

Introduction of Protractor as test automation framework for AngularJS applications

Justo Bustamante



Author(s) Justo Bustamante.	
Degree programme Business Information- and Communication Technology	
Report/thesis title Introduction of Protractor as test automation framework for AngularJS applications.	Number of pages and appendix pages 57 + 3
<p>The objective of this thesis was to describe the introduction of Protractor as test automation framework in an AngularJS development environment.</p> <p>This thesis was written in a diary-based format and contains journals reporting the weekly assignments and the challenges and solutions encountered during the eight weeks of the thesis-project.</p> <p>Thesis-project was conducted at KiKo Oy, a Finnish middle-sized IT company from the language technology field. KiKo Oy had recently launched the DictionaryPlus service that is built with an AngularJS web stack. KiKo Oy stakeholders advanced that Protractor should replace RobotFramework –the test automation tool at that time– because Protractor is believed to best support Angular-specific locator strategies that allow to test Angular-specific elements without any additional setup effort.</p> <p>The author works as Software Quality Assurance Engineer in KiKo Oy and has a solid understanding of the company’s business domain and its agile development practices. KiKo Oy’s CEO and R&D leader have commissioned the author to conduct this thesis-project.</p> <p>This diary-based thesis describes, at a high level, the project’s sequential stages, such as infrastructure preparation, Protractor’s installation, tests scripting, tests execution, and project’s closure evaluation.</p> <p>As result of the project, KiKo Oy stakeholders decided to withdraw RobotFramework from service and to adopt Protractor as the standard test automation framework for their AngularJS applications.</p>	
Keywords Protractor, automated testing, AngularJS, Continuous Integration, digital dictionary, RobotFramework.	

Table of Contents

1	Introduction	1
2	Starting situation	3
2.1	Analysis of my current job situation	3
2.1.1	Evaluation and development	5
2.2	Interest groups in the workplace	6
2.3	Interaction skills in the workplace	8
3	Diary entries	10
3.1	Week 01.....	10
3.2	Week 02.....	14
3.3	Week 03.....	19
3.4	Week 04.....	24
3.5	Week 05.....	30
3.6	Week 06.....	36
3.7	Week 07.....	41
3.8	Week 08.....	47
4	Discussion and conclusions	52
	References	55
	Appendices.....	58
Appendix 1.	Keywords.....	58

1 Introduction

The present thesis analyzes the process of introducing Protractor as the standard test automation framework in the AngularJS development environment from a Finnish middle-sized IT company.

This diary-based thesis was written during the Autumn semester of 2017 in the period covering from week 09 to week 16 inclusive. Journals amount to 40 work days and were written on a daily basis. Each journal describes the tasks corresponding to each project's work day as it's specified in the diary-thesis guidelines. At the end of each week, a weekly analysis was issued. Additionally, the cited bibliography was referenced accordingly as the project progressed.

During the project's eight-week duration, a number of activities were logged as entries in the thesis-diary. Those activities are related to the phases of introducing and implementing Protractor in the technical environment of the target organization. Project stages comprise action points ranging from technical infrastructure's preparation to actual Protractors' installation and its commissioning into service, as well as evaluations of Protractor's scalable potential.

Target company at which the project was conducted is KiKo Oy, a well-established software development company specialized in digital dictionaries and language technology. KiKo Oy is also a market leader in E-learning solutions for language teaching. Its main customers belong to the Finnish education sector, such as high schools, polytechnics, and universities. As a middle-sized IT company, KiKo Oy employs 34 people including software specialists, business experts, and dictionary computational linguists. KiKo Oy is the daughter company of a major Finnish consortium dedicated to the content- and publishing industry.

The author works in KiKo Oy as Software Quality Assurance engineer for 3.5 years now and has collaborated to the test automation with RobotFramework system. At the time of this thesis-project, KiKo Oy was undergoing a technological modernization affecting most aspects of its software engineering department. Those changes had an impact on the selection of programming languages and development environments. Regarding test automation solutions, there was a need to find out what benefits would Protractor offer to KiKo Oy in comparison to the preexisting RobotFramework, and how production development had to be re-organized to support Protractor's implementation. Those were the driving reasons for CEO and R&D leader to commission the author with the realization of the present thesis-project.

The Application Under Test (AUT) for this project is DictionaryPlus, a free online dictionary and word-learning service developed by KiKo Oy with its new AngularJS web stack. DictionaryPlus combines top quality word lists with a gamified approach to create an innovative and motivating way to learn multilingual words. Protractor was planned to interoperate with DictionaryPlus and its related entities in the Continuous Integration system.

For this thesis-project an *ad hoc* team of stakeholders was formed among developers, business specialists and Quality Assurance personnel within KiKo Oy. Depending on their skills, some of those stakeholders were assigned with tasks at some stage of the project. Key skills needed for this thesis-project were a deep understanding of software testing, agile development experience, familiarity with scripting in JavaScript, and a basic insight of project management. The special requirement was to have a solid understanding of the business domain in which the project is encompassed, namely: online digital dictionaries.

It's worth to mention that –as I write these words– this might be the first thesis dealing with Protractor to be published in the Theseus.fi repositories.

Author has included as an appendix a list of keywords that may help the reader to understand the technical terminology used in the present thesis.

As a side note, the target company's Non-Disclosure Agreement obligated the author to not to reveal the company's real name neither the brand-name of the application under test. For this thesis purposes, the company is mentioned under the alias **KiKo Oy** whereas the application under test is named as **DictionaryPlus**.

2 Starting situation

2.1 Analysis of my current job situation

My current job position is Software Quality Assurance Engineer in the company KiKo Oy. As a Software Quality Assurance Engineer, I'm in charge of all the testing-related matters and am a member of the Products and Services Development team. Among my core responsibilities are the following:

- Co-defining a feature's specifications (along with other stakeholders)
- Designing test scenarios
- Writing test plans
- Writing test cases
- Reporting the detected issues and tracking them until final fixing
- Defining test specifications for automated testing
- Scripting automated tests
- Executing automated tests
- Issuing the final test reports

My work cycle starts when the concept of a new product or feature is being designed. I co-participate in the planning sessions along with the Product Owner (PO) and the developer's staff. During those sessions, we establish the main functionalities (i.e. what the feature is supposed to do; and not less important: what it isn't supposed to do). The sessions outcome is a specifications document in which all possible use paths and use cases are defined. After the feature's use-flow logic is approved, I begin creating the test documentation for the official feature's specifications.

Next step is to identify all possible test scenarios. Because these are tightly related to the use scenarios, I must foresee all the possible situations in which users –who are granted with different roles– will interact with the feature. An efficient test scenario should cover not only the correct usage-flow but also the incorrect actions of so-called “fool users” trying to break the application.

A test plan is a wider, more encompassing document which includes all the instructions for testing each instance around the feature. I'm responsible for outlining in the test plan each topic involved in the test efforts, such as platforms, browsers, test cases, schedules, resources, etc. A considerable amount of coordination and time is invested in creating this document.

Once test documentation has been reviewed and approved, the actual testing can commence. I proceed to test with the two standard test approaches, namely manual and automated testing.

Manual testing can be executed with two distinct methods: it can be scripted (i.e. following a pre-determined step sequence) or it can be exploratory (i.e. going in a more intuitive or freestyle mode).

In the case of automated testing, I'm responsible for coding the test scripts in JavaScript. These scripts will be run in our test automation framework each time a code's update is merged to the master branch in the CI code base.

Each feature's functionality is exercised in creative ways trying to find any possible vulnerability or failure in its logic. Each detected defect is immediately reported to the developer's team for debugging and fixing. Once the defects are fixed they still must undergo regression testing as a verification measure.

If the final acceptance reports are satisfactory, I notify the PO, Lead developer and other stakeholders that testing activities have been completed and that the new feature works according to specification. Then the feature can be deployed to the production environment.

Documentation plays an important role during the testing activities. I'm in charge of creating, updating and managing test documents in special repositories for the entire organization to access. Most critical documents are the ones related to feature's specifications, meaning the technical details of the feature scope.

A special know-how is necessary for successfully accomplishing the test assignments. A test engineer must have a good knowledge of the organization's business domain. In the case of KiKo Oy, the main domain is digital dictionaries and language technology.

In order to reach my long-term professional goals within KiKo Oy, I need to gather other extra special abilities. It's indispensable to acquire more knowledge about computational linguistics, programming, project management, customer interface, usability, and team managing.

In the nowadays Finnish job market for software QA engineers, it's nearly an imperative to be a skilled programmer. Python, JavaScript, Java and Ruby are the most sought-after programming languages in QA positions. Moreover, it's a clear edge to have a workable knowledge of databases and agile practices. Fortunately, I have had the opportunity to develop the above-mentioned skills during my time in KiKo Oy.

KiKo Oy's new web stack is based on AngularJS technology which includes its own technical framework, components, and tools, such as Node.js, Karma Runner, Protractor, etc.

2.1.1 Evaluation and development

I set the theoretical scale of 100 points as the equivalent of an experienced performer. This is the referential basis for my own evaluation.

I consider myself as a skillful performer scoring around 80/100 points. I feel confident working nearly independently in the execution of my duties and have a solid knowledge of the tools and methodologies involved in the modern practices of software testing.

However, although I've been for over 3.5 years with KiKo Oy, I still must learn and develop my skills further. When carrying out programming tasks, I frequently must reach my colleagues asking advice on how to optimize my scripts' logic. Other topics that I don't master yet are the mechanics behind CI jobs and the configuration of test environments.

As to my professional development, I honestly believe that I have already covered two-thirds of the path towards becoming a senior test automation engineer. As a proof, I can demonstrate my newly acquired skills that enable me to test databases and to write complex scripts. My career has experienced an evident upgrade since I started my work relationship with KiKo Oy.

On the other hand, the upcoming awarding of my Haaga-Helia degree is an incentive for improvement in my professional life. My courses at Haaga-Helia have given me the necessary abilities and the theoretical background for understanding other business areas. Now I even feel confident acting as a junior project manager and taking care of tasks related to accounting, pricing and customer's service.

My performance has become more visible inside KiKo Oy to the point of me being incorporated into the R&D team. The latter means that I'm entitled to work in agile environments with heavily automated processes. Nowadays my supervisors and PO can appreciate that my contributions are increasing the added value in our services.

Sometimes, as an expression of trust in my judgement, the CEO of KiKo Oy has assigned me with special tasks about usability and user experience. Moreover, my opinion has been consulted by the marketing team during the planning sessions, because being a tester

makes me into an end-user's advocate with a rich insight into what users like and expect from our products and services.

Additionally, KiKo Oy has implemented the "*Kaikki myyntiin*" policy by which all the staff is encouraged to participate in the sales activities whenever it's applicable. Because of my gained expertise in that domain, I've been sometimes required to attend sales meetings along with salesmen and new or old customers.

Something similar occurs with the customer service's department. I've been given a trusted task of contacting customers who report a functional issue. I can directly request more specifics on the problem and keep them updated about the ongoing corrective work.

And again, it became clear that programming is the main area where I must dedicate more time and effort. Because technology evolves every day, it's nearly compulsory for an automation tester to stay up-to-date on the contemporary trends. New programming technologies bring new tools, frameworks or methodologies that directly impact on the current test practices.

2.2 Interest groups in the workplace

The KiKo Oy company splits internally into five main departments, that is Management, Marketing & Sales, R&D, Dictionary Content, and Customer service. Stakeholders from R&D (e.g. POs and programmers) and Customer service are the ones most closely related to my QA position.

Interaction among internal stakeholders occurs generally on a daily basis. My direct supervisor is the Product & Services Director, and she is instrumental in driving through different channels the dynamics of stakeholder's interaction. My relationship with other stakeholders occurs along the lifecycle of several projects going on within KiKo Oy. The Software QA engineer is tightly bound to the R&D's sphere.

In what concerns to Customer service, the support requests are the binding element in my relationship with stakeholders. Requests are made via emails or phone calls, or eventually via a Skype video conference.

As to external interest groups, it must be reminded that KiKo Oy is the daughter company of a major Finnish consortium. In that sense, the consortium branches and partners are among the main external stakeholders. Other external groups of interests are: (a) customers

from private and public sectors; (b) content providers, such as dictionary publishing houses, and; (c) 3rd party cooperation partners (e.g. TEKES agency).

The interest groups can be appreciated in figure 1.



Figure 1. shows the stakeholder groups which frequently interact with the QA engineer's position within KiKo Oy.

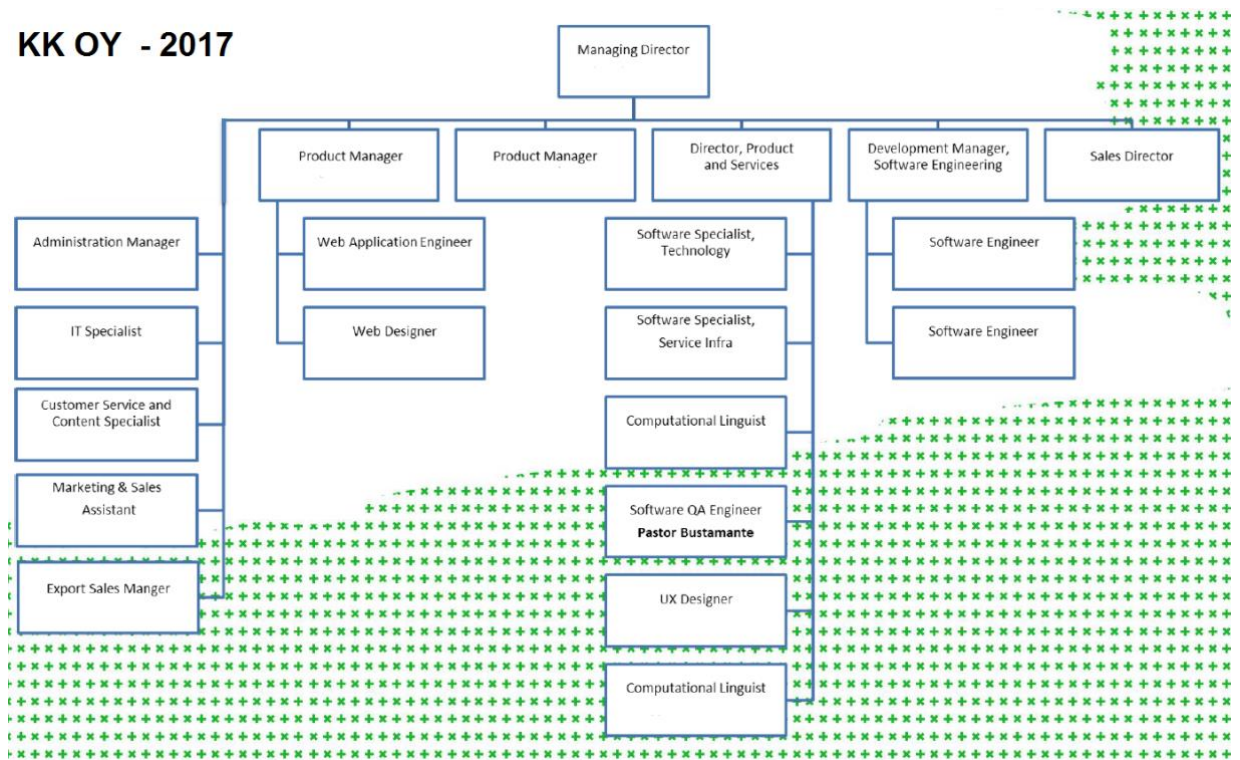


Figure 2. shows the Organization chart where the QA engineer's position can be appreciated within context.

2.3 Interaction skills in the workplace

Aside of Finnish, English has been adopted as the official work language by KiKo Oy.

KiKo Oy staff has a cross-cultural background of five different nationalities (e.g. Finnish, French, Peruvian, Italian, Pakistani). The majority of foreigner colleagues belong to the R&D department.

The bulk of our customers is located in Finland; therefore, the sales- and marketing activities are conducted almost entirely in Finnish.

A similar language's approach applies to customer service. Our major customers are the Finnish educational institutions. This means that written and oral communications are mainly in Finnish.

A different situation occurs when it comes to partners and content providers. Usually, their headquarters are located abroad (e.g. London, Madrid, Paris, Munich, etc.) and their contact persons prefer to communicate in English.

In my daily work, the communication flow is constant and swift. I reply to requests in the shortest time avoiding delays in the production schedules.

KiKo Oy's communication channels are online-based. Messaging tools are Skype (Premium account) and Slack instant messaging. Our email servers are configured to interoperate with the project management tools sending notifications to the involved recipients.

For documentation management, there's special conventions and guidelines in place. Because KiKo Oy's business domains (i.e. Dictionaries and Language technology) are highly specialized ones, we have created a proper terminology as well as a terms bank. This enables personnel to be on the same page and speak a common jargon.

In my daily QA work, I produce a considerable amount of documentation, such as test plans, test scenarios and technical reports that must undergo reviewing before approval. Readability and agility are the main principles to be kept in mind when writing QA documents.

Oral communication succeeds mainly in the daily Scrum meetings held at 10 o'clock every morning. Each member must report to the team about what he has done on the previous day, and what he plans to do in the present day, and flag any potential impediment.

Planning sessions are characterized by face-to-face communication among colleagues, usually speaking in English. Colleagues working remotely can participate via videoconference. Session's facilitator makes sure that the session's duration remains within the time-box's limit.

There have been challenges for me when trying to communicate in Finnish. Despite the efforts that I've done improving my Finnish skills, I still feel unable to write error-free documents or to understand Finnish bibliography. However, I continue practicing and enhancing my knowledge of technical Finnish.

3 Diary entries

3.1 Week 01

Mon 27-Feb-2017

On this date, I officially started driving the whole project of my thesis, namely: "Introduction of Protractor as test automation framework for AngularJS applications". The first initial step was to meet the general CEO once more and re-confirm with him that I had got the necessary approval and support from KiKo Oy's management for starting the project. I estimated a time-box of about 20 minutes for this meeting.

Also, in order to schedule a favorable date for the project's general presentation, I had to check the involved stakeholders' calendars hoping to assure their attendance.

Day's review:

My main goal was accomplished after some delays due to CEO's tight schedule. However, the meeting succeeded well and I received the official acknowledgement that my project was supported by the organization's management. This was an important first formality before kicking-off the project.

In addition, I quickly asked stakeholders about their time conveniences for holding the project's general presentation; we all agreed that Wed 08-Mar-2017 would be a suitable date.

As an aside, I was able to make some progress writing the project's paperwork, particularly the project's general presentation.

Tue 28-Feb-2017

Based on the progress done the previous day, I planned to draft all my data, texts and graphics in PowerPoint slides for the project's general presentation. I also wrote down some thoughts that could help me to make a convincing case in front of the stakeholders.

Day's review:

The main task today was a meeting with our Products & Services Director (P&SD). We discussed some particularities that could impact on the R&D team. She gladly accepted to become my supervisor during the project's realization. However, it's worth to remember that the R&D team will have the primary competency in the project.

P&SD asked me to include in the project's presentation certain technical topics that could eventually serve as a discussion basis with other stakeholders.

At the end of the day, besides accomplishing the planned tasks, I had the chance to give further explanations to colleagues who got in touch with me.

Wed 01-Mar-2017

I was still working on the general presentation for the kick-off session. I planned to spend several hours gathering data and covering the topics flagged by P&SD in the last meeting.

Another important task was to draft a survey asking stakeholders about their goals, expectations, and suggestions. Their feedback would help me to fine-tune my project's proof of concept.

Additionally, I booked a new meeting with the P&SD at her request. She informed me that the R&D leader had pointed out some valid topics and wanted to discuss them with me.

At the end of the day, I managed to finish the first documentation version. However, it was still far from being deemed as solid documentation.

Thu 02-Mar-2017

On this date, the meeting with the P&SD turned to be the main task. R&D leader's attendance was a great contribution and he shed light on some points that I had missed in the preliminary sessions. For example, he explained the feasibility of converting legacy test scripts (from RobotFramework) into a readable format for Protractor.

Not less important was to prevent the risk of resource shortages. R&D leader was concerned about bottlenecks resulting from R&D personnel being allocated to the project's activities. This issue demanded further and deeper discussion.

A series of other technical topics remained unaddressed. Hence, I had to approach my other R&D colleagues and ask their input.

I also spent some time drafting the project scope's documentation. It might take a couple of days before it's ready for review and approval.

At the end of the day, I felt that I had improved my skills for arranging meetings, discussing with stakeholders and processing feedback.

Fri 03-Mar-2017

As an action point derived from yesterday meeting, I arranged a work session with our IT-support specialist. The objective was to assess if our current IT capabilities wouldn't suppose any impediment for the project. A comprehensive evaluation made clear that our technological infrastructure (i.e. servers, licensing, tooling, work stations, etc.) was fit and compatible with the Protractor's implementation.

For streamlining the project's information exchange, I sketched a communication flow with different channels (e.g. emailing, group chatting, meeting scheduling, briefings, etc.).

I spent some time examining an issue flagged by a colleague about potential downtimes and platforms' stoppages due to Protractor-related modifications. The objective was to avoid any risk of conflicting configurations at the time of implementing Protractor dependencies in our servers.

Friday was not much productive due to the fact that some stakeholders preferred to work remotely from home.

Weekly analysis

The past week was obviously more interesting and exciting than any usual work week. The project's starting phase demanded around 75% of my work time. I spent that time writing documentation, arranging and conducting meetings, and collecting feedback from other stakeholders.

Although the aforementioned activities were familiar to me, it was now when I could apply discussion- and communication skills my own way and pace. I wanted to inject more dynamism in the meeting's practices looking forward to collecting information more efficiently.

One of the greatest satisfaction last week was to notice that the Protractor's introduction project was unanimously welcomed. It sparked interest and motivation in the R&D team because they saw Protractor as the right test tool to be used in the AngularJS web stack. Some previous feedback had already indicated that RobotFramework (despite it's a superb test automation tool) was showing some shortcomings when exercising AngularJS locators. Most developers believed that it was due time for introducing Protractor and replacing RobotFramework.

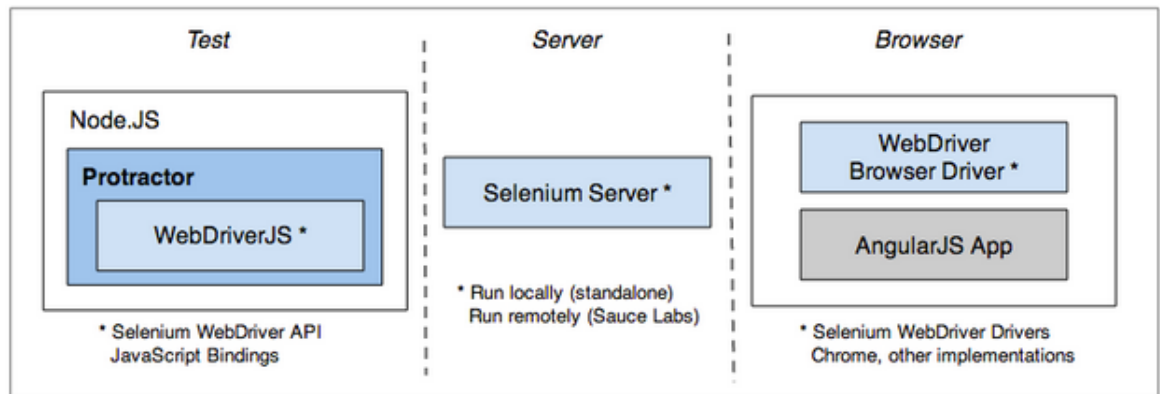


Figure 3. shows how Protractor works and interacts with AngularJS (workflow)

Stakeholders were also aware that some substantial server's configuration needed to be done. They knew that building a new test framework was going to cost time and effort and that my time availability would be diverted from my usual work. However, all the stakeholders shared the opinion that Protractor introduction must be given priority.

On the other hand, I identified some challenges that I had overlooked when drafting the project's proof of concept.

One main challenge was posed by the project-related documentation. Unlike the scope that I initially estimated, it seemed that a larger amount of documentation was necessary to satisfy all the information needs. Each of the stakeholders could request documentation from a certain point of view. Documentation could be of business, marketing or purely technical nature. I underlined that documentation had to be collaborative, meaning that documents had to be all the way commented, edited and approved by a group of stakeholders. Despite the collaborative benefits added to the documentation content, the downside is that collaboration demands more time and coordination. Enhancement of the

collaborative practices was filed as a future task; the goal here is to increase agility in documentation creation.

Another big challenge was personnel resourcing management. Even though the stakeholders were enthusiastic to help in the Protractor project, there's concern about potential problems caused by distracting resources from their usual duties. I explained to the affected parties that my recommendation was to avoid disrupting the normal performance of the R&D team. I assured them that the largest share of the project's effort, including coding and executing the test iterations, was to be carried out by me. In case I need advice or hands-on help from developers, I will look for the most convenient free slot in their schedules. Also, I made clear that the timeframe for the Protractor project (i.e. Feb-May 2017) was selected after a careful evaluation of KiKo Oy's roadmaps checking the months with less project's workload. However, I still foresee that further action plans will be necessary to address any resourcing issue that may arise.

Last but not least, I saw that there's room for improvement in how to manage delays. It's worth to remember that this Protractor project involves several stakeholders who have their own schedules and deadlines. Sometimes their replies or reactions are delayed because of more urgent matters. This should be seen as a potential difficulty in the project's lifecycle.

3.2 Week 02

Mon 06-Mar-2017

On this first day of the week, I had to address some organizational tasks that must be accomplished first before continuing with the project. These tasks were listed as follows:

- Meeting the R&D lead developer and other programmers for establishing the definitive resourcing needs
- Start drafting the Protractor's project roadmap

The first task took nearly the whole morning, and it resulted in a long discussion with the R&D developers. There were a few issues on personnel availability that needed to be clarified.

For the second task, I had to do a big amount of paperwork besides double-checking with other involved stakeholders the feasibility of the proposed timeline.

My day's review was positive. Despite of the demanding workload approaching and consulting several stakeholders, I managed to fulfill my planned tasks. Roadmap drafting was progressing, but not completed yet.

Tue 07-Mar-2017

Today I focused on three specific tasks, which I supposedly could get done by working entirely independently:

- Conducting the internal survey in which I asked from the involved groups additional ideas about the Protractor's project.
- Finishing the definitive version of the project's general presentation.
- Starting to outline the theoretical test automation environment. This means creating a structure with all the entities involved in the test framework, such as software, hardware and infra (i.e. processes, methodologies, best practices, conventions, schedules and communication channels, etc.).

Day's review:

The internal survey became a more complicated task than I thought it would be. It took a lot of time and hassle to get the reply from the invitees. Some of the surveyed people didn't answer the critical questions or even left the questionnaire untouched.

On the other hand, I finished the definitive version of the presentation for tomorrow Wed 08-Mar-2017. This task was considered as done.

Finally, I underestimated the complexity of outlining the theoretical test automation environment. I couldn't even start with it because there were many moving parts of deep technical nature. This task had to be postponed for a later date.

Wed 08-Mar-2017

Today I've got an extraordinary task to achieve, namely: to keep the general presentation about Protractor as test automation framework for AngularJS applications.

At 10.30 I started with the presentation at the KiKo Oy's big meeting room. It was a positive sign to see the whole group of stakeholders attending the presentation. Also, people from other departments were present listening and participating.

Day's review:

The Protractor's general presentation task was achieved completely, and I believe that it fulfilled each of its objectives (i.e. informing, engaging, and declaring the official status of the project).

I was very busy explaining the project's scope, as well as its potential benefits and challenges. Also, I felt satisfied with my performance as a presentation's facilitator, because I was able to sell the Protractor's project as an innovative idea to the rest of the organization. During the questions round, I managed to answer satisfactorily the multiple inquiries from the attendees.

Thu 09-Mar-2017

After the general presentation, some additional information pieces were still requested by certain stakeholders; consequently, I had to re-schedule my tasks for today.

CEO and R&D leader were interested in evaluating the advantages of applying Acceptance Test-Driven Development (ATTD) methodology (Jain, 2008) along with Protractor's implementation. They asked me to arrange a specific discussion on this topic.

That discussion disrupted my initial task schedule and raised several sub-topics to be taken into account.

The original Protractor's implementation plan envisioned that the current Test-Driven Development (TDD, 2017) would remain as the applicable methodology. The idea of replacing TDD by ATTD assumed to expand the project's scope to include Cucumber tool (Cucumber.io, 2017) as a new layer on top of the Protractor framework. This plan's change would add more software, more configuration work, and a greater learning curve as well. Its increased complexity can be appreciated in the next figure:

Protractor-Cucumber Test Automation Framework

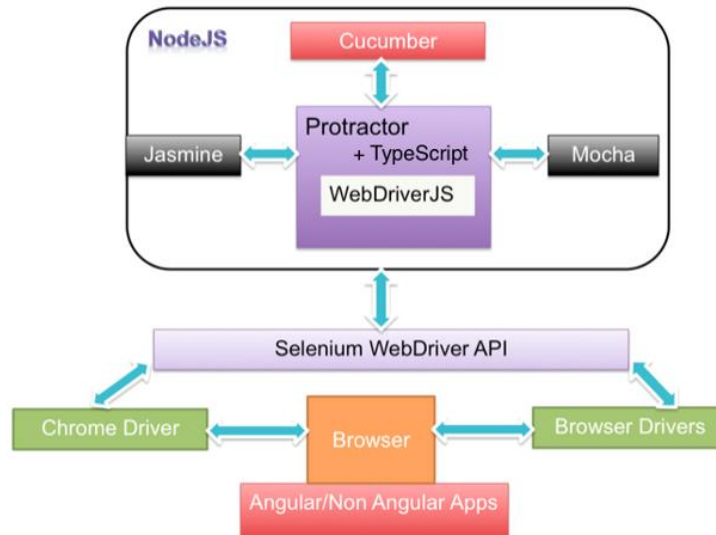


Figure 4. shows how an ATDD framework would look like with Cucumber on top

Day's review:

The main task of this day, even though it was not scheduled, was a long and exhausting one. Discussions on methodological issues were the main point. At the end of the day, I managed to validate my initial proposal of preserving TDD methodology for the Protractor's implementation. Today I took advantage of my negotiation skills and got the discussion resolved.

Fri 10-Mar-2017

For today I planned two main tasks in the technical preparation, namely:

- Getting the test server's availability ready
- Doing paperwork on pending topics (e.g. theoretical test framework and TDD workflows)

At the end of the day, only the first task was done. I worked along with our IT specialist who configured a test server where CI will intercalate with Protractor- and unit tests. This task allowed me to better understand how servers can act as test platforms.

On the other hand, the paperwork had to be postponed due to more urgent assignments of my usual test work (e.g. testing dictionary updates)

Weekly analysis

The past week had a key significance in the implementation of the Protractor's project.

Main goals had various aspects that can be identified as follows: (a) informing the whole organization about the project kick-off and scope; (b) making sure that the necessary human and technical resources were allocated for the upcoming project's phases; and (c) defining the test methodology upon which the Protractor tests will be executed.

The Protractor's general presentation had aimed to fulfill the first aspect of the main goal. During the presentation, I had the opportunity to go through the project's scope, timeline, objectives, potential benefits, and challenges.

Around 15 people attended the meeting in the big conference room at KiKo Oy premises. Attendees showed a clear interest in knowing the project's specifics. The subsequent Q&A round brought inquiries of diverse nature and served as a measurement of the interest given to the project.

As the test management guidelines state, a good test manager must be able to spark interest among colleagues and stakeholders when introducing innovative ideas. Presentations must contain clear concepts and include the proper facts and terminology. Language had to be clear and concise and understandable for any member of the organization, even though the topics in question were eminently technical.

After having held the general presentation and other small meetings, I felt that I learned to act as meeting facilitator and to keep better work sessions. This is undoubtedly a very valuable skill when pushing forward a large project inside an organization. The latter is in concordance to what Chris Anderson recommended in her article "*How to give killer presentations*" (Anderson, 2013).

Another crucial assignment last week was to make sure that all necessary human and technical resources had been allocated to the project. It was agreed that one programmer (or two if needed) would assist me whenever I encounter difficulties in writing test scripts. Moreover, it was agreed that my assistance requests will be given priority by the IT colleagues. IT support is pivotal granted the special needs of server's configuration. As a remark, by confirming the availability of human and technical resources, I secured the

convergence of two important foundations in a software project, as it's advised by Steve McConnell in his book "*Software Project – survival guide*" (McConnell, 1998).

I had to recognize again that thanks to my ICT studies at Haaga-Helia, I could clearly understand the planning of technical projects within a medium-sized organization.

Last but not least, I believe that I had improved my negotiations skills; it was demonstrated during the discussions on the test methodology to be applied in the Protractor's implementation. Even though CEO and R&D developers were evidently more interested in shifting to ATDD, I confronted their ideas and produced facts and data to back up that TDD was the right choice for the Protractor's introduction phase.

I kept in mind one principle that characterizes successful agile testing projects. This principle –as described in the book "*Agile Testing*" (Crispin & Gregory, 2009)– is to correctly prepare the methodology, infrastructure, and workflows for the upcoming agile testing practices.

3.3 Week 03

Mon 13-Mar-2017

On this first day of the week, I planned to resume defining the theoretical test automation environment (i.e. processes and methodology). This important task had been postponed due to other pre-conditions of the established schedule.

I subdivided it as follows:

- Outlining the processes for Protractor' usage
- Reutilizing/Adapting some practicable workflows from the current RobotFramework environment

The first subtask was accomplished almost completely. Protractor's processes were easy to depict as a concise flowchart for everybody to understand. My previous experience outlining software quality processes helped me with this assignment.

For the second subtask, I had to dig in our intranet legacy files searching adaptable workflows from the RobotFramework environment. There were some useful documents that I could reutilize to fit our current needs.

My day's review was positive. I felt that now I can work more independently, without bothering my colleagues or other stakeholders.

Tue 14-Mar-2017

Today I focused on three specific tasks that I expected to fulfill working independently, as I did yesterday:

- Finishing the first version of the theoretical test automation environment (including the chapter about best practices, conventions, and schedules)
- Finishing the definitive version of the Protractor's project roadmap
- Finishing the documentation about processes for Protractor's usage

Day's review:

Today I confirmed that independent work is a great satisfaction and a time-saving practice. I was able to set my own pace and arrange my day's program in the most suitable way. My original schedule with three specific tasks was fulfilled without running into any inconvenient factor. It was a productive day in terms of documentation, specifically in terms of processes and workflows.

Finally, I was satisfied with my improved skills for estimating time-boxes. Now I can allocate more accurately the right number of hours needed for a certain task. This was how I distributed the three tasks along my work day, and this planning proved to be realistic.

Wed 15-Mar-2017

Today I undertook the first software hands-on task, that is to say, the installation of WebStorm, the Integrated Development Environment (IDE). WebStorm is a proprietary IDE licensed by JetBrains, and with this IDE the Protractor test scripts are written and executed.

I went through all the stages of WebStorm installation, including the following action points:

- Downloading the free trial version (30 days) in two desktop computers with different platforms (i.e. Windows and Mac OS)
- Checking the installation dependencies
- Installing WebStorm
- Customizing the preferences of the Graphic User Interface

Day's review:

The entire installation process went smoothly and without delays. However, it required attention and analytical work, particularly when checking that the necessary dependencies were right in place. This was an important aspect because by doing so the risk of software's conflicts was avoided.

Among test experts, it's said that WebStorm is the optimal IDE for writing Protractor tests. For that reason, I wanted to take the most out of its advantageous features and spent some hours customizing the interface and optimizing the user preferences. JetBrains website (JetBrains, 2017) offers more instructions about WebStorm installation.

Thu 16-Mar-2017

After having installed WebStorm the previous day, it was the time for setting up Protractor in our test stations. This endeavor was planned as the main task for today.

I basically followed the installation instructions from Protractor's official website (ProtractorTest, 2017). As it was clearly stated in the tutorials, installation demanded the verification of prerequisites, such as the availability of Node.js and Java JDK in the test computers. The absence of those prerequisites would have made Protractor's operability impossible.

I can say that my technical skills were up to the task, even though I had to request advice from our IT specialist at some stages of the installation process. But it must be said that the configuration was done by other colleagues because this is an area I'm still not completely familiar with.

Day's review:

After several hours of dedicated work, I got Protractor up-and-running on my two test machines. Protractor *per se* doesn't have its own graphic interface, but it must be exercised through WebStorm when tests are executed or debugged.

Up to now, work has been progressing according to the project's roadmap.

Fri 17-Mar-2017

For today I planned the setup of Selenium Webdriver *locally* on the test stations (*locally* means the installation in individual test machines, not in the test server).

Selenium WebDriver is an essential dependency in the Protractor framework, and it must be properly installed before attempting any test execution.

Selenium WebDriver installation was a sensible task which demanded some level of expertise. The complexity of the setup process can be assessed by reading its installation tutorial (ProtractorTest, 2017).

I requested once more the support from our IT specialist. It became evident that configuration was an area in which I must develop further.

Day's review:

After several hours of dedicated work, it was possible to setup Selenium WebDriver and get it up-and-running according to the project's needs. This task was insightful and taught me some basics of software's setup and configuration.

At the end of this last day of the week, all of the necessary installations and setups had been completed.

Weekly analysis

The past week brought me closer to the test tools of the Protractor's test framework.

In the first two days of the week, I focused on some overdue organizational tasks, such as finishing the theoretical test automation environment and the processes documentation. They were placed high on the list of priorities. They all provided the right guidelines on how to proceed with a Protractors' environment. Processes show the right testing paths and guarantee the observance of procedures.

Having in place clear and enforceable processes was a matter of key importance in this project. It must be reminded that the TDD methodology ruled the test automation activities within the R&D team, and for that to happen it was necessary to define processes that work in tight coordination with the team's development methodology.

By reading books about agile testing and its implementation within test projects (Crispin & Gregory, 2009), I learned that every testing project must be paved with the proper

infrastructure. This infrastructure was of both theoretical- and technical type as it has been described in the assignments of the past week. Fortunately, during my technical courses at Haaga-Helia University, I learned how to assess and outline workable processes. Furthermore, my experience as professional software tester played a key role when designing best practices and procedures, let alone the numerous theoretical aspects of a test automation environment.

On the other hand, the past week put me in front of the tools going to be present in my daily test work, such as WebStorm, Protractor and Selenium WebDriver.

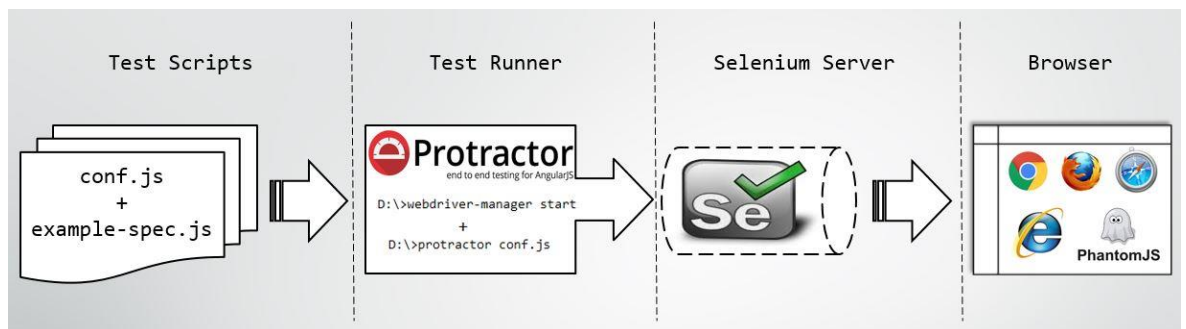


Figure 5. shows how the tools are wired together in a Protractor's framework

From the above array, only WebStorm was a proprietary software (yearly license costs around 125 USD). The other tools are open source meaning they are practically free of charge. In this sense, I followed the R&D policy of favoring open source tools over proprietary ones. The reason for such policy was not only saving money, but also the fact that open source software allows contributions and enhancement code from enthusiastic and collaborative programmers around the world. This is the case of some programmers in KiKo Oy staff who regularly collaborate with pull-requests to open source projects. The latter was a new insight for me and opened my view about the endless possibilities for contributing with testing-related projects in the future.

Last week's main goal was to have Protractor up-and-running locally on our test machines. And this was accomplished without much trouble or delays. This achievement was not the result of plain luck, but the consequence of proper task- planning and prioritizing. Specialized resources –like IT support persons– were there available whenever their help was requested. Efficient task scheduling and forecasting enabled me to avoid delays and bottlenecks in the resources availability. I was introduced to resources scheduling when

reading some books about software testing management (Farrell-Vinay, 2008), (Osterhage, 2015). Definitely, this has been one of the most valuable learnings about test project management.

Stakeholders were satisfied with the progress achieved last week. They were full of expectations on the new test automation framework.

Selenium WebDriver proved to be a powerful tool and showed its high performance when running test scripts at high speed. It means that test execution won't occupy the CI server for longer times during the integration jobs. Selenium library offered a wide array of useful functions, such as asserting, looping, counting, finding, and windows handling. This provided us with an enormous potential for multiple test solutions. Selenium WebDriver literature is well organized and retrievable from its official website. Learning and applying advanced Selenium commands in my test scripts can be deemed as a future assignment for me (SeleniumHq, 2012).

3.4 Week 04

Mon 20-Mar-2017

This week began with some configuration tasks.

I subdivided the aforementioned task as follows:

- Tooling the Continuous integration server (CircleCI server and GitHub)
- Configuring the technical environment (CircleCI, Protractor, Selenium)
- Smoke-testing the configurations (CircleCI and Protractor)

CircleCI is a continuous integration and delivery platform (CircleCI, 2017). Protractor was meant to interoperate with the CircleCI build system. Also, Protractor iterations were going to be intercalated with the Unit Testing routines in the CircleCI server.

Day's review:

My participation in the configuration tasks was limited. I acted more as an assistant or learner rather than as a QA engineer. Both R&D leader and the IT specialist took over the assignment. I just gave assistance and took notes about the procedures.

Configuration's smoke-testing was conducted by me. For that purpose, I executed a basic automated test script directly from my local test computer. After completing the corresponding build- and test iterations, CircleCI console finally threw the message "*test pass*". The former confirmed that the necessary entities were correctly wired together at the server and enabling Protractor tests.

Tue 21-Mar-2017

On Tuesday, an unplanned work assignment came up: our IT person decided to do a wide refactoring of the CircleCI configuration. He argued that a new optimized configuration would run integration jobs in a more efficient and time-saving fashion.

Besides the above, my main task for today was to design the applicable test plans.

Designing test plans included the following subtasks:

- Defining the scope of automated testing
- Defining test scenarios
- Defining the automatable test items (i.e. the most critical story-tests, and regression tests as well)
- Defining test environment, schedules, and responsibilities

A test plan is a document of essential importance in software testing. It identifies a range of functional and non-functional aspects that every tester must know before testing an application (Software testing fundamentals, 2017).

Day's review:

I decided to reuse and adapt some legacy test plans from the preexisting RobotFramework documentation. Test scenarios and test cases covering DictionaryPlus remained practically identical irrespectively of the test tool being used (e.g. Protractor or RobotFramework).

My experience in creating test documents helped me to quickly identify which aspects were applicable to our project and which weren't. This approach allowed me to save time and to get the task almost done. On the other side, the IT person finished the refactoring of CircleCI configuration. Now, both, build jobs and test iterations can be run faster.

Wed 22-Mar-2017

Today's main goal was to focus on the test cases. Test cases must follow the acceptance criteria derived from the main test plan.

Here it's worth to emphasize that Protractor's test cases had to be defined at the acceptance level; in other words, they were not unit- or integration tests.

Test cases had to be written for both regression and story/feature testing.

Today's goal included other tasks as well:

- Creating test data
- Creating stubs and test credentials (e.g. for authentication purposes)

Day's review:

I achieved only 70% of the planned work because it was necessary to add new test cases. Some changes were done in the DictionaryPlus interface, and those changes had to be included in the automated test coverage.

On the other hand, defining test cases was a task that required collaboration from other experts (e.g. GUI designers or business analysts) who knew best the GUI flows and the critical steps of certain business processes. Today was not possible to have those experts collaborating due to their tight schedules. I hoped that tomorrow they could reserve some hours for helping me with the test cases.

Thu 23-Mar-2017

Today's goal was to get the test cases written and reviewed, as well as to define the test exit criteria.

Fortunately, the required stakeholders were able to adequate their schedules to fit the documentation work of today.

Defining exit criteria –meaning the criteria of how much testing must be conducted and when the test activities can be considered as concluded— required a multidisciplinary discussion. In that sense, I arranged a meeting with other experts, such as graphic designer, business analyst, lead developer and computational linguist. I collected their opinions and drafted a document which was subsequently reviewed and officially approved.

Right after completing that task, I interviewed some stakeholders from customer service. Their recommendations and executable examples will serve as a basis for test cases enhancement.

Day's review:

Despite the big workload, at the end of the day, I managed to sort the impediments out and get the planned tasks done. Test cases were now defined, and test data was already updated and classified.

Reducing delays in the work with other stakeholders is something I felt satisfied with. Thanks to good planning and coordination I could arrange sessions with the required stakeholders without waiting long times. A key part of session planning is to avoid disturbing the stakeholders' work schedules.

Fri 24-Mar-2017

For today I planned to organize and distribute the documentation that I prepared in the previous days.

This was a task I could do independently. All documents and data tables were uploaded to our Intranet- and GitHub repositories and were shared with the involved stakeholders.

All the above documents constituted the test infra for Protractor's project.

Day's review:

The assigned tasks were fulfilled effortlessly and without disrupting other stakeholders. After having uploaded the documentation, I sent an email inviting the stakeholders to review the test infra and share with me their inquiries, concerns or suggestions.

Accomplishing of the Protractor-related tasks demanded the whole morning. The rest of the day was dedicated to attending normal KiKo Oy test assignments.

Weekly analysis

The past week represented the completion of the last project's phase prior to actual Protractor's test execution. Now all necessary entities (e.g. software, hardware, and infrastructure) have been installed and configured according to specifications.

This is a recap of what has been done in the last two weeks:

- IDE (WebStorm) was installed and customized in test computers
- Protractor was already installed and working in the test computers
- Continuous integration server was configured for CircleCI and Protractor
- Selenium WebDriver was already setup in CircleCI server and in both test machines
- Test plans and other documentation had been uploaded to a shared repository

All the above constituted a sort of test framework built around the Protractor tool. Even though a test framework's creation was not a specific goal, the current project required to do so because each test automation tool needs a proper technical infrastructure to correctly interoperate with other entities (Huiskonen, 2011)

From the above array, only WebStorm and Circle CI were proprietary software, and their monthly fees aren't expensive. The rest of the tools were open source which means they are practically free of charge. In this respect, I followed the R&D policy of favoring open source tools over proprietary software. Arranging this new automation framework represented to KiKo Oy only a minor money investment. In that sense, my Protractor's project can be categorized as a low-risk endeavor at least in budgetary terms. The former translates into less pressure on me because I'm not putting large amounts of money at stake.

About the specifics of last week.

I'm now convinced that I can dare to configure the CI server independently, without help. System tooling can appear as a complex task at first sight, but after reviewing the code snippets and familiarizing with the usual configuration elements, it looked like a doable work. I'm decided to study deeper these configuration aspects and try to do it by myself next time.

In the meanwhile, I continued collaborating effectively with the other project's stakeholders. Whenever I needed their support they showed their willingness to help me and push the work forward. I took advantage of the testing-oriented mindset which already existed in the KiKo Oy's staff. As I read in the *Changing Your Testing Mindset* presentation (Crispin & Gregory, 2014), collaboration and pair-working are key factors in fitting testing into agile iterations. In KiKo Oy's work culture, testing has been made a team affair that doesn't concern only testers. The fact that tester and programmers use the same tool incentivizes mutual and smooth cooperation. This positive situation makes easier to build a foundation of core agile practices. Agile team in KiKo Oy was already wise and expert on the needs of agile testing, and thus the cooperation tester-programmer was enhanced.

Creating test plans and expanding the story-related test cases was not a complex task. However, reusing and adapting legacy documentation took a considerable time. Even though I had an extensive experience in writing test plans, test scenarios and test cases, the new DictionaryPlus features represented a challenging work and required me to meet other stakeholders.

Among those stakeholders was the dictionary domain expert. She's a computational linguist who advised me on how to better test the dictionary's search modes. In turn, the graphic designer walked me through the paths of the DictionaryPlus interface. On the other hand, the business analyst pointed out the most critical elements that support our business core (i.e. dictionary selling), whereas R&D leader described the mechanics behind the new user stories.

Collaboration among our stakeholders could be labeled as good, and that was something we could take pride of at KiKo Oy. A good read on how to improve team collaboration can be found in the presentation titled *What Testers Can Learn to Collaborate Effectively with Programmers* (Crispin & Gregory, 2015). The advantage of having onboard experienced agile practitioners enabled our team to work faster and smoothly.

Dealing with the technical aspects of the last week demanded from me a high degree of technical awareness. It demonstrated that a test engineer must be able to understand the technical repercussion of introducing a new test tool into service. He or she must be able to read code alone or to pair with programmers to understand that code. It's a permanent learning process in which new skills must be used continually, incrementally and iteratively. A presentation shedding light on this subject is *Do testers have to code... to be useful?* (Crispin & Gregory, 2015).

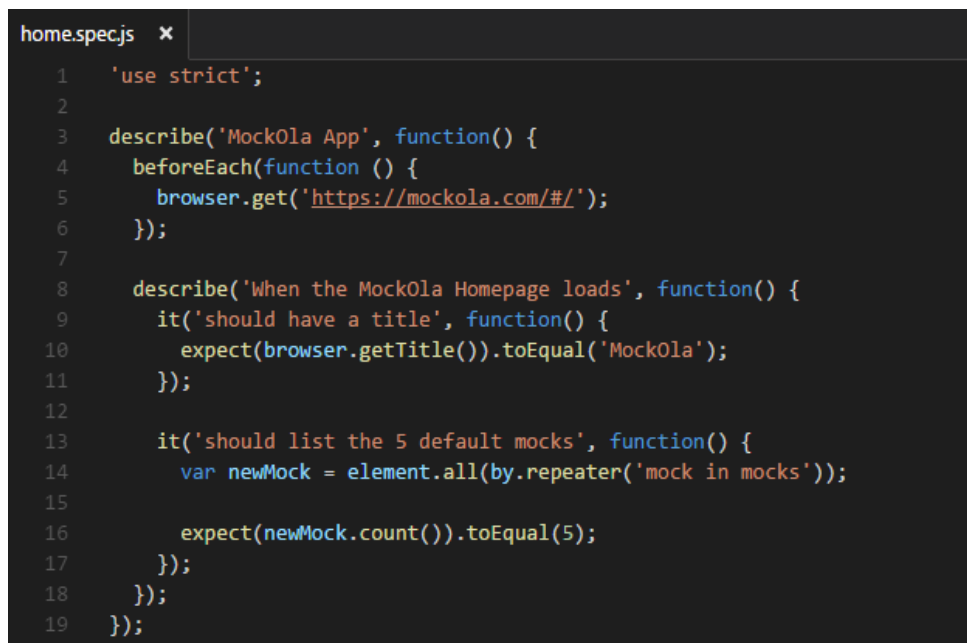
Ability for understanding, writing and refactoring code became mandatory in my professional activities as Software QA engineer. That's why I don't miss any opportunity to learn more and enhance my programming skills. For this Protractor project, my superiors at KiKo Oy had even offered to pay me some AngularJS online courses for reinforcing my coding abilities. My future personal goal after this project is to become a Software Development Engineer in Test (SDET) according to the new professional trends (SDET, 2017).

In this project, the pivotal importance of coding will be more visible in the upcoming weeks when the first test cases get automated.

3.5 Week 05

Mon 27-Mar-2017

The past week began with actual Protractor scripting. After quickly checking that I've got all the necessary tools and resources ready (Deshansh, 2017), I started scripting the automated test cases.



```
home.spec.js x
1  'use strict';
2
3  describe('MockOla App', function() {
4    beforeEach(function () {
5      browser.get('https://mockola.com/#/');
6    });
7
8    describe('When the MockOla Homepage loads', function() {
9      it('should have a title', function() {
10       expect(browser.getTitle()).toEqual('MockOla');
11     });
12
13     it('should list the 5 default mocks', function() {
14       var newMock = element.all(by.repeater('mock in mocks'));
15
16       expect(newMock.count()).toEqual(5);
17     });
18   });
19 });
```

Figure 6. shows a basic Protractor test case script

Week's work was distributed in test suites. Each day of the week revolved around a certain feature's module in the following order: (1) Monday: Authentication; (2) Tuesday: General UI; (3) Wednesday: Dictionary; (4) Thursday: Word lists; and (5) Friday: Profile page.

Games module was excluded from testing in the Protractor's project because the technology behind Games is more complex than simple HTML and requires a different test approach.

Today's task was to automate the Authentication module's test suite, specifically the test cases for logging in/out to DictionaryPlus.

As a side-note, the other part of authentication, i.e. Signing-Up, won't be automated for the time being but will be tested only manually.

Day's review:

Automating test cases for the log-in/out feature ended up being a challenging job. I must explain here that there are two ways to log in to DictionaryPlus: (a) by exercising the log-in modal, and; (b) by using Facebook social log-in that requires a special test account.

I managed to get the test logic relatively straightforward, but the difficult part was to write a script able to iterate through each test case. That iteration had to perform simple routines, such as filling in username and password to the modal fields, or simulate negative test cases (e.g. entering invalid credentials, rejecting an unregistered user, or typing in email addresses with the wrong format to trigger the expected error messages).

At the end of the day, I completed the first scripting task by working independently. All of the newly created test cases passed well in my local test machine.

Tue 28-Mar-2017

Given that Protractor could now log-in to DictionaryPlus and access its full GUI, it was time for scripting the next test suite and check that all graphic elements were right in place.

In that sense, today I focused on exercising all the elements of the general GUI. The tests of this suite could be classified as regression- rather than story tests.

Day's review:

The goal was to automate the GUI test cases specified by the test plan. Test scripts were meant to interact with each GUI's static or dynamic element.

As the first action, I mapped the locators from the Document Object Model (DOM) of the GUI. Then, I outlined the logical flow of the test cases. The GUI suite didn't require a test logic of high complexity, but it sufficed to go through the locators and make simple assertions (e.g. verify if an element was present or not).

After that, I wrote a test script exercising the navigation's links in all the modules (pages) of DictionaryPlus. Among the essential GUI elements were headers, banners, headlines, footers, navigation bar, log-in/log-out icons, language selector, dictionary search box, user avatar, and social media links.

At the end of the day and after much-dedicated work, I've got the GUI test suite up-and-running locally.

Wed 29-Mar-2017

Today's main task was to automate tests cases for the Dictionary module.

Dictionary is the core functionality of DictionaryPlus, and because of its technical particularities, the task turned to be especially complex.

Terms are searched in the multilingual dictionary supporting 45 different languages. This implied that all possible language combinations (e.g. language directions like FI-EN or FR-SV) had to be automatically tested as well.

Day's review:

I took advantage of some legacy testware that was written in the Domain Specific Language (DSL) of RobotFramework. While its algorithms and code logic were reusable, the test scripts had to be rewritten anew in JavaScript for Protractor.

Automating the dictionary searches required the implementation of complex loops. Each search action was meant to fetch test data, execute the lookup and return results. Programming those loops turned to be an assignment that surpassed my capabilities. I had to ask cooperation from a programmer colleague to get the script working as desired.

For the sake of clarity, I want to explain how this looping was expected to work:

- (1) Protractor takes a search word from a test data table
- (2) Protractor enters that word in the dictionary search box
- (3) Protractor hits the search button and triggers the search.
- (4) Once the result entry displays in the GUI, Protractor verifies that some elements (e.g. dictionary's copyright strings) are present in the resulting article.

On the other side, permuting through different language combinations required a more complex loop. But one more time, cooperating with colleagues made possible to come up with an efficient solution. This second loop iterated through the dictionary's language selector to cover all possible language combinations.

The task was completed late that day.

Thu 30-Mar-2017

Today's goal was to automate test cases for the Word Lists feature. This is perhaps the most innovative and marketable feature of the DictionaryPlus application.

As a background information, I must explain that DictionaryPlus gives users the option to create and customize word lists with their own headwords, translations, definitions and usage examples. Those word lists are stored and accessed from the cloud.

Day's review:

Today it was also necessary to request support from one of our skilled programmers. After some hours of pair-coding, the output was a large and complex script intended to cover the process of creating and populating word lists.

However, the test script behaved erratically at the first attempts; it was prone to break due to latency and timing issues. Finally, after doing some tweaks our programmers had the script working more or less reliably in the local test machines.

Yet all the involved programmers agreed that a more robust script must replace soon the current one. In consequence, I arranged for next week a code review session with programmers. The goal was to evaluate other solutions or refactor the current script.

While I felt satisfied with the work progressing according to the roadmap, the last tasks had revealed my shortcomings in programming. One more time it was made clear that I must improve my programming skills.

Fri 31-Mar-2017

Automating the test cases for the Profile page –the last DictionaryPlus module— was scheduled for today.

The profile page is a separate module in which the user's information is stored. User's scores and awards are visible at glance.

I expected to fulfill this task by working independently because the module doesn't contain intricate functionalities and its locators were easy to identify and exercise.

Day's review:

This time I was able to cope with the task alone. I wrote a straightforward script asserting the presence of the main elements (e.g. user's score, user's statistics, user's titles, user's engagement level, user's awards, user's achievements, stickers, pins, and thumbnails).

Test script worked well in the local test machine.

I also spent a few hours refining the Page Object Model (POM) repository. During test execution, this POM repository is intensively referenced by Selenium WebDriver because it contains the DictionaryPlus locators and the test methods (Guru99, 2015).

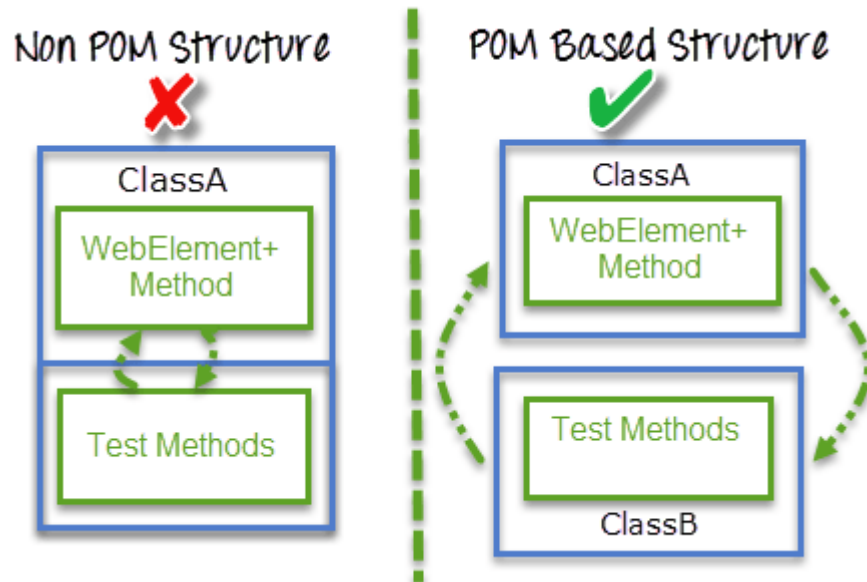


Figure 7. Page Object Model (POM) approach helps make the code more readable, maintainable, and reusable.

Weekly analysis

The past week was for me one of the most important in the project's lifecycle. I saw myself writing automated test cases in the new Protractor framework. And that was my great expectation since the beginning of this project.

But despite being a user-friendly tool, Protractor showed that it can be challenging as well.

Firstly, Protractor was still a relatively new tool in the testing scene. As I wrote these lines, there was not even a Protractor cookbook available in the market. Protractor's topics and best practices could only be found on the Internet. If a test engineer needed advice on how to resolve a certain scripting issue, he/she had to browse dozens of pages with thousands of different –and sometimes contradictory– views that must be filtered out until finding the right solution (EggHead.io, 2017).

Secondly, Protractor's community is constantly pushing upgrades. Each latest version may either bring new features or deprecate functionalities that test engineers are already familiar with. In consequence, learning curve increases unexpectedly.

Thirdly, Selenium WebDriver undergoes upgrading as well. Some scripts failed because the code semantics had been modified or even deprecated. In those cases, the scripts had to be rewritten anew.

It's important to explain that according to Protractor conventions, each test suite might include more than one file, like the following:

- A test suite file with the suffix ***spec.js*** (e.g. *generaltests.spec.js*) containing all the test case scripts.
- A POM file containing the locators and methods which were wired with the test case scripts.
- A configuration file, usually named as ***conf.js***
- Also, if test cases require test data, then it's necessary to import data tables formatted as Comma Separated Values (CSV).

Another sensitive task last week was to organize the POM repository (QAShahin, 2014). Enhancing GUI locators and methods to match the upcoming test scripts was a time-consuming work, but it had to be accomplished before writing more test suites. POM helps test engineers when writing scripts that exercise the same web elements repeatedly. Scripts maintenance becomes easier as well because any modification done in the POM files will be automatically reflected in the affected test scripts.

An important aspect of Protractor's configuration is to define the test browsers (ProtractorTest, 2017). Internet Explorer 11 (IE11) was set as the standard browser for automated testing. Reasons behind this decision are based on the analytics of our customer's browsers distribution. KiKo Oy's largest customers are Finnish state organizations, meaning that ministries, universities, and schools share a Common Operating Environment (COE) in which Windows 8 and Windows 10 are the standard platforms, whereas IE11 is the default browser.

Scripting on last week demanded accuracy and a problem-solving mindset. Some test suites required deep approach and complex coding. For testing some DictionaryPlus forms, drop-down menus, dynamic lists and dynamic graphic elements, I frequently resorted to loops referencing test data.

Below is a recap of what has been done last week:

- Test cases were configured to run against Internet Explorer 11.
- Repositories for POM were created and populated
- Five test suites have been written (i.e. Authentication, General UI, Dictionary, Word lists, Profile page)

As a personal thought, I can comment that this project was also testing my real capabilities as an automation engineer. I confirmed that problem-solving mindset and proactive attitude are key factors in this field. Sometimes it's necessary to look for solutions outside the office and to contact advisers in forums or specialized discussion groups. Because AngularJS and Protractor were all the time being subject of upgrading, it's mandatory to keep up with the innovation's pace.

3.6 Week 06

Mon 03-Apr-2017

This week began with the execution of Protractor tests in the real development environment. The first five suites were ready for running on the CI server.

The task for today –and for the rest of the present week– was to execute the automated test scripts and debug all the errors that may arise.

Day's review:

Before starting the actual test execution, I check-listed that CI server and test configuration were clear. Additionally, test suites were re-run locally to verify that everything is fine.

In order to avoid disrupting the CI work of programmers, we in the R&D team agreed that afternoons would be the best timeframe for executing the trial tests. Reason for that was that programmers merge the bulk of their pull-requests in the morning.

Around 13:00 o'clock, I merged my local scripts to the master branch in CI. I used GitHub Desktop for version control and source code management

At the first run, a considerable number of Protractor's tests failed. Several error messages were logged in the CI console ranging from "locator not found" to "timed-out". This was frustrating because the test suites had worked well locally, and I expected them to behave similarly in the CI job.

Next step was to debug the failed test scripts. I customized longer waits to the affected test actions and optimized the HTML locators until everything looked fine in the script. Then I executed locally the complete set of test suites, and they all passed.

Finally, I merged the updated suites to master branch hoping for better results. However, the CI job failed again. Although this time fewer tests failed, it caught my attention that among them were the same tests that I just had fixed.

I spent the rest of the day debugging the test scripts further.

Tue 04-Apr-2017

Today's main task was to keep on debugging the test scripts until getting them working in CI.

Day's review:

Failed tests required a closer examination. I repeated a few times the pattern of yesterday, in other words: running fixed tests and debugging failed tests. I tried to get this issue sorted out by working independently but to no avail.

In the case of GUI elements being reported as "not found" by Protractor, I tried enhancing the identification of AngularJS locators. I found a valuable tool known as *Elementor* available for free downloading (Dominguez, 2016). *Elementor* allowed me to easily identify alternative AngularJS locators. By selecting more robust locators a tester can write more reliable scripts.

However, despite of my best analysis and efforts the tests in question continued failing. I decided to ask advice from colleagues. It seemed that I was missing an important and characteristic property of AngularJS, that is: Asynchronous calls and workflows (The Net Ninja, 2016)

I had to invest some hours watching an AngularJS tutorial focusing on the concept of *Promises* (Red Ape EDU, 2013). It turned out that my approach for handling waits and sleeps was essentially wrong. The lesson was the following: when scripting tests for asynchronous AngularJS you must use *Promises* instead of just adding simple Selenium waits.

After replacing the Selenium waits by *Promises* in the script, I managed to get more tests passing. No need to say that using *Promises* became a matter of convention in my work.

I spent the rest of the day refactoring the test suites and adding *Promises* wherever applicable.

Wed 05-Apr-2017

Today's two main tasks were:

- Continuing debugging and fixing the failed test scripts
- Holding a code review session with two programmers (As a reminder, last week I arranged a meeting for enhancing some loops in the scripts)

Day's review:

I continued debugging and fixing test scripts. This work was a time-consuming activity and demanded many hours. Keep in mind that each CI job takes around 10 minutes including other integration routines, such as executing hundreds of unit tests against the master branch. I also had to wait for the most convenient moment before merging my local branch to master; not observing the latter could cause waiting queues and bottlenecks in the pipeline of pull-requests.

At the end of the day, I managed to get nearly all the test suites passing. It was also necessary to spend some time updating the POM and methods files.

In what concerns to the code review session, I actively cooperated with two programmers optimizing the scripts. Greatest challenge here was to optimize two loops iterating through the word lists. After two hours of pair-programming, we came up with an efficient solution. Now we have in place three middle-complex loops which interact well with the word list feature. This solution behaved well even in the CI job demonstrating to be a robust fix.

Tasks for today were accomplished after a long work day.

Thu 06-Apr-2017

As an ongoing task, today I continued running test cases and fixing failed scripts.

Also, I planned a meeting with R&D leader who wants to brief me about a new feature story. According to the Test-Driven Development (TDD) methodology, an automated test case must be in place running even before the feature was written. That's why I was asked to script tests prior to the feature's implementation.

Day's review:

At the morning, I debugged the failed scripts and refactored the ones which pass. Again, some locators and loops were originating challenges. I had to watch more videotutorials and to search tips in Protractor's forums. Finally, by the end of the day, I managed to get all the 35 test cases (contained in the five test suites) working and passing.

In the afternoon, I discussed with our R&D leader about the new feature to be added to DictionaryPlus. It's about a dynamic graphic element that will be included in the GUI. Based on the feature's specifications, I wrote a test script including simple assertions and some click-on actions. The actual feature was expected to be merged tomorrow. As long as the feature is not present and working in the GUI, the new test script will be failing.

One important extra task for today was to update the POM and methods files.

Also, I spent some time taking care of some chores. I updated the status of test cases in our Test Management tool. This basically means to mark the tests as "pass" or "fail" in our logs for other stakeholders to see.

Fri 07-Apr-2017

For today I planned to write a few more test cases trying to expand the test coverage. Additionally, some chores had to be taken care of.

Given that the planned tests didn't contain complex functions or brittle locators, I expected to achieve it by working independently.

Day's review:

As expected, I was able to cope with the tasks by working alone. I had written straightforward scripts including basic functions like clicks and assertions. Test coverage was increased from 35 to 45 tests distributed across the five test suites.

In addition, I continued developing the test script for the new GUI feature. Around 15.00 the feature's most stable version was merged to master, and automatically it was exercised by my test script. Seeing the "pass" message in the console was a satisfactory result because now I had 45 scripts running efficiently against the features under test.

Results could be assessed in the following perspective:

- Regression tests were up-and-running

- Story (new feature) test was up-and-running
- TDD methodology was correctly applied by writing automated tests prior to actual developing of a new feature.

Finally, among other chores, I updated the test manager logs and shared the test execution reports with the stakeholders.

Weekly analysis

The past week brought new insights about Protractor performance within an agile environment. After long hours of debugging and fixing scripts, it was very satisfactory to see the complete set of test suites passing in the CI jobs.

At that stage of the project, I could realize how valuable are the technical courses I took during my studies in Haaga-Helia University. The acquired knowledge of several technical aspects (e.g. programming, software projects, Scrum methodology, database design, and multicultural teamwork) allowed me to better organize the project's implementation.

The core of my expectations had been already fulfilled. I had delivered a fully functional automation framework which was built around Protractor and Selenium WebDriver, let alone that I have created the necessary documentation infra and the test artifacts.

However, the road had also shown some obstacles. During last week, it became clear that my AngularJS knowledge was far from being optimal to match a senior automation level. AngularJS –as JavaScript does– includes the peculiar characteristic of asynchronous invocations with *promises* (Hathi, 2015); this was a totally new topic to me and needed to be examined further. My test tactic of using simple Selenium waits instead of *promises* proved to be wrong and resulted in a waste of time and effort. I attempted to apply the same linear approach that I used with RobotFramework while missing that asynchronicity makes a significant difference between both systems. Due to asynchronous calls, my initial test cases were prone to fail particularly when one test action had other test action as pre-condition. This means for example that Protractor tried asserting the presence of the “*Welcome, you're logged in*” message even before the logging-in test was finalized. After receiving feedback and advice from colleagues, I learned the lesson about *promises* and subsequently changed the scripting approach. This change made tests more robust and less prone to fail. No need to mention that this approach was immediately adopted as a best practice in our test strategy.

For helping me improving my AngularJS skills, KiKo Oy has approved to pay me an advanced AngularJS training course in Udemy.com online academy.

Another important aspect that I observed last week was the interaction between Protractor and CircleCI. It must be noted that each new CI build takes around 10 minutes. During that time, programmers can't merge their code changes. In a normal work day, our programmers merge numerous changes to master branch, and each merging triggers a new CI job. In order to prevent any bottleneck's risk, the team's workflow must be kept agile and avoid unnecessary delays in the pipeline. One possible solution would be running the automated regression tests overnight when the CI jobs amount is minimal.

Using a valuable tool like code review sessions was a right step towards scripts optimization. By pair-reviewing code with other programmers, I could deeply analyze the script's structure and its efficiency. Scripts can be optimized or can be rewritten anew. If necessary, test engineer and developers can pair-program a new script solution. These hands-on sessions were a great opportunity for me to learn new coding tricks.

As mentioned in last week journals, I had Protractor executing two distinct types of tests, namely: (a) regression tests; and (b) story tests. As an explanatory note: "regression tests" are the ones exercising the features already deployed in previous releases, whereas "story tests" are the ones exercising a new feature being built along the sprint. According to agile principles, story tests must be completed within the sprint's duration. In DictionaryPlus context, each sprint's duration was set to one week.

Test-driven development (TDD) continued being the most advantageous methodology for KiKo Oy's needs. In DictionaryPlus software engineering, TDD is applied at both testing levels: (a) Unit (component) testing, and; (b) End-to-End (acceptance) testing. My mission was to script automated acceptance tests, and it might be somehow overlapping with the concept of Acceptance Test-driven Development (ATDD). However, consensus in our R&D team was that this flexible approach fits best our testing needs.

3.7 Week 07

Mon 10-Apr-2017

Week's main goals were to evaluate the potential Protractor's scalability and to measure the project's results. It's worth to remind that Protractor had been already evaluated before the project's start; and now it's time to evaluate it again after the introduction's implementation (Johnson, 2007).

In that sense, I planned to keep a preliminary meeting with the R&D team to both give and receive advice on how to approach the evaluation in question.

Additionally, today I planned to write a few more automated test cases for expanding the test coverage. I decided to challenge my coding skills and write some complex scripts including If/Else decision flows.

Day's review:

As an explanatory note: the upcoming evaluation will not solely assess Protractor tool, but also the other entities belonging to the test framework.

In a meeting with the R&D team, we all together established the indicators to be included in the evaluation report. Besides some key metrics listed in the standard guides for software evaluation (Illes, Hermann, Paech & Rückert, 2014), the R&D developers underlined that two important trials must be conducted:

- Protractor should work as a test tool for non-angular applications
- Protractor should be compatible with SauceLabs virtual platforms and browsers

The aforementioned trials demanded the preparation of some shortened test suites. Stakeholders expected a brief report on these assignments by the end of the week.

At the same time, I continued developing a few test cases with the purpose of enhancing my scripting skills. At the end of the day, I got some complex If/Else scripts working fine in the CI job.

Tue 11-Apr-2017

Today's main task was to assess the reliability of Protractor interacting with non-angular applications.

As a background information: KiKo Oy develops other online applications outside of the AngularJS stack. KiKo Oy stakeholders wanted to clarify how well Protractor would behave when testing those non-angular applications.

Day's review:

First, I wrote a basic test suite to be run locally against a KiKo Oy's non-angular application. The test suite contained simple test cases, such as logging-in, navigating through sections, and making a successful dictionary search.

As I expected, test execution failed against the non-angular website. Error messages in the console clearly stated that “no AngularJS could be found”.

Following instructions found in Protractor forums (Esteva, 2014), I set the test script to ignore AngularJS synchronization. There’s a few special JavaScript commands for that purpose. Depending on the requirements, a test engineer can select the most suitable command. For my task, I decided to employ the most usual of those commands, as depicted below:

```
beforeEach(function() {  
    return browser.ignoreSynchronization = true;  
});
```

Figure 8. shows a JavaScript command to ignore synchronization in a non-angular website.

The above snippet forced Protractor to stop waiting for AngularJS to load and finish its tasks. Then I updated my scripts to configure Protractor not to be excessive smart about the non-angular website.

At the end of the day, I had the Protractor test cases passing correctly against the non-angular website.

Wed 12-Apr-2017

Today’s main task was to ascertain if Protractor tests could be integrated to SauceLabs virtual platforms and browsers (Sypolt, 2016). Subtasks were broken down as follows:

- Opening a free trial account in SauceLabs
- Configuring integration between SauceLabs, Protractor and our local test machines
- Configuring the virtual platforms and browsers
- Running tests
- Evaluating virtual browsers’ performance and quality

Day’s review:

I opened a test account in SauceLabs cloud service granting me a 30-days free trial. SauceLabs offers a very useful collection of tutorials and guidelines. After some hours

reading how to do the integration, I managed to get SauceLabs wired with Selenium and my local test machine (NdManWar, 2017).

However, configuration proved to be not a simple task. It was necessary to make additional tweaks until getting the tests running. I asked advice from developers who helped me with the CI platform's configuration.

As a pre-condition for parallel test execution, I had to specify an array of platforms and browsers. I selected as platforms Windows 8, Windows 10 and Mac OS latest version, whereas Internet 11, Edge and Safari latest versions were chosen as browsers. In the mobile side, I selected Android (Chrome) and OS (Safari) latest versions. This selection was made in accordance with DictionaryPlus' browser compatibility matrix.

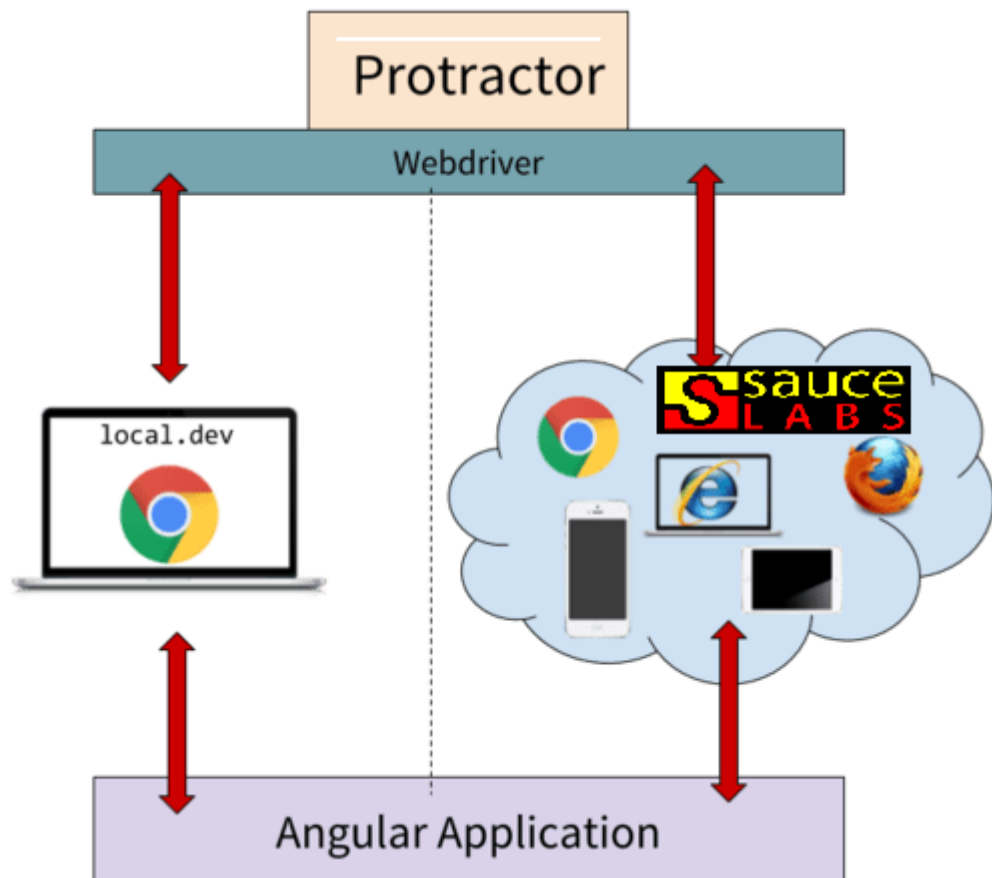


Figure 9. shows how Protractor interacts with SauceLabs virtual browsers

During the first executions, the test suite ran erratically due to virtual browsers' poor performance (Gouchenour, 2016). SauceLabs browsers –when no script optimization is done– might render very slowly causing the tests to time-out and fail. More and better configuration was needed to get the performance problem sorted out (Gouchenour, 2017).

On the other hand, SauceLabs offers great advantages like a wide scope of platforms and browsers, and the ability to run multiple tests parallelly.

Today's task was fulfilled, but some questions remained open.

Thu 13-Apr-2017

Today is the last day of this week, and I had to share my findings with the other stakeholders.

Day's review:

I issued two brief reports on the requested trials, meaning: Protractor and non-angular applications, and Protractor and SauceLabs, respectively.

Details of the above trials were discussed in a meeting with the rest of the stakeholders.

R&D members were satisfied with the trial about non-angular applications. This aspect will be counted in for finally establishing Protractor as the standard test tool in KiKo Oy. A versatile tool like Protractor will allow us to extend automated testing beyond the DictionaryPlus web stack.

Regarding SauceLabs trial, the browser performance was the main point of discussion. In testing-related terminology, such slowness issue can be labeled as a test risk. However, during the meeting, it was revealed that one of our developers had worked with SauceLabs in his previous job, and thus he offered his know-how for optimizing the test scripts (Gouchenour, 2017).

A third evaluation report included diverse aspects, such as (a) support for data-driven test cases (Vasanth, 2015); (b) defect management tool; (c) tests maintenance; (d) usability of the framework; (e) manuals and guides; (f) online user communities; and (g) up-front and indirect costs.

Other important key indicators have addressed secondary questions, such as (a) amount of tests that can be included in a test suit; (b) strived number of tests needed for proper coverage; and (c) amount of CI iterations that should be run per day.

Fri 14-Apr-2017

National holidays.

Weekly analysis

As described in the last journals, the main week's goal was to evaluate the overall potentiality of the new test automation framework.

I also provided some figures about Protractor's usage in the AngularJS web stack. Most important indicators, such as costs, maintenance, and time- and resources consumption were of special interest for the stakeholders.

Assessing Protractor ability to interact with non-angular applications was of key importance for the R&D team. Even though this ability had been verified by reading Protractor's literature, the stakeholder's expectation was to see it working in a real KiKo Oy's development environment. Evaluations demonstrated that Protractor can flawlessly run tests against other KiKo Oy's non-angular products and services.

Another evaluation subject was the interoperability between Protractor and SauceLabs virtual platforms and browsers. Evaluation's outcome was positive and demonstrated that Protractor's compatibility was expandable to virtual mobile platforms (i.e. tablets and smartphones). SauceLabs advantage is to run test suites parallelly and to cover the full scope of DictionaryPlus browsers. Until now testing with mobile devices has been done manually, which is a time-consuming activity that usually requires extra testers.

Another SauceLabs side-benefit is costs saving in hardware. Acquiring and maintaining a collection of physical test devices (e.g. DictionaryPlus required 11 different devices of desktop- and mobile type) represent money expenses, let alone the man-hours spent in upgrading its software configuration. Now thanks to SauceLabs, testers have at their disposal all the necessary platforms in the cloud.

However, there's still a few challenges with the performance of SauceLabs virtual browsers. Frequently their slow rendering has caused the tests to time-out and fail. After a brief research in the Protractor forums, I discovered that this issue was not unfamiliar to the tester community and that its solution required special configuration and code's optimization.

Another peripheral issue was to rewrite the test scripts to be run in the small-sized screens of virtual tablets and phones. It's because locators change their positions in the responsive GUI when displaying in mobile screens, whereas locators in desktop screens usually keep a fixed position in the GUI.

SauceLabs is a premium service. Before purchasing a license, the Protractor's stakeholders will decide yet which account type meets best our test needs.

On Thursday –last day of the week— I worked hard preparing the final reports on other Protractor’s metrics. This task allowed me to dive deeper into performance indicators and standard evaluation criteria, such as direct/indirect costs, maintenance, training possibilities, and time management. Here’s where I realized that metrics play a key role in any software-related project. I had reviewed those concepts already during my studies in Haaga-Helia University, but I didn’t have any practical experience with them until now.

In the meanwhile, Protractor has been running flawlessly in the CI platform every time our developers have merged new code to master branch.

3.8 Week 08

Mon 17-Apr-2017

National holidays.

Tue 18-Apr-2017

The first thing to do this week –the end week of Protractor’s project— was to sit together with the whole stakeholder’s group and take an official decision on the project’s outcome.

That final decision was going to imply other administrative assignments, such as managing documentation and legacy files.

Day’s review:

At 10.am the stakeholder’s group held a decision meeting on Protractor’s introduction. The CEO of KiKo Oy was present as well. Each stakeholder had already reviewed all the reports that I had submitted last week.

After having a brief discussion and keeping a round of questions, the final decision was taken: Protractor’s introduction project had produced all the necessary proof and data to make an informed decision on the new automation tool. It was unanimously decided to establish Protractor as the new test automation tool for the DictionaryPlus web stack.

CEO has congratulated the involved stakeholders and thanked especially to me for the smooth conduction of the project. CEO has also committed himself to share the project's details with the Management Board in our parent company.

This decision implied a few action points to do, such as dismantling RobotFramework and archiving its related testware in a specific repository. After discussing some final Protractor's topics, the R&D developers and I planned the distribution of the above-mentioned tasks.

For this day, no additional task was logged.

Wed 19-Apr-2017

As agreed yesterday at the decision meeting, RobotFramework had to be retired from our R&D environment. This measure included two clearly distinguishable tasks

- Withdrawing RobotFramework from the CI server
- Removing other RobotFramework-related instances (e.g. Selenium, RIDE, and its reporting tool)

Day's review:

My participation in the above-described tasks was only marginal. Main work was overtaken by our R&D leader who is an expert configuring the CI platform.

As an explanatory note: RobotFramework had to be dismantled because two test automation tools shouldn't unnecessarily coexist in the same CI platform. Keeping Protractor and RobotFramework doing the same work would be a senseless decision resulting in a waste of time and resources.

It took a couple of hours until RobotFramework was completely removed from the CI platform. A lot of complementary verification was done to assure that nothing has been broken in the process.

It also took some hours to remove RobotFramework dependencies from the R&D test environment. Selenium configuration was cleaned out from the CI server, whereas RIDE (RobotFramework Integrated Development Environment) was discontinued. Also, the associated report-generation tool was uninstalled.

After verifying that nothing has been broken by the last changes, I e-mailed a notification to the other stakeholders.

Thu 20-Apr-2017

Today main task was to manage the legacy artifacts from the RobotFramework test environment.

Legacy artifacts can be understood as the whole testware that is used in a specific test environment. It can be test code, scripts, documentation, test data, associated applications, etc.

RobotFramework had been used in KiKo Oy for over five years, and during that time the QA personnel had produced a large number of legacy artifacts. These artifacts had to be correctly classified and archived in a specific repository.

Day's review:

Today I worked along with our IT specialist. We held a brief meeting to plan the management of the legacy artifacts.

The task became a time-consuming activity. It was necessary to browse through several files scattered all over the directories. After some hours, I compiled and classified over 100 artifacts, such as test scripts, specifications, test plans, test studies, data tables, guidelines, etc. Some of them were obsolete and others were fairly new or still under works.

At the end of the day, all legacy artifacts were backed-up and archived in a special Intranet repository. They remained accessible to every member of the R&D if eventually some artifact must be revisited.

The last subtask was to delete all legacy items from the development environment, particularly from GitHub and Project Management tool.

A last complementary verification was done on the cleaned system. Finally, I reported to other stakeholders that RobotFramework had been officially retired.

Fri 21-Apr-2017

Today was the official end of the Protractor's introduction project. All important tasks and decisions had been already accomplished. At this point, only one task remained open, namely:

- Making an inventory of the test hardware (e.g. tablets, phones and desktop computers) which became redundant after SauceLabs adoption

As a side note, it must be stated that the former RobotFramework environment was capable of supporting SauceLabs as well. However, for some reason, the integration was not advanced and certain browsers were still tested manually with physical devices.

Day's review:

Today I required the cooperation from our IT specialist one more time. It was not a hard day because there were not many devices to be accounted for.

We made an inventory of all the physical devices utilized for manual testing. Five laptops among Windows- and Mac OS types, and three tablets (Android and iPad) and three smartphones (Android and iPhone) were listed, re-labeled and stored in the general shelf of the IT department.

Those devices can eventually be used for testing or any other activity, but they won't be upgraded or configured further. Also, the new Quality Assurance policy states that no additional device will be purchased for testing purposes. All members of the R&D were encouraged to shift to SauceLabs cloud service. Developers and testers must use SauceLabs virtual browsers for debugging their code or testing (manually and automatedly) DictionaryPlus.

Final reports and e-mails were sent to stakeholders declaring that the eight-week Protractor's introduction project has come to an end.

Weekly analysis

It was the final week of the Protractor's introduction project. After eight weeks of hard and dedicated work, the main goal had been achieved and all the stakeholders were satisfied with the results.

As described in the last week's journals, the tasks were more of administrative nature rather than technical.

Last week's highlight was the final decision meeting on Protractor's acceptance. Even though I was confident that Protractor's approval was a matter of fact, it was a satisfactory moment when the CEO and other stakeholders validated my project's idea and congratulated me for the efficient conduction of the project.

All the reports that I prepared before the meeting had helped the CEO and stakeholders to take a favorable final decision on Protractor. Metrics and other figures were key factors when assessing the project's results.

At the final meeting, I honestly expected some critical feedback on the project, or some opinions favoring RobotFramework over Protractor in what's called "resistance to change". However, all the stakeholders, as well as non-involved personnel, expressed enthusiastically their support to the new test automation tool.

Removing each of the RobotFramework-related dependencies was a hand-made endeavor. It was necessary to review each configuration file at CircleCI before proceeding to clean.

Similar difficulties were encountered when uninstalling and removing other software that won't be needed in the new Protractor's framework.

Managing test devices which were made redundant by SauceLabs was not a complex task, but it demanded time and cooperation. Here again, I needed the support of our IT person because the task was basically about hardware management.

Acquiring and maintaining test devices represented appreciable expenses in money and man-hours. Each test computer costs around 600 euros, whereas new smartphones can cost between 500-700 euros. Tablets were in the price range of 200-350 euros.

License's costs must be calculated in. In the case of Windows desktops, KiKo Oy had to pay for the operative system licenses. DictionaryPlus's cross-browser support specified at least three Windows versions (i.e. Win7, Win8, and Win10).

Last but not least, test devices required frequent software upgrade and configuration. This work translates into precious man-hour costs.

Now, by using SauceLabs virtual browsers, KiKo Oy can reduce some testing-associated costs.

I'm glad that the Protractor's introduction project has been satisfactorily completed. In the last eight weeks, I have gained a valuable hands-on experience on test project management. This experience complements well with the theoretical knowledge that I learned in Haaga-Helia University. Both, theory and experience leverage my professional goals.

4 Discussion and conclusions

When I started this project eight weeks ago, I was fully aware that my skills and knowledge were going to be challenged. Whilst I had gathered a solid basis in automated testing during my professional trajectory, the thesis-project goals were beyond my real technical abilities. I knew that an intensive hands-on learning cycle was waiting for me.

My ICT studies in Haaga-Helia had provided me with the basic capabilities for launching the Protractor's project. On the technical side, I learned about Java programming, databases, hardware and networks. On the organizational side, I got familiarized with Scrum, Software Project management, communication, and documentation, among other disciplines. All the aforementioned know-how served me as an optimal launchpad for bigger challenges out of the classroom.

Since the project's start, I had identified programming as the area with the most room for improvement. Being not an actual programmer, I was compelled to expand my coding skills while conducting the project. Consequently, I read books and signed-up to online courses. Also, videotutorials and queries in specialized forums were key in my learning endeavors.

But the main sources of advice were my colleagues from the R&D team. Gathering new knowledge and applying it to real life situations –as for example writing test scripts— was a productive and satisfactory activity.

However, despite of my learning advances along the thesis-project, it turned crystal clear that expanding my programming skills would be a never-ending task. Every day brand-new technologies are popping up in the market which translates into new test tools and new technical approaches and new best practices. A senior test engineer must be capable of understanding and use –if required— those modern technologies.

I estimate that my knowledge has increased around 25%, which is a considerable upgrade. From being a junior test scripiter, I have evolved into a test engineer that now is able to write

scripts of medium-high complexity. Besides my previous knowledge of Java and Python languages, I have gained a deep understanding of AngularJS which appears to be a very popular technology among web developers.

Along the project, I had the opportunity to actively participate in the scrum planning and to have a say in the agile testing practices for DictionaryPlus. I was given the responsibility to adapt the TDD methodology rules –as much as it’s theoretically permitted– to fit our needs. As a result, an acceptance testing layer was aggregated to TDD.

Besides technical skills, the importance of communication is pivotal when conducting a software-related project. At the initial stage, my abilities as project manager were very limited. However, as project moved forward I learned to effectively utilize communication levels and channels in the best viable way. Other stakeholders were acting as my internal customers and were constantly requiring updated information. Requests, reports, and feedback had to be delivered on time to keep the work pace and to avoid delays or misunderstandings. Smooth interaction and cooperation with other KiKo Oy’s specialists were key factors for the project’s success.

Another factor under the spotlight was time management. Project’s roadmap had a tight schedule –perhaps too tight— with milestones serving as precondition for other milestones. Any unexpected delay would have created bottlenecks and put the roadmap in risk. I utilized time-boxes as a strategy for time’s management. Time-boxes facilitated the prioritizing of the project’s activities but required careful analysis to estimate the stakeholders’ response times and to forecast the prerequisite tasks’ completion date. Also, when I scheduled meetings or distributed tasks, I always booked the most convenient slot in the stakeholders’ calendars to avoid being disruptive.

Writing a diary-type thesis was a very innovative approach to me. However, my preexisting experience with documentation helped me to better report my activities in the journals. Diary format made easier to follow the status of ongoing activities and to analyze the work done in the week. A diary also constitutes a clear time log that can be shared with stakeholders.

I also learned how to advance concepts for new projects. I believe that by proposing my ideas with solid grounds and through the right channels, I can incentivize a change inside the organization. Decision makers feel positive about innovation proposals, especially when the intrapreneurs show enthusiasm and courage to validate their ideas. Organizations want to see beneficial, feasible ideas being driven forward by encouraged people. For me, this is a learning that I definitively will use in my future professional challenges.

Positive feedback from CEO and colleagues was a rewarding source of motivation. During my learning stages, I welcomed critics and recommendations because I consider them important means for improvement. Furthermore, I appreciated the trust and the encouraging feeling present in my work environment. Congratulations and words of encouragement when due are a driving force for any employee.

My plans for future development are focused on becoming a Software Development Engineer in Test (SDET) which is the professional line being followed by numerous test engineers of the Finnish test automation community.

Software testing is a professional area with a broad scope of specialization fields. Besides automated testing, there are other exciting paths to consider. I'm also interested in becoming a specialist in Penetration testing (i.e. testing of security weaknesses in software). That's how the future looks like in my area, and now I feel better prepared than before for taking on these new challenges.

References

- Anderson, Chris. *How to give a killer presentation*. Harvard Business review. 2013. URL: <https://hbr.org/2013/06/how-to-give-a-killer-presentation>. Accessed: 11-Mar-2017
- CircleCI. 2017. URL: <https://circleci.com/>. Accessed: 20-Mar-2017
- Crispin, L. & Gregory, J. *Changing Your Testing Mindset*. Booster Conference, Bergen, Norway. 2014. URL: <https://www.slideshare.net/lisacrispin/changing-your-testing-mindset-booster-conference-bergen-norway-2014>. Accessed: 25-Mar-2017
- Crispin, L. & Gregory, J. *Do testers have to code... to be useful?*. 2015. URL: <https://www.slideshare.net/lisacrispin/do-testers-have-to-code-to-be-useful>. Accessed: 25-Mar-2017
- Crispin, L. & Gregory, J. *What Testers Can Learn to Collaborate Effectively with Programmers*. 2015. URL: <https://www.slideshare.net/lisacrispin/what-testers-can-learn-to-collaborate-effectively-with-programmers-others>. Accessed: 25-Mar-2017
- Crispin, Lisa & Gregory, Janet. *Agile testing*. Addison-Wesley. 2009.
- Cucumber. 2017. URL: <https://cucumber.io/>. Accessed: 09-Mar-2017
- Deshansh. Protractor Testing Tool for End-to-end Testing of AngularJS Applications. 2017. URL: <http://www.softwaretestinghelp.com/protractor-testing-tutorial/>. Accessed: 27-Mar-2017.
- Dominguez, Andres. Elementor. *In GitHub*. 2016. URL: <https://github.com/andresdominguez/elementor>. Accessed: 04-Apr-2017
- EggHead.io. Learn Protractor Testing for AngularJS. 2017. URL: <https://egghead.io/courses/learn-protractor-testing-for-angularjs>. Accessed: 01-Apr-2017
- Esteva, Santiago. Protractor - Testing Angular and Non-Angular Sites. 2014. URL: http://ng-learn.org/2014/02/Protractor_Testing_With_Angular_And_Non_Angular_Sites/. Accessed: 11-Apr-2017
- Farrell-Vinay, Peter. *Manage Software Testing*. Auerbach. 2008
- Gouchenour, Phil. Running Protractor Tests. 2016. URL: <https://support.saucelabs.com/hc/en-us/articles/225100507-Running-Protractor-Tests>. Accessed: 12-Apr-2017

- Gouchenour, Phil. Setting Up AngularJS and Protractor Tests. 2017. URL: <https://support.saucelabs.com/hc/en-us/articles/225100407-Setting-Up-Angular-js-and-Protractor-Tests>. Accessed: 12-Apr-2017
- Gouchenour, Phil. Tips for Lean, Speedy Tests with Sauce Labs. 2017. URL: <https://wiki.saucelabs.com/display/DOCS/Tips+for+Lean,+Speedy+Tests+with+Sauce+Labs>. Accessed: 13-Apr-2017
- Guru99. Page Object Model (POM) & Page Factory in Selenium: Complete Tutorial. 2015. URL: <http://www.guru99.com/page-object-model-pom-page-factory-in-selenium-ultimate-guide.html>. Accessed: 31-Mar-2017
- Hathi, Rajeev. Understanding Asynchronous Invocation With Promise Using AngularJS. 2015. URL: <https://www.webcodegeeks.com/javascript/angular-js/understanding-asynchronous-invocation-promise-using-angularjs/>. Accessed: 08-Apr-2017
- Huiskonen, Jukka. *Designing automated module testing framework*. Haaga-Helia, 2011. URL: http://www.theseus.fi/bitstream/handle/10024/34468/Jukka_Huiskonen.pdf?sequence=1. Accessed: 25-Mar-2017
- Illes, T., Herrmann, A., Paech, B. & Rückert, J. Criteria for Software Testing Tool Evaluation. A Task Oriented View. 2014. URL: https://www.researchgate.net/publication/228345981_Criteria_for_Software_Testing_Tool_Evaluation_A_Task_Oriented_View. Accessed: 10-Apr-2017
- Jain, Naresh. ATDD - Acceptance Test Driven Development. 2008. In *Slideshare*. URL: <https://www.slideshare.net/nashjain/acceptance-test-driven-development-350264>. Accessed: 09-Mar-2017
- JetBrains. Download WebStorm. 2017. URL: <https://www.jetbrains.com/webstorm/download/#section=windows>. Accessed: 15-Mar-2017.
- Johnson, David W. How to evaluate testing software and tools. 2007. URL: <http://searchsoftwarequality.techtarget.com/tip/How-to-evaluate-testing-software-and-tools>. Accessed: 10-Apr-2017
- McConnell, Steve. *Software Project Survival Guide*. Microsoft Press. 1998.
- NdManWar. JS-Protractor-Selenium. In *GitHub*. 2017. URL: <https://github.com/saucelabs/sample-test-frameworks/JS-Protractor-Selenium>. Accessed: 12-Apr-2017
- Osterhage, Wolfgang W. *IT Quality Management*. Springer-Verlag. 2015

ProtractorTest. Setting Up the Selenium Server. 2017. URL: <http://www.protractortest.org/#/server-setup>. Accessed 17-Mar-2017

ProtractorTest. Tutorial, Step 3 – changing the configuration. 2017. URL: <http://www.protractortest.org/#/tutorial>. Accessed: 01-Apr-2017

ProtractorTest. Tutorial. 2017. URL: <http://www.protractortest.org/#/tutorial>. Accessed: 15-Mar-2017

QAShahin. AngularJS Protractor Tutorial 07 - Using Page Object Pattern. 2014. URL: https://www.youtube.com/watch?v=ln_jaC11SAA&t=3s. Accessed: 01-Apr-2017

Red Ape EDU. Deferred and Promise objects in AngularJS. In *YouTube*. 2013. URL: <https://www.youtube.com/watch?v=FsOmdsICNYY>. Accessed: 04-Apr-2017

SDET, Software Development Engineer in Test. (n.d.). In *Wikipedia*. URL: https://en.wikipedia.org/wiki/Software_Development_Engineer_in_Test. Accessed: 25-Mar-2017.

SeleniumHq. WebDriver: Advanced Usage. URL: http://www.seleniumhq.org/docs/04_webdriver_advanced.jsp. Accessed: 18-Mar-2017

SoftwareTestingFundamentals. 2017. URL: <http://softwaretestingfundamentals.com/test-plan/>. Accessed: 21-Mar-2017

Sypolt, Greg. Get Started with Protractor Testing for AngularJS. 2016. URL: <https://saucelabs.com/blog/get-started-with-protractor-testing-for-angularjs>. Accessed: 12-Apr-2017

The Net Ninja. Asynchronous JavaScript #1 - What is Asynchronous JavaScript?. In *YouTube*. 2016. URL: <https://www.youtube.com/watch?v=YxWMxJONp7E>. Accessed: 04-Apr-2017.

TTD, Test Driven Development. (n.d.). In *Wikipedia*. URL: https://en.wikipedia.org/wiki/Test-driven_development. Accessed: 09-Mar-2017

Vasanth, Daniel. Protractor - E2E Testing includes Data driven - For AngularJS. In *YouTube*. 2015. URL: <https://www.youtube.com/watch?v=yK6z0zqMiQs>. Accessed: 13-Apr-2017

Appendices

Appendix 1. Keywords

Acceptance test level	A formal testing with respect to user needs, requirements, and business processes conducted to determine whether a system satisfies the acceptance criteria.
AngularJS	AngularJS (commonly referred to as "Angular" or "Angular.js") is a complete JavaScript-based open-source front-end web application framework mainly maintained by Google
Asynchronous flow (JavaScript)	Occurrence of events independently of the main program flow and ways to deal with such events without the program blocking to wait for results
Acceptance test–driven development (ATDD)	A development methodology based on communication between the business customers, the developers, and the testers. It uses runnable specifications.
Test-driven development (TDD)	A software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved to pass the new tests
Build automation	A process of automating the creation of a software build and the associated processes including: compiling computer source code into binary code, packaging binary code, and running automated tests.
CI job (or CI project)	A user-configured description of work which Continuous Integration system should perform, such as building a piece of software, etc.
CircleCI	A continuous integration and delivery platform which gives automated testing and continuous integration tools.
Continuous Integration (CI)	A practice of merging all developer working copies to a shared mainline several times a day
End-to-End (E2E) testing	A methodology used to test whether the flow of an application is performing as designed from start to finish

GitHub	A web-based Git or version control repository and Internet hosting service. It offers all of the distributed version control and source code management
Graphical user interface (GUI)	A type of user interface that allows users to interact with applications through graphical icons and visual indicators
Integrated Development Environment (IDE)	A software application that provides facilities for programmers. An IDE normally consists of a source code editor, build automation tools and a debugger.
Page Object Model (POM)	A design pattern to create Object Repository for web UI elements and methods which perform operations on those WebElements.
Protractor	An Open Source End-to-End test framework for AngularJS applications. It runs tests against an application running in a real browser, interacting with it as a user would.
Research and development (R&D)	A department staffed by engineers and tasked with innovating and developing new products,
RobotFramework	A generic, keyword-driven test automation framework for acceptance testing. It uses tabular test data syntax.
SauceLabs	A cloud-hosted, web and mobile application automated testing platform. It provides virtualized platforms and browsers.
Selenium Webdriver	A tool for writing automated tests of websites. It aims to mimic the behavior of a real user, and as such interacts with the HTML of the application.
Software Quality Assurance	Means of monitoring the software engineering processes and methods used to ensure quality.
Technical (web) stack	A collection of software required for Web development. It usually contains an operating system, a programming language, database software and a Web server.

Test automation	The use of special software (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes with predicted outcomes.
Test coverage	Technique which determines whether our test cases are actually covering the application code and how much code is exercised
Test scenario	A high-level classification of test requirements grouped depending on the functionality of a module
Test script	A set of instructions that will be performed on the system under test to test that the system functions as expected. It can be executed manually or automatically
Test suite	A collection of test cases that are intended to be used to test a software program to show that it has some specified set of behaviors.
Unit (component) testing	A testing method by which individual units of source code are tested to determine whether they are fit for use.