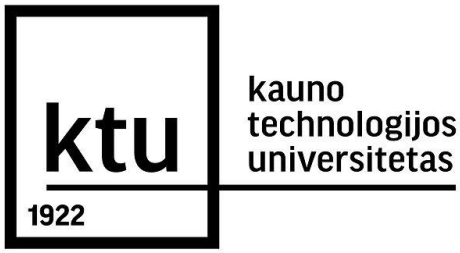

INFORMATIKOS FAKULTETAS

T120B162 Programų sistemų testavimas

1 laboratorinis darbas

Studentė: Gabija Šeškauskaitė, IFF-0/2
Dėstytojai: doc. Šarūnas Packevičius
lekt. Dominykas Barisas

KAUNAS 2023

Turinys

1. Įvadas

Šiame skyriuje pateikiamas įvadas į laboratorinį darbą bei supažindinama su testuojama sistema.

1.1. Laboratorinio darbo tikslai ir uždaviniai

1) Išsiaiškinti testavimo plano struktūrą;
2) Išsiaiškinti, kokie punktai turi būti išpildyti bei kokie punktai yra svarbiausi, norint sudaryti tinkamą programinės įrangos testavimo planą;
3) Sudaryti testavimo planą norimai ištestuoti sistemai;
4) Paruošti laboratorinio darbo ataskaitą.

1.2. Testuojamos sistemos aprašymas

Testuojamas žaidimas kryžiukai-nuliukai. Žaidimas yra žaidžiamas dviejų žaidėjų. Vienas žaidėjas lentelės langelius pildo X, kitas – O. Žaidėjas pirmas užpildęs eilutę, stulpelį ar įstrižainę savo ženklais laimi.

Žaidimas nereikalauja registracijos, užtenka vienam žaidėjui nurodyti žaidimo kambario pavadinimą ir slapyvardį, o kitas žaidėjas įveda žaidimo kambario pavadinimą kuriame nori žaisti ir savo slapyvardį. Žaidėjai taip pat gali pasirinkti kokio dydžio žaidimo lentą nori naudoti, 3x3 arba 4x4. Taip pat galima pridėti kliūtis, W blokuotų langelį kaip siena, o B pašalintų padėtą ženklą.

1.3. Testuojamos sistemos architektūra

Kliento pusė (ang. Front-End) – sukurta naudojant JavaScript ir jQuery technologijas.

Serverio pusė (angl. Back-End) – sukurta naudojant SignalR hub, ASP.NET Core.

2. Sistemos testavimo tikslai

1) Užtikrinti, kad sistema veikia taip, kaip nurodyta specifikacijoje, t.y., atitinka tiek funkcinius, tiek nefunkcinius reikalavimus, išpildo visus kokybės užtikrinimo aspektus bei tenkina visus numatytus panaudojimo atvejus taip, kad klientas galėtų naudotis sistema be atsirandančių klaidų ar sutrikimų;
2) Testavimo metu ne tik užtikrinti tai, kad sistema veiktų taip, kaip buvo numatyta specifikacijoje, bet ir identifikuoti esamas naujas ar galimai sistemos plėtros laikotarpiu atsirasiančias klaidas, apie jas pranešti savo komandai bei jas stengtis ištaisyti iki tol, kol sistema bus pradėta naudotis, ar bent jau numatyti galimas sistemos korekcijas plėtros laikotarpiu, kad surastos naujos klaidos būtų ištaisytos arba jų būtų išvengta.

3. Testavimo prioritetai

Šiame skyriuje pateikiama sistemos testavimo prioritetai bei apibrėžiamas klaidų klasifikavimas pagal jų prioritetus bei svarbą.

3.1. Sistemos testavimo prioritetai

Sistemos testavimo prioritetai yra tokie (nuo svarbiausio iki mažiausiai svarbaus):

1) Pagrindinė žaidimo funkcionalumas - tai apima žaidimo lentos rodomąją dalį, žaidimo eigos logiką ir žaidimo taisykles. Svarbu užtikrinti, kad žaidimas būtų įmanomas, o žaidėjai galėtų atlikti leistinus ėjimus ir žaisti be klaidų.
2) Žaidimo naudotojo sąsaja (UI) - tai apima grafinį žaidimo dizainą, naudotojų sąveikavimą su žaidimu ir užtikrintų nesudėtingą žaidimo naudojimą žaidėjams.
3) Žaidimo greitis - tai apima, kaip greitai žaidimas reaguoja į žaidėjų ėjimus, kaip greitai vykdomi serverio skaičiavimai ir kaip greitai pateikiama informacija žaidėjams.

Pačios sistemos testavimo prioritetai ir klaidų, atsirandančių sistemos kūrimo/naudojimosi metu prioritetai skirstomi kiek skirtingai. Programinės įrangos testavimo procese sistemos klaidos turi du pagrindinius vertinimo kriterijus:

1) Sistemos klaidos prioritetas – tai dydis, kuris parodo, kokia tvarka turi būti ištaisomos atsiradusios sistemos klaidos (ar klaida turi būti ištaisyta dabar, ar gali kiek palaukti). Prioritetas yra nustatomas remiantis kliento reikalavimais. Klaidos prioritetų kategorijos yra pateiktos 2 lentelėje.

1 lentelė. Sistemos klaidų prioritetų klasifikacija

Klaidos prioritetas	Klaidos prioriteto reikšmė	Klaidos prioriteto apibūdinimas
1	Aukštas (High)	Klaida yra pastebima, tačiau jos ištaisymas gali palaukti (jei yra klaidų su aukštesniu prioritetu).
2	Vidutinis (Medium)	Klaida turi būti pataisyta įprasto plėtros procesu metu. Klaidos ištaisymas gali palaukti iki kitos programinės įrangos sukūrimo versijos.
3	Žemas (Low)	Klaida turi būti ištaisyta kaip tik galima greičiau, nes ji daro didelę įtaką sistemos veikimui – sistema naudotis negalima tol, kol ji nebus ištaisyta.

2) Sistemos klaidos svarba – tai dydis, kuris parodo, kaip stipriai klaida gali paveikti sistemą ir jos veikimą.

2 lentelė. Sistemos klaidų svarbos klasifikacija

Klaidos svarba	Klaidos svarbos reikšmė	Klaidos svarbos apibūdinimas
S1	Critical	Klaida nutraukia visos sistemos arba kai kurių jos dalių darbą ir/ar stipriai paveikia bei sugadina sistemos duomenis. Tam, kad pasiektume norimą rezultatą, sistemoje nėra jokių alternatyvių funkcijų.
S2	Major	Klaida nutraukia visos sistemos arba kai kurių jos dalių darbą ir/ar stipriai paveikia bei sugadina sistemos duomenis, tačiau norimam rezultatui pasiekti sistemoje yra kitų būdų.
S3	Moderate	Klaida, kuri nenutraukia sistemos ar jos dalių darbo, tačiau dėl jos sistema pateikia klaidingus rezultatus.
S4	Minor	Klaida, kuri nenutraukia sistemos darbo ar kitaip neigiamai nepaveikia naudojimosi sistema ir norimi rezultatai gali būti pasiekiami kitais būdais.
S5	Cosmetic	Klaidos, kurios nedaro įtakos sistemos veikimui ar naudojimuisi sistema bei yra susijusios su sistemos išvaizda.

4. Prielaidos (būtinos sąlygos testavimui atlikti)

Šiame skyriuje aprašomos būtinos sąlygos, kurios užtikrins sklandų testavimo plano vykdymą.

1) Testuojamos sistemos dokumentacijoje turi būti aprašyti funkciniai ir nefunkciniai reikalavimai;
2) Turi būti numatyta klaidų taisymo strategija (nustatyti klaidų klasifikavimo požymiai – kurios klaidos turi būti taisomos pirmiausia, kaip turi būti skirstomos klaidos bei jų taisymo prioritetai);
3) Turi būti numatyta sistemos testavimo strategija (kokios sistemos vietos, funkcionalumas turi būti testuojamos pirmiausia);

4) Turi būti paruoštas naudotojų sistemos priėmimo testavimo planas (kas/kokiu metu bus suteikiama testuoti galutiniam naudotojui, kokiu būdu bus gaunamas grįžtamasis ryšys apie sistemos veikimą);
5) Prieš kiekvieną konkretų funkcionalumo testavimą, sistemoje privalo būti realizuotas atitinkamas funkcionalumas;
6) Turi būti įdiegta bei tinkamai sukonfigūruota testavimo aplinka.
7) Turi būti paruošta aplinka sistemos klaidų registravimui, darbų paskirstymui (pvz., Jira, GitHub).

5. Testavimo aplinka

Testavimo scenarijai bus atliekami naudojant tokią aparatinę/programinę įrangą (minimalūs reikalavimai):

1) Nešiojamas kompiuteris;
2) Procesorius: i7, 16GB RAM
3) Microsoft Windows 10 OS;
4) Įrašytos naujausios populiariausių interneto naršyklių (Mozilla Firefox, Google Chrome) versijos.

6. Testavimo technikos

Tam, kad sistema būtų ištestuota tinkamai, neužtenka jai parinkti tik vienos kažkurios rūšies testus, turi būti taikomos skirtingos testavimo technikos, vykdomi skirtingi testavimo tipai. Mūsų testuojamai sistemai bus taikomos tokios testavimo technikos:

1) Vienetų (unit) testai – testai, kurie testuos konkrečias kodo vietas, funkcinius vienetus (pvz., atskiras funkcijas, klasių dalis ir pan.). Šie testai bus rašomi programuotojų ir bus kuriami lygiagrečiai kuriant sistemos funkcijas. Šių testų kūrimui numatyta naudoti xUnit biblioteką;
2) Static testai – kurie urie tikrins programos kodą be jo vykdymo. Šie testai bus naudojami funkcinių klaidų aptikimui, kurios gali būti rastos analizuojant kodą be jo paleidimo.
3) Performance testai – testai, kurie padės ištestuoti sistemos elgseną didinant duomenų apkrovas, užklausų kiekį, taip ištestuojant sistemos stabilumą, patikimumą, resursų panaudojimą bei padės identifikuoti sistemos spragas, kurių iš esmės negali parodyti kitų rūšių testai.

7. Testavimo grafikas

3 lentelėje pateiktas kuriamos sistemos testavimo grafikas numatomomis datomis, kada turi būti vykdomas konkretus testavimo etapas.

3 lentelė. Kuriamos sistemos testavimo grafikas

Testavimo procesas	Nuo	Iki
Unit testai	2023-10-18	2023-10-27

Static testai	2023-10-27	2023-11-10
Performance testai	2023-11-10	2023-11-24

8. Testavimo rizikos

Šiame skyriuje pateikiami galimi keblumai, kurie gali atsirasti ir sutrikdyti normalią testavimo proceso eigą. Galimos testavimo rizikos ir jų sprendimo būdai:

1) Blogai sudarytas testavimo grafikas – gali atsitikti taip, kad testavimo grafike nurodyto testavimo procesų laiko gali neužtekti tinkamam sistemos testavimui. Šiai problemai ištaisyti būtų galima perdėlioti testavimo grafiką prailginant atitinkamų testavimo procesų periodus ar pasamdant naujų resursų į komandą, kad testavimą būtų galima suspėti atlikti per numatytą laiką;
2) Neigiami priėmimo testų rezultatai – galutiniams naudotojams (klientams) gali netikti galutinis sistemos vaizdas, naudotojo sąsaja, scenarijų bei užduočių vykdymas naudojantis sistema, todėl gali tekti perdaryti ne vieną sistemos komponentą ar sistemos išvaizdą. Galimam šios problemos sprendimui tinkamas variantas – glaudus bendradarbiavimas su galutiniu naudotoju, jam vis parodant ir duodant išmėginti sistemos funkcionalumą, kad grįžtamasis ryšys būtų gaunamas kiek galima anksčiau.
3) Darbuotojų kompetencijos stoka – tai glaudžiai siejasi su atliekamo testavimo kokybe bei galutinio testavimo rezultatu, nes daugiau patirties turintys darbuotojai gali išvengti kai kurių klaidų, kurios galbūt prasprūstų pro akis mažiau patirties turintiems darbuotojams. Taip pat mažiau patirties turintys darbuotojai yra linkę mažiau tiksliai įvertinti numatomus darbus bei jų vykdymo trukmę, kas gali daryti didelę įtaką testavimo grafikui. Todėl reikia stengtis, kad komandoje būtų bent po vieną mid-senior lygio programuotoją ar testuotoją, kuris galėtų padėti šių klaidų išvengti.
4) Kritinių klaidų neidentifikavimas – turimos omeny tos klaidos, kurios turi didelę reikšmę sistemos veikimui ir kurios testavimo proceso etape nebuvo surastos. Tokio klaidos dažniausiai išryškėja jau galutiniam naudotojui naudojantis sistema. Todėl reikia stengtis testavimo procesus vykdyti nuosekliai bei siekti kuo didesnio sistemos kodo padengimo testais, taip sumažinant tokių klaidų atsiradimo tikimybę.
5) Sistemų (testavimo aplinkų) trikdžiai – testavimo metu gali atsirasti techninių sistemos trikdžių ar programinės įrangos gedimų, kurie prailgintų numatytą testavimo procesų užbaigimo laikotarpį. Tokias problemas numatyti sunku, tačiau jų tikimybę galima sumažinti naudojantis kuo naujesne aparatine įranga, legalia programine įranga bei pasirinkamus tinkamus paslaugų (pvz., debesų kompiuterijos ar interneto) tiekėjus.

9. Testavimo scenarijai

Šiame skyriuje pateikiami testavimo scenarijai, kurie bus vykdomi testuojant kuriamą sistemą.

9.1. Žaidimo veikimo testavimo scenarijai

9.1.1. Žaidimo pradžios scenarijai

Feature: Žaidimo pradžios scenarijai

1. **Scenario: Žaidimo pradžia ir laimėtojo nėra**
    - *Given*: Žaidimo 3x3 lentelė tuščia
    - *When*: Žaidėjas "X" atlieka pirmą ėjimą
    - *Then*: Žaidimas tęsiasi ir nėra laimėtojo
2. **Scenario: Žaidimo pradžia ir žaidėjas "X" laimi**
    - *Given*: Žaidimo 3x3 lentelė tuščia
    - *When*: Žaidėjas "X" atlieka ėjimus ir laimi
    - *Then*: Žaidimas baigiasi, ir žaidėjas "X" laimi
3. **Scenario: Žaidimo pradžia ir žaidėjas "O" laimi**
    - *Given*: Žaidimo 3x3 lentelė tuščia
    - *When*: Žaidėjas "O" atlieka ėjimus ir laimi
    - *Then*: Žaidimas baigiasi, ir žaidėjas "O" laimi
4. **Scenario: Žaidimo pradžia ir laimėtojo nėra**
    - *Given*: Žaidimo 4x4 lentelė tuščia
    - *When*: Žaidėjas "X" atlieka pirmą ėjimą
    - *Then*: Žaidimas tęsiasi ir nėra laimėtojo
5. **Scenario: Žaidimo pradžia ir žaidėjas "X" laimi**
    - *Given*: Žaidimo 4x4 lentelė tuščia
    - *When*: Žaidėjas "X" atlieka ėjimus ir laimi
    - *Then*: Žaidimas baigiasi, ir žaidėjas "X" laimi
6. **Scenario: Žaidimo pradžia ir žaidėjas "O" laimi**
    - *Given*: Žaidimo 4x4 lentelė tuščia
    - *When*: Žaidėjas "O" atlieka ėjimus ir laimi
    - *Then*: Žaidimas baigiasi, ir žaidėjas "O" laimi
7.

9.1.2. Ėjimo scenarijai

Feature: Žaidimo ėjimo scenarijai

4. **Scenario: Žaidėjas "X" atlieka ėjimą**
    - *Given*: Žaidimo lentelė turi langelį, kuriame jau yra "X"
    - *When*: Žaidėjas "X" bando atlikti ėjimą toje pačioje vietoje

- *Then*: Žaidimas eina toliau, ir "X" nepraranda savo eilės

5. **Scenario: Žaidėjas "O" atlieka ėjimą**
    - *Given*: Žaidimo lentelė turi langelį, kuriame jau yra "O"
    - *When*: Žaidėjas "O" bando atlikti ėjimą toje pačioje vietoje
    - *Then*: Žaidimas eina toliau, ir "O" nepraranda savo eilės

9.1.3. Baigimo scenarijai

Feature: Žaidimo baigimo scenarijai

6. **Scenario: Lygiosios rezultatas (lygiosios)**
    - *Given*: Žaidimo lentelė užpildyta, bet nėra laimėtojo
    - *When*: Žaidimas pasibaigia, ir nėra laimėtojo
    - *Then*: Žaidimas baigtas lygiosiomis
7. **Scenario: Žaidėjas "X" laimi horizontaliai**
    - *Given*: Žaidimo lentelėje "X" laimi horizontaliai
    - *When*: Žaidimas pasibaigia
    - *Then*: Žaidėjas "X" laimi
8. **Scenario: Žaidėjas "O" laimi horizontaliai**
    - *Given*: Žaidimo lentelėje "O" laimi horizontaliai
    - *When*: Žaidimas pasibaigia
    - *Then*: Žaidėjas "O" laimi
9. **Scenario: Žaidėjas "X" laimi vertikaliai**
    - *Given*: Žaidimo lentelėje "X" laimi vertikaliai
    - *When*: Žaidimas pasibaigia
    - *Then*: Žaidėjas "X" laimi
10. **Scenario: Žaidėjas "O" laimi vertikaliai**
    - *Given*: Žaidimo lentelėje "O" laimi vertikaliai
    - *When*: Žaidimas pasibaigia
    - *Then*: Žaidėjas "O" laimi
11. **Scenario: Žaidėjas "X" laimi įstrižai**
    - *Given*: Žaidimo lentelėje "X" laimi įstrižai
    - *When*: Žaidimas pasibaigia
    - *Then*: Žaidėjas "X" laimi
12. **Scenario: Žaidėjas "O" laimi įstrižai**
    - *Given*: Žaidimo lentelėje "O" laimi įstrižai
    - *When*: Žaidimas pasibaigia
    - *Then*: Žaidėjas "O" laimi

10. Testavimo rezultatai bei testavimo išbaigtumas

Testavimo proceso pabaigoje turi būti pasiekti tokie rezultatai:

1) Užtikrinta, kad testuojama sistema yra stabili naudotis;
2) Paruoštas užbaigtas testavimo planas;
3) Paruoštas užbaigtas sistemos naudotojų vadovas;
4) Paruoštos kiekvieno testavimo grafike nurodyto testavimo proceso ataskaitos;
5) Užbaigta testuojamos sistemos specifikacija (kuri galėjo keistis po naudotojų atidavimo testų, jei buvo atliekami kokie nors specifikacijoje nenumatyti sistemos pokyčiai).

Sistemos testavimas laikomas išbaigtu, kai:

1) Sistemos užsakovas (jei toks yra) ir sistemos kūrimo komanda tarpusavyje susitaria, jog sistemos testavimas ir jo rezultatai tenkina abi šalis;
2) Kai sistemoje nėra likusių užregistruotų jokių didesnės reikšmės klaidų (svarba S3 ir aukštesnė);
3) Visi automatiniai testai grąžina teigiamą rezultatą (testai įvykdomi sėkmingai);
4) Testuojamo kodo padengimas siekia bent 90%;
5) Kai sistemoje atsirandančių klaidų skaičius tendencingai mažėja ir artėja prie 0;
6) Kai sistema savo darbą atlieka taip pat stabiliai, nepriklausomai nuo augančio sistemos naudotojų skaičiaus.

11. Išvados

Laboratorinio darbo metu buvo:

1) Išanalizuota testavimo plano struktūra;
2) Išskirti svarbiausi testavimo planą sudarantys punktai, jie detalizuoti bei pritaikyti testuojamai sistemai;
3) Sudarytas testuojamos sistemos testavimo planas;
4) Paruošta laboratorinio darbo ataskaita.