8] B. Oliinyk and V. Oleksiuk, "Automation in software testing, can we automate anything we want?," in *CS&SE@SW 2019*, 2019.

[9] K. Wiklund, S. Eldh, D. Sundmark, and K. Lundqvist, "Impediments for software test automation: A systematic literature review," *Software: Testing, Verification, and Reliability,* vol. 27, no. 8, 2017.

[10] P. Runeson, "A survey of unit testing practices," *IEEE Software,* vol. 23, no. 4, pp. 22-29, 2006.

[11] L. Williams, G. Kudrjavets and N. Nagappan, "On the Effectiveness of Unit Test Automation at Microsoft," in *2009 20th International Symposium on Software Reliability Engineering*, Mysuru, India, 2009.

[12] M. Chalashkanov, "Evaluation of Test Algorithms Using Different Programming Languages," in *Sixth International Scientific Conference Telecommunications, Informatics, Energy and Management - TIEM 2021*, 2021.

[13] A. Contan, C. Dehelean, and L. Miclea, "Test automation pyramid from theory to practice," in *2018 IEEE International Conference on Automation, Quality, and Testing, Robotics (AQTR)*, Cluj-Napoca, Romania, 2018.

[14] E. Borjesson and R. Feldt, "Automated System Testing using Visual GUI Testing Tools: A Comparative Study in Industry," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, Montreal, QC, Canada, 2012.

[15] B. Haugset and G. K. Hanssen, "Automated Acceptance Testing: A Literature Review and an Industrial Case Study," in *Agile 2008 Conference*, Toronto, ON, Canada, 2008.

[16] E. Alégroth, M. Nass and H. H. Olsson, "JAutomate: A Tool for System- and Acceptance-test

Automation," in *2013 IEEE Sixth International Conference on Software Testing, Verification, and Validation*, Luxembourg, Luxembourg, 2013.

[17] F. Khorram, J.-M. Mottu and G. Sunyé, "Challenges & opportunities in low-code testing," in *MODELS '20: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, Virtual Event, Canada, 2020.

[18] C. A. Fortunato, F. Furtado, F. Selleri, I. d. F. Junior and N. L. Júnior, "Quality Assurance in Agile Software Development: A Systematic Review," *Brazilian Workshop on Agile Methods,* pp. 142-148, 2016.

[19] N. Walkinshaw, Software Quality Assurance - Consistency in the Face of Complexity and Change, Springer Cham, 2017.

[20] S. Goericke, The Future of Software Quality Assurance, Springer Nature, 2020.

[21] B. Majeed, S. K. Toor, K. Majeed, and M. N. A. Chaudhary, "Comparative Study of Open Source Automation Testing Tools: Selenium, Katalon Studio & Test Project," in *2021 International Conference on Innovative Computing (ICIC)*, Lahore, Pakistan, 2021.

[22] D. KAKARAPARTHY, "OVERVIEW AND ANALYSIS OF AUTOMATED TESTING TOOLS: RANOREX, TEST COMPLETE, SELENIUM," *International Research Journal of Engineering and Technology (IRJET),* vol. 4, no. 10, pp. 1575-1579, 2017.

[23] P. A. F. P. R. O. R.-V. A. M. Tanja E. J. Vos, "testar – scriptless testing through graphical user interface," *Software: Testing, Verification, and Reliability,* 2021.

[24] L. V. Hufkens, "Evolutionary Scriptless Testing," in *Research Challenges in Information Science. RCIS 2022*, 2022.