RANKINIO IR AUTOMATINIO TESTAVIMO TYRIMAS

Vytaras Valatkevičius, Dmitrij Šešok
Vilniaus Gedimino technikos universitetas
E. p.: valatkevicius@stud.vgtu.lt; dmitrij.sesok@vgtu.lt

Įvadas

Programinės įrangos testavimas yra programos ar sistemos vykdymo procesas, siekiant nustatyti programinės įrangos klaidas. Galima teigti, kad tai procesas, kuriuo patvirtinama ir patikrinama, kad programinė įranga, sistema ar produktas atitinka verslo bei techninius reikalavimus. Tai reiškia, kad ji veikia taip, kaip tikėtasi, ir gali būti įgyvendinta pagal tas pačias charakteristikas, kitaip tariant, testavimo metu palyginami faktiniai rezultatai su laukiamais.

Efektyvumas ir produktyvumas atlieka svarbų vaidmenį atliekant programinės įrangos testavimą. Testo efektyvumas yra santykinis testavimo strategijos sugebėjimas rasti klaidas programinėje įrangoje [1]. Testo produktyvumas – tai bandomosios programinės įrangos klaidos santykinė kaina [2]. Viena iš pagrindinių problemų programinės įrangos pramonėje yra testavimo laikas. Laikas reiškia pinigus, taigi visas procesas yra brangus [3]. Todėl, siekiant sumažinti testavimo išlaidas, būtina sumažinti rankinio testavimo laiką, o tai, tikėtina, gali padėti padaryti testavimo atvejų automatizavimas.

Tikslas – atlikti realaus projekto rankinio ir automatinio testavimo įtakos kainai ir laikui tyrimą bei pagal gautus rezultatus pateikti rekomendacijas, padėsiančias pasirinkti testavimo būdą.

Uždaviniai: 1) išnagrinėti panaudos atvejų taškų panaudojimą rankinio testavimo pastangų įvertinimui; 2) išnagrinėti COCOMO II modelio taikymą automatinio testavimo pastangų įvertinimui; 3) įvertinti rankinio ir automatinio testavimo pastangas taikant panaudos atvejų taškus ir COCOMO II modelį; 4) atlikti rankinio ir automatinio testavimo pastangų palyginimą.

Tyrimo metodai: mokslinės literatūros analizė, duomenų susisteminimas ir apibendrinimas.

Rankinio testavimo pastangų apskaičiavimo metodika

Tam, kad galėtume palyginti rankinį testavimą su automatiniu, turime turėti įverčius, kuriuos lyginsime. Tam puikiai tinka laikas ir kaina. Apskaičiavę rankinio testavimo laiką, galėsime natūraliai sužinoti ir rankinio testavimo kainą. Rankinio testavimo laikui apskaičiuoti naudosime panaudos atvejų taškus (angl. *Use Case Points*). Panaudos atvejų taškai yra programinės įrangos vertinimo metodika, naudojama programinės įrangos kūrimo pastangų prognozavimui, tačiau taip pat puikiai tinkanti testavimo pastangų apskaičiavimui [4].

Panaudos atvejų taškai

Analizėje naudojamas vienas pastangų kintamasis MTE_i – tai rankinio testavimo pastangos, kur i – versijos numeris. Rankinio testavimo pastangos yra apskaičiuojamos naudojant 1 lygtį:

$$MTE_i = AUCP = UUCP\,(0{,}65 + (0.01 \cdot TEF)). \quad (1)$$

Čia: AUCP – pritaikyti panaudos atvejų taškai (angl. *Adjusted Use Case Points*), TEF – techniniai ir aplinkos verčių koeficientai (angl. *Technical and Environmental Factor*), UUCP – nepritaikyti panaudos atvejų taškai (angl. *Unadjusted Use Case Points*). UUCP apskaičiuojamas taip:

$$UUCP = UAW + UUCW. \quad (2)$$

Čia: UAW – nepritaikytų dalyvių svoris (angl. *Unadjusted Actor Weights*), UUCW – nepritaikytų panaudos atvejų svoris (angl. *Unadjusted Use Case Weights*) [5].

Turint TEF, UAW ir UUCW galima apskaičiuoti AUCP. Tačiau, kad gautume galutines pastangas, mums reikia AUCP padauginti iš konversijos koeficiento. Šis konversijos koeficientas nurodo žmogaus darbo valandas [6]. Pavyzdžiui, *Pastangos* = *AUCP* · 2, kur 2 – žmogaus darbo valandos, per kurias jis įvykdo vieną tašką, šiuo atveju suplanuoja, aprašo ir įvykdo testus.

Norėdami apskaičiuoti kainą, turime tiesiog padauginti gautas pastangas iš darbo vietos kainos: *Kaina* = *Pastangos* · *DarboVietosKaina*.

Automatinio testavimo pastangų apskaičiavimo metodika

Analizėje naudojami penki pastangų kintamieji: MTE_i – rankinio testavimo pastangos, TME_i – automatinio testavimo atvejų palaikymo pastangos, TAE_i – testų automatizavimo pastangos, TTE_i – bendros testavimo komandos pastangos, *STE* – bendros visos programinės įrangos testavimo pastangos visoms versijoms (i – versijos numeris).

Rankinio testavimo pastangos yra apskaičiuojamos naudojant panaudos atvejų taškus. Kai mes bandome automatizuoti testo atvejį, sukuriame visiškai naują programą, o šios visos naujos programinės įrangos kūrimo išlaidos yra lygios testų automatizavimo pastangoms, kurias galima apskaičiuoti naudojant COCOMO-2 modelį [7]. Šis modelis skirtas programinės įrangos sąnaudoms įvertinti [8].

Visiško testavimo automatizavimo scenarijuje MTE_i apskaičiuojamas iš 1 lygties, bet tik naujoms funkcijoms. Norėdami išbandyti senąsias funkcijas dabartinėje versijoje, turėtume naudoti testų atvejus, kurie buvo automatizuoti ankstesnėje versijoje. TAE_i apskaičiuojamas naudojant ekologinių projektų COCOMO-2 modelį, kaip nurodyta 3 lygtyje [8]. Taip pat daroma prielaida 4-oje lygtyje, kad 30 % pastangų naudojama palaikant anksčiau parašytus automatizuotus testus [7].

$$TAE_i = 2{,}4(KLOC)^{1{,}05} \quad (3)$$

$$TME_{i+1} = 0{,}3(TAE_i) \quad (4)$$

$$TTE_i = MTE_i + TME_i + TAE_i \quad (5)$$

$$Pastangos = STE_i = \sum_{i=1}^{k} TTE_i \quad (6)$$

Rankinio ir automatinio testavimo pastangų apskaičiavimas realiam projektui

Nagrinėsime lojalumo sistemos dviejų WEB servisų rankinį ir automatinį testavimą. Ištirsime, kiek reikia laiko atlikti regresijos testavimui, bei įvertinsime, kada atsipirks automatizuoti testai. Taip pat, kiek laiko reikės skirti testavimui naudojant automatizuotus testus.

Rankinis testavimas

Pirmuoju atveju išsiaiškinsime rankinio testavimo trukmę. 1–3 lentelėje pateikti koeficientai parinkti atsižvelgiant į [4] rekomendacijas.

Nustatome nagrinėjamų WEB servisų dalyvius ir jų koeficientus (1 lentelė):

1 lentelė. ***Nepritaikyti dalyviai su koeficientais (UAW)***

Dalyvis	**Koeficientas**
POS sistema	3
Kiosk sistema	3
WEB sistema	3
Mobile programa	3
Iš viso	**12**

Antrame žingsnyje nustatome naudojimo atvejų skaičių su koeficientais (2 lentelė):

2 lentelė. ***Naudojimo atvejų skaičius (UUCW)***

Naudojimo atvejis	**Tipas**	**Koeficientas**
Bonus points	Average	2
Stickers	Average	2
Tare coupons	Complex	3
Value coupons	Simple	1
E-gift cards	Average	2
E-voucher	Simple	1
Segments	Average	2
Promotions	Complex	2
Registration	Average	2
Card replacement	Simple	3
Family Sharing	Simple	1
Iš viso		**21**

Apskaičiuojame UUCP:

$$UUCP = UAW + UUCW = 12 + 21 = 33. \quad (8)$$

Pateikiame techninius koeficientus ir aplinkos vertes su priskirtais svoriais (3 lentelė):

3 lentelė**.** ***Techniniai ir aplinkos verčių koeficientai (TEF)***

Apibūdinimas	**Vertė**	**Svoris**	**Praplėsta vertė**
Test tools	5	3	15
Documented inputs	5	5	25
Development environment	2	0	0
Test environment	3	4	12
Test-ware reuse	3	3	9
Distributed system	4	3	12
Performance objectives	2	1	2
Security features	4	0	0
Complex interfacing	5	3	15
Iš viso			90

Apskaičiuojame pritaikytus panaudos atvejų taškus (AUCP):

$$AUCP = UUCP \cdot (0{,}65 + (0{,}01 \cdot TEF)) = 33 \cdot (0{,}65 + 0{,}01 \cdot 90) = 51{,}15. \quad (9)$$

Turėdami AUCP galime apskaičiuoti žmogaus pastangas, kurias jis užtruktų testuodamas rankiniu būdu. Teigiame, kad vienas panaudos atvejo taškas atitinka dvi darbo valandas:

Pastangos = 51,15 · 2 = 102,3 (*žmogaus darbo valandos*). (10)

Sakykime, kad žmogus vidutiniškai per mėnesį dirba 21 dieną, o darbo vietos kaina yra 2000 eurų. Tokiu atveju gauname, kad regresijos testavimas kainuoja:

Kaina = 1217,86 (eurų).

Automatinis testavimas

Apskaičiuojame testų automatizavimo pastangas iš 3 lygties, kur koeficientai yra gauti iš daugybės projektų analizių. KLOC yra projekto kodo pristatytų eilučių skaičius, kuris išreikštas tūkstančiais. Šiuo atveju skaičiuosime ne testų automatizavimo pastangas versijai, o visas reikalingas testų automatizavimo pastangas. Dėl to, kad žinome kodo eilučių skaičių, kuris reikalingas dviem nagrinėjamiems WEB servisams, t. y. 3200 kodo eilučių:

$TAE = 2{,}4(3{,}2)^{1{,}05} = 8{,}14$ (*žmogaus darbo mėnesių*). (11)

Jeigu testų kodas nesukurtas, galime apsiskaičiuoti LOC (kodo eilutės (angl. *Lines of Code*)) reikšmę, pasitelkdami bendrus funkcijų taškus ir programavimo kalbos koeficientus [9]. LOC reikšmę galima konvertuoti į KLOC.

Iš 11 lygties matome, kad mums prireiks 8,14 žmogaus darbo mėnesių, kad automatizuotume du WEB servisų testus. Jeigu laikysime, kad darbo vieta kainuoja tiek pat, kiek ir rankinio testavimo atveju, tuomet automatizavimo kaina bus:

Kaina = 8,14 · 2000 = 16 280 (*eurų*) (12)

Kad apskaičiuotume, kiek reikės skirti laiko testų palaikymui po automatizavimo, turime apskaičiuoti, kiek vidutiniškai buvo sukoduojama eilučių per darbo mėnesį:

$$LOC = \frac{3200}{8.14} = 393{,}12 \text{ (}eilučių/mėn.\text{)}. \quad (13)$$

Pavertę LOC į KLOC gauname, kad per darbo mėnesį sukoduojama 0,39312 kodo eilučių. Kadangi žinome, kiek sukoduojama per mėnesį eilučių, galime apskaičiuoti testų automatizavimo pastangas mėnesiui:

$TAE_{mėn} = 2{,}4\ (0{,}39312)^{1{,}05} = 0{,}90$ (*žmogaus darbo mėnesių*). (14)

Iš 4 lygties galime apskaičiuoti automatinio testavimo palaikymo vidutiniškas pastangas kiekvienam būsimam mėnesiui:

TME = 0,3 · 0,90 = 0,27 (*žmogaus darbo mėnuo*). (15)

Rankinio ir automatinio testavimo palyginimas

Analizuojant rankinio ir automatinio testavimo trukmę ir kainą reikia įvertinti tai, jog kiekvienoje naujoje versijoje bus sukuriamas naujas funkcionalumas arba praplečiamas esamas. Dėl to prie kiekvienos naujos versijos iš projekto vykdymo patirties pridedame 4 darbo valandas, o tai yra 0,02 žmogaus darbo mėnesio. Ta pati reikšmė turi būti pridedama ir prie automatinio testavimo, kadangi naujo funkcionalumo testai bus automatizuoti tik kitoje versijoje, o dabartinėje versijoje reikia naują funkcionalumą tikrinti rankiniu būdu. Kad sužinotume, kiek pastangų reikia skirti naujų testų automatizavimui, reikia apskaičiuoti, kiek sukoduojama eilučių 4 val. testavimui:

$LOC_{naujų\ testų} = 4 \cdot 3200/102{,}3 = 125{,}12$ (*eilučių*). (16)

Naujų testų automatizavimui reikalingos pastangos:

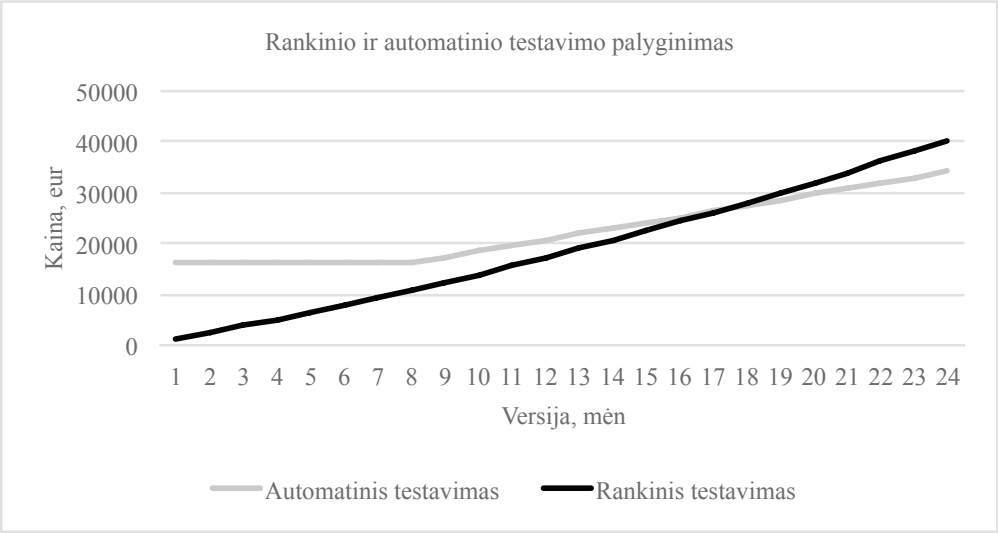
$TAE_{naujų\ testų} = 2{,}4\ (0{,}12512)^{1{,}05} = 0{,}27$ (*žmogaus darbo mėnuo*). (17)

4 lentelėje pateikiami daliniai rezultatai, kurie nusako, kaip didės rankinio testavimo laikas ir kaina kiekvienoje versijoje, taip pat automatinio testavimo kūrimo laikas ir kaina. Matome, kad automatinio testavimo procesas iki 9 versijos buvo kuriamas, o po jos vyksta pilnas automatinių testų naudojimas, palaikymas ir naujų testų automatizavimas.

4 lentelė. ***Rankinio ir automatinio testavimo rezultatai***

	Rankinis testavimas		Automatinis testavimas				
Versija	**Rankinio testavimo pastangos**	**Rankinio testavimo kaina**	**Rankinis testavimas**	**Testų palaikymas**	**Naujų testų automatizavimas**	**Laikas**	**Automatinių testų kaina**
i+1	0,61	1220	-	-	-	8,14	16280
i+2	0,63	1260	-	-	-		
i+3	0,65	1300	-	-	-		
i+4	0,67	1340	-	-	-		
i+5	0,69	1380	-	-	-		
i+6	0,71	1420	-	-	-		
i+7	0,73	1460	-	-	-		
i+8	0,75	1500	-	-	-		
i+9	0,77	1540	0,02	0,27	0,27	0,56	1120
i+10	0,79	1580	0,02	0,27	0,27	0,56	1120
i+11	0,81	1620	0,02	0,27	0,27	0,56	1120
i+12	0,83	1660	0,02	0,27	0,27	0,56	1120

Toliau paveiksle pavaizduotas rankinio ir automatinio testavimo palyginimas ilguoju laikotarpiu (24 versijos = 24 mėnesiai). Matome, kad pradiniu atveju testų automatizacija kainuoja daug daugiau ir užtrunka ilgiau negu testuojant rankiniu būdu. Po 8 versijos sukūrus automatizuotus testus sutaupoma jau daug laiko testuojant kiekvieną versiją, o 17 versijoje rankinio testavimo kaina susilygina su automatiniais testais ir nuo kitų versijų automatinis testavimas kainuoja jau mažiau. Tai reiškia, kad automatiniai testai laiko atžvilgiu atsiperka po 9 mėnesių, kainos atžvilgiu – po 18 mėn.



Pav. Rankinio ir automatinio testavimo palyginimas

Išvados

Kuo ilgiau sistema vystoma, testuojant rankiniu būdu, tuo ilgesnis laikas reikalingas tam, kad atsipirktų automatinių testų įvedimo strategija.

Siūlomos rankinio ir automatinio testavimo kaštų vertinimo metodikos leidžia labai greitai (per kelias darbo valandas) įvertinti rankinio ir automatinio testavimo panaudojimą konkrečiam jau išvystytam verslo projektui ir įvertinti atsiperkamumą laiko bei kainos atžvilgiu.

Literatūra

1. A Survey of Effective and Efficient Software Testing, [interaktyvus], [žiūrėta 2018 m. kovo 22 d.]. Prieiga per internetą: http://www.micsymposium.org/mics2015/ProceedingsMICS_2015/Mailewa_2D1_41.pdf.
2. Cost effective software test metrics, [interaktyvus], [žiūrėta 2018 m. kovo 22 d.]. Prieiga per internetą: https://www.researchgate.net/publication/228658128_Cost_effective_software_test_metrics.
3. Dasso A., Funes A., 2007, *Verification, Validation and Testing in Software Engineering*. Argentina: Universidad Nacional de San Luis.
4. An Alternative Approach to Test Effort Estimation

Based on Use Cases, [interaktyvus], [žiūrėta 2018 m. vasario 4 d.]. Prieiga per internetą: http://ieeexplore.ieee.org/document/4815360/.
5. Simplifying effort estimation based on Use Case Points, [interaktyvus], [žiūrėta 2018 m. vasario 5 d.]. Prieiga per internetą: https://pdfs.semanticscholar.org/6d73/e7227e8adfce3f8a0399d8712bd36d70a020.pdf.
6. Use Case Point and e-Use Case Point method of software effort estimation: A critical performance comparison, [interaktyvus], [žiūrėta 2018 m. vasario 4 d.]. Prieiga per internetą: https://www.researchgate.net/publication/279956164_Use_Case_Point_and_e-Use_Case_Point_method_of_software_effort_estimation_A_critical_performance_comparison.
7. The Impacts of Tet Automation on Software's Cost, Quality and Time to Market, [interaktyvus], [žiūrėta 2018 m. sausio 14 d.]. Prieiga per internetą: https://www.sciencedirect.com/science/article/pii/S1877050916001277.
8. Optimizing Basic COCOMO Model Using Simplified Genetic Algorithm, [interaktyvus], [žiūrėta 2018 m. vasario 5 d.]. Prieiga per internetą: https://www.sciencedirect.com/science/article/pii/S1877050916311723.
9. Cost Estimation, [interaktyvus], [žiūrėta 2018 m. gegužės 9 d.]. Prieiga per internetą: http://people.cs.ksu.edu/~padmaja/Project/CostEstimate.htm.

Summary

RESEARCH OF MANUAL AND AUTOMATED TESTING

V. Valatkevičius, D. Šešok

This article provides two methods for manual and automated testing effort assessment. Using these methods, the influence on time and cost of manual and automated testing was compared. A study with a real, developed project has revealed that in the long run, automated tests help reduce the time and cost of the testing process.

Keywords: *manual testing, automated testing, use case points, COCOMO II.*

Santrauka

RANKINIO IR AUTOMATINIO TESTAVIMO TYRIMAS

V. Valatkevičius, D. Šešok

Šiame straipsnyje pateikiamos dvi metodikos, kuriomis įvertinama rankinio ir automatinio testavimo pastangos. Taikant šias metodikas buvo palyginta rankinio ir automatinio testavimo įtaka laikui ir kainai. Atliktas tyrimas su realiu, išvystytu projektu parodė, kad ilguoju laikotarpiu automatiniai testai padeda sumažinti testavimo proceso laiką ir kainą.

Prasminiai žodžiai: *rankinis testavimas, automatinis testavimas, panaudos atvejų taškai, COCOMO II.*

Įteikta 2018-04-11
Priimta 2017-05-14