Using Newsletters to Analyze Curated Software Testing Content

Philip De Munck[1]

Supervisor(s): Andy Zaidman[1]**, Baris Ardic**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Philip De Munck
Final project course: CSE3000 Research Project
Thesis committee: Andy Zaidman, Baris Ardic, Koen Langendoen

Abstract

As software and systems continue to get more complex, software testing is an important field to ensure that software functions properly. Every day information about software testing is being discussed on the internet via blog posts, discussion boards, and more. This information is scattered among many different websites, making it hard to access. To analyze software testing content published on the internet, newsletters curated by members of the field and reflective of industry trends were used. This analysis provides a broad overview of what software testing-related content is being discussed on the internet. Common problems discussed in newsletters include properly maintaining tests, working with and fixing flaky tests, and properly analyzing test results. Javascript and Typescript are the most popular programming languages discussed, while the web is also the most popular platform. When looking at test types, automated tests are frequently discussed, followed by end-to-end tests and unit tests. Common techniques and strategies discussed include API testing, the use of continuous integration, and the use of continuous deployment. Selenium, Cypress, and the Gherkin syntax are the most frequently discussed tools and technologies. Finally, opinionated articles tend to be most common, followed by articles that introduce a technology and articles that explain a concept.

1 Introduction

Software Testing is a critical part of the software development life cycle [1]. The consequences of not testing properly can range from minor irritations to complex failures with significant effects [2]. One example of a major failure could be the Ariane 5 rocket that failed less than a minute after launch due to a software failure [3]. Due to many factors, software bugs tend to become more expensive to fix the later they are discovered [2]. Proper testing earlier in the development process can therefore help to reduce the impact of these bugs.

Due to the importance of software testing in ensuring the proper functioning of software, detailed research must be conducted to understand the current state of the industry. Furthermore, this research and the associated output can be used to further our understanding of the direction that software testing is taking.

Information about software testing is discussed frequently in various formats, such as academic papers, online blogs, physical conferences, and more. On the internet, information is constantly being published, updated, and revised. This begs the question: *What software testing knowledge is currently being discussed on the internet?* By identifying what is being discussed, we can foster a better understanding of current issues, trends, and common tools within the software testing space.

Newsletters were chosen as the subset of the internet to be analyzed since they are usually curated by a member of the field in question. By using resources posted to software testing newsletters, we know that the resource was seen as valuable since it was chosen to appear in an edition of a newsletter. Although some newsletters are more popular and are published more frequently than others, all newsletters selected have been publishing editions for over a year and have a publicly accessible website.

To answer the question set out in this introduction, the following four research questions (RQs) have been defined:

1. What platforms, languages, and test types are discussed in newsletters?
2. What common problems related to software testing are discussed in newsletters?
3. What software testing tools, techniques, and strategies are discussed in newsletters?
4. What types of software testing resources are shared in newsletters?

In Section 2, the related work will be discussed. Section 3 will discuss the methodology for the research. Section 4 will discuss the results of the research. A discussion about the results will take place in Section 5 while the main conclusions will be drawn in Section 6. Section 7 will discuss threats to the validity of this research. Finally, Section 8 will discuss how the research was conducted responsibly and in accordance with common principles.

2 Related Work

Limited research has been done on newsletters, especially those in the software sector. However, general research has been conducted on news aggregators (sites that aggregate/distribute existing content, possibly after curating it), which newsletters are a part of. Furthermore, software testing knowledge present on the internet has been analyzed before in different forms. This section serves to discuss the previous work that discusses software testing knowledge present on the internet.

Similar to what is being analyzed in this thesis, Aniche et al. also analyzed programming-related news aggregators and investigated, among other aspects, the characteristics of content posted to these aggregators [4]. A survey conducted in the aforementioned paper indicated that the interviewees found programming-related news aggregators to provide relevant content. Could it be possible for software testing newsletters (a sort of content aggregator) to also provide valuable insights on the state of the software testing industry? The analysis in this thesis serves to answer this question, among others.

Chowdhury and Landoni analyzed news aggregator services and found that since news is published and distributed through many channels, services that aggregate this content can make it much simpler for a user to find relevant news [5]. Since software testing resources are published in many places on the web, newsletters may be able to serve as the aggregator, providing easier access for software testing practitioners to find relevant content.

Additionally, Florea and Stray gathered and coded information regarding the most in-demand skills for software

testers [6]. Using a similar coding approach to what is used in this paper, Florea and Stray used job advertisements as a basis for their analysis [6]. They concluded that employers are looking for certain skills, including but not limited to, test automation, functional testing, and performance testing [6].

Finally, Pagano and Maalej conducted a study that explored how developers blog [7]. They discovered, among other things, that developers' blogs tend to describe things abstractly and at a higher level. Blog aggregators were also used as a starting point from which they were able to find blog posts, similar to how newsletters are used as the starting point in this research. They also observed that blog posts tend to consist of "short documentations, tutorials, and howtos" [7]. Software testing newsletters contain many types of resources, blog posts included.

3 Methodology

This section describes the methodology used during the various steps of the research process. Each step of the research process is individually described and provides details about how it can be replicated.

3.1 Source Discovery

During the beginning of the research process, various search engines were used to discover which websites contained software testing knowledge. An example of a Google search used would be: "best websites for software testing discussion". The outcome of these searches provided a broad overview of sites containing knowledge and discussion of software testing and related fields.

After discovering the various types of websites where software testing knowledge existed, the focus of newsletters was chosen. Newsletters are similar to regular publications such as newspapers and news websites but a curator is required to select the content that will be used in the newsletter [8]. Newsletters can be general or specific to a certain topic and are delivered to readers at certain intervals (weekly, etc.) [8].

Since newsletters contain a variety of information curated by an author, they were selected as an optimal source of information about software testing. Since this research aims to analyze software testing knowledge discussed on the internet, curated newsletters provide an overview of what people are reading within the community.

3.2 Source Selection

This research focuses on articles published online since they frequently focus on explaining or discussing one particular topic relevant to the field. Other media such as videos and forum posts were in most cases discarded.

After newsletters were chosen as the source of information, the selection of newsletters to analyze began. Since all newsletters are different, certain sections were excluded from the analysis in the research. Information about specific sources has been gathered and is up to date as of May/June of 2023.

To select relevant newsletters, comparison articles [1] were considered and search engines were used to find the most commonly discussed newsletters. Five newsletters were selected for further analysis. All of the newsletters selected were either present on the first page of results or discussed in comparison articles on the first page of results for the Google query: "best software testing newsletters" [2].

LambdaTest is a platform that allows testers to orchestrate and execute their tests online [9]. LambdaTest also curates and publishes the "Coding Jag" newsletter [3] weekly, having so far published 137 issues with the first issue dating back to September 4th, 2020. This newsletter contains the sections: news, performance, automation, tools, others, and events. The "others" section was excluded from the analysis due to it mainly containing video content that was difficult to process. In addition, content in the "events" section was excluded as it focused on advertising conferences/events which did not directly discuss software testing information.

Ministry of Testing is a community of over 10,000 software testers that contains a forum ("The Club") and a learning environment ("The Dojo") [10]. It also publishes the "Your Weekly Testing News" newsletter [4], currently on issue 422 with the first issue being published on June 11th, 2019. This newsletter has many sections that tend to change each week. "Helpful business posts" was the only section included since it contained articles related to software testing. Most other sections in this newsletter were focused on events, forum posts about learning, and job advertisements, among other topics.

Software Testing Weekly is, as the name implies, a software testing-related newsletter [5] published weekly with over 6,000 readers [11]. It has currently published 166 editions and published its first edition on January 10th, 2020. This newsletter also contains various sections that change from week to week, but this analysis only included the sections: news, automation, tools, and books. These sections, unlike others, mostly published articles related to software testing in different forms. Other sections were not included since they focused on video or joke content.

Trending In Testing is a newsletter [6] aimed at fulfilling the needs of software testers around the world [12]. It has so far published 84 editions at irregular intervals with its first issue dating back to July 6th, 2021. This newsletter contains the sections: testing, automation, security, api, tools, software bugs, and miscellaneous. All of these sections were included since they discussed software testing content. The preamble, however, was excluded since it mainly consisted of articles that did not discuss software testing information.

Software Testing Notes is a newsletter [7] read by over 1,000 users consisting of various posts related to software testing [13]. This newsletter is published weekly and has

[1] https://www.lambdatest.com/blog/latest-software-testing-newsletters

[2] https://web.archive.org/web/20230624133434/https://www.google.com/web/20230624133434/https://www.google.com/search?client=firefox-b-d&q=best+software+testing+newsletters

[3] https://www.lambdatest.com/newsletter

[4] https://www.ministryoftesting.com/newsletter

[5] https://softwaretestingweekly.com

[6] https://trendingintesting.com/category/weekly-newsletter

[7] https://softwaretestingnotes.com

so far published 85 editions, having started on March 15th, 2021. Various sections are used in this newsletter that change depending on the edition. This analysis considered the sections: testing, automation, performance, security, and accessibility. These sections were included since they all discussed various aspects of the software testing industry. Other sections were not included since they contained jokes, unrelated bonus content, and resources that did not discuss software testing-related information.

3.3 Data Collection

The data collection process started by collecting the four most recent editions from each of the five newsletters mentioned in Section 3.1. 240 resources were downloaded from 20 total newsletter editions.

During the data collection process, each resource mentioned in an included category of a newsletter edition was downloaded as a PDF using the "Save to PDF" print destination in Firefox. Certain articles were downloaded in a simplified format (stripping extra formatting, including manual removal of HTML/CSS) to make the analysis process using software more efficient.

An important consideration made during the data collection process was what to do with resources that appeared in multiple newsletters. In this case, duplicate resources were left in but they were all tagged identically. Since a resource being present in multiple places implies that it is more popular, this was even more of a reason to include them in the data set [14] with identical tags.

After downloading the PDF, each article was recorded in a database that included the source, edition number, link, publish date, the section it was present in, and whether the resource had previously been used in another newsletter. This database also includes links to all editions used, publish dates of each edition, start dates of each newsletter, and more utility information.

3.4 Data Analysis

To analyze the data present in each of the resources, grounded theory was used. Grounded theory is a methodology using a set of methods to code data and eventually create a theory based on the data analyzed [15]. Open coding, one of the methods part of grounded theory, was initially used to begin coding the data. Open coding is the practice of breaking the data down into small parts which can then be given a code that represents the main idea [15]. Axial coding, which groups many codes into one overarching category, was then conducted [15].

AtlasTI, a software used to ease the process of qualitative data analysis, was the main piece of software used to analyze the resources [16]. Using the grounded theory approach mentioned in the previous paragraph, each resource was analyzed and codes were applied to corresponding topics. Using axial coding, the codes were sorted into seven top-level categories.

A rule used during the analysis was that each tag was only allowed to be applied at most once to each resource. This is due to the fact that writers tend to mention certain words more or less depending on their writing style, which would bias the outcome. Only allowing one of each tag per resource ensures that all mentions of a certain item are tracked fairly and cannot be skewed by an article that mentions it many times.

The **article_type** category represents the type of each resource in the dataset. Based on how the resource presents its information, how the author writes this information, whether different technologies are compared, and more, a specific tag is assigned. This tag category, unlike any of the others, is mandatory. Therefore, exactly one *article_type* tag must be assigned to each resource. A special case with this tag is that if a resource is classed as a tech comparison, technologies are not tagged due to the large number of technologies usually mentioned briefly in such resources.

The **lang** category represents resources that show an example in a specific programming language or discuss a specific programming language. For example, a code sample programmed in JavaScript would have the *js* tag applied. Resources are allowed to have multiple different *lang* tags to represent all languages discussed.

The **platform** category represents the platform that the article discusses. Examples of platforms include web, mobile, iOS, and Android. An article that discusses mobile testing would be tagged with the *mobile* tag while an article that discusses android testing would be tagged with the *android* tag. This means that the sum of the iOS tags and Android tags will not equal the number of mobile tags. Multiple different *platform* tags can be applied to a single resource if that article discusses multiple platforms.

The **prob** category represents common problems and solutions discussed in a resource. For example, articles that discuss test maintenance would have the *maintain_tests* tag assigned. Multiple problems can be discussed in an article, therefore multiple different *prob* tags can be applied to a single resource.

The **tech** category is the largest and contains all tags related to specific technologies. For example, articles that discuss web automation using Selenium would have the *selenium* tag applied. Since multiple technologies can be discussed in a resource, multiple different *tech* tags can be applied to a single resource.

The **technique** category contains all tags related to specific testing techniques and strategies. Articles that discuss test-driven development, for example, would have the *tdd* tag applied. Each resource can have multiple different *technique* tags.

The **test_type** category contains tags representing all types of tests. For example, an article discussing the usage of unit tests would be given the *unit* tag. Multiple different *test_type* tags can be assigned to a single resource.

Figure 1 provides a sample of a resource that has had some tags applied. In this case, multiple technique and technology tags were applied. In addition, a *test_type* tag was applied to show that automated tests were discussed. Not shown in this snapshot is the required tag in the *article_type* category

4 Results

After assigning tags to each of the resources downloaded, the frequencies of each tag could be analyzed. This section will

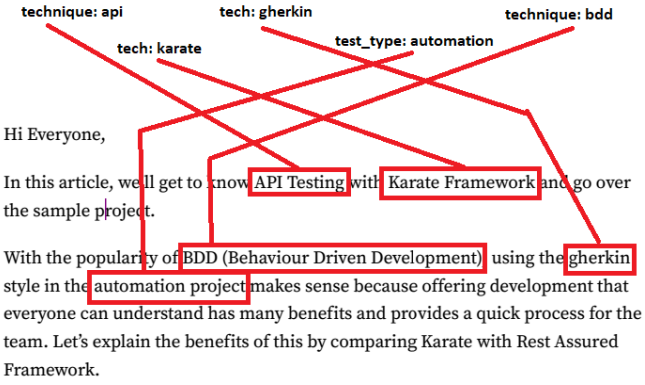

Figure 1: An example of a resource with tags applied [17].

focus on presenting and visualizing the results gathered from the research. Only tags that occurred 6 or more times (double the maximum number of duplicate articles) will be shown and discussed. This condition has been implemented to ensure that the popular tags discussed later come from a variety of sources. The elements shown in the referenced figures will be discussed in the next subsections based on their frequency (highest to lowest). For more detail, a description of all the individual tags, their frequency, and their purpose can be found in the data set [14] accompanying this thesis.

4.1 What platforms, languages, and test types are discussed in newsletters?

Resources shared in software testing newsletters discuss many different types of platforms, languages, and test types. Understanding the common platforms, programming languages, and test types discussed in these resources may allow researchers to better understand the direction the industry is going in. This section will individually discuss each of these factors and how they are discussed in resources.

Figure 2 showcases the various platforms discussed in resources shared in newsletters. Most resources tend to focus on web testing. However, mobile testing was also a minor focus, with Android more than twice as popular as iOS.

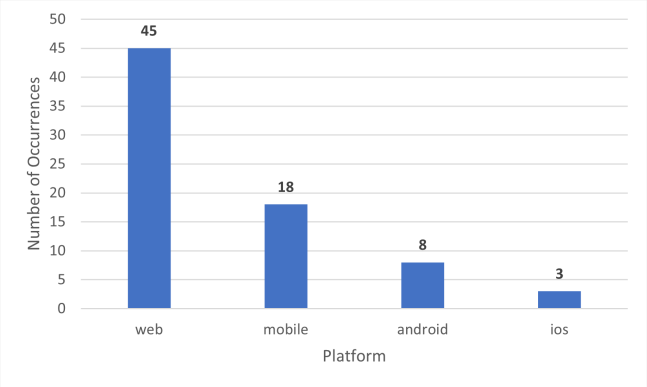

Figure 2: Frequency of different platforms.

Figure 3 shows common programming languages discussed in resources. JavaScript/TypeScript were the most commonly discussed, almost twice as much as the next most popular language, Java. Python was also commonly discussed.

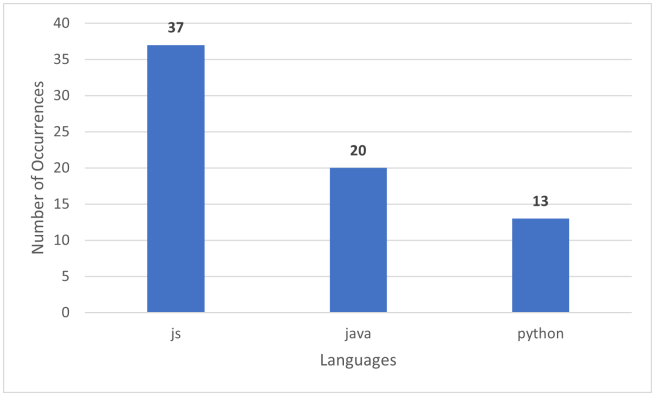

Figure 3: Frequency of different programming languages.

Figure 4 showcases common types of tests discussed in software testing-related content. By far the most commonly discussed test type was automated tests. Following this, end-to-end tests, unit tests, integration tests, and functional tests were discussed. Manual testing, which is only reliant on the tester, was also commonly discussed. In addition, performance testing, regression testing, exploratory testing, acceptance testing, penetration testing, and system testing were discussed.

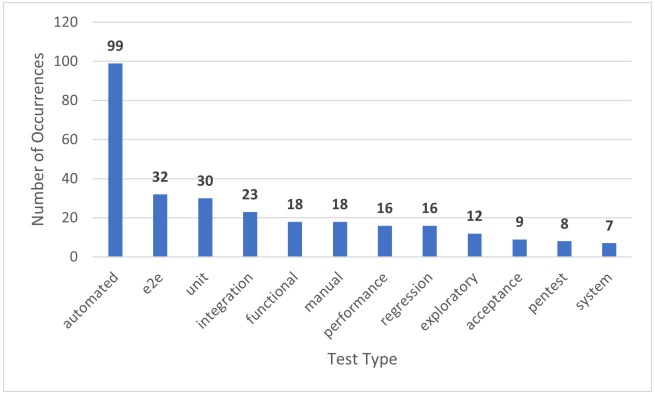

Figure 4: Frequency of different test types.

4.2 What common problems related to software testing are discussed in newsletters?

The graph shown in Figure 5 showcases common problems and their frequency. Understanding common problems encountered in the industry may allow others to create solutions that address these issues. This section will focus on these common problems discussed in newsletter resources.

Properly maintaining tests is the most common issue discussed in resources. As a project expands, tests can become redundant and difficult to maintain. Especially in larger

companies, multiple testing environments and procedures can make maintaining testing suites difficult.

The next most common problem is flaky tests and figuring out how to properly fix them. One common cause of flaky tests can come from using web automation frameworks such as Selenium, where a frequently changing web page may cause false test results. Since flaky tests can have many underlying causes, multiple solutions exist. For example, using a different test framework or adding explicit wait periods for UI tests can help avoid flaky tests. Scientific research was also not discussed in newsletter resources, which may have provided more complex approaches for dealing with flaky tests.

Furthermore, another common problem discussed is how to properly analyze test results. Test results can often be hard to interpret if they only contain information about pass/failure rates. Many resources presented in newsletters discuss methods to make test reports easier to understand for other testers and clients alike [18]. Customizing the framework used for testing can allow for better reports to be generated. In addition, data visualization can create reports that are easier to comprehend.

Another common problem discussed in resources was proper system observability. If a system does not have proper logging or metrics available, it can be more difficult to debug a failing test or analyze failures that should have been tested in the first place. By ensuring that a system is observable, tests can be easily added when failures are detected, and failing tests can more easily be fixed by analyzing logs and metrics.

Next, preventing over-automation is also a commonly discussed issue. For certain applications where the user interface frequently changes, creating an automated UI test may be a waste due to all of the maintenance required when the UI changes [19]. Some tests are more unstable than others or may require human judgment which makes automated testing not applicable [19]. Due to these and other reasons, many resources discussed facing this issue.

Ensuring that an application can perform well is a critical part of testing. Many resources discussed using testing (especially performance/load testing) to ensure that the application could hold up under load.

In addition, properly sourcing data to use during testing is another common issue. Creating a data set similar to production can be quite challenging in a complex system. It is critical to use data that reflects the real world in order to ensure that problems users encounter will be detected during testing.

Finally, choosing the right testing framework is another common problem that testers face. Testing frameworks can often make writing tests much easier with quality-of-life improvements. Testing frameworks can also be focused on a certain domain, making it critical to select the proper one for the use case presented.

4.3 What software testing tools, techniques, and strategies are discussed in newsletters?

The graph shown in Figure 6 shows common techniques, strategies, and their frequency. The graph shown in Figure 7 showcases technologies and their frequency. Understanding common techniques and strategies that are discussed may allow testing practitioners to find new ideas that will help them test more effectively. In addition, understanding common technologies used can help researchers better gauge the state of the industry. This section will discuss the tools, techniques, and strategies shown in the figures.

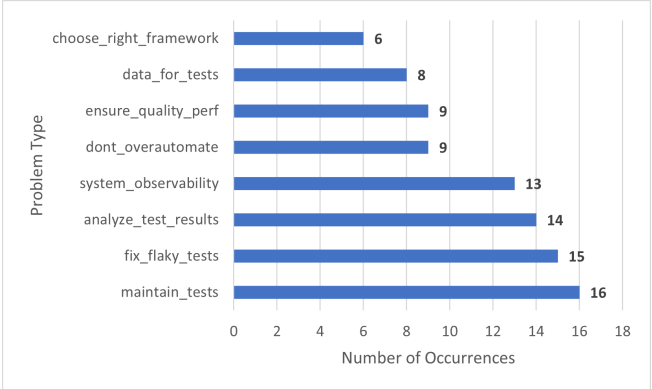

Figure 5: Frequency of different problem types.

Techniques and Strategies

API testing is the most common technique discussed in resources presented in newsletters. Companies are rapidly adopting APIs since software in today's age often utilizes APIs [20]. Because of this, properly testing APIs and ensuring that they function properly is a key technique.

Continuous Integration (CI) is the next most common technique used during the testing process. Integrating the testing process into a pipeline when committing code to a repository is essential to automate and speed up the testing process.

Continuous Delivery (CD), oftentimes grouped with continuous integration, is another common technique used during the testing process. Integrating tests into the continuous delivery pipeline can ensure that released software will be as bug-free as possible.

Behavior Driven Development (BDD) is another common technique discussed in resources. BDD is a collaborative approach where members of an agile team work together to create testing scenarios [21]. This approach is mentioned in many resources as being a great way to define new test scenarios.

Next, the usage of the agile methodology during development and testing is also a common technique. Using this technique during testing can allow testers to incrementally design and create test cases.

The Page Object Model (POM) is also a popular technique used when creating automated web tests using a framework like Selenium. Using the POM pattern entails creating separate classes for each web page, allowing for easier maintenance of tests as the application changes [22].

The usage of low-code tools is another common technique seen in resources. Low-code tools can allow those less familiar with programming to design test cases. This can make testing more accessible for systems with many collaborators of different experience levels.

Testing in parallel is another key technique that can make the testing process much more efficient. When parallel runs

are feasible, they can significantly increase the speed at which tests complete. However, the feasibility of running tests in parallel depends on many external factors such as databases, the test framework used, and more.

Shift-left testing involves moving testing to an earlier part of the development process [23]. By doing so, issues can be caught quicker, saving the team resources and time [23].

Cross Browser Testing (CBT) is a key technique that works by testing a web application in many different browsers since every browser may behave slightly differently when rendering a website [24].

The use of mocks is another common technique that works by creating a fake version of something to use during a test. Using mocks can often help when testing a complex service that is hard to replicate in a test environment.

Test Driven Development (TDD) is a technique used in testing where tests are implemented before the actual software is programmed [25]. Using TDD may cause developers to focus more on tests since they are being coded before the actual software. This may in turn make testing more effective.

Finally, the use of the testing pyramid is another common technique discussed. This technique is used in test automation and consists primarily of three layers: unit, service, and user interface [26]. Creating tests across all three of these layers helps to ensure that an application is properly tested.

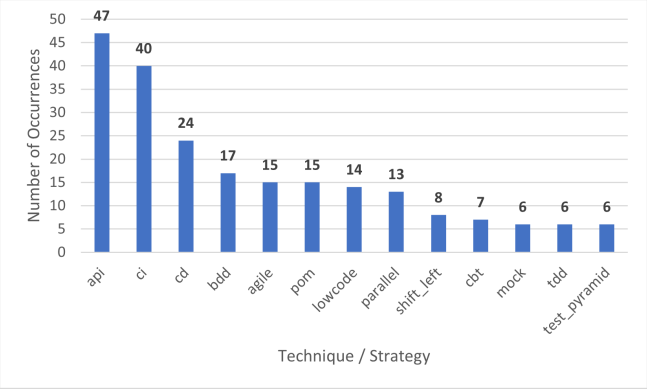

Figure 6: Frequency of different techniques and strategies.

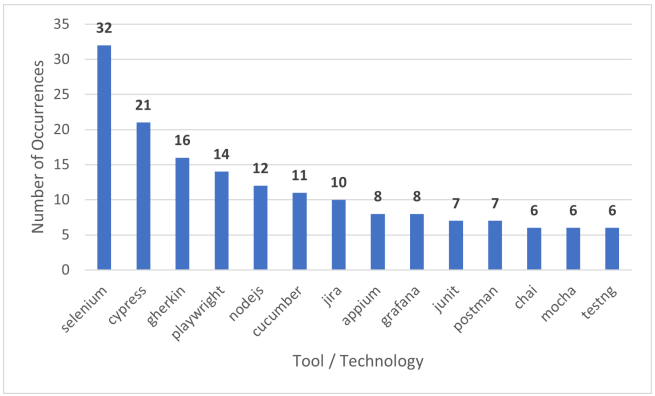

Figure 7: Frequency of different tools and technologies.

Tools and Technologies

Selenium, Cypress, and Playwright are the first, second, and fourth technologies respectively discussed in the resources. Selenium used to be the main contender for web automated testing but Cypress and Playwright (other web automation testing frameworks) have emerged [27]. These frameworks have been used as web applications have evolved due to technological advancements [27].

Gherkin is another common technology discussed in various resources. According to Jones, "A Gherkin (Given-When-Then-And) scenario is non-technical, clear to read, and applied to describe test cases" [21]. The gherkin syntax, also used in combination with BDD, can be used to better create tests with many stakeholders involved [21].

Next, NodeJS is also a commonly discussed technology. NodeJS is frequently used as a backend for web servers. In addition, many testing tools, for example Selenium, can be installed via NodeJS.

Cucumber allows testers to define tests using BDD and human-readable language [28]. Using this technology can also allow non-technical users to understand what behaviors a test might cover [28].

Jira is a tool that is extremely common for managing tests and allows users to track bugs [6]. In addition, Jira was the second most requested tool when looking at job advertisements for software testers [6].

Appium is a platform used to automate the testing of user interfaces [29]. As UIs become more complex, automated tools can help testers more effectively design tests.

Grafana is an analytics platform frequently used to analyze performance. Analytics are key to making system metrics and performance observable.

JUnit is a Java-based unit testing framework.

Postman is a commonly used tool in the development space for testing API endpoints. Testers may be able to use this tool to quickly test endpoints and various other features of an API they are testing.

Chai is a BDD-focused assertion library used in combination with many JS-based testing frameworks [30].

Mocha is a Javascript testing framework.

TestNG is a Java-based testing framework.

4.4 What types of software testing resources are shared in newsletters?

The graph shown in Figure 8 showcases article types and their frequency. Understanding the ways content is shared with newsletter readers may help future researchers when looking further into online software testing discussions. This section will discuss the common article types shown in this figure.

Opinion (*opinion*) articles are the most common in the data set. An opinion article is categorized as a resource that significantly discusses the author's opinion on a certain aspect of software testing. Therefore, these articles are not as objective as can be. An example of this category would be an article explaining why testing is important, according to the author.

Tech introduction (*tech_intro*) articles are the second most common in the data set. These articles serve to introduce a

specific technology to the reader. An example of this category would be an article introducing a reader to Cypress and instructing them on how to set up a basic test.

Concept explanation (*concept_explain*) articles follow tech introduction articles in popularity. These articles focus on explaining a concept to the reader without necessarily using a specific technology. An article explaining the different types of tests (unit vs. end-to-end) without focusing on specific technologies would be considered a concept explanation.

Following concept explanations, tech comparison (*tech_compare*) articles are next in terms of popularity. These articles focus on comparing and contrasting various technologies within a specific sub-field of software testing. An article comparing various automated web testing frameworks (Selenium, Cypress, Playwright, etc) would be considered a tech comparison.

Finally, explainer (*explainer*) articles were not able to fit into any of the previously defined categories. These articles may be vague, not completely related to software testing, or have other distinguishing features that make them hard to categorize.

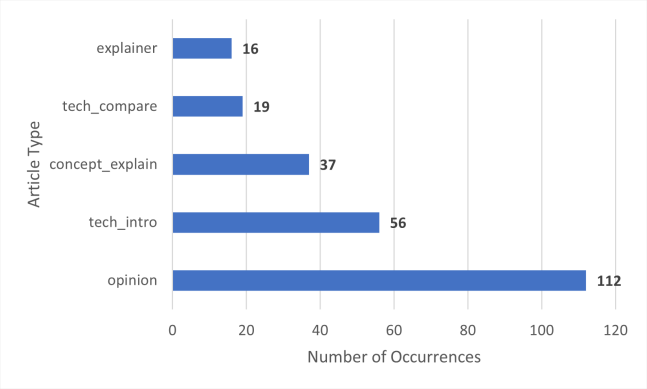

Figure 8: Frequency of different article types.

5 Discussion

Based on the results gathered, the research questions can be re-analyzed and connections can be discovered.

With regard to tools and technologies mentioned in RQ3, frameworks used to automate web testing (Selenium, Cypress, and Playwright) topped the frequency list (1st, 2nd, and 4th place, respectively). Many of the resources in the selected newsletters focus on automation testing, more specifically web automation testing, which is why these specific frameworks were commonly discussed.

In addition to this, the results from RQ1 also show that JavaScript and TypeScript emerged as the most popular languages. Automated and end-to-end tests were also the two most popular test types. This, combined with the fact that the web platform was the most popular, suggests that web automation testing is a common type of testing used in practice today.

Looking further into RQ3, some of the popular techniques are also related to web testing. API testing and POM usage were both frequently mentioned in the selected resources and are used when conducting web tests. However, many of the popular techniques are platform-agnostic. The use of CI/CD, BDD, agile, low code programs, and more can be used in many types of tests, and not just with web testing. This suggests that in most cases, the techniques discussed in the selected resources tend to focus on those that can be applied to many different fields and approaches. However, it may also be the case that techniques in general tend to be more abstract and therefore can apply to more domains.

When looking at RQ2 and problems that were often discussed, all of those discussed were quite abstract and not related to a specific type of test or platform. Properly analyzing test results and fixing flaky tests can be applied to almost any domain in software testing, for example. This may suggest that oftentimes, the problems faced by testing practitioners in a certain context can be extrapolated to a more abstract platform that would also affect them in another context.

In terms of the resource categories discovered in RQ4, opinion resources were the most prevalent, followed by tech introduction, concept explanations, tech comparisons, and explainer resources. The frequency of opinion resources implies that much of the testing content presented through newsletters is based on personal experience. Many resources published in newsletters shared opinions based on using techniques and technologies in practice while testing real applications.

6 Conclusions and Future Work

Curated software testing content presented in newsletters provided a broad basis from which information could be extracted. By utilizing grounded theory, tags and categories were created to organize the information presented in the resources in line with the four research questions proposed at the beginning of this thesis.

After utilizing grounded theory to analyze the data, the frequencies of popular items were noted. In terms of platforms discussed, testing on the web was most common, followed by mobile (with Android being twice as common as iOS). JavaScript/TypeScript were the most common programming languages discussed, followed by Java and Python. In addition, many test types were discussed in newsletter resources, including automated, end-to-end, unit, integration, functional, manual, performance, regression, and more. When referring to resource types, opinion pieces tended to be most frequent, followed by tech introduction, concept explanation, tech comparison, and finally, explainer resources. Various common problems were also found, which included: maintaining tests, fixing flaky tests, properly analyzing test results, ensuring that the system is observable, preventing over-automation, ensuring quality performance, finding proper data to use for tests, and choosing the proper framework for testing. Various common technologies were also found to appear in these resources, such as web automation frameworks (Selenium, Cypress, Playwright, Appium), utility applications (Postman, Grafana, Jira), testing frameworks (JUnit, Chai, Mocha, TestNG), Gherkin, NodeJS, and Cucumber. Finally, various techniques and strategies such

as API testing, the use of continuous integration and deployment, behavior-driven development, agile development, page object model, low-code tools, testing in parallel, and more were commonly discussed in the analyzed resources.

This thesis provides a detailed look into what is commonly discussed in online resources. Analyzing newsletters specifically allowed for the analysis of curated content that was directly delivered to software testers. This research and its associated data set [14] can serve as a jumping-off point for future research into software testing trends discussed among testing practitioners on the internet.

This work focused on analyzing resources within software testing-related newsletters. Future work can expand on this research by using more resources from the newsletters already identified, using a different analysis method, or using a similar analysis technique but finding new newsletters. In addition, this work used grounded theory to manually tag and organize the data. Using more automated or data-driven approaches may help to more effectively analyze a large amount of data. This could enable a more comprehensive analysis of resources present in software testing newsletters.

7 Limitations and Threats to Validity

Although this research was conducted with the utmost care, there are still elements that may threaten the validity of the research. This section will discuss the limitations that may affect the validity of this research.

Regarding external validity, the resources analyzed in this research may not be available in the future. Although the links were saved in the data set [14], authors may delete, alter, or remove the content at a later point in time. This may make it difficult to replicate certain parts of the research process.

The manual inspection and tagging of resources is also an external threat to validity since it is subjective. A researcher attempting to replicate this research may have difficulty obtaining the same data set or similar conclusions depending on how they tag information. This manual inspection and tagging may also be a threat to the internal validity of the research due to the subjectivity.

8 Responsible Research

Following the principles of responsible research is critical when conducting any sort of research. In this thesis, a large amount of data was collected, analyzed, and discussed. To ensure that this has been done properly, this section will discuss measures taken to ensure that research was conducted responsibly.

The concept of replication is critical in ensuring that the research was conducted responsibly. This entails that another researcher should be able to replicate this study and obtain similar results. To ensure that this experiment can be replicated effectively, Section 3 clearly outlines the methodology and all steps taken to collect and analyze the data. However, since the resources are tagged based on personal opinion, exact replication of the results in this thesis may not be possible.

In addition to the concept of replication, publicly accessible data is also critical to allow for further research to be conducted. A data set [14] has been published alongside this thesis which provides detailed information about the tags, resources, and other key portions of the research. This public data set has been shared to allow others to validate the work presented here and build upon it in future research.

Due to the many newsletters and resources used, some data was left out due to it not being applicable. In cases where data was left out, this is clearly explained in the requisite section with a proper reason. Excluding data to come to a better conclusion is not acceptable and therefore all exclusions are detailed clearly in this thesis. For example, the newsletter sections that were included/excluded for each newsletter are detailed in Section 3.2.

References

[1] M. E. Khan and F. Khan, "Importance of software testing in software development life cycle," *International Journal of Computer Science Issues (IJCSI)*, vol. 11, no. 2, p. 120, 2014.

[2] G. Fraser and J. M. Rojas, *Software Testing*. Cham: Springer International Publishing, 2019, pp. 123–192. [Online]. Available: https://doi.org/10.1007/978-3-030-00262-6_4

[3] M. Dowson, "The ariane 5 software failure," *SIGSOFT Softw. Eng. Notes*, vol. 22, no. 2, p. 84, mar 1997. [Online]. Available: https://doi.org/10.1145/251880.251992

[4] M. Aniche, C. Treude, I. Steinmacher, I. Wiese, G. Pinto, M.-A. Storey, and M. A. Gerosa, "How modern news aggregators help development communities shape and share knowledge," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 499–510. [Online]. Available: https://doi.org/10.1145/3180155.3180180

[5] S. Chowdhury and M. Landoni, "News aggregator services: user expectations and experience," *Online Information Review*, vol. 30, no. 2, pp. 100–115, 2006.

[6] R. Florea and V. Stray, "The skills that employers look for in software testers," *Software Quality Journal*, vol. 27, no. 4, pp. 1449–1479, Dec 2019. [Online]. Available: https://doi.org/10.1007/s11219-019-09462-5

[7] D. Pagano and W. Maalej, "How do developers blog? an exploratory study," in *Proceedings of the 8th Working Conference on Mining Software Repositories*, ser. MSR '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 123–132. [Online]. Available: https://doi.org/10.1145/1985441.1985461

[8] N. Seely and M. Spillman, "Email newsletters: An analysis of content from nine top news organizations," *Electronic News*, vol. 15, no. 3-4, pp. 123–138, 2021. [Online]. Available: https://doi.org/10.1177/19312431211037681

[9] [Online]. Available: https://www.lambdatest.com/about

[10] [Online]. Available: https://www.ministryoftesting.com/about-us

[11] D. Dylowicz, "The best software testing news." [Online]. Available: https://softwaretestingweekly.com/

[12] May 2021. [Online]. Available: https://trendingintesting.com/about-trending-in-testing/

[13] [Online]. Available: https://softwaretestingnotes.com/

[14] P. De Munck, "Data underlying the bachelor thesis: Using newsletters to analyze curated software testing content," 2023. [Online]. Available: https://doi.org/10.4121/9e59a43d-474b-46c2-9e29-c7c0b21bd6b4

[15] M. Vollstedt and S. Rezat, *An Introduction to Grounded Theory with a Special Focus on Axial Coding and the Coding Paradigm*. Cham: Springer International Publishing, 2019, pp. 81–100. [Online]. Available: https://doi.org/10.1007/978-3-030-15636-7_4

[16] May 2023. [Online]. Available: https://atlasti.com/

[17] B. Akkaya, "Api testing with karate framework," Apr 2023. [Online]. Available: https://medium.com/insiderengineering/api-testing-with-karate-framework-d62d4135447b

[18] J. Hodehou, "How to generate xml reports in pytest?" Apr 2023. [Online]. Available: https://www.lambdatest.com/blog/xml-reports-in-pytest/

[19] O. Cepeda, "When you should not automate your tests," Apr 2023. [Online]. Available: https://medium.com/@ocepeda34/when-you-should-not-automate-your-tests-cae66c52ea4d

[20] I. Gino, "Council post: Why almost every company is now an api company," Sep 2021. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2021/09/16/why-almost-every-company-is-now-an-api-company/

[21] R. Jones, "Tips for building an effective test automation framework from scratch," Nov 2022. [Online]. Available: https://www.accelq.com/blog/tips-to-build-effective-test-automation-framework/

[22] E. Red, "5 must-know cypress testing strategies for software engineers," May 2023. [Online]. Available: https://blog.devops.dev/5-must-know-cypress-testing-strategies-for-software-engineers-1477938669b

[23] A. Bhasin, "Introduction to shift left testing," Mar 2023. [Online]. Available: https://dzone.com/articles/introduction-to-shift-left-testing

[24] A. Mesbah and M. R. Prasad, "Automated cross-browser compatibility testing," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 561–570. [Online]. Available: https://doi.org/10.1145/1985793.1985870

[25] E. Guerra and M. Aniche, "Chapter 9 - achieving quality on software design through test-driven development," in *Software Quality Assurance*, I. Mistrik, R. Soley, N. Ali, J. Grundy, and B. Tekinerdogan, Eds. Boston: Morgan Kaufmann, 2016, pp. 201–220. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128023013000090

[26] A. Contan, C. Dehelean, and L. Miclea, "Test automation pyramid from theory to practice," in *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, May 2018, pp. 1–5.

[27] C. Cunha, "Testing applications with cypress," Apr 2023. [Online]. Available: https://www.getxray.app/blog/testing-applications-with-cypress-xray

[28] N. Li, A. Escalona, and T. Kamal, "Skyfire: Model-based testing with cucumber," in *2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, 2016, pp. 393–400.

[29] [Online]. Available: https://appium.io/docs/en/2.0/

[30] [Online]. Available: https://www.chaijs.com/