# Disszertáció

# Tartalomjegyzék

# Bevezető

Probléma: a problémát a munkahelyen fedeztem fel, a jelenlegi munkafolyamatot szeretném optimalizálni autómatizált teszteléssel.   
Röviden a munkám: Product Lifecycle Management szoftvert szabunk testre klienseknek. A szoftver neve Windchill, ami Spring keretrendszerre épülő webes applikáció.   
A probléma pedig ott adódik, hogy a fejlesztéseinket tesztelni kell. Jelenleg kézzel csináljuk ezt. Ez lassú. És valahányszor release-re kerül a sor, újra kell csinálni kézzel minden tesztet, hogy meggyőződjünk róla, hogy minden új funkció az elvártnak megfelelően üzemel és a már meglévő funkciókat nem befolyásolják. Ezt a teszt környezeten egy teszter végzi. Ezen rengeteg idő jelentős részét meg lehetne takarítani, ha a teszteket autómatizálnánk. Plusz, mivel ugyanazt a Windchill-t több projektben is használjuk, ezért a tesztek nagy része újrahasznosítható lenne minimális módosításokkal. A másik probléma pedig az, hogy nem lehet elmenteni és újra futtatni a teszteket, mert ugye azt mindig kézzel csináljuk. Itt jön képbe az autómatizált tesztelés. (Selenium.) Autómatizálhatjuk a teszteket, szimulálhatjuk a végfelhasználó viselkedését és akárhányszor újra futtathatjuk azokat. Ennek meg van az az előnye is, hogy így megbizonyosodhatunk arról, hogy egy már meglévő funkciót nem rontottunk el egy új fejlesztéssel.

A Cucumber pedig azért hasznos választás, mert így a tesztek olvashatóak lesznek nem csak a fejlesztők, hanem a menedzserek számára is.

PTC Community oldalon publikálhatom a kutatást.

A cég belső portálján is publikálhatom az eredményeket.

Cél: hogy bebizonyítsam az autómatizált tesztek hozzájárulnak a szoftver minőségének javulásához és ezzel időt, energiát és nem utolsó sorban pénzt takaríthatunk meg. Valamint választ adni arra a kérdésre, hogy mikor érdemes autómatizálni a teszeket

# A kutatott technológiák bemutatása

### Autómatizált tesztelés frameworkok

Előnyök, hátrányok

### Selenium

Mi a Selenium?

A Selenium egy keretrendszer, melynek segítségével autómatizálhatjuk a webböngészőt. Hogy mit kezd ezzel a hatalommal a felhasználó, az teljesen rajta áll. Első sorban a webes alkalmazások tesztelés céljából történő automatizálására szolgál, de nem merül ki ennyiben. Például az unalmas web alapú adminisztratív teendőket is automatizálni lehet vele. [1]

Nagyon gyakran emlegetik a Seleniumot és a WebDrivert egymás mellett. Aki most találkozik először a kifejezésekkel, bizonyára összezavarodhat, hogy mi a különbség a kettő között. Régen a Selenium és a WebDriver két különálló projekt volt, a lényege mindkettőnek ugyanaz volt: hogy irányíthassuk a böngészőt, csak másképp volt megvalósítva. [2] Később, 2007-ben a két projekt egyesült és kiadták a Selenium 2.0 –t, amely tartalmazta mindkét projektet. [1]

Miért Selenium?

Azért döntöttem Selenium mellett, mert figyelembe kell venni, hogy a csapat miben tapasztalt. Javaban van jártasságunk, ezért ésszerű döntés lenne, ha a teszteket is ugyanabban a nyelvben és környezetben tudnánk írni, továbbá a Selenium jól integrálható JUnit-al és Maven-el. A tanulási nem meredek azok számára, akiknek van jártassága Java-ban és objektum orientált programozásban. Szempont volt még a költség is: mivel a Selenium nyílt forráskódú és bárki számára ingyenesen elérhető, ezért erre nem kell plusz költségekkel számolni. Selenium további előnyei közé tartozik, hogy sok böngészőt támogat. Mivel a Windchill hivatalosan az InternetExplorer és a Google Chrome böngészőt támogatja, fontos szempont, hogy a böngészők közötti különbségek ne jelentsenek gondot.

A WebDriver miatt képes a végfelhasználó tevékenységét imitálni a böngészővel, mint például billentyű leütések, drag-and-drop, vissza gomb

Plusz pont a Seleniumnak, hogy nagy a felhasználótábora, így jó eséllyel találunk megoldást a gyakori problémákra. Az iDatalabs szerint a Selenium piaci részesedése több, mint 27%. [3]



Ábra 1 - iDatalabs kutatásai szerint több, mint 29000 cég használja a Seleniumot [3]

Mert Selenium Grid segítségével párhuzamosan is elvégezhetőek a tesztek, bár ez a továbbfejlesztésben játszana fontos szerepet.

Selenium IDE

Selenium WebDriver

Selenium Grid

Selenium Remote Control

### Cucumber

BDD – Behaviour Driven Development

### Page Object Model

# Szakirodalmi tanulmány

Kiss Tibor küldi

# Megvalósított rendszer bemutatása

Rendszer arhitektúra

- hibák

- sebesség

- jmeter - terhelhetőség

- comparatív összehasonlítás, keresni a piacon

- hogyan lehetne tesztelni azt, hogy a meglévő megy-e még ha hozzáadunk újat

- response idő, érdekesség szempontjából processzor idő

- adatbázis keresés, optimalizálás

- lekérdezések eredményei és időtartama

- eredmények naplózhatósága verziók között, összehasonlítás az előző verziókkal, mi csökken, mi nő, hatékonyság

# Következtetések és tapasztalatok

Mikortól érdemes autómatizálni a teszteket?

# Továbbfejlesztési lehetőségek

# Idézett forrásmunkák

|  |  |
| --- | --- |
| [1] | „SeleniumHQ,” SeleniumHQ, [Online]. Available: https://www.seleniumhq.org/. [Hozzáférés dátuma: 02 2019]. |
| [2] | B. Hansen, „Automated Web Testing with Selenium and WebDriver Using Java,” Pluralsight, 22 09 2017. [Online]. Available: https://app.pluralsight.com/library/courses/automated-web-testing-selenium-webdriver-java/table-of-contents. [Hozzáférés dátuma: 01 2019]. |
| [3] | P. Sams, Selenium Essentials, Livery Place, 35 Livery Street, Birmingham B3 2PB, UK: Packt Publishing Ltd., 2015. |
| [4] | C. Tudose, „Java: BDD Fundamentals,” Pluralsight, 11 01 2019. [Online]. Available: https://app.pluralsight.com/library/courses/java-bdd-fundamentals/table-of-contents. [Hozzáférés dátuma: 02 2019]. |
| [5] | „iDatalabs,” iDatalabs, [Online]. Available: https://idatalabs.com/tech/products/selenium/. [Hozzáférés dátuma: 05 2019]. |