Tallinna Ülikool

Informaatika Instituut

Kasutajaliidese testid programmeerimisoskuste hindamiseks

Bakalaureusetöö

Autor: Mait Mikkelsaar

Juhendaja: Jaagup Kippar

Autor: ..... ,, ... ,, 2014

Juhendaja: ..... ,, ... ,, 2014

Instituudi direktor: ..... ,, ... ,, 2014

Tallinn 2014

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

.....	.....
(kuupäev)	(autor)

Sisukord

Sissejuhatus

Rakenduste loomisel kirjutatakse *unit teste*, millega saab kontrollida meetodite ja klasside funktsionaalsust. Samuti kasutatakse *Unit teste* koodi hindamiseks. Mitmed veebikeskkonnad võimaldavad koodi hinnata, näiteks CodeEval. Samalaadse keskkonna on loonud ka Karmo Rosental enda bakalaureuse töös „Programmeerimisoskuste hindamise veebikeskkond“ (Rosental, 2013). Nimetatud keskkonnad tegelevad funktsioonide testimisega, mitte terviklahenduse valideerimisega. See võib tekitada olukorra, kus meetodid on korrektsed, aga neid ei ole õigesti kasutatud. Autor leiab, et on vaja leida moodus, kuidas hinnata terviklahendust. Selleks, et vaadata, kas rakendus käitub nii nagu reaalsetes olukordades peaks, tuleb kasutusele võtta kasutajaliidese testid. Nende abil saab hinnata kuidas käitub veebirakendus, kui kasutaja seda kasutab.

Bakalaureusetöö eesmärgiks on luua kasutajaliidese testide komplektid, millega on võimalik hinnata programmeerimisoskust. Täpsemalt on tegemist näidisülesannete ja neile vastavate testidega. Õppuritele antakse ette ülesannete kirjeldused, mille põhjal nad peavad valmis saama veebirakenduse. Lahenduse valmides antakse testkomplektile lahenduse internetiaadress, mille peale rakendus teostab ridu kasutajaliidese teste. Rakenduse töö lõppedes tagastatakse tulemus, kus on kirjas kuidas programmeerijal läks, ja millised on probleemsed kohad. Vastavalt tulemusele on võimalik hinnata lahendaja oskusi.

Idee töö teema valikuks sai alguse pärast Selenide andmeteegiga tutvumist. Autor osales Rosentali veebikeskkonna testimises ja proovis programmi meetodite hindamist. Pool aastat pärast seda tutvus autor Selenidega, mis tegeleb kasutajaliidese automaattestidega, ja leidis, et seda teeki saab kasutada programmeerimisülesannete hindamisel. Rääkides enda ideest juhendajale, leidis juhendaja, et selline rakendus võib huvi pakkuda Kutsekojale. Suheldes Kutsekoja esindajaga jõuti järeldusele, et selline kutsetestide automaatse hindamise võimalus on vajalik.

Võõrkeelsete lühendite loetelu

Loetelu on koostatud tuginedes Vallaste sõnastikule (Vallaste, 2013).

API – *Application Programming Interface*, rakendusliides, programmiliides, API-liides. Arvuti operatsioonisüsteemiga või rakendusprogrammiga määratud reeglistik, mille alusel rakendusprogramm kasutab operatsioonisüsteemi või teise rakendusprogrammi teenuseid.

Ajax – *Asynchronous JavaScript And XML*, asünkroonne JavaScript ja XML. Sellega tähistatakse selliste interaktiivsete veebirakenduste loomise meetodit, kus toimub jooksev kulissidetagune andmevahetus brauseri ja veebiserveri vahel nii, et kasutaja iga liigutuse peale pole vaja kogu veebilehte uuesti alla laadida.

jQuery – skriptikeele JavaScript andmeteek. jQuery võimaldab lihtsamini kirjutada JavaScript'i koodi - nimelt teeb see lihtsamaks veebilehtede manipulatsiooni, sündmuste käitlemise, animatsiooni ja Ajaxi kasutamise.

Draiver – sündmusi töötleva programmi põhisilmus, käske vastuvõttev ja neid täitmiseks väljasaatev kood.

JVM – *Java Virtual Machine*, Java virtuaalmasin. Tarkvara, mis interpreteerib Java programmikeeles kirjutatud programme, mis on kompileeritud baitkoodideks ning salvestatud ".class" laiendiga failidena.

XPath – *XML PATH Language*, XSL'i komponent, mida kasutatakse sildistatud XML-elementide identifitseerimiseks. Seda kasutatakse ka teheteks arvudega ja manipulatsioonideks stringidega.

HTML – *HyperText Markup Language*, hüpertekst-märgistuskeel. Enimlevinud kodeerimissüsteem (tekstivorming) veebidokumentide loomiseks. HTML koodid ehk märgendid määravad ära selle, kuidas veebileht arvutiekraanil välja näeb.

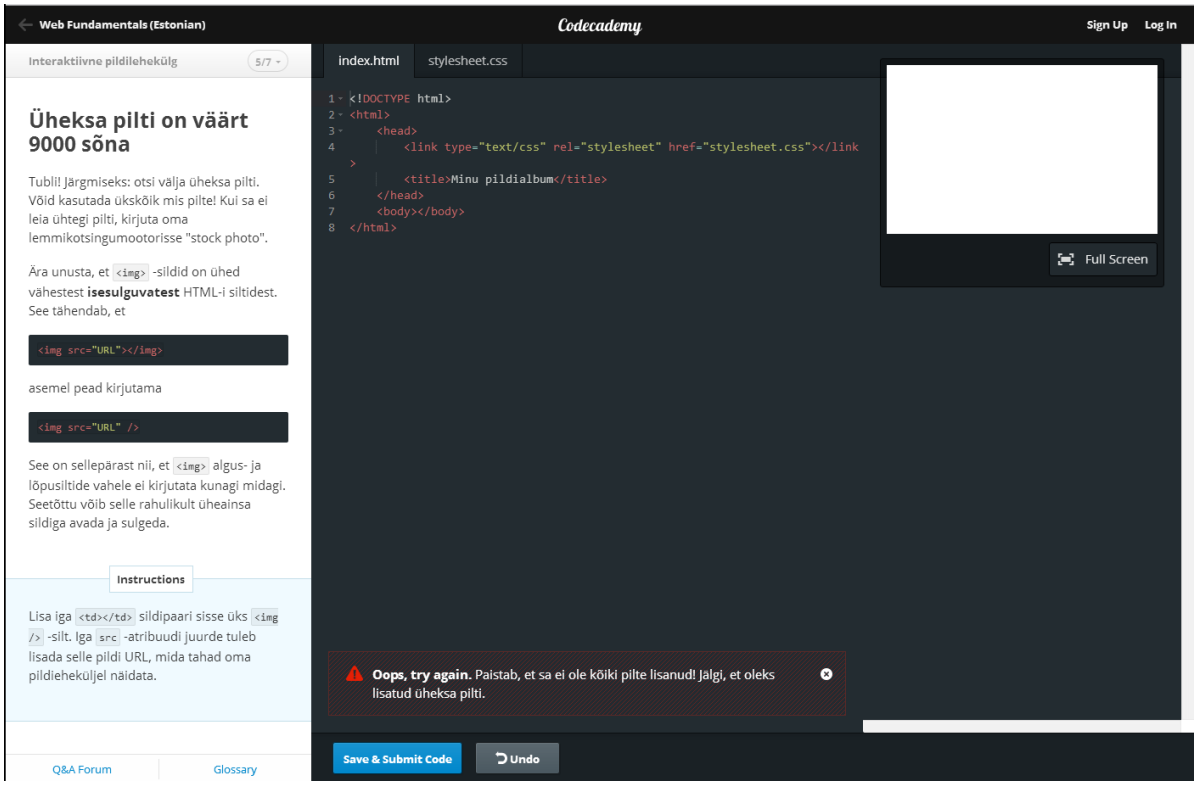
1 Sarnased rakendused

Koodi automaattestimisega tegelevad keskkondi on mitmeid, aga ükski selline keskkond ei tegele ülesannete kasutajaliidese testimisega. Kuna sarnased kasutajaliidese testimise keskkonnad puuduvad, siis autor toob välja mõned koodi funktsionaalsust testivad meediumid.

1.1. Codecademy

Codecademy on ettevõte, mis keskendub haridusele ja mille eesmärgiks on luua parim õppimise kogemus. Codecademy pakub veebipõhist haridust, kus kommuuni liikmed on loonud kümneid tuhandeid kursuseid ja neid kursuseid on läbitud miljoneid kordi (Codecademy, 2014).

Codecademy kursused on valdavalt ingliskeelsed. Teise keelena on kasutusel eesti keel. Eestikeelsed kursused on HTML/CSS ja JavaScript, mis on kättesaadavad aadressitel http://www.codecademy.com/tracks/web-et ja http://www.codecademy.com/tracks/javascript-et. Kursuse ülesannete tekstid on eestikeelsed, aga raamistik ümber on ikkagi ingliskeelne, seega esinevad kursuse kasutajaliideses osad erinevates keeltes (Joonis 1).



Joonis 1. Codecademy

Kuna eesmärgiks on luua rakenduse kasutajaliidest hindavad testid ja nimetatud keskkond on õpetava suunitlusega, siis ei ole antud meediumit mõtet vaadata funktsionaalsuse osas . Põhjus, miks autor võttis Codecademy uurimise objektiks, on keskkonna eestikeelsete ülesannete analüüs.

1.2. Codility

Codility keskkonna eesmärgiks on aidata ettevõtetel palgata paremaid programmeerijaid ning ülemaailmne programmeerimisoskuste edendamine. Kasutades automaatteste ja hindamisteenuseid võimaldab Codility kokku hoida värbajate aega uute töötajate palkamisel. Kasutusel olevad automaattestid sõeluvad välja kuni 90% kandidaatidest, mis hoiab kokku aega ja tagab kompetentsemad personali liikmed. Keskkond on tasuline, millel on üks tasuta näidisülesanne.

Codility kasutajad saadavad kandidaatidele testi, mille nad saavad kokku panna programmeerimiskeelele vastavatest ülesannetest. Lahendatud ja hinnatud testi kohta saab värbaja vaadata aruannet (Joonis 2). Aruandes on kirjas näiteks kandidaadi punktide arv, tema tulemus võrreldes teiste testi lahendajatega, lahendus ja veel palju detailset informatsiooni (Codility Ltd., 2013).

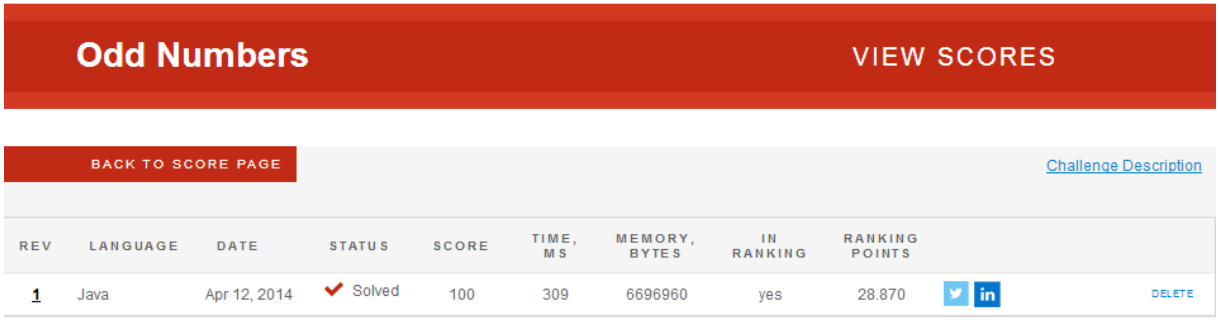
Joonis 2. Codility aruanne

1.3. CodeEval

CodeEval on programmeerijate poolt kasutatav keskkond, kus nad saavad demonstreerida enda oskusi. Arendajad saavad osaleda rakenduste arendamise võistlustel ja võita seeläbi auhindu. Samuti annab lahenduste loomine võimaluse tööandjatele muljet avaldada. Arendajad asuvad ka CodeEval edetabelites, kus nad saavad ennast võrrelda teiste kasutajatega. Tööandjatele on aga ülesannete koostamine ja võistluste loomine võimaluseks end parimatele programmeerijatele tutvustada.

CodeEval keskkonnas oli veebilehe külastamise hetkel ligikaudu 140 ülesannet, mis olid jaotatud kolmeks raskusastmeks. Ülesanded on kirja pandud lühidalt ja seega on neist lihtne aru saada. Kasutajal on võimalik enda poolt loodud ülesande lahendus kirja panna veebiredaktoris või laadida üles lahenduse faili. Lahenduse saab esitada 17 erinevas programmeerimiskeeles: C, C#, C++, Objective-C, Java, JavaScript, Python, Ruby, PHP, Perl, Go, Haskell, Scala, Clojure, Bash, Lua ja Tcl. Lai programmeerimiskeelte valik toob endaga kaasa ka probleemi, nimelt on erinevatel keeltel erinevad vormistamise nõuded. Vormistamisel peab järgima vastava keele nõudeid ja need nõuded ei kuulu lahenduse juurde (CodeEval, 2014).

Peale lahenduse esitamist saab kasutaja näha raportit, kus on kirjas mitu punkti ta antud ülesande eest sai. Raport sisaldab lisaks ka lahenduse koodi tööaega ja kasutatud mälu mahtu (Joonis 3).



Joonis 3. CodeEval lahenduse raport

2 Kasutajaliidese testide nõuded

Selles peatükis annab autor ülevaate rakenduse nõuetest, mis on saadud Kutsekoja esindajalt ning on lisatud täiendavad nõuded autori enda poolt.

2.1 Baasnõuded

Koostatav kasutajaliidese testimise rakendus on suunatud Tarkvaraarendaja kutsestandardi taseme 4 testimiseks. Täpsemalt, rakendub graafiline automaattestimine antud kutsestandardi punktile B.2.7 oskuste h-lõikele: „oskab kasutada mõne põhivoolu programmeerimiskeelt tasemel, mis võimaldab iseseisvalt olemasolevat koodi lugeda ja uut koodi luua“ (Sihtasutus Kutsekoda, 2011).

Kutsekoja esindaja nõuded (U. Mets, isiklik side, 11. märts, 2014):

- Ülesande keerukus, mis vastaks kutsekooli tasemele
- Piiratud lahendamise aeg – maksimum 2 tundi
- Rohkem kui üks ülesanne

2.2 Täiendavad nõuded

Baasnõuded on väga üldsõnalised ja seega lisas autor täiendavad nõuded, mis tema arvates on vajalikud hea testkomplekti puhul:

- Rakenduse tööle panemine peab olema lihtne
- Testid on iseseisvad ja ei sõltu eelnevatest testidest
- Testida saab erinevates programmeerimiskeeltes loodud veebilehtesid
- Testid on lihtsasti mõistetavad nii hindamise läbiviijale kui ka õppurile

3 Testide rakenduse kavandamine

Rakenduse loomiseks valis autor Java programmeerimiskeele ja Selenide andmeteegi, mis on ühe kõige populaarsema kasutajaliidese testimise arendustööristade komplekti, Seleniumi, kapsel. Arendustööriistade valikus mängis rolli Selenide lihtsus ja arusaadavus.

3.1 Selenide

Selenide on Seleniumi veebiliidese pakend, mis võimaldab järgnevaid eeliseid (Solntsev, 2014):

- Sisutihe testide API
- Ajaxi tugi
- Veebilehitseja automaatkäivitus
- jQuery stiilis elementide valimine

Need olid näited Selenide eelistest, mis on välja toodud andmeteegi ametlikul veebilehel. Tegelikkuses on neid eeliseid veelgi – üheks tähtsamaks võimaluseks peab autor automaatset ekraanitõmmise piltide tegemist. Pilt tehakse hetkel kui test läbi kukub. Testimise lõppedes saab iga läbikukkunud testi kirjelduse juurde vaadata ka ekraanipilti, mis on jäädvustatud läbikukkumise hetkel. Lisaks pildile salvestatakse ka leht ülesmärkimiskeeles HTML, mis annab ülevaate HTML koodist, mis pildi salvestamise hetkel näha oli.

Kasutades sama eesmärgi saavutamiseks Seleniumi ja Selenide, siis tulemuseks on, et Selenidega kirjutatud kood on oluliselt lühem ja autori arvates ka arusaadavam (Koodinäide 1). Juurde tulevatest võimalustest toob autor välja ka elemendi leidmise teksti abil – võimalus, mis Seleniumi puhul puudub, kui just ei võta juurde kasutusele Xpath'i (Koodinäide 2) (Codeborne).

```
SELENIUM:
try{
      WebElement element = driver.findElement(By.id("lahku-nupp"));
      fail("Element ei tohiks eksisteerida: " + element);
}catch(WebDriverException itsOk) {}

SELENIDE:
$("# lahku-nupp").shouldNot(exist);
```

Koodinäide 1. Seleniumi ja Selenide koodi mahu võrdlus

SELENIUM:

```
(puuudub)
```

SELENIDE:

```
$(byText("Ostan heas korras Chrysleri võre")).should(exist);
```

Koodinäide 2. Selenide ja Seleniumi erinevus elemendi leidmisel

3.2 Näidisülesanded

Kutsekojal puuduvad standardsed kutseoskuste hindamise ülesanded. Olemasolevate testide puudumisel koostas autor näidisülesanded, mis sobiksid oskuste hindamiseks ning kasutajaliidese testid, mis neile ülesannetele vastavad. Autor koostas kolme erinevat tüüpi ülesanded:

- olemasoleva koodi täiendamine,
- uue, lihtsa ja väikese veebirakenduse loomine,
- uue, keerukama ja mahukama veebilehestiku loomine.

Ülesannete kirjeldused on detailsed ja pikad, kus on kirjas lisaks ülesande üldisele kirjeldusele ka elementide id-de ja klasside nimed. Elementide nimed tuleb ette anda kasutajaliidese testide tõttu – pikemalt selgitab autor seda kasutajaliidese testide ülesehituse juures.

Koostatud rakendused ei ole mõeldud koheselt programmeerimisoskuste hindamise juures rakendamiseks, vaid annavad ülevaate, kuidas kasutajaliidese testide põhjal hindamine käib.

3.2.1 Lihtsa veebirakenduse loomine

Lihtsa veebirakenduse näiteks võttis autor tollikalkulaatori loomise. Lahendusega saab arvutada tolle sentimeetriteks ja vastupidi. Vigase sisendi korral annab ka veateate. Rakendus asub ühel lehel ja sellel on 5 elementi – sisendi kast, rippmenüü konverteerimise valiku tegemiseks, sisestuse nupp, vastus ja veateade.

3.2.2 Keeruka veebilehestiku loomine

Keeruka veebilehestiku loomise eesmärgiks on valmis saada autoosade müümise ja ostmisega tegelev veebilehestik. Lehestik koosneb üheksast lehest. Lehed, kus on vaja sisestada autoosa või auto mark, on sama ülesehitusega, nagu ka ostu ja müügi lehed. Erineva sisuga lehti on seega seitse. Lehestiku jaoks on vaja teha kaks rolli – administraator ja kasutaja. Kummalgi rollil on nende hallatavad lehed.

Ülesandega saab kontrollida nii sisselogimise süsteemi loomise oskust kui ka andmebaasi kasutamise oskust. Andmebaasi juures tuleb kasutada nii andmetabeli manipulatsioone kui ka mitme tabeli sidumist.

Antud ülesanne on aeganõudev ja sellest tulenevalt tuleb enne sarnase ülesande kasutusele võtmist programmeerimisoskuste hindamise juures teha mitmeid katseid hindamaks selle sobivust.

3.2.3 Olemasoleva koodi täiendamine

Koodi täiendamise ülesandeks on postkontorite võrgustiku veebilehestiku loomine, täiendades olemasolevat koodi. Ette antud olemasolevas lehestikus on paki sisestamise leht, selle asukoha vaatamise leht, keskkontori –ja ühe postkontori leht. Juurde tuleb teha kuus kontori lehte, mis kõik põhinevad olemasoleval postkontori lehel. Kontorite lehtedel on paki väljastamine ja saatmine keskkontorisse. Keskkontori lehel saab väljastada ja lisaks ka edasi saata harukontoritesse.

Ülesande raskuspunktiks on andmebaasiga ühendamine. Näidislehtedel on paigas vormid ja tabelid, mis tähendab, et lahendajal tuleb lisada vaid andmete näitamine õigetel kohtadel.

3.3 Testide ülesehitus

Autor võttis eesmärgiks luua kasutajaliidese testid, millega saab kontrollida veebilehti, sõltumata programmeerimiskeelest, mida on kasutatud lehtede loomiseks. Testid otsivad elemente id ja klasside kaudu, sellest tulenevalt ei ole vahet, millist keelt on kasutatud lehe loomiseks. Ainuke vahe on veebilehtede faililaiendites ja sellest tulenevalt on käivitamisel vaja faililaiend parameetrina kaasa anda.

Koostatud kasutajaliidese testid otsivad elemente id ja klassi kaudu. Selenide andmeteek võimaldab kasutada oluliselt rohkem märgendeid aga ühtse stiili hoidmiseks ja selle jaoks, et ülesande kirjeldus ei veniks liiga pikaks peab piirama erinevate märgendite kasutamist. Id-d on kasutusel üksikute elementide kättesaamiseks; klassid on kasutuses tabelitest ja loenditest elementide saamiseks.

Nimedes on autor kasutanud Google HTML/CSS Style Guide juhiseid, mis tagavad koodi parema loetavuse ja kvaliteedi. Teisisõnu on id ja klasside nimedes kasutatud väikeseid tähti ja sõnade vahel on sidekriips (Google, 2014).

Tulenevalt java testimise andmeteegi JUnit eripäradest ei ole võimalik üheselt ja selgelt määrata testmeetodite käivitumise järjekorda. Alates JUnit versioonist 4.11 on võimalik testide klassi ees kasutada märget *@FixMethodOrder*, mis võimaldab muuta testmeetodite järjekorda (JUnit team, 2012).

Eelnevalt mainitud märke juures tuleb täpsustada millist kolmest võimalikust sorteerimismeetodist on kasutatud (JUnit, 2012):

- DEFAULT
  Sorteerib testmeetodid deterministlikusse kuid mitte ettearvatavasse järjekorda
- JVM
  Jätab testmeetodid JVM'ist tagastatud järjekorda.
- NAME_ASCENDING
  Seab testmeetodid, nende nimede järgi, tähestikulisse järjekorda.

Autor leiab, et antud sorteerimise võimalused ei ole sobivad ja seega on loodavad testid iseseisvad ehk nad ei ole sõltuvad eelmistest testidest. Hindamise seisukohalt on ka õigem, et kasutajaliidese testid ei oleks üksteisest sõltuvad. Näiteks ei jää nii ülejäänud punktid saamata, kui loogilises järjekorras esimene test peaks ebaõnnestuma.

Testide komplektid on pakitud jar faililaiendiga pakkidesse, mida saab käsurealt käima lasta. Käivitatava jar paki eelis on see, et failid on ühes kohas ja käivitamine toimub ühe lihtsa käsureaga. Nimetatud käsureast on pikemalt kirjas rakenduse arenduse peatükis.

3.4 Tulemuse kuvamine

Koostatavas rakenduses kuvatakse tulemus konsooli aknasse. Autor ei pea vajalikuks pöörata rõhku testide tulemuse kuvamise dekoreerimisele tarkvara alfaversioonis. Tagasisides saab hindaja näha järgnevaid kirjeid:

- Testide koguarv
- Läbi kukkunud testide arv

Eelnevale teabele järgneb iga läbi kukkunud testi nimi ja selle täpsem kirjeldus:

- Vea põhjus
- Vea kirjeldus

- Eeldus, missuguses olekus element on
- Vea hetkel jäädvustatud ekraanipilt

4 Rakenduse arendus

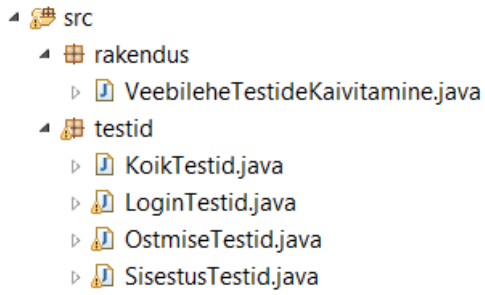
Selles peatükis kirjeldab autor, kuidas ta koostas kasutajaliidese testid programmeerimisoskuste hindamiseks.

4.1 Arenduskeskkonna seadistamine

Arenduskeskkonnaks valis autor Eclipse versiooni 4.2.1. Rakenduse loomise hetkel oli väljas ka versioon 4.3.2, aga uuema väljalaske kasutusele võtmine ei anna juurde funktsionaalsust, mida antud testide komplekti koostamisel kasutada (The Eclipse Foundation, Inc.,2014).

Testide kirjutamiseks lisab autor Selenide andmeteegi. Selenide'ga tutvumise käigus selgus, et vajalik on lisada ka Seleniumi ja sellega kaasa tulevad teegid (Selenium, 2014).

Loodud projekti failid on jaotatud kahte paketti: rakendus ja testid (Joonis 4). Esimeses paketis on rakenduse osa mis käivitab testid ja annab väljundi vastavalt testide tulemusele. Teine pakett sisaldab ühte või mitut testide faili – olenevalt testitavast ülesandest.



Joonis 4. Projekti paketid

4.2 Arenduse protsess

Arenduse protsessi võib jagada nelja etappi: testide komplekti arendus, esmane testimine, rakenduse täiendamine, teine testimine. Esimeses arenduse astmes koostas autor keeruka veebilehestiku loomise ülesande, ülesande ühe võimaliku lahenduse ja selle ülesande lahendust kontrolliva testide komplekti. Näidislahendused on lisatud bakalaureuse tööga kaasas olevale andmekandjale.

4.2.1 Testide komplekti arendus

Esimeseks ülesandeks, millele teste kirjutama hakata, valis autor auto varuosade veebilehestiku loomise. Lehestikku saab kasutada ainult sisse loginud kasutaja – seega tuli esmalt kontrollida, kas logimine töötab. Kuus esimest testi seda kontrollivadki. Testid ise näevad välja suhteliselt lihtsad ja loetavad (Koodinäide 3).

```
@Test
public void AdminLoginSucceedTest() {
	open("/");
	$("#kasutajanimi").setValue("Juku");
	$("#parool").setValue("kala");
	$("#login").click();
	$("#vale-kasutaja").shouldNotBe(visible);
	$("#login-div").shouldBe(present);
	$("#login-div").shouldHave(text("Tere, Juku"));
	$("#lahku-nupp").click();
}
```

Koodinäide 3. Kasutajaliidese testi näide

Järgnevalt kirjutas autor testmeetodid andmete sisestamiste kohta. Need testid kontrollivad, kas andmeid saab baasi sisestada ja kas sisestused kuvatakse õigetes kohtades. Üheteistkümnenda testi kirjutamise juures, kus tuleb kasutada mõlemat rolli, mõistis autor, et koodi lihtsustamise mõttes tuleb sisselogimise jaoks teha eraldi meetod (Koodinäide 4).

```
 private void adminLogin() {
	open("/");
	$("#kasutajanimi").setValue("Juku");
	$("#parool").setValue("kala");
	$("#login").click();
}
```

Koodinäide 4. Sisselogimise meetod

Auto varuosade veebilehestiku jaoks koostas autor kokku 18 testi, millega hinnata lehestiku vastavust nõuetele. Nii mitme kasutajaliidese testi läbimine võtab palju aega, siis autor jaotas need kolmeks, et vajadusel käivitada ka väiksemat testide üksust. Need kolm osa on login-, sisestuse- ja ostmise testid (Joonis 4). Login ja sisestuse sisu on seletatud eelmistes peatükkides, ostmise failis on autoosade ostmise ja müümise kontroll, mis hõlmab peaaegu kõiki muid teste.

Ülesande võib lahendada erinevates programmeerimiskeeltes, seega tuli lisada ette antav parameeter, mis tähistab faililaiendit. Lisatud sai ka piltide kataloogi nimetamise- ja veebilehitseja valiku võimalus. Selenide toetab küll mitut erinevat veebilehitsejat, aga automaatselt leiab ta üles ainult vaikelehitseja– Firefox– draiveri. Teiste jaoks on vaja ette anda draiveri asukoht. Autor võimaldas Chrome veebilehitseja kasutamise, mille asukoht antakse jällegi ette parameetriga.

Selleks hetkeks oli testimise rakenduse käivitamine võimalik ainult läbi arenduskeskkonna - järgmine samm oli võimaldada see tegevus käsurealt. Rakenduse käivitamise klassiks on VeebileheTestideKaivitamine, kus peale testide jooksutamist saab kätte läbi kukkunud testide loendi ja see töödeldakse sobivaks väljundiks. Väljundis on kirjas vea põhjus, -kirjeldus, eeldus, mida oodati, ja ekraanipildi nimi.

Kogu projekt on kokku pakitud käivitatavaks jar failiks, mille saab avada käsurealt. Käsureal tuleb ette anda järgnevad parameetrid: lehestiku kausta aadress, piltide kausta nimi (kausta puudumisel tekitatakse kaust juurde), faililaiend, testide komplekti number ja kui ei ole soovi kasutada Firefox lehitsejat, siis ka Chrome *draiver*'i aadress (Näide 1). Testide komplekti numbrid on 0 kuni 4, mis on vastavalt kõikide-, sisse logimise-, sisestuse- ja ostmise testide numbrid.

```
java -jar AutoLehestikuTestid.jar http://localhost/kutsetestid/autoosad
pildid .php 0 C://chromedriver.exe
```

Näide 1. Autolehestiku testide käivitamise käsurea näide

4.2.2 Esmane testimine

Autor testis programmeerimisülesande korrektsust ja kasutajaliidese testide arusaadavust Tallinna Haapsalu Kolledžis. Testgrupiks oli 15 liikmeline veebiprogrammeerimise kursust lõpetav õppurite rühm. Testimine kestis ligikaudu 6 tundi, mille lõpus saatsid tudengid autorile vabas vormis tagasiside koos valminud veebilehestiku lingiga.

Testimine toimus 28. märtsil 2014. aastal. Autor andis testijatele kätte autoosade veebilehestiku näidisülesande detailse kirjelduse ja kasutajaliideste testide faili. Koos testide failiga tuli kaas anda ka käsurea rida, millega saab testid käima lasta. Autor andis ka jooksvalt juhiseid, kuidas probleemseid olukordi lahendada.

Esimese tunni jooksul, kui ülesannet lahendati, sai selgeks, et kogu veebilehestiku koostamine osutub liiga mahukaks ja aeganõudvaks. Nagu mainitud kestis testimine 6 tundi ja selle aja jooksul suutis enamik õppureid tööle saada logimise ja mõned sisestuse osad.

Ülesande kirjelduse juures oleks pidanud olema ka üldisem kirjeldus. Üldisem kirjeldus oleks andnud õpilastele ülevaate sellest, mis on oodatav lõpptulemus. Esialgne kirjeldus oli detailne ja mitmel tudengil jäi arusaamatuks millist veebilehestikku oodati.

Autor eeldas, et ülesannete lahendamise raskuspunktiks saab testide tulemuste mõistmine. Kohapeal olles tuli korduvalt lahti mõtestada testi tulemustes kirjeldatut. Tagasisisdes aga mainiti enim, et ülesande kirjeldusest oli raske aru saada ja testid olid arusaadavad. Antud õpilased ei olnud varem kokku puutunud id'de ja klassidega, mis raskendas ka ülesande kirjelduse mõistmist. Sellest saab järeldada, et ülesannete korrektseks lahendamiseks peab olema varasem kokkupuude nii id kui klassidega.

4.2.3 Rakenduse täiendamine

Esimese asjana täiendas autor ülesannete kirjeldust, kus parandas pisivead ja lisas algusesse oodatava tulemuse lühikirjelduse. See muudab ülesande teksti pikemaks aga see on vajalik, et oleks lihtsalt võimalik hoomata tervet ülesannet.

Testides ei tulnud erilisi muutusi teha. Ainuke muutus on sisestuste kontrollimise juures. Päringu tulemusena on tavaliselt viimane sisestus ka viimasel kohal, kui ei ole kasutatud mingit sorteerimise- või muud järjekorda muutvat funktsiooni. Eelnevalt kontrollisid testid, et kas viimane sisestus on ka tabelis viimasel kohal olemas, siis nüüd muutis autor seda selliselt, et kontrollitakse, kas tabelis üldse on vajalikud andmed olemas.

Peale keeruka veebilehestiku loomise ülesande täiendamist võttis autor ette lihtsa rakenduse loomise, milleks oli tollikalkulaatori tegemine. See rakendus pidi olema suheteliselt kiiresti lahendatav aga samas peab saama sellele kirjutada mitmeid teste, et hinnata lahendaja oskusi. Lihtsa veebirakenduse testimine hõlmab vähem kasutajaliidese teste ja seega on ka testkomplekti käivitamine lihtsam. Parameetriteks tuleb kaasa anda faili asukoht, ekraanipiltide kausta nimi ja kui on soov kasutada Chrome veebilehitsejat, siis ka selle veebilehitseja draiveri asukoht (Näide 2).

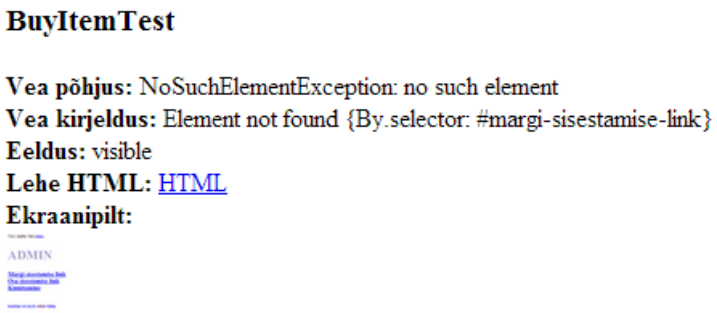
```
java -jar TolliKalkulaatoriTestid.jar
http://localhost/kutsetestid/tollikalkulaator pildid C://chromedriver.exe
```

Näide 2. Tollikalkulaatori testide käivitamise käsurea näide

Viimaseks ülesandeks, mis koostatud sai, oli koodi täiendamise ülesanne. Eesmärks oli valmis saada postkontorite võrgustiku lehestik. Lahendajale on ette antud 4 veebilehte. Paki sisestamise- ja väljastamise leht, keskkontor ja üks harukontori lehtedest, mille põhjal saab koostada ka ülejäänud harukontorite lehed. Nendel neljal lehel on olemas kõik vajalikud

elemendid, aga neil puudub funktsionaalsus. Kõikidel vajalikel elementidel on olemas id'd ja klassid, seega ei tohiks probleemiks tulla, et arendaja seob need valede elementidega. Lehed on kirjutatud php's seega on käsurida, mis käivitamisel tuleb ette anda, sarnane tollikalkulaatori omaga (Näide 2), muutes ära testkomplekti nime.

Autor leidis, et testide tulemuste näitamist tuleb paremaks muuta, kuna nende lugemine käsurealt on raskendatud. Paremaks mõistmiseks kirjutab rakendus ülesmärkimiskeeles HTML vastusete faili, kus on kirjas tulemus ja vigade detailsem kirjeldus (Joonis 5). Läbi kukkunud testide kirjelduse juurde on lisatud ka vea hetkel salvestatud lehe HTML kood. Hindamiseks ei ole vigade detailne kirjeldus oluline, aga kui arendajaga tekib arutelu lahenduse üle, siis on lihtne vea koht üle vaadata. Tulemuste juures ekraanipildile vajutades avaneb pilt suuremalt. Loodud lehelt on tulemuste lugemine lihtsam ja kiirem.



Joonis 5. Läbi kukkunud test

4.2.4 Teine testimine

Teiseks testgrupiks valis autor mugavusvalimiga Tallinna Ülikooli tudengid. Grupis on esimese, teise ja kolmanda kursuse tudengid. Tagasiside andis 3 kolmanda kursuse tudengit. Lahendused olid kirjutatud nii php's, javascriptis kui ka .NET'is ja huvitav oli ka see, et üks tudeng kasutas andmebaasi asemel andmete hoiustamiseks sessiooni.

Töö kirjutamise hetkeks oli võimalik testida kolme tudengi lahendusi. Esimene tudeng lahendas kõik kolm ülesannet, teine lahendas koodi täiendamise ja lihtsa veebirakenduse ülesande ja kolmas ainult lihtsa veebirakenduse oma.

Lihtsa veebirakenduse loomises, milleks oli tollikalkulaatori tegemine, ei saanud keegi rohkem kui pooltest testidest läbi. Kontrollteste oli kaheksa ja seega, kui tuleb sisse lihtne viga, näiteks rippmenüü valikute nimetused, siis mõjutab see suuresti lõpptulemust. Autor proovis lahendused käsitsi läbi testida ja leidis, et lahendused andsid küll õigeid vastuseid aga

ei vastanud ülesande kirjeldusele. Kuigi nii väheste lahendajate põhjal ei saa seda kindlalt väita, jääb autorile mulje, et kõige raskemaks osutub ülesande kirjelduse mõistmine.

Teine testimine kestab edasi ka peale antud bakalaureusetöö esitamist.

4.3 Hindamine

Programmeerimisülesannete lahendamine on kutsestandardi omistamise üks osa ja seega ei hakka autor keskenduma õppuritele hinnangu andmisele. Antud programmikoodi automaattestimine annab kutsekomisjonile teada, kui suures mahus on isik ülesannet lahendanud ja komisjon saab selle teadmise lisada teiste kompetentsusnõuate arvestamise juurde, mille põhjal saab väljastada kutsekvalifikatsioon.

5 Edasiarendus

Bakalaureuse töös valminud kasutajaliidese testide komplekt on algeline versioon sellest, milline see võiks lõpuks välja näha. Esimese asjana tuleks igale testile anda punktide arv, vastavalt selle raskusastmele. Hetkel on kõik testid ühe väärtusega, mis oskuste hindamise seisukohalt on väär.

Järgnevalt võib mõelda läbi kukkunud testide kirjelduse täiendamisele näiteks täpsem veateade ja lehekülje aadressi lisamisele. Vastuste leht tuleb viia ühtsele keelele, olgu selleks siis eesti-, vene-, või inglise keel. Autor ise leiab, et mõistlik oleks viia vigade kirjelduse osa inglise keelele, kuna koodi lugemine inglise keeles on harjumuspärane, ja tulemuse osa, kus on kirjas punktid või läbitud testide arv, esitada eesti keeles.

Edasi oleks autoril soov panna rakendus serverisse ja teha sellele keskkond. Kasutajaliidese kaudu saab rakendusele ette anda veebilehe aardessi ja pärast testide läbiviimist on võimalik vaadata tulemust. Tulemuses on kirjas saadud punktide arv ja lahendamise kuupäev, sarnaselt CodeEval keskkonnale. Lisaks võiks olla iga lahenduse juures välja toodud, kui mitmes tulemus on, võrreldes teiste arendajatega, nagu on välja toodud Codility's.

Vajalik on suurem ülesannete ja nendele vastavate kasutajaliidese testide komplektide hulk. Raskuspunktiks on ülesande sõnastamine selliselt, et seda ei ole võimalik mitmeti mõista. Teine testimine tõi välja, et isegi lühikeste ülesannete kirjelduste ja lihtsate rakenduste puhul mängib sõnastus olulist rolli.

Kokkuvõte

Bakalaureusetöö eesmärgiks oli arendada programmeerimisoskust hindavate kasutajaliidese testide komplekt. Eesmärgi saavutamiseks lõi autor näidisülesanded, mille põhjal tuleb lahendajatel luua veebirakendused. Järgnevalt koostati testide komplektid, mis kontrollivad lahenduse vastavust ülesande kirjeldusele. Peale lahenduse testimist saab anda hinnangu programmeerija oskustele.

Autor teostas 2 testimist, et hinnata ülesannete ja vastavate kasutajaliidese testide sobilikkust. Esimeses testimises käsitleti keeruka veebilehestiku loomist ja selles osales 15 õppurit. Testimise tulemuste põhjal viidi sisse vajalikud muudatused tarkvaras ja alustati teise ringiga tarkvara elutsüklist. Teine testimine on käimas töö valmimise hetkel ja autor kirjeldas tulemusi, mis selleks hetkeks olid selgunud. Testimiste järgi võib öelda, et testkomplekte on lihtne kasutada ja annavad adekvaatset informatsiooni lahenduste korrektsuse kohta. Lahendajate arvates võiks sellist hindamist rakendada ka tulevikus ja samalaadsete lahenduste abil teostada loengutes testidel põhinevat tarkvara arendust.

Töö käigus loodud testkomplektid ja ülesanded vajavad täiendusi ja sellega autor jätkab. Edasiarenduse ideed on välja toodud töö lõpus. Kindlasti on vaja pidada nõupidamisi kutsekomisjoniga, et luua sobivamad ülesanded kutseoskuste hindamiseks. Vaja on luua täpsemad standardid, millised ülesanded ja oskused on vajalikud arendajale, et ta vastaks tarkvaraarendaja tasemele. Praegune kutsestandardi eeskiri on liiga üldsõnaline.

Käesoleva töö tegemisel leidis autor, et samalaadseid testide komplekte saab kasutada ka kursuste eksamite või kontrolltööde raames loodud rakenduste töö kontrollimisel. Selline testimine lihtsustab ja kiirendab oluliselt õppejõu tööd lahenduste korrektsuse hindamisel.

Summary

User Interface Tests for Assessment of Programming Skills

The objective of this bachelor thesis is to develop a set of user interface tests to assess programming skills. The idea came in collaboration with Kutsekoda to develop automated tests to assess created web solutions.

Kutsekoda does not have standard exercises for programmers therefore the author first created three example assignments. The goal of the first one was to create a complicated website and the second one was to create a simple web app. In the third excercise the programmer gets an incomplete website and has to finish it.

The author created user interface tests to evaluate the solutions that the programmers create for the aforementioned excercises. Tests are written in Java using Selenide library. When the tests are done, the program creates an answer file where is written the score and a detailed description of the failed tests.

Created test sets are a work in progress and they still needs a lot of improvement before they can be used in real life. The author envisions these being used not only by Kutsekoda but also by lecturers to assess students' exams and tests.

Kasutatud kirjandus

Vallaste. (2014). Külastatud 12. aprillil, 2014, aadressil http://www.vallaste.ee.

Codecademy. (2014) *Learn to code | Codecademy*. Külastatud 12. aprillil 2014, aadressil http://www.codecademy.com.

Codility Ltd. (2014). *Automated tests of programming skills. Assessment of software developers. Recruitment software. Codility*. Külastatud 12. aprillil 2014, aadressil https://codility.com.

Codility. (2012). *How to create a Codility programming test?*. Külastatud 12. aprillil 2014, aadressil http://www.youtube.com/watch?v=X7AdDvi8fdY.

CodeEval. (2014). *CodeEval - Evaluations Made Easy*. Külastatud 12. aprillil 2014, aadressil https://www.codeeval.com.

Sihtasutus Kutsekoda. (2011). *Kutsestandard. Tarkvaraarendaja, tase 4*. Külastatud 12. aprillil 2014, aadressil http://kutsekoda.ee/et/kutseregister/kutsestandardid/10415268/lae/tarkvaraarendaja-tase-4-11pdf.

Solntsev, A. (2014). *Selenide: concise UI tests in Java*. Külastatud 12. aprillil 2014, aadressil http://selenide.org/index.html.

Google. (2014). *Google HTML/CSS Style Guide*. Külastatud 12. aprillil 2014, aadressil http://google-styleguide.googlecode.com/svn/trunk/htmlcssguide.xml?showone=ID_and_Class_Name_Delimiters#ID_and_Class_Name_Delimiters.

JUnit team. (2012). *Summary of changes in version 4.11*. Külastatud 15. aprillil 2014, aadressil https://github.com/junit-team/junit/blob/master/doc/ReleaseNotes4.11.md#test-execution-order.

JUnit. (2012). *MethodSorters (JUnit API)*. Külastatud 15. aprillil 2014, aadressil http://junit.czweb.org/apidocs/org/junit/runners/MethodSorters.html.

The Eclipse Foundation, Inc. (2014). *Eclipse - The Eclipse Foundation open source community website*. Külastatud18. aprillil 2014, aadressil https://www.eclipse.org/.

Selenium. (2014). *Selenium - Web Browser Automation*. Külastatud 19. aprillil 2014, aadressil http://docs.seleniumhq.org/.

Codeborne. (2014). *Selenide vs Selenium · codeborne/selenide Wiki · GitHub*. Külastatud 26. aprillil 2014 aadressil https://github.com/codeborne/selenide/wiki/Selenide-vs-Selenium.

Rosental, K. (2013). . Külastatud 10. aprillil 2014, aadressil http://minitorn.tlu.ee/teemaderegister/get_file.php?id=234&name=karmo_rosental.pdf.

Lisad

1. Ülesanded

Näidisülesannete kirjeldused asuvad tööga kaasas oleval andmekandjal ja aadressil http://minitorn.tlu.ee/~mait/bakalaureus/%dclesanded.

2. Koodi täiendamise ülesande lähtefailid

Koodi täiendamise ülesande lähtefailid asuvad tööga kaasas oleval andmekandjal ja aadressil http://minitorn.tlu.ee/~mait/bakalaureus/postkontor_l%e4htefailid.zip.

3. Kasutajaliidese testid

Ülesannete korrektsust kontrollivad testkomplektid on saadaval tööga kaasas oleval andmekandjal ja aadressil http://minitorn.tlu.ee/~mait/bakalaureus/Testid.

4. Kasutajaliidese testide lähtekood

Ülesannete lahendusi kontrollivate testide lähtekood asub tööga kaasas oleval andmekandjal ja aadressil http://minitorn.tlu.ee/~mait/bakalaureus/L%e4htekood.

5. Ülesannete võimalikud lahendused

Näidisülesannete võimalikud lahendused on saadaval tööga kaasas oleval andmekandjal ja aadressil http://minitorn.tlu.ee/~mait/bakalaureus/Lahendused.