


Improving UI Test Automation using Robotic Process Automation

Marina Cernat¹, Adelina-Nicoleta Staicu² and Alin Stefanescu¹ ^a

¹Department of Computer Science and Research Institute of the University of Bucharest, University of Bucharest, Romania

²Cegeka, Romania

Keywords: Robotic Process Automation, RPA, UI Testing, Test Automation.


Abstract: Robotic Process Automation (RPA) is now one of the fastest growing segments in enterprise software. This technology uses so called “software robots” that can mimic humans interacting with various applications at the UI level. Thus, RPA achieves automation of various UI scenarios, without writing dedicated software to implement them. In this position paper, we open a discussion on the opportunities and challenges of using RPA to improve the test automation.

1 INTRODUCTION

Robotic Process Automation (RPA) (van der Aalst, 2018; Syed, 2020) is one of the strongest contenders in the enterprise software market, being also the fastest growing segment in 2019 (Gartner, 2019). The technology found a sweet spot in enterprise automation by emulating repetitive tasks done by a human operating a computer. The first-class citizen in this context is a so-called “software robot” that interacts with the system at the UI level, e.g., clicking or filling web forms with data copied from various data sources that are accessible through the front-end. Being in principle agnostic to the technology behind the UI, RPA can be easily deployed and integrated in the workflows of a company including legacy applications that do not offer proper APIs. In fact, one of the promises of RPA is that it is cheaper and quicker to implement compared to writing a dedicated software. More precisely, the most suitable tasks for RPA are those that are repetitive, but not frequent and structured enough to economically justify full enterprise automation. On the other end of the spectrum are those tasks that require a lot of creative human input, are ad-hoc, or too infrequent to be worth the investment in automation (van der Aalst, 2018). In fact, due to its nature, UI testing is an area that would be amenable to RPA automation, and this is the very idea that we explore in this paper.

Software testing is a very important stage in software development, with a proportion of 25% of

IT spending dedicated to QA and testing (Sogeti, 2019). However, less than 20% of the testing processes are automated, so there is a lot of potential to improve the state-of-the-practice. In fact, the need for test automation is reflected in its healthy growth – the automation testing market is estimated to double its size from USD 12.6 billion in 2019 to USD 28.8 billion in 2024 (Markets and Markets, 2019). Testing activities are very diverse: from unit testing, to integration testing, to UI testing. While unit testing is almost entirely automatic, UI testing still involves a lot of manual testing, with testers clicking through interfaces following certain scenarios for acceptance testing or using their intuition in exploratory testing. In general, automating UI testing is regarded as challenging (Aho, 2018) due to fragile scripts when the interface is changing, high maintenance (Alegroth, 2016), difficulty in defining the test oracles (Memon, 2013), estimating the costs of automation (Dobslaw, 2019), or combinatorial explosion of the states to be explored (Nguyen, 2014; Vos, 2015). Moreover, other complications appear when the UI testing needs to deal with scenarios involving several applications or even remote ones. In this situation, one does not have the same control of the GUI elements as when only one web application is under test (via e.g., Selenium). RPA may provide a solution to some of these problems since it works very well at the UI level, in a heterogeneous environment, is in many cases scriptless (thus easier to learn), more stable, and

 <https://orcid.org/0000-0002-8418-2643>

profits from recent advanced features such as computer vision. Industry already started to understand its potential benefits to UI test automation (Bhukan, 2017; Murphy, 2019), even though RPA cannot be applied to UI testing out-of-the-box (Ariola, 2019). We hope that we spark the interest of the academic community to investigate and combine state-of-the-practice in RPA with state-of-art in testing to improve test automation (Arcuri, 2018).

The structure of the paper is the following: After a short introduction in RPA, we describe two usage scenarios examples. Then, we discuss opportunities and challenges as well as related work. Finally, we conclude with future work and an appendix.

2 A SHORT INTRO TO RPA

Robotic Process Automation is the technology that enables the automation of a process by imitating and integrating the actions of a person who interacts with digital systems when executing a process. Note that the RPA does not replace existing systems. Instead, it works with the current enterprise landscape and executes well-established actions. RPA interacts with the system in the same way a human would, but faster, at a lower cost, and with less errors. This technology offers the possibility to integrate modules developed in programming languages such as VB.NET, C#, Python, or Java.

The RPA tool market is highly dynamic with several companies competing in a fast-expanding environment (Gartner, 2019). The biggest players are UiPath, Automation Anywhere, and Blue Prism. In the rest of the paper, we will refer to the UiPath platform, because it is currently the leader in the field and, also, the second author of the paper has 2+ years of industrial experience using its technology.

The main components of UiPath platform are:

- **UiPath Studio:** An IDE that relies on Microsoft Workflow Designer and Microsoft Visual Studio tools and uses the Visio-style process views to graphically model workflows as sequences, flowcharts, or state machines. It gives the user an ergonomic experience. Built on .Net framework, UiPath Studio allows working with all types of variables, arguments, reading data from several formats (Excel, PDF, common databases, Word, desktop or web-based applications), writing data, creating reports, handling keyboard strokes and mouse clicks, as well as Optical Character Recognition (OCR).

- **UiPath Robot:** The workflows created in UiPath Studio are executed by a UiPath robot. There are two main types of robots: attended and unattended, the difference being that the former requires human inputs at some point during the execution.
- **UiPath Orchestrator:** A component that manages the UiPath robots across various platforms.

Due to requests and needs from the users, RPA has become, in the past years, a conglomerate of different technologies that are combined in order to help and ease the development of an RPA project. Advanced plugins, including machine learning, cognitive automation and computer vision, can now be used by the robots. Also, third party integration with popular cognitive services from Microsoft, Google, IBM are readily available.

Another challenge and opportunity for RPA is the automation in virtual or remote environments a.k.a. VDIs (virtual desktop interfaces). There is a clear growth in VDI usage among enterprise customers, proving a need of quick and stable UI automations for technologies such as Citrix, VMware and Windows Remote Desktop. UiPath was among the first ones to implement it using latest breakthroughs in computer vision research. Thus, it solved for the robot the difficult problem to have obtain the underlying properties of UI elements (buttons, text fields, etc.). In VDI, this ability is damaged because of the lack of a traditional interface since the robot only obtains an image of a remote desktop. Mixing computer vision, machine learning, OCR, text fuzzy matching and a multi-anchoring system, robots now automatically recognize on-screen elements, not relying on IDs, hidden properties or metadata. Also, this solution for VDIs can be extended also to work for similar situation of scanned PDFs, Microsoft Silverlight, images etc.

Yet another area of great progress of RPA recently is that of processing and understanding unstructured information, especially text, through data mining, natural language processing, and machine learning.

These RPA advancements could be very helpful when solving some of the problems of UI testing.

3 A SIMPLE EXAMPLE OF TESTING USING RPA

To illustrate better how RPA could be used for test automation, we will provide a small example. Assume that we want to test two calculator apps, one installed locally (e.g., the Windows calculator app),

but also one online (e.g., the calculator that appears on top of the Google page when you search for “calculator”). A scenario that a human would follow to test them could be: take two numbers, for instance 1 and 2, add them on the calculator (desktop and online) and verify if the output of the calculator is indeed the expected value of 3.

To automate this in RPA, we quickly build a robot that can test in parallel both calculators using test data extracted from an Excel spreadsheet. The spreadsheet contains test data with operands and an operator (1+2, 5+7, 3*8) and the expected computation result for them (3, 12, 24, respectively). The robot can implement two test cases, one for each app, in the usual way of doing testing, by reading the data, executing the application under test and comparing the output with the expected values. If the values are equal, the test has passed, otherwise it failed. The graphical model of the robot is provided in Fig. 1 in the Appendix, where we have the sequence with the steps. When it runs, the robot will do the following

- Step 1 (test setup): It reads the values from Excel (see Fig. 2 in the Appendix) and keeps them into an internal data table variable.
- Step 2: It opens the desktop and online calculator in the same time, but in different threads.
- Step 3 (test execution): For each row of the data table (which contained the test data, 1, +, and 2), the robot performs the encountered operation by clicking the buttons of the calculator, then it takes the obtained result and writes them back in the Excel spreadsheet (to be used for debugging, if the tests fail). See Fig. 3 in the Appendix.
- Step 4: It closes both calculators.
- Step 5 (test assert): It compares the column of the actual results and the column of the expected results from the spreadsheet and put into the “Status Desktop”, respectively “Status Online” columns, the test results for each test being “Passed”, if the results were equal, and “Failed” otherwise. See Fig. 4 in the Appendix.

Thus, the robot is graphically modelled and will automatically click through the interfaces of a desktop app, but also of a web app in the browser, implementing a data-driven test suite and writing the results back into a file for the test reports. Note that everything is done at the UI level and the robot seamlessly switches between applications (Excel, desktop app, web browser). This would not be possible using a classical UI web testing framework such as Selenium, which would only be able to deal with the web app in the browser, but not with the desktop app.

4 AN END-TO-END EXAMPLE OF TESTING USING RPA

To understand how RPA may improve UI testing, we exemplify a more complex testing scenario that uses several applications and technologies.

Assume we want to test various functionalities of a back-office banking application. An unattended robot could be implemented to do the following tasks.

First of all, we want to test if the application is working on several web browsers, so the robot should successfully login regardless of the used browser. The robot keeps an oracle application testing suite logging if the test passed or not in an Excel format. If the login is successful, the test has passed, else the test failed.

After testing the login functionality, the robot tests the ability of internal money transfers in the application. So, it transfers an amount of money to another account, then it checks if the money arrived. If the transfer is successful, the test passed, and the result is saved. After doing the payment, the bank clerk, and now also the robot, should be able to see and save the payment confirmation in PDF format. So, the robot also tests this functionality, opening the menu where it finds the payment confirmations and saves the most recent file. In order to check if the PDF file contains the fields that represent the amount of money sent and the account to which the money was transferred, the robot opens the PDF file and checks the corresponding fields. If they exist and are correct, the test passed.

After this check, we move forward and test an application that loads these PDF files in a database.

First, the robot checks the database connection. Then, the robot checks if the files are correctly loaded in the database. For all the files previously processed, the robot interrogates the database and checks if the actual result corresponds to the expected result.

Last but not least, the robot will access for some final verifications a sensitive, more restricted area of the bank system remotely through a VDI. In this case, the robot has access only to screenshots of the remote applications, without access to the logical elements of the UI. However, also in this case the robot can use the latest computer vision RPA add-ons (see previous section) and smoothly solve the given tasks.

We notice that writing test scripts to implement all these test cases is not at all easy, whereas using the advanced capabilities of RPA, the task becomes feasible.

5 OPPORTUNITIES AND CHALLENGES OF USING RPA FOR TESTING

As already hinted until now, we consider that there are many opportunities of using RPA to improve UI test automation.

First of all, RPA is much more accessible than test scripting or programming. Most RPA tool providers, including UiPath, offer some products that do not require technical knowledge (see the lightweight UiPath StudioX solution aimed at non-technical users). Therefore, businesspersons, field experts or manual testers without technical knowledge can automate some of their UI testing tasks. Thus, using RPA, we can imagine that acceptance tests can be automated easier and at a lower cost even by the end users (for example, the doctors from a hospital) with some training before, because they are the experts in the respective field and this is the best way to check if the tested application or usage scenario is correct.

Then, a robot can access through the UI any tools available irrespective if APIs are provided or not. Thus, if certain tools, including testing tools, are used inside an organization, the robot can use them to achieve its goal. An RPA robot could implement test scenarios that are more flexible, maintainable, stable, easier to integrate (ERP legacy systems, CRM, calendar, email, PDF, VDIs), and also accurate. This was proved by the fast adoption of RPA in enterprise automation, but we expect that this will still hold in the UI test automation domain (in fact, that could constitute a good topic for an empirical software engineering research project).

Last but not least, using RPA for testing could capitalize on the high growth and advanced developments of the RPA tool providers. They are now in a race to include as many sophisticated features as possible, and UI testing could benefit from them, since the UI testing tool providers are not so fast in implementing the latest technologies, especially in the field of AI. So, the robots will become smarter in several dimensions and this could clearly benefit a UI testing process deploying them.

However, with great opportunities come also challenges. First of all, the RPA as-is must be adapted to the UI testing requirements. This means that one should define a suitable test infrastructure, RPA tools must be better integrated with existing test tools for test management, reporting, test execution, and other test automation tools such that the strengths of all are combined. Then, one should see what can be further automated such that state-of-the-art in testing can be

used also in the context of RPA-based testing. So, we should investigate how model-based testing, search-based testing, automatic test generation, keyword-driven and data-driven testing, fuzz-testing, exploratory testing, to name a few, could be embedded into RPA test robots. Also, the challenges of UI testing (Aho, 2018) should be revisited, investigating which of them could be solved by RPA-based UI testing. Last but not least, in order to be attractive to industry, cost models that include RPA licenses should be devised such that the final testing toolchain and process will provide a good return-on-investment (Dobslaw, 2019).

6 RELATED WORK

We could not find any reference in the academic literature reading the idea of using RPA technologies to automate UI testing. This is understandable since the RPA is a very recent technology that only recently reached maturity (van der Aalst, 2018) and whose challenges are started to be discussed (Syed, 2020).

There are only a handful of research papers that may be relevant to our discussion involving RPA for quality assurance. (Lübke, 2016) discussed the possibility of using BPMN for test case design, which is similar to the graphical modeling of RPA robots. (Moffitt, 2019) proposed an approach of using RPA for auditing, which resembles in some way a testing process. (Beschastnikh, 2017) envisaged a framework that deploys “bots” to do code analysis during the software development process, but they do not work at UI level and do not use RPA. (Enoiu, 2019) investigated the general concept of test agents that could distributively and cooperatively run a test plan. It is an interesting idea to adapt for the context of RPA test robots. (White, 2019) studied the problem of UI testing for VDIs and implemented a solution to extract UI test relevant information from images. This could complement the current RPA plugins that do the same task.

We analysed the industrial UI test tools. We tried to be as comprehensive as possible in our comparative analysis and we downloaded (whenever possible) the tools and experimented with them in order to understand their main features, strengths and weaknesses. We do not have space in this short paper to discuss the findings, but we only mention those that offer interesting or advanced features that could be used in combination with an RPA-based test framework: Selenium (see also (Besant Technologies, 2019)), AppliTools, RobotFramework, Power Automate, Eggplant, EyeAutomate, Squish,

Tricentis, Mabl, test.ai, apptest.ai, Functionize, testim.io, as well as academic tools such as GUITAR (Nguyen, 2014) and TESTAR (Vos, 2015).

Note that some of the RPA tools evolved from UI testing tools, e.g., Automation Anywhere (which in the meanwhile is not active anymore in the testing market), while UI testing companies, e.g., Tricentis (Murphy, 2019), Leapwork, want to enter the RPA market. On the other hand, some RPA companies such as UiPath are starting to provide RPA-based testing solutions¹.

7 CONCLUSIONS

The goal of this short paper is to promote the idea of using the latest RPA technologies and research in order to improve the state-of-the-art in UI test automation. Based on our initial investigations, we believe that there is a great potential in this idea both from an academic as well as industrial point of view.

As future work, there are many aspects that we plan to address. First, we will implement several complex test scenarios using UiPath tooling to make an inventory of strengths and weaknesses with respect to testing. Then, we will try to enhance them by integrating state-of-the-art tools and approaches from the testing research (see also the discussion at the end of Section 5), but also existing commercial UI tools (see the tool list in Section 6). Some of the topics to focus on are: the use of AI to obtain a “smarter” test robot; the generation of test robots using process discovery and understanding (Gao, 2019) (see also UiPath Explorer component); but also the development of methods to test the RPA implementations themselves (see also (Cewe, 2018)). Last but not least, we will perform all the above in the context of a collaboration with UiPath, which showed interest to provide feedback, access to tooling and industrial use cases to validate the resulting ideas and prototypes.

ACKNOWLEDGEMENTS

This work was supported by a grant of Romanian Ministry of Research and Innovation CCCDI-UEFISCDI, project no. 17PCCDI/2018. We also thank to Ingo Philipp, Vice President at UiPath, for inspiring discussions on the topic of the paper.

¹ <https://www.uipath.com/product/test-suite>

REFERENCES

- van der Aalst, W., Bichler, M., A. Heinzl, A., 2018. Robotic Process Automation. *Business & Information Syst. Eng.* 60 (4), pp. 269-272, Springer.
- Aho, P., Vos, T., 2018. Challenges in Automated Testing Through Graphical User Interface. In *Proc. of ICST Workshops 2018*, pp. 118-121, IEEE.
- Alegroth, E., Feldt, R., Kolstrom, P., 2016. Maintenance of automated test suites in industry: An empirical study on visual GUI testing. *Information and Software Technology*, 73, pp. 66–80, Elsevier.
- Arcuri, A., 2018. An experience report on applying software testing academic results in industry: we need usable automated test generation. *Empirical Software Engineering* 23 (4), pp. 1959-1981, Springer.
- Ariola, W., 2019. RPA for Software Test Automation: Not So Simple. *CIO Magazine*. Online at: <https://www.cio.com/article/3409056/rpa-for-software-test-automation-not-so-simple.html>
- Besant Technologies, 2019. RPA vs Selenium. *Industry blog post*, 2019. Online at: <https://www.besanttechnologies.com/rpa-vs-selenium>
- Bhukan, S., 2017. Robotic Process Automation and the Testing Future. *Industry blog post*. Online at: <https://www.testingbits.com/robotic-process-automation-and-the-testing-future>
- Beschastnikh, I., Lungu, M., Zhuang, Y., 2017. Accelerating Software Engineering Research Adoption with Analysis Bots. In *Proc. of ICSE-NIER, 2017*, pp. 35-38, IEEE.
- Cewe, C., Koch, D., Mertens, R., 2018. Minimal Effort Requirements Engineering for Robotic Process Automation with Test Driven Development and Screen Recording. In *Proc. of BPM'18 Workshops*, LNBIP vol. 308, pp. 642-648, Springer.
- Dobslaw F., et. al, 2019. Estimating Return on Investment for GUI Test Automation Tools. arXiv report no. CoRR abs/1907.03475, 12 pp.
- Enoiu E., Frasheri, M., 2019. Test Agents: The Next Generation of Test Cases. In *Proc. of NEXTA'19*, ICST Workshops 2019, pp. 305-308, IEEE.
- Gao, J., van Zelst, S., Lu, X., van der Aalst, W., 2019. Automated Robotic Process Automation: A Self-Learning Approach. In *Proc. of OTM Conferences 2019*, LNCS 11877, pp. 95-112, Springer.
- Gartner, 2019. Magic Quadrant for Robotic Process Automation Software. *Market research report*, no. G00379618.
- Lübke D., van Lessen, T., 2016. Modeling test cases in BPMN for behavior-driven development. *IEEE Software* 33 (5), pp. 15-21, IEEE.
- Moffitt, K., Rozario, A., Vasarhelyi M., 2019. Robotic Process Automation for Auditing. *J. of Emerging Technologies in Accounting* 15 (1), pp. 1-10.
- Markets and Markets, 2019. Automation Testing Market by Component, Endpoint Interface, Organization Size,

Vertical, and Region - Global Forecast to 2024. *Market research report* no. TC 6163.

Memon, A., Banerjee, I., Nagarajan, A., 2013. What Test Oracle Should I Use for Effective GUI Testing? *In Proc. of ASE 2003*, pp. 164-173, IEEE.

Murphy, T., 2019. Test automation + RPA,” *Tricentis industry webinar* Online at: <https://www.tricentis.com/resources/test-automation-rpa>

Nguyen, B., Robbins, B., Banerjee, I., Memon, A., 2014. GUITAR: an innovative tool for automated testing of GUI-driven software. *Automated Softw Engineering* 21 (1), pp. 65-105, Springer.

Sogeti, 2019. World Quality Report 2019-2020, 11th edition. *Market research report*.

Syed R. et al., 2020. Robotic Process Automation: Contemporary themes and challenges. *Computers in Industry* 115.

Vos, T., Kruse, P., Condori-Fernández, N., Bauersfeld, S., Wegener, J., 2015. TESTAR: Tool support for test automation at the user interface level. *Int. Journal of Information System Modeling and Design* 6 (3), pp. 46-83, IGI Global.

White, T., Fraser, G., Brown, G., 2019. Improving random GUI testing with image-based widget detection. *In Proc. of ISSTA 2019*, pp. 307-317, ACM.

APPENDIX

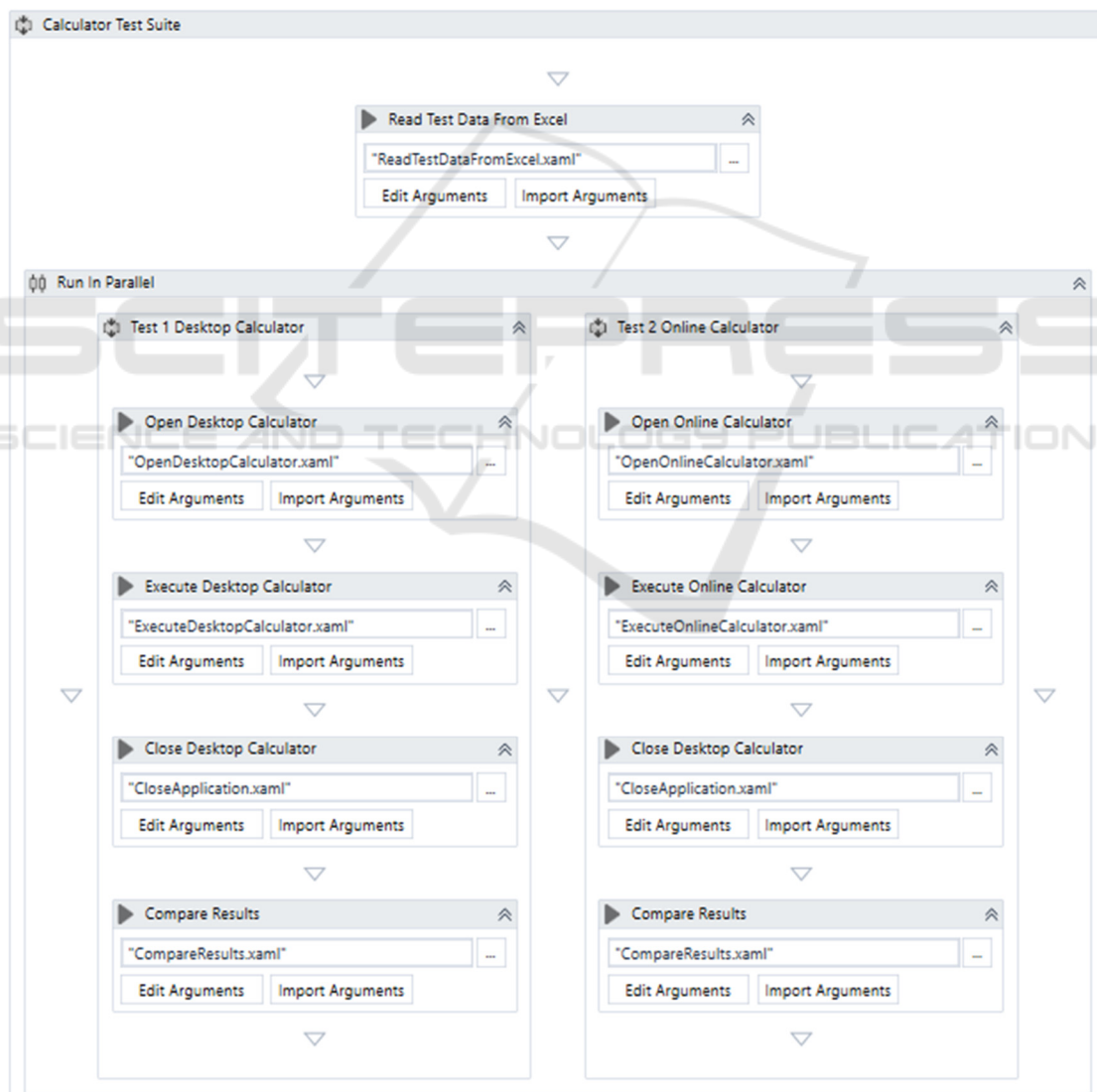


Figure 1: Implementation of a test robot in UiPath for a test suite with two test cases: the one on the left to verify the desktop calculator and the one on the right to verify an online calculator.

Test data 1	Operator	Test data 2	Expected Result	Status Desktop	Actual Result Desktop Calculator	Status Online	Actual Result Online Calculator
1	+	2	3	Passed	3	Passed	3
5	+	7	12	Passed	12	Passed	12

TestDataSheet +

Figure 2: Excel file containing the test data: The first input data in the first 3 columns and output data in the 4th column. The rest of the columns are used to record the results of the tests.



Figure 3: The RPA sequence for automating the calculation 1+2 in the desktop calculator (the screenshots show the clicks performed by the robot).

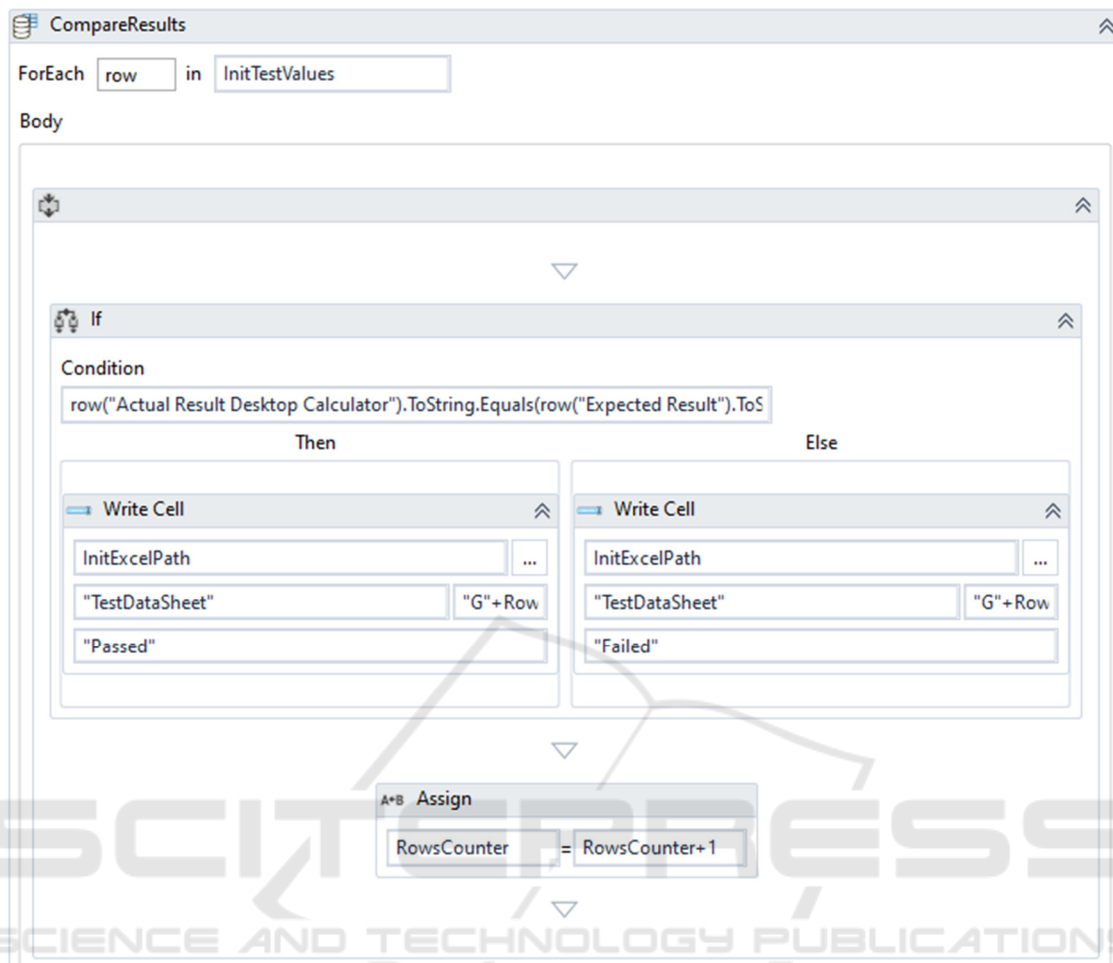


Figure 4: The implementation of the comparison between the expected value (oracle) and the calculated value.