SQA

case studie

S67-3

2021

Inhoud

[The case 2](#_Toc66439644)

[Onderzoeksvragen 3](#_Toc66439645)

[Strategie 4](#_Toc66439646)

[Resultaten 5](#_Toc66439647)

[1. Wat is een automated acceptance test? 5](#_Toc66439648)

[2. Welke tools/framework zijn beschikbaar om acceptance tests automatisch uit te laten voeren? 5](#_Toc66439649)

[3. Wat zijn de resultaten van de prototypes zoals vermeld in de casus approach? 6](#_Toc66439650)

[Resultaat TestCafe: 7](#_Toc66439651)

[Resultaat Cypress.io: 8](#_Toc66439652)

[4. Op welke manier zou een tool/framework uit het prototype meegenomen kunnen worden naar een bestaande CI/CD pipeline samen met static code analysis en code coverage? 9](#_Toc66439653)

[Conclusie 11](#_Toc66439654)

[Bronnen 12](#_Toc66439655)

[Aanbevelingsrapport 13](#_Toc66439656)

# The case

Author: Leon Schrijvers and Nico Kuijpers

Date: November 27, 2020

As a software developer you may already know that clients often don't know what they really want in a project. This results in vague requirements like 'everything the old system does, but then better'. Good luck team!

A friend of yours graduated a couple of years ago and is currently working on a web application project. He is very enthousiastic about the project and the commitment of the Product Owner. The Product Owner has a clear product vision and is fairly able to explain what the requirements are about. Furthermore, the Product Owner is involved in the project, in such a way that he performs a manual UAT at the end of each sprint. In this way he can easily determine if the delivered functionality is accepted. Great!

After delivering several sprints, the project gets bigger and bigger, and the Product Owner has a hard time to manually test all delivered user stories and the overall functionality of the project as well. Furthermore, his involvement in the project decreases a bit because another project has come up that requires a lot of focus. This leads to several issues:

* User stories aren't tested anymore by the PO and are, according to the DoD, never done;
* The PO realizes that too late the delivered functionality doesn't really add up to his initial ideas;
* This results in a lot of new requirements, rework and discussion about when a user story will be accepted.

A similar situation may happen to your group project in Semester 6! In previous projects you have written a lot of use cases or acceptance criteria to prevent discussion on the acceptance of user stories. However, in your current project you are worried that the PO is not enough involved to review and approve all this.

You have heard something about automated acceptance testing, which may be a solution in this situation. Also, the PO mentions that reports on static code analysis and code coverage should be prepared for each sprint delivery. How can you convince the PO that your software not only fulfils the required functionality, but is also of good quality?

As you are always excited about experimenting with new techniques, you want to start right away with the newest tooling and ideas. You propose a spike to investigate the subject. But you also know that spikes can be very expensive in a scrum project, so you know that this requires a structured approach which enables you to deliver value to the project within a reasonable amount of time. You come up with the following approach

* Investigate what automated acceptance testing is and how it can be used to reduce the amount of discussion with the PO about the acceptance of user stories;
* Investigate what tools/frameworks are available for your development platform (long list);
* Based on some substantiated product selection strategy, select two of these tools that are most promising (short list);
* For these two tools, implement a prototype which demonstrate the possibilities of automated acceptance testing;
* Come up with a plan that describes the new approach regarding the process of user story acceptance and generating reports on static code analysis and code coverage. Think about how to involve the PO in this new approach and about how this new approach can be integrated into the existing CI/CD pipeline;
* Write a report that summarizes your findings, including a recommendation on how to continue with software quality assurance in the project.

# Onderzoeksvragen

Omdat er in de casus ook al een approach vermeld staat is het verstandig om hier de vragen op te baseren. In dit document willen we bereiken dat we een aanbeveling kunnen doen over hoe de software quality assurance in het project in de toekomst gewaarborgd kan worden. De hoofdvraag luidt als volgt:

*“Hoe kunnen geautomatiseerde acceptance tests in een project geïntegreerd worden om de software quality in dat project te verzekeren?”*

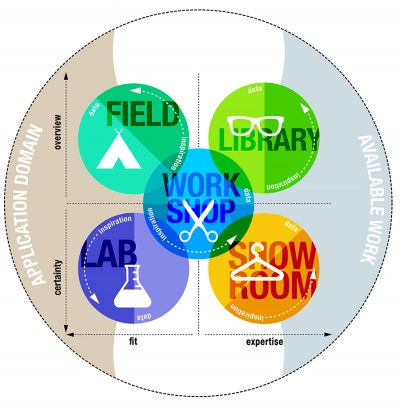
Om deze hoofdvraag te kunnen beantwoorden hebben we deelvragen opgesteld welke gebaseerd zijn op de approach uit de casus. Deze deelvragen zijn hieronder te vinden:

1. Wat is een automated acceptance test?
2. Welke tools/frameworks zijn beschikbaar om acceptence tests automatisch uit te laten voeren?
3. Wat zijn de resultaten van de prototypes zoals vermeld in de casus approach?
4. Op welke manier zou een tool/framework uit het prototype meegenomen kunnen worden naar een bestaande CI/CD pipeline samen met static code analysis en code coverage?

Naast een samenvatting van de antwoorden op de onderzoeksvragen, zal er in de conclusie ook een vermelding naar een aanbevelingsrapport te vinden zijn. In dit aanbevelingsrapport staat een advies over hoe in de toekomst verder te gaan met geautomatiseerde acceptance tests.

# Strategie

Het verrichten van het onderzoek zal gebeuren met behulp van het DOT Research framework. Voor het gebruik van het framework is het van belang dat er verschillende methodes worden gebruikt. Deze methodes komen uit verschillende strategieën en worden toegepast om je onderzoek te valideren.



Om de onderzoeksvraag te beantwoorden hebben wij gekozen voor de volgende strategieën met daarbij de volgende methodes.

|  |  |  |
| --- | --- | --- |
| Field | Library | Workshop |
| Focus group | Community research | Prototyping |
|  | Available product analysis | IT architecture sketching |

# Resultaten

## 1. Wat is een automated acceptance test?

Verschillende partijen geven verschillende definities van wat (automated) acceptance testing precies inhoudt. Zo geeft de firma Delicious Brains aan dat een acceptance test alleen het testen is van de final build om te zien of het klaar is voor de klant.

*“acceptance testing is the process of testing the final build of a forthcoming plugin release to make sure it is ready for our customers. In other (larger) companies it’s when the “customer” or their representative tests the release candidate and hopefully signs-off on it, “accepting” the release. Once accepted, it’s released into production (or in our case uploaded to our server and released to our customers).” -* (Ian Jones, 2020)

Een groot verschil ten opzichte van unit tests is dat een acceptance test wordt uitgevoerd vanuit het perspectief van de klant. Hierbij wordt gekeken of alle acties en scenario’s precies doen wat ze zouden moeten bereiken en hier niet van af wijken. Wat wel vergelijkbaar is met een unit-test is dat een acceptatietest over het algemeen een binair resultaat heeft. Slagen of mislukken.

## 2. Welke tools/framework zijn beschikbaar om acceptance tests automatisch uit te laten voeren?

Tijdens het onderzoek is er gekeken naar tools die end-to-end testing mogelijk maken. Door het te gebruiken van deze tools kunnen de paden test worden die de eindgebruiker zal doorlopen. Deze tests kunnen ook gekoppeld worden aan een CI/CD tool waardoor de resultaten meegenomen kunnen worden in het overwegen of de build is geslaagd.

* Cypress.io
* Nightwatch.js
* Selenium
* TestCafe
* Katalon Studio
* UFT One
* TestComplete
* SoapUI

## 3. Wat zijn de resultaten van de prototypes zoals vermeld in de casus approach?

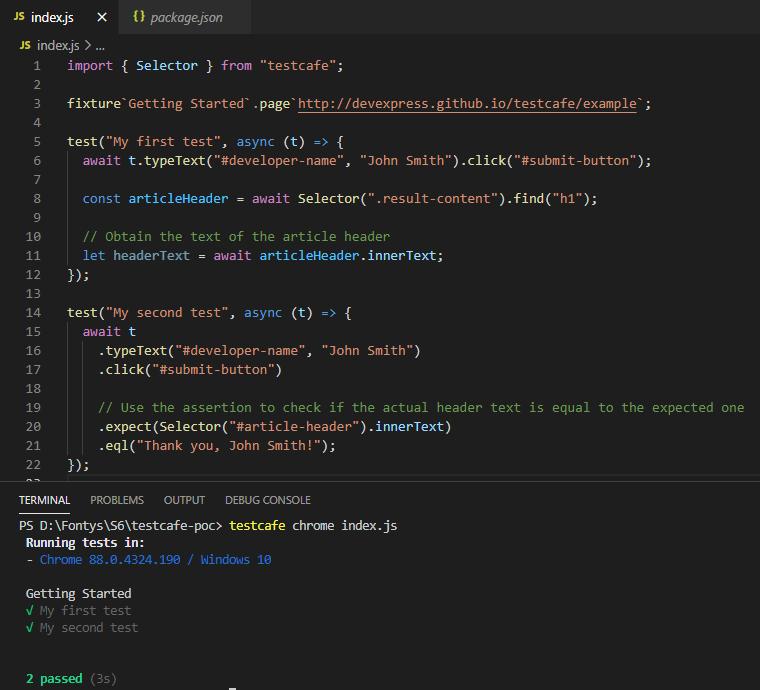
Hier hebben we gekozen om zowel TestCafe als Cypress.io te testen. Deze prototypes zijn succesvol gerealiseerd. Op basis van de prototypes kan er gezegd worden dat beide tools geïmplementeerd zouden kunnen worden voor deze casus. In de conclusie doen we een aanbeveling over welke het beste zou zijn.

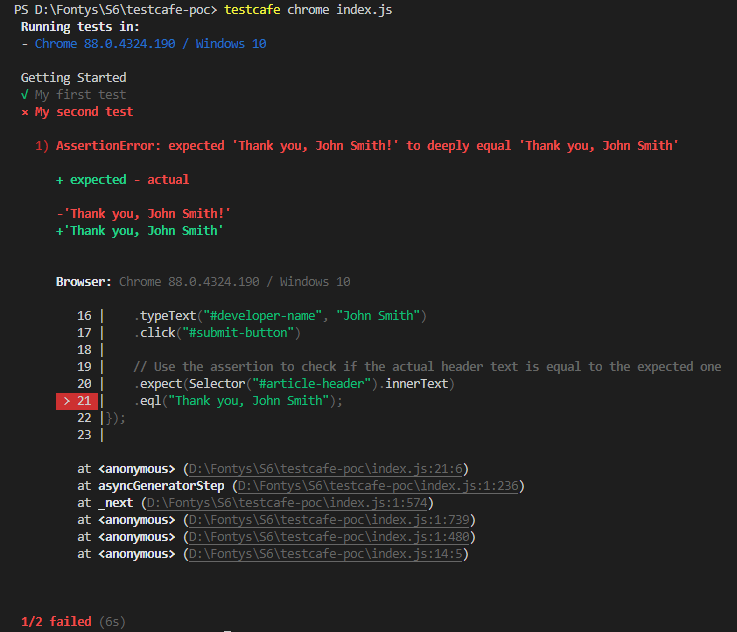
De keuze tussen de tools is gebaseerd op de volgende criteria:

|  |  |  |
| --- | --- | --- |
| CRITERIA | TestCafe | Cypress |
| Besturinssystemen | Windows, Linux, MacOs | Windows, Linux, MacOs |
| Technische kennis benodigd | Medium tot advanced | Medium tot advanced |
| Kosten, per maand | Gratis | Gratis |
| Open Source | Ja | Ja |
| Dashboard/UX | Command line | Command line, Dashboard |
| Browser support range | Uitstekend | Goed |
| Gebruiksgemak | Goed | Goed |
| Programeer taal ondersteuning | Uitstekend | Goed |

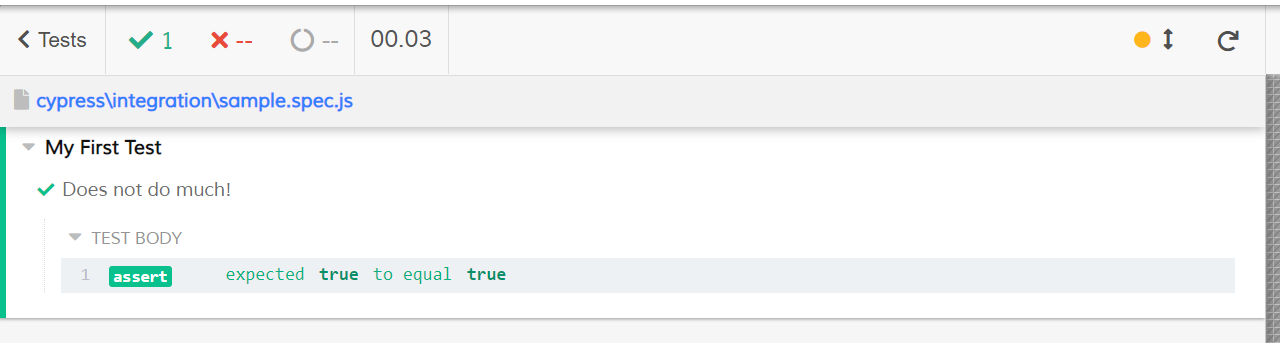
Hieronder zijn de prototype resultaten te vinden. Voor beide frameworks zijn screenshots te vinden van hoe de automatische tests geïmplementeerd zouden kunnen worden en wat hier het resultaat van is.

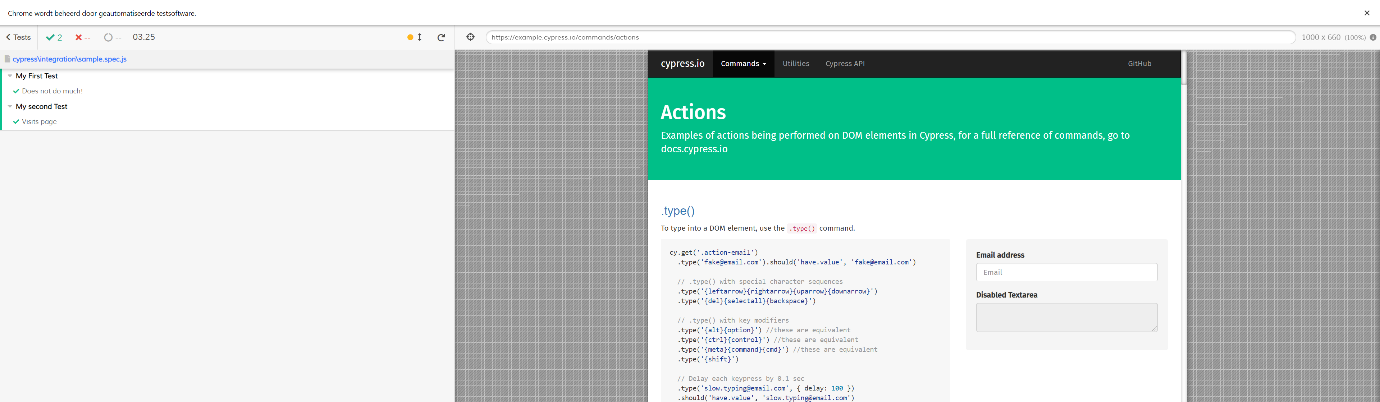
### Resultaat TestCafe:

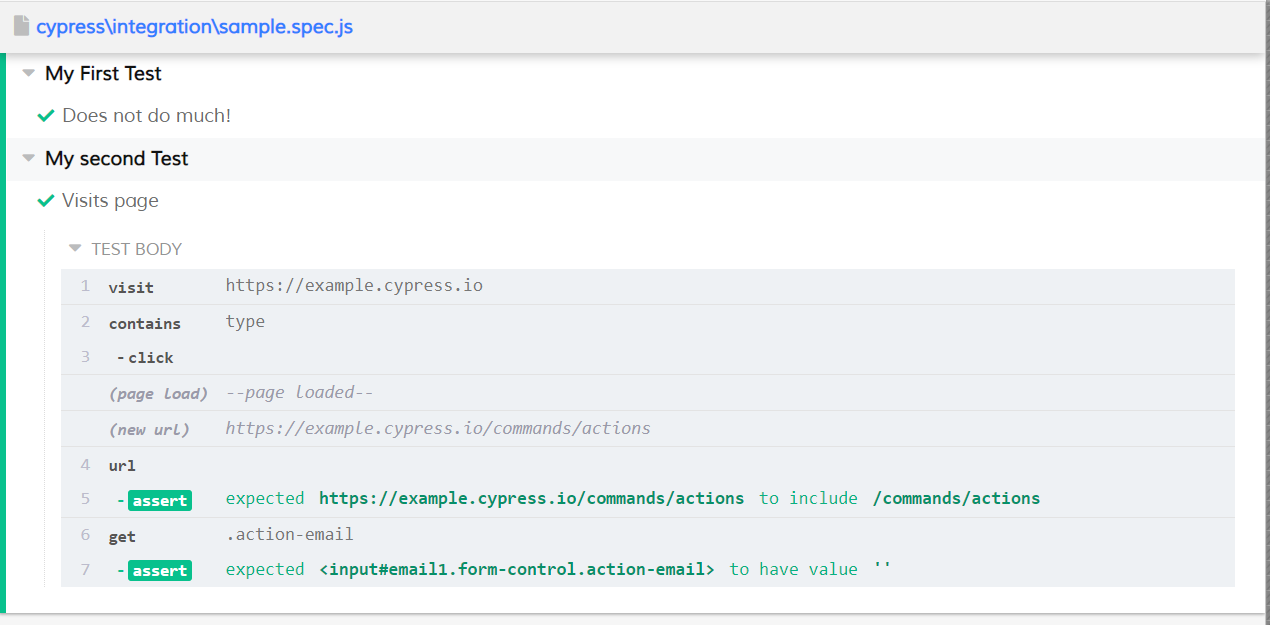




### Resultaat Cypress.io:







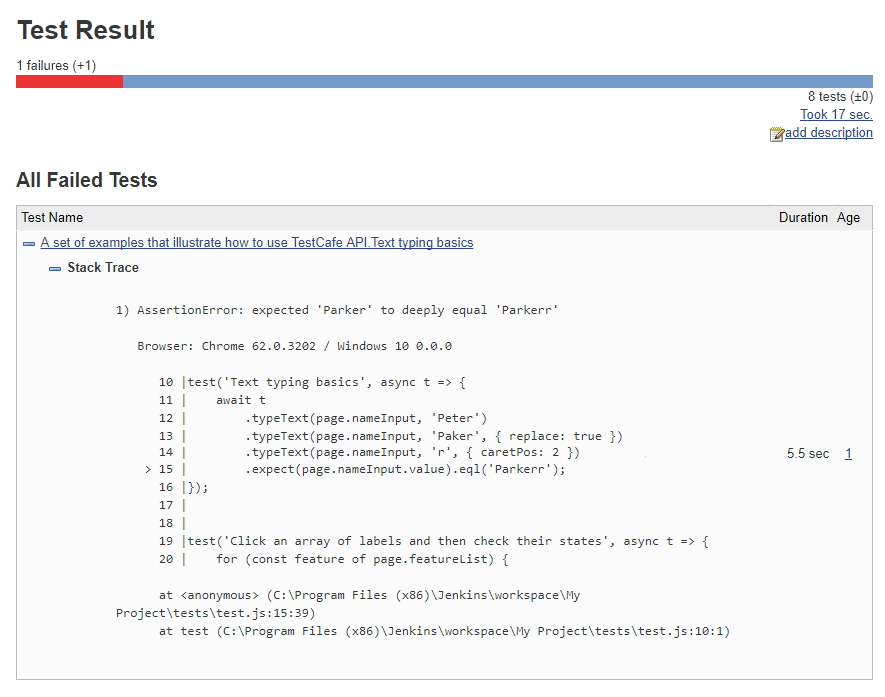
## 4. Op welke manier zou een tool/framework uit het prototype meegenomen kunnen worden naar een bestaande CI/CD pipeline samen met static code analysis en code coverage?

In deze deelvraag behandelen we het 5e approach puntje zoals vermeld in de casus. Een mogelijke manier om een Product Owner te betrekken bij sprint opleveringen terwijl de PO weinig tijd heeft, is door hem een overzichtelijk rapport te geven met de resultaten van de sprint. In dit geval is er behoefte aan een overzicht met acceptence tests, static code analysis en code coverage.

Omdat we als groep al met static code analysis en code coverage hebben gewerkt raden we hiervoor aan om codecov.io en SonarCube te gebruiken.

Binnen Jenkins is het mogelijk om TestCafe te implementeren in de CI/CD pipeline. Dit kan gedaan worden door binnen Jenkins de repository toe te voegen, een commando toe te voegen zodat Jenkins weet dat dit getest moet worden en vervolgens de test(s) te runnen. Hierna zal Jenkins laten zien hoeveel tests er zijn en hoeveel hiervan geslaagd zijn.

*“Jenkins displays a test run report with overall results. Click the test name for information about a specific test.*

*” (Integrate TestCafe with Jenkins | TestCafe, z.d.)*

Vervolgens kunnen de resultaten van deze tests in een kort rapport naar de PO gestuurd worden als de PO wegens tijdgebrek of andere redenen zelf niet aanwezig is om de user acceptance tests uit te voeren.

We zijn van mening dat alhoewel dit een prima oplossing is dat we alsnog zouden willen aanraden om de acceptance tests handmatig uit te voeren indien mogelijk. Zo is het ook zo dat de meeste bedrijven in het bedrijfsleven geen gebruik maken van geautomatiseerde user acceptance tests.

*“Only 3% of all testing teams use automation in their UAT process, according to TestDrive UAT. This number illustrates a general trend wherein most teams rely on manual testing.” -* (AgreeYa, z.d.)

Ook heft TestCafe het voordeel dat deze in veel verschillende bestaande CI pipelines geïntegreerd kan worden. TestCafe is beschikbaar voor de volgende CI frameworks:

* AppVeyor
* Azure DevOps
* Bitbucket Pipelines
* CircleCI
* CircleCI + LambdaTest
* GitHub Actions
* GitHub Actions + BrowserStack
* Gitlab
* Jenkins
* TeamCity
* Travis
* Travis + Sauce Labs

# Conclusie

*“Hoe kunnen geautomatiseerde acceptance tests in een project geïntegreerd worden om de software quality in dat project te verzekeren?”*

Geautomatiseerde acceptatietesten kunnen in een project geïntegreerd worden als de Product Owner zelf weinig tijd heeft om bij het project betrokken te zijn. Er zijn meerdere manieren om deze acceptatietesten te automatiseren in een bestaand project. Hier zijn ook meerdere tools/frameworks voor beschikbaar. Cypress.io en TestCafe zijn uit al deze tools/frameworks geselecteerd als meest geschikt. Op veel gebieden gaan de twee tools gelijk op, maar TestCafe is door het realiseren van prototypes als beste van de twee gebleken. Dit is gebaseerd op twee zaken: browser support range en programmeertaal ondersteuning.

Ook is het mogelijk om de acceptatietesten nog verder te automatiseren door deze te integreren in een CI-pipeline. Dit is bijvoorbeeld mogelijk met Jenkins door slechts een paar commando’s aan Jenkins toe te voegen. Jenkins geeft vervolgens een kort rapport met een overzicht van de tests wat naar de PO gestuurd kan worden.

Ondanks dat acceptatietesten geautomatiseerd op te zetten zijn, zijn er maar weinig bedrijven die hier ook echt gebruik van maken. Het gaat er tenslotte om of de klant tevreden is met de release van het project.

# Bronnen

Agile Alliance. (onbekend). *“What is Acceptance Testing?”* Geraadpleegd op 11 maart 2021 van:  
[https://www.agilealliance.org/glossary/acceptance/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa\_book~'aa\_event\_session~'aa\_experience\_report~'aa\_glossary~'aa\_research\_paper~'aa\_video)~tags~(~'acceptance\*20test))~searchTerm~'~sort~false~sortDirection~'asc~page~1)](https://www.agilealliance.org/glossary/acceptance/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'acceptance*20test))~searchTerm~'~sort~false~sortDirection~'asc~page~1))

Jones, I. (2017, 15 augustus). *“Behind-the-Scenes: How We’re Automating Acceptance Testing”*. Geraadpleegd op 11 maart 2021 van:  
<https://deliciousbrains.com/how-were-automating-acceptance-testing/>

TestCafe. (onbekend). *“Integrate TestCafe with Jenkins”*. Geraadpleegd op 11 maart 2021 van:  
<https://devexpress.github.io/testcafe/documentation/guides/continuous-integration/jenkins.html>

Bijlagen

## Aanbevelingsrapport

We raden aan om een automated acceptance test te integreren in een bestaande CI/CD pipeline als de Product Owner (PO) wegens omstandigheden niet elke sprintoplevering kan bijwonen. Op deze manier kan er toch gecommuniceerd worden naar de PO door een rapport te mailen met de code coverage, static code analysis en automated acceptance tests.

Deze statistieken kunnen allemaal gegenereerd worden door deze pipeline. Als aanbeveling willen we in dit rapport TestCafe naar voren brengen. Deze kan geïntegreerd worden in vele continuous integration systems. Een lijst met opties en eventuele instructies is te vinden op de volgende pagina: <https://devexpress.github.io/testcafe/documentation/guides/continuous-integration/>.

De reden dat we TestCafe aanraden is omdat deze heel eenvoudig is om op te zetten. Ook is de browser ondersteuning heel groot. In tegenstelling tot veel andere tools biedt TestCafe ook ondersteuning voor Internet Explorer. Dit is een browser die in 2021 toch nog door ruim 2% word gebruikt. *(https://kinsta.com/browser-market-share/)*

Daarnaast is de CI Pipeline integratie erg eenvoudig en kan dit zonder extra drivers of aanvullende installaties van derde partij werken. Het enige nadeel aan TestCafe is dat er geen Time travel module in TestCafe verwerkt zit. Hiermee zal de tool automatisch laten zien in welke meest recente rapport deze test wel is geslaagd. Met deze informatie kan de ontwikkelaar kijken waarom het in een oudere build nog wel werkte en wat er sindsdien is aangepast. Vanuit het oogpunt van de ontwikkelaar is dit een groot pluspunt maar voor de PO heeft dit geen toegevoegde waarde.