Projektförslag

Jämförelse av automatiserade testverktyg via prestandautvärdering i form av CPU-, RAM-förbrukning och exekveringstid

Användarnamn: a23fanro
Namn: Fanny Rosell
Kurs: Webbutveckling - Forskning och utveckling
Datum: 2025-12-15

Bakgrund

Tanken med examensarbete är att utvärdera två testverktyg, Selenium och Playwright, för att jämföra deras prestanda i from av CPU-, RAM-förbrukning och exekveringstid vid genomförande av tester på webbapplikationer.

Mätning av exekveringstiden kan motiveras som relevant mätdata då exekveringstiden påverkar hur lång den totala tiden blir för testning, och därmed för hela utvecklingsprocessen. När tester exekveras snabbare behöver utvecklare i mindre utsträckning vänta på att fortsätta sitt arbete i väntan på testresultat, vilket kan påskynda utvecklingen. Kortare exekveringstid kan också argumenteras för att genomförandet av fler tester kan ske, vilket det med bidrar till effektivitet och lägre utvecklingskostnader.

CPU- och RAM-användning kan motiveras som relevanta mätvärden eftersom de utgör en del av de systemresurser som automatiska tester förbrukar. Om resursanvändningen skiljer sig signifikant mellan de två olika testverktygen kan detta indikera skillnader i användning av resurser. Högre resursförbrukning i samband med testning kan enligt Zaidman (2024) kopplas till ökad energianvändning och därmed möjligtvis ha en högre miljöpåverkan.

Testautomation är processen att utföra automatiska tester på mjukvara för att verifiera ett programs beteende. Enligt Jose (2021) är det en viktig metod för att minska testarbetet. Butt et al. (2023) menar att det minskar tiden för test och kostnaden under ett mjukvaruprojekts utveckling. Inom webbutveckling kan testautomation användas för att verifiera funktionalitet och flöde på en webbsida. Både Jose (2021) och Butt et al. (2023) menar att manuella tester i vissa fall kan ersättas av automatiserade tester.

Playwright är ett verktyg för skapande av automatiserade tester som enligt Playwright (u.å) är flerspråkigt, där Typescript, JavaScript, Python, .NET och Java stöds. Playwright är byggt på Node.js och använder en egen webbläsarautomation som styrs av deras egna protokoll och API:er, vilket innebär att Node.js behöver vara installerat för att kunna använda Playwright.

Selenium är ett verktyg för automatisering av tester som stödjer flera programmeringsspråk, bland annat det tänkta Javascript. För att genomföra testautomation med Selenium så används WebDriver-standarden med tillhörande protokoll för att styra webbläsaren, till skillnad från Playwright som har en egen webbläsarautomation. Vid användning av Selenium med Javascript körs testerna i en Node.js-miljö, vilket innebär att Node.js behöver vara installerat för att kunna starta och kontrollera webbläsare i WebDriver.

WebDriver är ett gränssnitt som enligt W3E (u.å) med hjälp av fjärrstyrning möjliggör inspektion och kontroll av webbläsare. Protokollet fungerar oberoende av plattform och språk och gör det möjligt för externa program (exempelvis Selenium) att styra webbläsarens beteende. Funktionaliteten är byggd på DOM-element som manipuleras och kontrollering av webbläsarens tillstånd och interaktioner. Verktyget är i första hand avsett för automatiserad testning, men kan även användas för generell webbläsarautomation.

Node.js är enligt OpenJS Foundation (u.å) en JavaScript-miljö som gör det möjligt att exekvera JavaScript på en server eller dator, det gör det möjligt för både Playwright och Selenium WebDriver att bland annat starta webbläsare, köra tester och skicka nätverksförfrågningar.

npm är enligt sin egen dokumentation (2023) världens största programvaruregister och används av open source-utvecklare över hela världen för att dela och återanvända paket. npm består bland annat av ett kommandogränssnitt, som är den huvudsakliga metoden för utvecklare att integrera med npm från terminalen. Registret är en stor offentlig databas med JavaScript-programvara och tillhörande metadata. npm kommer användas tillsammans med Playwright för att ladda ner och installera paket och dess beroenden.

Problemformulering

Problemet är att automatiserade tester tar tid att exekvera, vilket kan förlänga den totala tiden för projektet och skapa ‘dötid’ för utvecklare som måste vänta på resultatet från test innan mer utveckling kan ske, som i sin tur kommer kosta mer pengar för projektet då utvecklingstiden blir längre.

Det andra problemet är att automatiska tester förbrukar systemresurser, vilket kan öka till högre elektricitetsförbrukning, som enligt Zaidman (2024) kan vara en direkt inverkan på ett projekts miljöpåverkan.

En studie gjord av Mobaraya och Ali (2019) visar att längre exekveringstid kan leda till högre resursförbrukning och kan uppstå till följd av faktorer som flöden och dynamiskt innehåll på webbapplikationer, där de jämför Cypress, ett annat testverktyg, mot just Selenium.

Baserat på problemen ovan kan valet av testverktyg vara mer eller mindre viktigt, och denna kommande undersökningen som är planerad för examensarbetet kan indikera vilken av testverktygen som passar sig bättre att använda vid automation av tester för webbapplikationer.

Hypotesen är att ett av de två verktygen, Selenium WebDriver eller Playwright, kommer visas ha en bättre prestanda, det vill säga kortare exekveringstid och mindre resursförbrukning, än den andra.

Nollhypotes är att det inte finns någon signifikativ skillnad på exekveringstid eller resursförbrukning mellan Selenium WebDriver och Playwright vid körning av liknande tester.

Metod

Metod: Experimentell. Planerar skapa så snarlika testfall som möjligt med hjälp av två olika testverktyg. Utvärdering av flakiness om det finns tid och möjlighet. Flakiness är huruvida tester faktiskt genomförts eller ej. Enligt Berndtsson et al. (2008) kan en experimentell metod användas för att undersöka hur förändringar i utvalda variabler påverkar ett systems egenskaper, i detta fall testverktygets prestanda, för att utreda hypotesen och problemformuleringen.

Testerna kommer utföras genom så kallade ‘end-to-end’ tester, som är en metod för att validera en hel applikations flöde från början till slut. Sannolikt kommer testerna vara relativt enkla, exempelvis inloggning, sökning och klick. End-to-end testning är enligt IBM (u/å) en testmetod som arbetar för att validera en hel applikations flöde från början till slut. Testerna förväntas skrivas med JavaScript i VS Code, och utvärderingen av exekveringstid och resursanvändning kommer mätas med ett greasemonkey-script eller i själva testskriptsen i samband med exekvering.

Motivering: Flera utvärderingar av prestanda mellan testverktyg har gjorts, bland annat av Pelivani et al. (2021) i studien “A Comparative Study of Automation Testing Tools for Web Applications” där motivering för att manuella testers omvandling till automation, och av Shokeh et al. (2025) i studien ‘A Comparative Evaluation of Cypress and Katalon for Web Application Test Automation’ där skapandet av testfall tydliggörs och exekveringstid studeras, samt av Sani et al. (2024) i Empirical Analysis of Widely Used Website Automated Testing Tools’**,** deras studie behandlar mer om användarvänlighet, funktionalitet och plattformsstöd, alltså fler faktorer utöver exekveringstid och prestanda, vilket är något jag inte tänkt ha med i min studie, men innehållet av deras studie är fortfarande värt att ta i akt.

Har tankar om att genomföra testerna på en liten webbsida, samt en mellan och en stor. Detta kommer utvärderas med hjälp av hur många DOM-element som existerar på sidan.

Etiska aspekt
1. Påverkan på systemet som testas: vid prestandatest kan flera tester köras, då kan webbsidan testet utföras på skapa mycket trafik, vilket kan göra att sidan körs långsammare eller blir mindre nåbar för faktiska användare som försöker bruka sidan.

2. Vid jämförelse av två verktyg (selenium och playwright) måste vara rättvis, försäkring om att båda verktygen ges rätt förutsättningar och testar delar som inte är specifikt bättre på ett av dem jämfört med den andra.

3. Test där känslig information behövs, ex inloggningsuppgifter, behöver ses över så ingen information läcker eller kan användas av obehöriga. Exempelvis kan konton som användas skapas för teständamål, där ingen faktiskt känslig information anges, så som bankuppgifter och/eller postadress.

+ Tidpunkt då tester utförs kan påverka, samma tidpunkt för testerna, inte olika pga trafik
+ samma hårdvara för verktygen, inte ena på laptop och andra på stationära

Referenser (Alfabetisk ordning)

Alégroth, E., Feldt, R. och Kolström, P. (2016) ‘Maintenance of automated test suites in industry: An empirical study on Visual GUI testing’, *Information and Software Technology*, 73, ss. 66–80. doi:10.1016/j.infsof.2016.01.012.

Berndtsson, M., Hansson, J., Olsson, B. & Lundell, B. (2008). *Thesis Projects: A Guide for Students in Computer Science and Information Systems*. London: Springer.

Butt, S., Khan, S.U.R., Hussain, S. och Wang, W.-L. (2023) ‘A conceptual model supporting decision-making for test automation in Agile-based software development’, *Data & Knowledge Engineering*, 144, 102111. doi:10.1016/j.datak.2022.102111.

IBM (u.å.) *End-to-end testing*. Tillgänglig på: https://www.ibm.com/think/topics/end-to-end-testing (Hämtad: 13 december 2025).

Jose, B. (2021) *Test automation: A manager’s guide*. London: BCS, The Chartered Institute for IT.

Mobaraya, F. och Ali, S. (2019) ‘Technical analysis of Selenium and Cypress as functional automation frameworks for modern web application testing’, i *Proceedings of the 9th International Conference on Computer Science and Application Engineering*. AGI Education Ltd. doi:10.5121/csit.2019.91803.

OpenJS Foundation (u.å.) *Node.js*. Tillgänglig på: https://nodejs.org/en (Hämtad: 13 december 2025).

Pelivani, E., Besimi, A. och Cico, B. (2021) ‘A comparative study of automation testing tools for web applications’, i *Proceedings of the 10th Mediterranean Conference on Embedded Computing (MECO 2021)*, 7–10 juni 2021, Budva, Montenegro. Budva: IEEE. doi:10.1109/MECO52532.2021.9460242.

Playwright (u.å.) *Playwright*. Tillgänglig på: https://playwright.dev/ (Hämtad: 9 december 2025).

Sani, B. och Jan, S. (2024) ‘Empirical analysis of widely used website automated testing tools’, *EAI Endorsed Transactions on AI and Robotics*, 3. doi:10.4108/airo.7285.

Selenium (u.å.) *Selenium*. Tillgänglig på: https://www.selenium.dev/about/ (Hämtad: 9 december 2025).

W3C (u.å.) *WebDriver*. Tillgänglig på: https://www.w3.org/TR/webdriver2/ (Hämtad: 13 december 2025).

Zaidman, A. (2024) ‘An inconvenient truth in software engineering? The environmental impact of testing open source Java projects’, i *Proceedings of the IEEE/ACM International Conference on Automation of Software Test (AST 2024)*. New York: IEEE/ACM. doi:10.1145/3644032.3644461.