Leotta, Maurizio; García, Boni; Ricca, Filippo; Whitehead, Jim. (2023). Challenges of End-to-End Testing with Selenium WebDriver and How to Face Them: A Survey. *2023 IEEE 16th International Conference on Software Testing, Verification and Validation, 16–20 April 2023, Dublin, Ireland: Proceedings*, Piscataway, NJ: IEEE. Pp.: 339-350.

DOI: https://doi.org/10.1109/ICST57152.2023.00039

Challenges of End-to-End Testing with Selenium WebDriver and How to Face Them: A Survey

Maurizio Leotta[1], Boni García[2,3], Filippo Ricca[1], Jim Whitehead[4,5]

[1] Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS), Università di Genova, Italy
[2] Universidad Carlos III de Madrid, Madrid, Spain
[3] Open Source Program Office, Sauce Labs Inc., San Francisco, CA, USA
[4] Baskin School of Engineering, University of California, Santa Cruz, CA, USA
[5] AI Research, Development, and Experimentation (AIX) Team, Sauce Labs Inc., San Francisco, CA, USA
maurizio.leotta@unige.it; boni.garcia@uc3m.es; filippo.ricca@unige.it; ejw@ucsc.edu

***Abstract*—Modern web applications are complex and used for tasks of primary importance, so their quality must be guaranteed at the highest levels. For this reason, testing techniques (e.g., end-to-end) are required to validate the overall behavior of web applications. One of the most popular tools for testing web applications is Selenium WebDriver. Selenium WebDriver automates the browser to mimic real user actions on the web.**

While Selenium has made testing easier for many Teams worldwide, it still has its share of challenges. To better understand the challenges and the corresponding solutions adopted we decided to undertake a personal opinion survey from the industry (in total with 78 highly skilled participants) with a focus on the Selenium ecosystem.

The results allow understanding which challenges are considered more relevant by professionals in their daily practice and which are the techniques, approaches, and tools they adopt to face them. Therefore, this study is useful to (1) practitioners interested in understanding how to solve the problems they face every day and (2) researchers interested in proposing innovative solutions to problems having a solid industrial impact.

***Index Terms*—End-to-End Testing, Web Testing, Selenium WebDriver, Personal Opinion Survey, Challenges.**

I. Introduction

End-to-end (E2E) testing of web applications, a type of black box testing based on the concept of test scenario, turned out to be a lot effective for improving the quality of the applications under test. For this reason, many software companies around the world have adopted it. Nowadays, Selenium WebDriver [1] is considered the de-facto library for developing E2E tests for web applications. Selenium [2] is an umbrella project to provide browser automation features to impersonate users who interact with browsers such as Chrome, Firefox, and Edge automatically. A recent study in the context of software testing by Cerioli et al. [3] appoints Selenium as one of the most used and valuable testing library today.

Selenium is one of the best tools for automating E2E test cases, but like anything good, it has some drawbacks too. Without a doubt, Selenium makes the testing phase more straightforward, but there are some challenges software developers and testers face while using it [4]. Among these, one particularly insidious and treated in the academic field is flakiness [5], a condition in which test cases fail for no apparent reason in a non-deterministic way.

This paper presents the results of an online survey carried out with professionals to understand better and characterize the most relevant challenges in E2E testing with Selenium WebDriver. A total of 78 experienced participants from 28 countries and four different continents completed this survey in 2022. The results reveal the main challenges developers face and how practitioners try to solve the problems (e.g., flakiness and fragile tests) that arise daily using Selenium.

The paper is organized as follows: Sect. II provides the essential background about E2E testing and presents the challenges considered in this survey. Sect. III describes the design and the procedure of the personal opinion survey we conducted. Sect. IV presents the results and lists the possible threats to validity. Finally, Sect. V reports the related literature while Sect. VI concludes the paper.

II. Challenges

This section briefly introduces the 13 challenges we investigated with our personal opinion survey. The 13 challenges were selected using a multi-step approach: (1) gray literature analysis and (2) thoughtful selection. So as a first step, we searched on the web for various articles (gray literature including many blogs, tutorials, etc.) to try to understand what are, according to the practitioners, the most challenging aspects in E2E web testing with a particular focus on tests implemented using Selenium. In particular, we performed a search on Google using the following query "challenges problems limitations E2E web testing Selenium WebDriver". We analyzed the first 30 articles and noted down the challenges they describe.

As a second step, we refined challenges by removing those that have a too technical focus (e.g., handling pop-up windows, handling captcha images, implementing data-driven testing) or can be considered out of the scope of this study (e.g., no support to desktop and mobile applications testing, defining naming conventions in the test code). This step is essential since we have to limit the challenges to analyze in order to maintain the questionnaire reasonably short. Finally, we double-checked and refined/rephrased the challenges list relying on

our experience in the E2E arena, which is witnessed by 100+ published articles in the field, including one of the first works on web testing (ICSE 2001 [6]), a book [7], and also the professional activity of two of us. To get additional feedback from Selenium experts, we shared the found challenges with the Selenium TLC (Technical Leadership Committee), which is the board responsible for the high-level technical guidance of the project. Thus, we are confident that the majority of the most relevant challenges were included in the final list: this has also been confirmed by the survey participants since none of them suggested including other challenges in our analysis.

In the following, we briefly introduce each challenge without providing possible solutions to solve or at least mitigate them, which will instead be described together with the survey results.

Slowness. E2E web test scripts are often considered slow to run, i.e., they take a long time to complete [4]. This is certainly true compared to other automated tests, such as the popular unit tests. Since E2E tests act through the web app interface rendered through a browser and validate all the layers from which a web app is composed, it makes the execution of E2E tests slower compared, for example, with unit tests taking care of validating only a method of a particular class.

Brittleness. E2E web test scripts are often considered brittle [8], i.e., fragile when web app evolution occurs. Specifically, test scripts may become broken when a web application evolves to accommodate requirement changes, bug fixes, or functionality extensions. When this issue occurs, test scripts typically cannot locate some links, input fields, and submission buttons, and software testers must repair them. This task is tedious and expensive since it has to be performed manually by testers.

Flakiness. E2E web test scripts are often considered flaky, i.e., they can break randomly without an apparent reason. The fact that a test can non-deterministically pass or fail when executed on the same version of the web application, without any change in both the app and the test code, can waste a lot of time for the testers trying to debug a non-existent fault in the code [9]. For this reason, flaky tests are considered insidious and dangerous [10].

Maintainability. Maintaining E2E test scripts (i.e., the activities required to keep them operative after they are deployed) is often considered troublesome. Depending on the kind of maintenance task performed on the web application, a tester has to execute a series of test script repair activities that can be categorized into two types: logical and structural [8]. Logical changes involve the modification of the web application functionality. The tester has to modify one or more steps of the broken test scripts to repair the test scripts. An example of a change request needing a logical repair activity is enforcing security through stronger authentication and thus adding a new web page containing an additional question displayed to the user when she clicks on the login button. Structural changes involve the modification of the web page layout/structure only. For instance, the login button's string on a web page may change from "Login" to "Authenticate." Often, the impact of a structural change could be considered smaller than that of a logical change. This is because a structural change requires "simply" modifying localization lines to repair the test scripts. The problem is that, in real cases, even a simple layout change (a kind of structural change) in the web application can potentially impact hundreds of localization lines [11] across many test scripts, making the maintenance time-consuming.

Asynchronous. E2E test scripts must manage asynchronous interactions, which are often considered problematic to handle. Asynchronous content (e.g., AJAX) allows the web page to retrieve small amounts of data from the server without reloading the entire page. When the test script has to interact with content modified by this asynchronous mechanism, it needs to wait till the asynchronous content has been updated (i.e., the response has been received from the server and the page updated accordingly). The main challenge in handling Asynchronous calls is knowing the waiting time for the web page content modified by such calls [7].

Time-consuming. The development of E2E web test scripts is a complex and time-consuming task [4]. This is due to the fact that to simulate user interaction with the web application, E2E tests must follow paths in the navigation graph of the web application. So, for each step of the tests, it is necessary to understand which web page it occurs, which web elements to interact with, write appropriate commands to interact with them, and so on. Since the code in Selenium WebDriver is written using classic programming languages such as Java or Python, this task can be very demanding, especially if not performed following the best practices (such as relying on specific design patterns for structuring the source code or adopting automated, robust locators generators).

Cross-browser. Cross-browser testing consists in reusing the same test logic for verifying web applications using different combinations of web browsers. [12]. Even if all web browsers nowadays support the common web standards developed by the World Wide Web Consortium (W3C), browsers can still render web pages differently. These differences in both rendering and functionality can be due to different factors, such as (1) different default settings of the browser/operating system (e.g., the default font used by a browser); (2) different user-defined settings, e.g., the screen resolution; and (3) different engines used to process web programming languages.

Failure analysis. Failure analysis (also known as troubleshooting) is the process of analyzing a failed test to understand the reason for its failure. In the context of E2E test scripts, the reasons for a failure can be many, such as problems in the functionality implementation, variations/anomalies in the web pages structure, communication problems between the layers that compose the web application, problems in any of the layers, and many others [4]. Furthermore, the failure of a test script could be caused either by the last step performed by the test (a more straightforward case to analyze) or by a previous step (more challenging to analyze). For these reasons, understanding the root cause of a failing E2E test script can be very complicated and challenging.

Infrastructure. The browser infrastructure required by Selenium scripts (e.g., browsers, drivers) is costly to set up and maintain. A test suite using the Selenium WebDriver requires

an intermediate component called *driver* in the Selenium jargon to control the browser and interact with the web application. Each browser vendor provides a specific driver for each browser's version (e.g., for Chrome version 107, ChromeDriver 107.0.5304.62 must be used). Thus, to run and keep a test suite updated over time, it is necessary to download the driver corresponding to the browser version and keep the versions browser-driver aligned during the natural evolution of the browsers since modern browsers automatically upgrade to the next stable version [13] [14]. Developers usually perform this alignment task manually, which becomes even more tiring (and expensive) if the developed test suite is multi-browser.

Scalability. Managing a large Selenium suite can be complex, particularly when this is executed in parallel. A test suite for a large web application often contains many test scripts, especially if a high coverage of the functionalities is required. Furthermore, each test script can be very complex and take a non-trivial execution time to complete (in the order of minutes or more). For this reason, parallel execution is often employed [7], highlighting also the problem of dependencies between test scripts.

Assertability. Selenium script assertions are mainly DOM-based, and it is challenging to create other assertion types (e.g., visual properties checking) [15]. Though many functional checks can be made through assertions based on data present in the DOM, this is by far more difficult (if possible) for different kinds of checks. For example, in those predicates related to the look-and-feel of the pages, the level of accessibility, or complex assertions about videos, interactive maps, etc.

Documentation. Selenium is a complex library that can be used in a multitude of contexts and to automate many kinds of tasks for different types of applications. In certain cases, the official documentation [16] could not be enough to make the most of Selenium.

Support. As in the documentation case, Selenium users may need support to solve specific problems when implementing complex scripts. It is unclear whether official Selenium support [17] (including user groups, chat rooms, Slack, etc.) is enough to get the most out of Selenium.

III. Study Definition, Design, and Procedure

We are interested to understand how Selenium experts perceive and face the 13 challenges described in Section II. Thus, we can define the **goal** of the survey as follows: *better understand the challenges of end-to-end testing with Selenium in order to address them effectively*.

In this work, we mainly take the perspective of *Industrial practitioners* interested in understanding more about the various challenging aspects (and the corresponding solutions) of the Selenium adoption and, more in general, the E2E web testing practice. Moreover, we also take the perspective of *Researchers* interested in understanding which are the challenges considered most important by the practitioners in order to propose innovative solutions for them, as well as *Teachers* to offer Software Testing courses providing in-depth information on how to solve the challenges considered most relevant by practitioners.

A. Research Questions

Given the above goal, the survey aimed at addressing the following *research questions*:

RQ1: *What are the most relevant challenges as perceived by Software Testers in Selenium-based E2E Web Testing?*

RQ2: *How is each challenge faced?*

RQ3: *Does the perceived importance of the challenges change according to the size of the Company where the survey participant works?*

B. Target Population and Sample Identification

The target population is the set of individuals to whom the survey applies. In our case, the population comprises professionals with good knowledge and experience with Selenium.

To build up a sample, according to [18], [19], we used *convenience* and *responded-driven sampling* [20]. More in detail we:

– invited the professionals attending the 2022 Selenium Conference [21] to participate in our survey. The Selenium Conference is a non-profit, volunteer-run event presented by members of the Selenium Community. The goal of the conference is to bring together Selenium developers and enthusiasts.

– asked our industrial partners for contacts of experts in the field of E2E web testing and individually invited the suggested practitioners to participate in our survey.

– posted the request for participation on the official Twitter and LinkedIn communication channels of the Selenium project.

Thus, the context consists of professionals with a potentially relevant hands-on experience in E2E testing with Selenium.

In total, we received 78 complete responses to our survey. Unfortunately, it is impossible to determine precisely how many people have been reached by our invitation messages and advertisements, so we cannot calculate the response rate. The same problem is also present in other software engineering surveys (e.g., in [22]).

C. Data Collection

Since the targeted population is distributed worldwide, we decided to collect responses via an online questionnaire from the end of July 2022 to September 2022. Using a web-based tool simplifies and speeds up the questionnaire completion with clear advantages regarding the number of responses obtained [23]. The online questionnaire has been developed and published using Google Forms [24].

D. Questionnaire Design

The questionnaire is organized into four sections: (1) survey description, (2) personal information (optional personal contacts + four questions [A-D]), (3) questions on the 13 challenges, and (4) a final open comment.

Each question in section (3) is subdivided into two parts. First, we asked the participants to evaluate the perceived

relevance of the challenges using a five-point Likert scale (1=Fully disagree, 2=Disagree, 3=Not sure, 4=Agree, 5=Fully agree). This part was mandatory. Second, we asked the participants to explain their experience in limiting/facing the challenge. This part was optional.

To harvest more answers, we decided that the questionnaire should take approximately 15 minutes to complete (long questionnaires get fewer answers than short questionnaires [25]), and we designed it accordingly.

In the questionnaire, reported in Table I, we included a question for each challenge described in Section II.

E. Survey Execution

The procedure followed to prepare, administer, and collect the questionnaire data has five main steps:

1) *Preparation and Design of the Questionnaire*. We started to define an initial version of the questionnaire agreed upon among the authors of this work. This required several iterations to ensure that we included all the relevant challenges concerning the current state-of-the-practice in Selenium-based E2E web testing. We are confident that we selected the most relevant challenges as explained in Section II.
2) *Pilot Study*. A pilot study was performed before executing the survey (i) to tune the questionnaire and (ii) to reduce the ambiguities contained in the questions. An industrial IT professional and a university professor completed a preliminary questionnaire version and provided their judgment. Following the suggestions of the two contacted experts, minor changes to the questionnaire were made. After this pilot study, we concluded that the survey was well-suited for Selenium professionals and that the questions were clear enough.
3) *On-line Deployment*. Once the questionnaire was refined after the pilot study, it was deployed online using Google Forms, as explained before.
4) *Monitoring*. During the data capture phase, our research group monitored the progress of the questionnaire submission.
5) *Data Analysis*. After questionnaires had been collected, simple analyses were performed to answer the research questions. Given the nature of this survey, which is mainly descriptive (it describes some conditions or factors found in a population in terms of its frequency and impact [26]), we applied quite exclusively descriptive statistics and showed our findings through charts. Anonymized raw data are available at: https://sepl.dibris.unige.it/2022-SeleniumSurvey.php

IV. Results

In this section, we first present some information about the respondents' background, in particular by analyzing the answers to questions A-D (see Table I, 76 responses received). Then, we analyze the results associated with the 13 questions (one for each challenge) to answer the RQs of this study.

A. Respondents' Background

To give an overview of the respondents' background, we summarized the answers to questions A-C in the first section of the questionnaire using pie charts.

ID	Question
A	How many years have you been working in the field of Web Test Automation? *(optional)*
B	How would you rate your knowledge of Selenium WebDriver? *(optional)*
C	Which is the size of your company? *(optional)*
D	In which country are you currently working? *(optional)*
1	**Slowness**. Selenium scripts are slow to run, i.e., take a long time to complete. *Is this challenge relevant?* [five-point Likert scale] *(mandatory)* *Explain your experience in limiting/facing this challenge* [open text] *(optional)*
2	**Brittleness**. Selenium scripts are brittle, i.e., fragile when web apps evolution occurs. *Is this challenge relevant?* [five-point Likert scale] *(mandatory)* *Explain your experience in limiting/facing this challenge* [open text] *(optional)*
3	**Flakiness**. Selenium scripts are flaky, i.e., can break randomly without a clear reason. *Is this challenge relevant?* [five-point Likert scale] *(mandatory)* *Explain your experience in limiting/facing this challenge* [open text] *(optional)*
4	**Maintainability**. The maintenance of Selenium scripts (i.e., the activities required to keep these scripts operative after they are deployed) is a troublesome task. *Is this challenge relevant?* [five-point Likert scale] *(mandatory)* *Explain your experience in limiting/facing this challenge* [open text] *(optional)*
5	**Asynchronous**. Selenium scripts must manage asynchronous interactions, which are problematic to handle. *Is this challenge relevant?* [five-point Likert scale] *(mandatory)* *Explain your experience in limiting/facing this challenge* [open text] *(optional)*
6	**Time-consuming**. The development of Selenium scripts is a difficult and time-consuming task. *Is this challenge relevant?* [five-point Likert scale] *(mandatory)* *Explain your experience in limiting/facing this challenge* [open text] *(optional)*
7	**Cross-browser**. Cross-browser testing implemented with Selenium (i.e., reusing the same test logic for verifying web applications using different browsers) is complex. *Is this challenge relevant?* [five-point Likert scale] *(mandatory)* *Explain your experience in limiting/facing this challenge* [open text] *(optional)*
8	**Failure analysis**. Failure analysis of Selenium scripts (also known as troubleshooting) is difficult, since the underlying cause of a failed test might be unclear. *Is this challenge relevant?* [five-point Likert scale] *(mandatory)* *Explain your experience in limiting/facing this challenge* [open text] *(optional)*
9	**Infrastructure**. The browser infrastructure required by Selenium scripts (e.g., browsers, drivers) is costly to set up and maintain. *Is this challenge relevant?* [five-point Likert scale] *(mandatory)* *Explain your experience in limiting/facing this challenge* [open text] *(optional)*
10	**Scalability**. Managing a large Selenium suite is complex (e.g. when using parallel execution). *Is this challenge relevant?* [five-point Likert scale] *(mandatory)* *Explain your experience in limiting/facing this challenge* [open text] *(optional)*
11	**Assertability**. Selenium script assertions are mainly DOM-based and it is challenging to create other assertion types (e.g., visual properties checking). *Is this challenge relevant?* [five-point Likert scale] *(mandatory)* *Explain your experience in limiting/facing this challenge* [open text] *(optional)*
12	**Documentation**. Selenium documentation is not clear, complete, and does not cover all important topics. *Is this challenge relevant?* [five-point Likert scale] *(mandatory)* *Explain your experience in limiting/facing this challenge* [open text] *(optional)*
13	**Support**. Selenium user communities (e.g., user group, Slack) are not welcoming, and do not answer my questions quickly and helpfully. *Is this challenge relevant?* [five-point Likert scale] *(mandatory)* *Explain your experience in limiting/facing this challenge* [open text] *(optional)*

TABLE I
QUESTIONNAIRE (PERSONAL INFO AND 13 CHALLENGES)

Fig. 1 summarizes the participants' years of experience focusing on the specific context of the web test automation. We can observe that more than one-quarter of the participants have a long experience in the field, exceeding ten years (red). The majority (about 40%) of the participants still have relevant experience between 5 and 10 years (yellow). Finally, another quarter has between 1 and 4 years of experience (blue), and

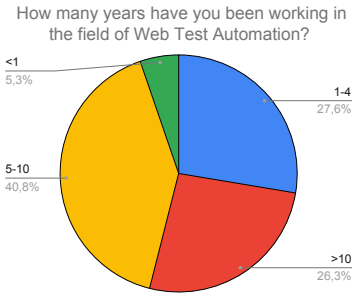

Fig. 1. Characteristics of the Survey Participants: Years of Experience

just a few participants have less than one year of experience (green). We are satisfied with these results because we were able to intercept many highly experienced professionals.

The subsequent pie chart, reported in Fig. 2, clearly confirms that we have been able to select and invite participants that perceive their knowledge of Selenium WebDriver as relevant. Indeed, more than 47% of the participants rated their Selenium experience as "Advanced" (red), while about 45% as "Intermediate" (blue). The few remaining participants (about 8%) rate their knowledge as "Basic" (yellow).

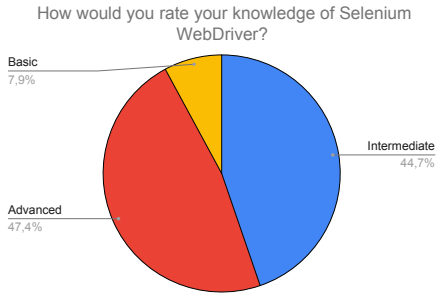

Fig. 2. Characteristics of the Survey Participants: Selenium Knowledge

The last pie chart reported in Fig. 3 provides an overview of the size of the companies where the participants work.

Most of the responses were obtained by practitioners working in Large companies (i.e., 250+ employees), totaling about 55% of the answers (blue). Then, the second larger group is the one of the participants working in Medium-sized companies (i.e., 50-249 employees), corresponding to about 28% of the answers (red). Finally, only a small fraction of the participants

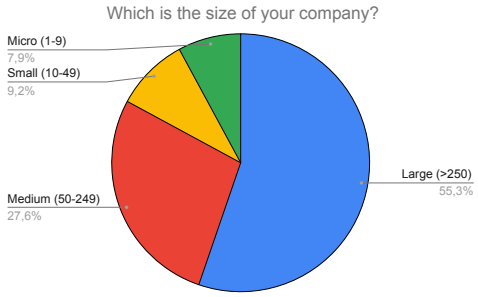

Fig. 3. Characteristics of the Survey Participants: Company Size

work in Small (i.e., 10-49 employees) and Micro companies (i.e., 1-9 employees), respectively, about 9% (yellow) and 8% (green) of all the participants.

Finally, we analyzed the geographic distribution of the participants. The respondents to our survey are from 28 different nationalities including United States, India, Spain, United Kingdom, Italy, Germany, Finland, France, Belgium, Brazil and many other.

B. Results for RQ1 - What are the most relevant challenges?

Fig. 4 provides the data to answer **RQ1**. In particular, it reports the 13 challenges ordered by perceived importance. In the second column, the average value (over 78 responses) of the perceived importance is reported for each challenge, while the third column reports the median values.

Looking at the figure, we can observe a full-point difference between the challenge considered more important, i.e., the management of Asynchronous content by the web test scripts, and the one considered less critical, the Selenium Documentation. Such a difference is not huge but still significant on a five-point scale. The average scores assigned to the various challenges are in the interval [2.38-3.41], where the central value of the scale is 3. So we can say that the various challenges do not obtain a strongly polarized evaluation shared among all the participants of our survey (how could it be, for example, a 4.5 or 1.5 point score).

We speculate that the obtained results could be motivated by the fact that the relevance of the various challenges may vary depending on the context where each participant works and the specific characteristics of the web applications she usually tests. For example, a participant who in her daily practice tests an application where asynchronous content is scarce or even absent might find the Asynchronous challenge, and probably in part also the Flakiness challenge, irrelevant in her experience. Similarly, a participant who is involved in the long-term testing of an industrial application where the identifiers of the web elements are well defined and consistently maintained across releases (e.g., a web application having meaningful and stable IDs tag values in the HTML code), could

	Average	Median
Asynchronous	3,41	4
Brittleness	3,31	3
Flakiness	3,27	4
Assertability	3,19	3
Scalability	3,05	3
Slowness	2,92	3
Failure analysis	2,92	3
Maintainability	2,79	3
Time-consuming	2,77	3
Infrastructure	2,69	2
Cross-browser	2,60	2
Support	2,44	2
Documentation	2,38	2

Fig. 4. Challenges ordered by perceived importance: Average and Median values from the five-points Liker scale

perceive the challenges relating to Brittleness and (partially) Maintenance to be insignificant in her working experience.

In conclusion, the three most relevant challenges are the management of Asynchronous content, the Brittleness and the Flakiness of the test scripts. On the contrary, the participants appear to be quite satisfied with the Support and the Documentation of the Selenium ecosystem (and so they do not perceive them as challenging problems). In general, the results provide a clear ranking among the 13 challenges.

The final open comment in the survey suggests the correct selection of the 13 challenges as no participants added new challenges to our initial set.

C. Results for RQ2 - How is each challenge faced?

Given the results described to answer **RQ1**, it becomes interesting to analyze the distribution of the evaluations provided by the participants to each challenge. Moreover, we summarize the respondents' comments concerning their experience facing each specific challenge. Let's analyze them in the same order as given in Fig. 4.

The distributions of the answers to the first four challenges are reported in Fig. 5. Starting from the top-left, we have Asynchronous, Brittleness, Flakiness, and Assertability. Looking at the answers distributions, we can observe that the ranking scores derive from different assignments on the five-point Likert scale. Indeed, in the case of Brittleness and Assertability, we observe a maximum peak for the "Not Sure" choice and, respectively, very low and quite low values for the negative choices. So we can say that a large number of participants (about 30%) do not have a strong opinion about these two challenges (the ones voted "Not sure") or simply do not have a direct experience with them. On the contrary, when such challenges are experienced, the majority of the participants agree with the fact that they must be considered relevant problems (in both cases, the scores obtained by "Fully agree" is more than twice the ones of "Fully disagree"). The distributions of Asynchronous and Flakiness have a different shape instead. They both show the maximum peak in the positive votes area. Thus, in this case, many participants experienced them and strongly agreed on the relevance of these two challenges.

Let us now analyze the participants' textual comments on these four challenges.

Concerning the management of ***Asynchronous*** content by the web test scripts, many participants provided additional info concerning their experience in facing this challenge. In general, they state to rely on various kinds of waiting strategies to ensure that the test script interacts with the web application only when the asynchronous content is available. Most of the respondents adopt the various waiting strategies available in Selenium, such as the implicit, explicit, and fluent waits [4] targeting the web element to interact with, and they are satisfied with them. The remaining respondents implement custom waiting strategies that include some kind of validations on the web page asynchronous content before executing the next Selenium action. Also, in our opinion adopting advanced waiting strategies is a crucial point to manage Asynchronous content correctly, as recently experienced in an industrial case study [27].

Several respondents report on different strategies to overcome the ***Brittleness*** of the test scripts challenge, i.e., the fact that they are fragile when a web app evolves. One common solution is to avoid the problem by coordinating the test team with the development team. Thus, it is possible to decide on a localization strategy and insert meaningful anchors in the web elements of interest during web app development (usually through the ID tag value). This way, the localization remains stable across the subsequent web app versions. Similarly, some respondents also propose providing developers with direct feedback on the E2E web tests. This direct feedback to the front-end engineers makes it possible if a test breaks due to the evolution of the app, to make them aware of the problem they caused immediately. Others propose the usage of the Page Object (PO) pattern [28], a quite popular web test design pattern, which aims at improving the maintainability of the test and reducing the duplication of code. A PO is a class that represents the web page elements as a series of objects and encapsulates the web page's features into methods. Adopting the PO pattern allows testers to follow the separation of concerns design principle since the test scenario is decoupled from the implementation. Indeed, all the implementation details are moved into the POs, which represent bridges between web pages and tests, with the latter only containing the test logic. Thus, all the functionalities to interact with a web page are offered in a single place, the PO, and can be easily called and reused within any test script, reducing the maintenance effort when a locator is broken. The use of the PO pattern reduces the coupling between web pages and tests, promoting reusability, readability, and maintainability of the test suites [11]. For this reason, many participants adopted it as a solution to other challenges, as explained in the rest of this section. Other respondents find this challenge very hard to solve, particularly when testing web

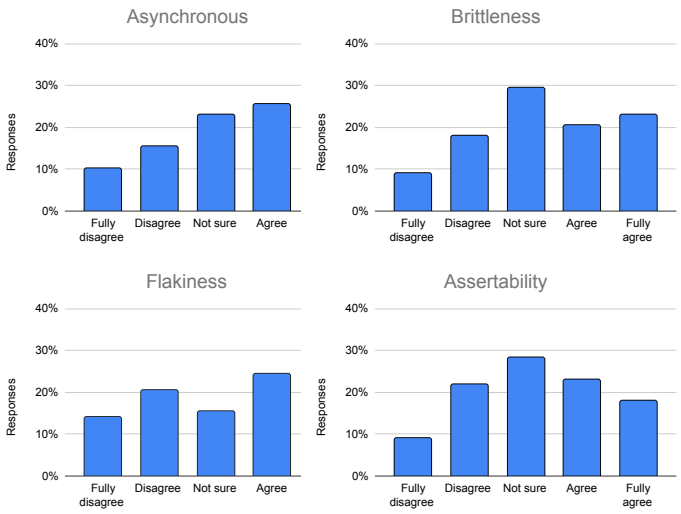

Fig. 5. Details of the answers distribution for the challenges ranked in the top four positions.

apps that do not have meaningful IDs. This way, they report that this often requires adopting complex XPath expressions which prove to be extremely fragile during the evolution of the web application. We fully agree with this statement, and indeed we have already proposed multiple solutions to reduce the Brittleness problem when XPath locators are used, such as ROBULA [29], SIDEREAL [30], and SIMILO [31]. Respondents also propose using multiple locators as a backup strategy: if one fails to retrieve the target web element, the other(s) can be employed. Also, in this case, we agree with such an idea since, in the literature, some proposal to combine multiple locators already exists, including MULTILOCATOR [32], self-healing locators, or similar ideas available in some commercial tools such as Katalon [33], Testim [34], and Screenster [35].

Concerning the ***Flakiness*** of the test scripts, i.e., the fact that they can break randomly without an apparent reason, respondents provided similar solutions as to the Asynchronous challenge. For example, understanding well the various waiting strategies available in Selenium and use them appropriately. We agree that the two challenges are related, and the incorrect management of the asynchronous content can cause flakiness in the test scripts. In general, since it is difficult to be sure to have developed flakiness-free test scripts, some respondents propose to re-execute each test script a predefined number of times before committing it in the master test suite. This should help to increase the reliability of the developed test scripts. Again, we agree with the practitioners' solution: indeed, recently, we proposed a tool, SLEEPREPLACER, that helps to optimize the waiting strategies of the test scripts by improving the test suite code and then validating such changes by re-executing multiple times the test scripts to reduce flakiness [36]. Similarly, in the presence of test scripts characterized by a high flakiness level, some respondents propose to automatically re-execute the failing test scripts to increase the likelihood of detecting only actual failures (indeed, a test that fails due to flakiness is likely to pass in one of the subsequent re-executions). Some respondents suggest using the JavaScriptExecutor to execute actions on web elements that proved to be a source of flakiness in previous runs when using the standard Selenium commands. Respondents find flakiness a complex challenge, especially when working with a cloud service provider In that case, potentially, every test creates even hundreds of HTTP requests from the WebDriver client to the server. In such a scenario, some of these requests probably fail. One solution reported is to keep test cases short and atomic because the longer a test case runs, the higher the risk of flakiness.

Concerning ***Assertability***, i.e., the possibility of implementing assertions in Selenium beyond the classical DOM-based ones, the majority of the respondents suggest integrating Selenium with other tools such as Percy [37], a visual testing tool able to handles different tasks such as capturing and rendering screenshots, as well as detecting and notifying visual changes in the web pages. A similar approach is Sikuli [38], [39], a tool that uses image recognition techniques to identify web elements (and any other element on the screen) and can be used together with Selenium WebDriver [8]. Some respondents find this a limitation of Selenium since some other tools natively support various types of assertions beyond the classical DOM-based ones (e.g., Applitools Eyes [40], Nightwatch VRT [41], or the Jest Framework [42]).

The distributions of the answers to the challenges ranked from position five to position eight are reported in Fig. 6. Also, in this case, we can observe a case (Scalability) showing a maximum peak in the positive votes. The other three challenges instead have the maximum peak on the "Disagree" choice. The votes assigned to the challenge Failure Analysis are quite balanced around the average value.

Let us analyze the textual comments on these four challenges provided by the participants.

Concerning ***Scalability***, i.e., the execution of complex Selenium test suites, for example, adopting parallelism, respondents state that it is essential to start the development of the test suite bearing in mind the need for parallel execution. Indeed, in their experiences, it is far more difficult to refactor an existing sequential complex test suite to support parallel test script execution strategies, mainly due to the dependencies among test scripts. We fully agree with the respondent. Removing the dependencies among test scripts is a must for parallelizing an existing test suite and is, in general, a complex task since dependencies can even be hard to find. To this end, a tool to discover them, named TEDD, has been recently proposed [43]. Some respondents report the problem of executing the test scripts in parallel on the same web app instance since state interferences are likely to happen. In our experience, it is better to clone the instance of the web app (for example, using Docker) to make the various parallel executions autonomous and avoid any interference (see, for example, the tool STILE [44]). This is also suggested by other participants who stated to use containers. Another tool that eases Selenium scalability is Selenium-Jupiter [14], a JUnit 5 extension that provides seamless integration with Selenium WebDriver and Docker. Selenium-Jupiter can be used to implement E2E tests by using

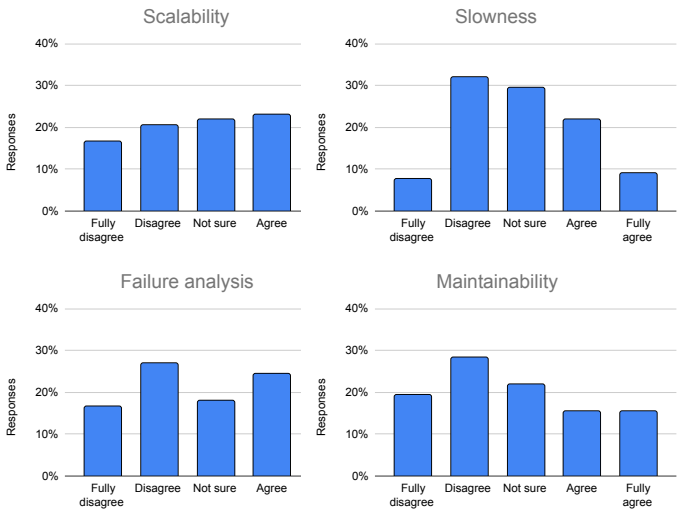

Fig. 6. Details of the answers distribution for the challenges ranked from position 5 to position 8.

multiple browsers in Docker containers. Many respondents report that they have adopted Selenium Grid [45] for parallel execution. Some participants recognize the scalability problem and suggest a drastic solution: reducing the number of E2E test scripts to run. This way, they prefer to take the risk of lowering the test coverage rather than spending a relevant effort for maintaining and executing large test suites.

Concerning ***Slowness*** of the Selenium test scripts, i.e., they take a long time to complete, some respondents stated that this problem is due to the fact that browsers themselves are the bottleneck. Also, the usage of non-optimized waiting strategies (e.g., thread sleep commands) is seen as a cause for the perceived slowness of the Selenium test scripts. We agree on that since using, for instance, fixed-time waiting strategies (as commonly done by inexperienced web testers) can lead to a relevant increment in the execution time [36]. Another cause of the perceived slowness is that in many cases each test script execution requires a new browser instance to be started. This way, respondents report that they try to reuse the same browser instance as much as possible across different test script executions to reduce the set-up time. This can be a relevant portion of the total execution time when test scripts are relatively short, such as testing simple functionalities. Some respondents suggest executing test scripts in headless mode. In this case, the browser does not display the UI, so the potentially heavy rendering process of the web pages is not executed. Another solution proposed by the respondents involves the minimization of the steps performed by Selenium, trying to do equivalent actions through APIs: for example, the portions of the test scripts navigation flow that are common to many of them (and so not part of the testing process for such a particular test script) such as login, logout, navigation, setting up, cleaning up, etc. can be replaced by APIs calls (when possible).

Concerning ***Failure analysis***, i.e., the troubleshooting that helps to pinpoint the underlying cause of a failed test scripts, the respondents suggest using logging libraries such as Log4j [46], ReportNG [47], and ReportPortal [48] that help in collecting detailed information concerning the failure. Many respondents find the common stack-trace messages not very informative and sometimes confusing. Browser interaction is perceived to lead to obscure error messages when something goes wrong. Respondents find the problem exacerbated by running test scripts on remote Grid/Cloud: they state that this, in their experience, usually requires a lot of additional investigations and efforts. The respondents also suggest tweaking the error messages so that if a test script breaks, it is easier to understand what happened and the cause of failure. Another proposed solution includes taking screenshots or HTML dumps when a problem occurs to inspect the page visually and better understand the cause of the problem. Some respondents suggest taking even video recordings of the failing test scripts so that replicating the problem and the subsequent failure analysis becomes more straightforward. Recently the tool BrowserWatcher [49] has been proposed to monitor web browsers such as Google Chrome, Mozilla Firefox, or Microsoft Edge and perform advanced log gathering and analysis that can also help in understanding the reasons for a failure.

Concerning ***Maintainability***, i.e., the set of activities required to keep the test scripts working after they are deployed, many respondents state that the maintenance effort absolutely depends on the design patterns and architecture of test scripts/suite, and so highlight the importance of developing reusable atomic components. Many of the respondents suggest using the Page Object pattern [28]. We agree with such suggestions since there is strong empirical evidence of the benefits deriving from adopting the Page Object pattern [8], [50]. For this reason, tools for the automatic generation of page objects have also been proposed, e.g., APOGEN [51]. Another design pattern that aims to improve reusability and maintainability is the Screenplay pattern. Screenplay is a user-centered pattern that uses actors, tasks, and goals to define tests in business terms rather than interactions [52]. Other participants also pinpoint the problems due to managing the continuously updated browsers' drivers, Selenium versions, and ever-changing browser versions. To mitigate this problem recently, the tool WebDriverManager has been proposed [13] and evaluated [53].

The distribution of the answers to the challenges ranked from position nine to position eleven is reported in Fig. 7. These challenges obtained lower votes and so, as expected, show a different shape in their distributions. In all cases, the maximum peak is on the negative side of the five-point Likert scale. Participants perceive Time-consuming and Cross-browser as less relevant challenges in their experience, while Infrastructure has a slightly contrasted evaluation with relatively high levels on both the extremes of the scale.

Let us analyze the textual comments provided by the participants on these three challenges.

Concerning ***Time-consuming*** challenge, i.e., that the development of Selenium test scripts is a complex and time-consuming task, the respondents suggest a solution similar to the one proposed for the Maintainability challenge. Thus, relying on helper methods that can be used for repeated actions helps

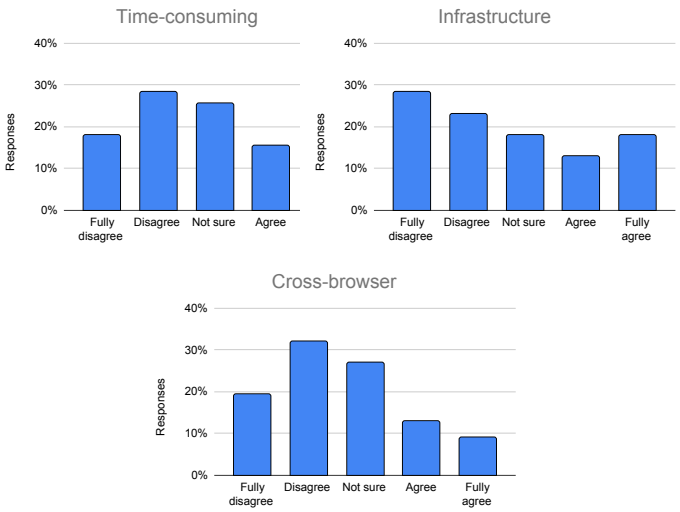

Fig. 7. Details of the answers distribution for the challenges ranked from position 9 to position 11.

overcome this challenge. Also, adopting the Page Object pattern can reduce the development effort in case of test suites targeting a strong coverage of the web application under test [54] (and so having reuse of the methods in the Page Objects). They also highlight the efforts required to manually define robust XPath locators (to this end, see the possible solutions for the automatic generation of robust locators already presented for the Brittleness challenge). Moreover, the respondents mentioned the effort required to set up the test suite (for instance, using WebDriverManager, like in the Maintainability challenge).

Concerning ***Infrastructure*** challenge, many respondents state to use also, in this case, WebDriverManager to mitigate the problems concerning the management of the infrastructure configurations. Other respondents suggest relying on container-based solutions and Docker to create ready-to-use and easily reusable configurations. Another possible solution is to outsource the infrastructure management and rely on the many available cloud vendors [55]–[57] that provide ready-to-use testing infrastructures on the cloud.

Concerning ***Cross-browser*** challenge, i.e., reusing the same test logic for verifying web applications using different browsers, respondents state that today their practice is to rely on cloud-based solutions such as SauceLabs [55], LambdaTest [56], and BrowserStack [57]. According to the respondents, in the last years, cloud solutions gradually replaced the need to manage multiple in-house versions of the test suites tailored for different browsers. They also found that the end of the IE (Internet Explorer) browser support has reduced the need for separate test script versions to test the same functionality. This is also due to the fact that, in general, W3C standardization helped to uniform browsers' behavior (and so few differences are required in the test scripts logic).

Finally, the distribution of the answers to the challenge ranked in the last two positions is reported in Fig. 8. In both cases, the distribution shapes are clearly skewed toward the negative values. So they do not represent a relevant problem for most respondents.

Let us analyze the textual comments provided by the participants on these two final challenges.

Concerning ***Support*** challenge, i.e., the difficulty of finding help in Selenium user communities such as user groups and Slack, the respondents state that the Selenium community is supporting, kind, and welcoming. The way, Selenium contributors on Slack and Google groups are perceived to do a great job of helping Selenium users. Respondents also appreciated the organization of the Selenium Conference [58], which helps deepen the knowledge of various topics related to the Selenium ecosystem.

Concerning ***Documentation*** challenge, i.e., the quality of the Selenium documentation, respondents find it quite comprehensive and clear. A few respondents suggest providing documentation in additional forms, such as video tutorials or webinars, since this could help better understand complex functionalities.

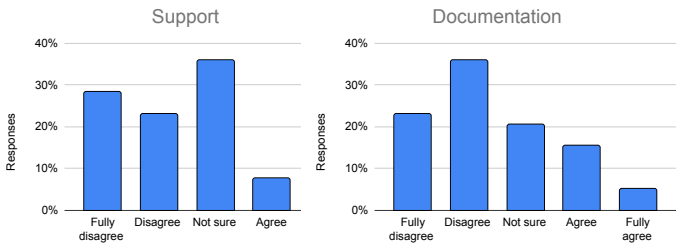

Fig. 8. Details of the answers distribution for the challenges ranked in position 12 and 13.

D. Results for RQ3 - Company size and perceived relevance

To answer **RQ3** we computed the ranking of the 13 challenges by perceived importance and partitioned for company size where the respondents work. Fig. 9 shows the results. We maintained the ranking ordering provided in Fig. 4 to answer **RQ1**: thus, each inversion in the green-white-red color scale indicates an inversion in the ranking w.r.t. the overall ranking computed for **RQ1**. For example, see Infrastructure in Micro companies or Assertability in Small companies.

We can observe that the overall ranking generally looks similar across different company sizes: the challenges perceived as the most important are still in the first positions when partitioning for company size, and the same hold for the ones in the last positions. However, there are some exceptions.

The column showing the answers of participants working in Large companies is the one that looks more similar to the overall results: the average value over all the 13 challenges is 2.96, while the overall data used to answer **RQ1** scored a very close 2.90. In this case, we can see some exchanges in the ranking position, but they are limited to relatively nearby positions (e.g., Failure Analysis).

In the case of the Medium companies, we can see a similar ranking. However, the values are generally lower (in this case, the average is 2.64), indicating that the participants assigned lower evaluations to all the challenges w.r.t. the overall participants. In this case, we can observe some significant changes in the ranking. For instance, Maintainability is the challenge that scored the lowest value.

The participants working in Small companies are slightly more positive, scoring an overall average value of 2.81. In

	Micro (1-9)	Small (10-49)	Medium (50-249)	Large (250+)
Asynchronous	3,50	3,43	2,86	3,60
Brittleness	3,67	3,29	2,86	3,40
Flakiness	3,00	3,29	2,71	3,50
Assertability	3,33	2,57	2,90	3,33
Scalability	3,00	2,86	2,71	3,17
Slowness	2,67	3,14	2,67	2,95
Failure analysis	3,50	3,14	2,90	2,71
Maintainability	3,33	2,57	2,29	2,90
Time-consuming	2,33	2,29	2,71	2,83
Infrastructure	3,67	2,43	2,43	2,62
Cross-browser	2,50	2,86	2,52	2,50
Support	2,50	2,43	2,38	2,52
Documentation	2,50	2,29	2,38	2,45

Fig. 9. Challenges ordered by perceived importance and partitioned by respondent's company size

this case, we observe multiple changes in the ranking as highlighted by the alternate red and green values (see, for instance, Assertability and Time-consuming).

Finally, the participants working in Micro companies are the most positive (see the greener colors), scoring an average value of 3.04. We can observe multiple and relevant changes to the ranking order. For instance, Infrastructure becomes the most relevant change together with Brittleness, while Slowness is one of the challenges that obtained the lower score.

Providing a detailed interpretation of the obtained results is complex since they can be due to many, possibly contrasting, factors. For example, the respondents from Micro companies could provide certain evaluations as they work (1) to ensure the quality of a small web app created in their company, or on the contrary, (2) as outsourced testers for bigger companies and therefore they perceive the challenges of Medium/Large companies. To keep the questionnaire short, we avoided asking too detailed questions about the participant's work situation.

To conclude, the interesting aspect is that by partitioning the results provided by disjoint groups of participants, very similar challenges' rankings are obtained: this is very positive in order to assess the generalizability of the results.

E. Threats to Validity

In our opinion the main threats to validity of this study are the following:

– *Sampling techniques*. Making the survey available online can result in having answers possibly by unqualified participants. To mitigate this threat, we distributed the survey to selected groups with a specific professional interest in Selenium.
– *Challenges selection*. Some important challenges may be missing. To reduce these threats, as reported in Section II, we systematically searched for the most relevant challenges reported by practitioners on the web. We refined the list according to our experience in the E2E field.
– *Terminology*. Some terms can have different interpretations among practitioners. Therefore, each term used in the questionnaire was carefully evaluated thanks to the judgments/comments provided in the pilot study.
– *Sample size and not uniform geographic distribution of the data points*. Our sample size is in line with the previously performed software testing surveys [4], [59]. Clearly, having more data points is desirable but not easy to achieve given the specific scope of the survey and the selection due to the highly desired knowledge level of the participants. Also, further geographically distributed data points are highly desirable to generalize our findings, but we have already covered many countries.
– *Self-exclusion.* We cannot exclude that some participants could have avoided answering because they considered participating in a survey useless. Self-exclusion is a well-known problem in Internet surveys, in particular when advertised online as we did.

V. Related Work

Concerning the related works focusing on analyzing and proposing technical solutions to the various challenges, we have already mentioned many of them in the results section. Here, for space reasons, we limit our analysis to a few similar works adopting the survey/content analysis empirical strategy in the context of testing.

Garcia et al. [4] present a descriptive survey, with a total of 72 participants from 24 countries, aimed to understand how the community uses Selenium and its ecosystem. By ecosystem, they mean the various components, tools, and other interrelated elements sharing the same technological background. Unlike our work, they do not focus on the challenges and the possible ways to face them. The number of participants is comparable with ours.

Kanij et al. [59] describe a survey, with a total of 104 participants, on the importance of various factors that influence effective testing, including testing-specific training, experience, skills, and human qualities like dedication and general intelligence. The survey responses strongly suggest that while testing tools and training are important, human factors were similarly considered highly important. This survey is very different from ours since the scope is far broader. Also, in this case, the number of participants is comparable.

Cerioli et al. [3] applied content analysis to about five million job advertisements taken from a popular Web job search engine. Among the various findings, it emerges that the most valuable testing tools, frameworks, and libraries are Selenium, JUnit, and Cucumber for both Testers and Coders. This strengthens the importance of focusing our analysis on Selenium in the context of E2E web testing.

VI. Conclusion and Future Work

In this paper, we presented the results of a personal opinion survey with 78 highly skilled participants from the industry, analyzing the most relevant challenges in E2E testing focusing on Selenium. The results allow understanding: (a) which are the main challenges developers face and (b) how practitioners solve the problems (e.g., flakiness and fragile tests) that arise every day using Selenium. In our opinion, this second point can be beneficial for the reader, that can find in this paper a summary of the most important solutions (as suggested by the 78 participants) adopted in the industry to overcome the various challenges, plus some suggestions deriving from our academic and industrial experience.

In future work, we plan to extend this personal opinion survey in several directions, including more participants and extending the survey to other countries. Moreover, we plan to deepen the most relevant solutions to the various challenges through interviews with industry practitioners and having specific skills (such as experts in test parallelization, test development, or cross-browser testing).

Acknowledgement. This work was partially supported in part by the Ministerio de Ciencia e Innovación-Agencia Estatal de Investigación (10.13039/501100011033) through the H2O Learn project under Grant PID2020-112584RB-C31, and in part by the Madrid Regional Government through the e-Madrid-CM Project under Grant S2018/TCS-4307.

REFERENCES

[1] "Selenium WebDriver," https://www.selenium.dev/documentation/webdriver/.

[2] "Selenium," https://www.selenium.dev/.

[3] M. Cerioli, M. Leotta, and F. Ricca, "What 5 million job advertisements tell us about testing: a preliminary empirical investigation," in *Proceedings of 35th ACM/SIGAPP Symposium on Applied Computing (SAC 2020)*. ACM, 2020, pp. 1586–1594. [Online]. Available: https://doi.org/10.1145/3341105.3373961

[4] B. García, M. Gallego, F. Gortázar, and M. Munoz-Organero, "A survey of the selenium ecosystem," *Electronics*, vol. 9, no. 7, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/7/1067

[5] M. Barboni, A. Bertolino, and G. D. Angelis, "What we talk about when we talk about software test flakiness," in *International Conference on the Quality of Information and Communications Technology*. Springer, 2021, pp. 29–39.

[6] F. Ricca and P. Tonella, "Analysis and testing of web applications," in *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*, 2001, pp. 25–34.

[7] B. Garcia, *Hands-On Selenium WebDriver with Java: A Deep Dive Into the Development of End-To-End Tests*. O'Reilly Media, Incorporated, 2022.

[8] M. Leotta, D. Clerissi, F. Ricca, and P. Tonella, "Approaches and tools for automated end-to-end web testing," *Advances in Computers*, vol. 101, pp. 193–237, 2016. [Online]. Available: https://doi.org/10.1016/bs.adcom.2015.11.007

[9] W. Lam, K. Muşlu, H. Sajnani, and S. Thummalapenta, "A study on the lifecycle of flaky tests," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1471–1482. [Online]. Available: https://doi.org/10.1145/3377811.3381749

[10] Q. Luo, F. Hariri, L. Eloussi, and D. Marinov, "An Empirical Analysis of Flaky Tests," in *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE'14. ACM, 2014, pp. 643–653.

[11] M. Leotta, D. Clerissi, F. Ricca, and P. Tonella, "Capture-Replay vs. Programmable Web Testing: An Empirical Assessment during Test Case Evolution," in *Proceedings of 20th Working Conference on Reverse Engineering (WCRE 2013)*. IEEE, 2013, pp. 272–281. [Online]. Available: https://doi.org/10.1109/WCRE.2013.6671302

[12] A. Mesbah and M. R. Prasad, "Automated cross-browser compatibility testing," in *Proceedings of the 33rd International Conference on Software Engineering*, 2011, pp. 561–570.

[13] B. García, M. Munoz-Organero, C. Alario-Hoyos, and C. D. Kloos, "Automated driver management for selenium webdriver," *Empirical Softw. Engg.*, vol. 26, no. 5, sep 2021. [Online]. Available: https://doi.org/10.1007/s10664-021-09975-3

[14] B. García, C. Delgado Kloos, C. Alario-Hoyos, and M. Munoz-Organero, "Selenium-jupiter: A junit 5 extension for selenium webdriver," *Journal of Systems and Software*, vol. 189, p. 111298, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121222000516

[15] B. Garcia, F. Gortazar, L. Lopez-Fernandez, M. Gallego, and M. Paris, "Webrtc testing: challenges and practical solutions," *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 36–42, 2017.

[16] "Selenium Documentation," https://www.selenium.dev/documentation/.

[17] "Selenium Support," https://www.selenium.dev/support/.

[18] M. Q. Patton, *Qualitative Evaluation and Research Methods*. SAGE Publications, inc, 1990.

[19] B. Kitchenham and S. L. Pfleeger, "Principles of Survey Research (part 5): Populations and Samples," *ACM SigSoft Software Engineering Notes*, vol. 27, no. 5, pp. 17–20, 2002.

[20] S. Baltes and P. Ralph, "Sampling in software engineering research: A critical review and guidelines," *Empirical Softw. Engg.*, vol. 27, no. 4, jul 2022. [Online]. Available: https://doi.org/10.1007/s10664-021-10072-8

[21] "Selenium Conference 2022 - July 29-30, 2022," https://2022.seleniumconf.in/.

[22] T. C. Lethbridge, "A Survey of the Relevance of Computer Science and Software Engineering Education," in *Proceedings of the 11th Conference on Software Engineering Education and Training*, ser. CSEET 1998. IEEE, 1998, pp. 56–66. [Online]. Available: http://dl.acm.org/citation.cfm?id=522339.794252

[23] A. Jedlitschka, M. Ciolkowski, C. Denger, B. Freimut, and A. Schlichting, "Relevant information sources for successful technology transfer: A survey using inspections as an example," in *Proceedings of 1st International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM 2007. IEEE Computer Society, 2007, pp. 31–40. [Online]. Available: http://dx.doi.org/10.1109/ESEM.2007.73

[24] "Google Forms," https://www.google.com/forms/.

[25] D. S. Walonick, *Survival Statistics*. StatPac, Inc., 1997.

[26] B. A. Kitchenham and S. L. Pfleeger, "Personal opinion surveys," in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. I. K. Sjoberg, Eds. Springer London, 2008, pp. 63–92.

[27] D. Olianas, M. Leotta, F. Ricca, and L. Villa, "Reducing flakiness in End-to-End test suites: An experience report," in *Proceedings of 14th International Conference on the Quality of Information and Communications Technology (QUATIC 2021)*, ser. CCIS, A. C. R. Paiva, A. R. Cavalli, P. Ventura Martins, and R. Pérez-Castillo, Eds. Springer, 2021, vol. 1439, pp. 3–17. [Online]. Available: https://doi.org/10.1007/978-3-030-85347-1_1

[28] M. Fowler, "PageObject," http://martinfowler.com/bliki/PageObject.html.

[29] M. Leotta, A. Stocco, F. Ricca, and P. Tonella, "ROBULA+: An algorithm for generating robust XPath locators for web testing," *Journal of Software: Evolution and Process (JSEP)*, vol. 28, no. 3, pp. 177–204, 2016. [Online]. Available: https://doi.org/10.1002/smr.1771

[30] M. Leotta, F. Ricca, and P. Tonella, "SIDEREAL: Statistical adaptive generation of robust locators for Web testing," *Journal of Software: Testing, Verification and Reliability (STVR)*, vol. 31, 2021. [Online]. Available: https://doi.org/10.1002/stvr.1767

[31] M. Nass, E. Alégroth, R. Feldt, M. Leotta, and F. Ricca, "Similarity-based web element localization for robust test automation," *ACM Transactions on Software Engineering and Methodology (TOSEM)*. [Online]. Available: https://doi.org/10.1145/3571855

[32] M. Leotta, A. Stocco, F. Ricca, and P. Tonella, "Using multi-locators to increase the robustness of web test cases," in *Proceedings of 8th IEEE International Conference on Software Testing, Verification and Validation (ICST 2015)*. IEEE, 2015, pp. 1–10. [Online]. Available: https://doi.org/10.1109/ICST.2015.7102611

[33] "Katalon," https://katalon.com/.

[34] "Testim," https://www.testim.io/test-automation-tool/.

[35] "Screenster," https://screenster.io/.

[36] D. Olianas, M. Leotta, and F. Ricca, "SleepReplacer: A novel tool-based approach for replacing thread sleeps in selenium webdriver test code." *Software Quality Journal (SQJ)*, 2022. [Online]. Available: https://doi.org/10.1007/s11219-022-09596-z

[37] "Percy," https://percy.io/.

[38] T.-H. Chang, T. Yeh, and R. C. Miller, "Gui testing using computer vision," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 1535–1544. [Online]. Available: https://doi.org/10.1145/1753326.1753555

[39] T. Yeh, T.-H. Chang, and R. C. Miller, "Sikuli: Using gui screenshots for search and automation," in *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 183–192. [Online]. Available: https://doi.org/10.1145/1622176.1622213

[40] "Applitools Eyes," https://applitools.com/platform/eyes/.

[41] "Nightwatch VRT," https://github.com/Crunch-io/nightwatch-vrt.

[42] "Jest Framework," https://jestjs.io/.

[43] M. Biagiola, A. Stocco, A. Mesbah, F. Ricca, and P. Tonella, "Web test dependency detection," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 154–164. [Online]. Available: https://doi.org/10.1145/3338906.3338948

[44] D. Olianas, M. Leotta, F. Ricca, M. Biagiola, and P. Tonella, "STILE: a tool for parallel execution of E2E webtest scripts," in *Proceedings of 14th IEEE International Conference on Software Testing, Verification and Validation (ICST 2021)*. IEEE, 2021, pp. 460–465. [Online]. Available: https://doi.org/10.1109/ICST49551.2021.00060

[45] "Selenium Grid," https://www.selenium.dev/documentation/grid/.

[46] "Log4j," https://logging.apache.org/log4j.

[47] "ReportNG," http://reportng.uncommons.org.

[48] "ReportPortal," https://reportportal.io/.

[49] "BrowserWatcher," https://github.com/bonigarcia/browserwatcher.

[50] M. Leotta, D. Clerissi, F. Ricca, and C. Spadaro, "Improving Test Suites Maintainability with the Page Object Pattern: An Industrial Case Study," in *Proceedings of 6th International Conference on Software Testing, Verification and Validation Workshops (ICST 2013 Workshops)*. IEEE, 2013, pp. 108–113. [Online]. Available: https://doi.org/10.1109/ICSTW.2013.19

[51] A. Stocco, M. Leotta, F. Ricca, and P. Tonella, "APOGEN: Automatic page object generator for web testing," *Software Quality Journal (SQJ)*, vol. 25, no. 3, pp. 1007–1039, 2017. [Online]. Available: https://doi.org/10.1007/s11219-016-9331-9

[52] D. Yuniasri, T. Badriyah, and U. Sa'adah, "A comparative analysis of quality page object and screenplay design pattern on web-based automation testing," in *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, 2020, pp. 1–5.

[53] M. Leotta, B. García, and F. Ricca, "An empirical study to quantify the setup and maintenance benefits of adopting WebDriverManager," in *Proceedings of 15th International Conference on the Quality of Information and Communications Technology (QUATIC 2022)*, ser. CCIS, A. Vallecillo, J. Visser, and R. Pérez-Castillo, Eds. Springer, 2022, vol. 1621, pp. 31–45. [Online]. Available: https://doi.org/10.1007/978-3-031-14179-9_3

[54] M. Leotta, M. Biagiola, F. Ricca, M. Ceccato, and P. Tonella, "A family of experiments to assess the impact of page object pattern in web test suite development," in *Proceedings of 13th IEEE International Conference on Software Testing, Verification and Validation (ICST 2020)*. IEEE, 2020, pp. 263–273. [Online]. Available: https://doi.org/10.1109/ICST46399.2020.00035

[55] "SauceLabs," https://saucelabs.com/platform/cross-browser-testing.

[56] "LambdaTest," https://www.lambdatest.com/.

[57] "BrowserStack," https://www.browserstack.com/.

[58] "Selenium Conference," https://twitter.com/seleniumconf.

[59] T. Kanij, R. Merkel, and J. Grundy, "A preliminary survey of factors affecting software testers," in *2014 23rd Australian Software Engineering Conference*, 2014, pp. 180–189.