Tallinna Tehnikaülikool
Infotehnoloogia teaduskond
Arvutiteaduse instituut
Üldinformaatika õppetool

Testandmete generaatori realiseerimine graafilise kasutajaliidese testimiseks

Bakalaureusetöö

Üliõpilane: German Mumma
Üliõpilaskood: 112165IAPB
Juhendaja: Maili Markvardt

Tallinn
2014

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

---------- ----------

(*kuupäev*) (*allkiri*)

Annotatsioon

Käesolevas töös, „Testandmete generaatori realiseerimine graafilise kasutajaliidese testimiseks“, realiseerin testandmete generaatori, mis loob testandmeid, rakendades ekvivalentsiklassidesse jaotumise ja piirjuhtude analüüsi testimistehnikaid. Nimetatud programmi realiseerimiseks, kasutan ABAP programmeerimiskeelt programmi loogika ja SAP WebDynpro ABAP kasutajaliidese jaoks. Realiseeritud testandmete generaatori väljundit, milleks on JSON formaadist fail testidega, on võimalik implementeerida automaattesti skriptides, et vähendada käsitööd kasutajaliideste testimisel ning ekvivalentsiklassi ja piirjuhtude testi loomisel.

Abstract

In my bachelor thesis “Realization of Test Data Generator for Testing Graphical User Interfaces”, I will realize a test data generator which supports test data creation accord to the equivalence partitioning and boundary value analysis testing techniques. Creating equivalence partitions and boundary values for testing graphical user interfaces can be a time consuming task. By introducing the test data generator into test development life cycle, manual workload can be reduced while still preserving high probability of defect detection.

Lühendite ja mõistete sõnastik

ISTQB	International Software Testing Qualifications Board. Nimetatud organisatsiooni eesmärk on pakkuda rahvusvaheliselt tunnustatud ja järjekindlad kvalifikatsioonid tarkvara testimises.
HTML	HyperText Markup Language Standard markeerimiskeel veebilehtede loomiseks.
HTML5	HyperText Markup Language 5 Markeerimiskeel, mida kasutatakse sisu struktureerimiseks ja esitamiseks internetis.
JSON	JavaScript Object Notation Avatud standardiga formaat, mis kasutab inimloetavat teksti andmeobjektide edastamiseks.
ABAP	Advanced Business Application Programming Saksa firma SAP AG standardne programmeerimiskeel SAP R/3 süsteemide arendamiseks.

Jooniste nimekiri

Tabelite nimekiri

Sisukord

Sissejuhatus

Tarkvara testimine on tähtis osa tarkvara arenduse elutsüklist. Tarkvara testimisele kuuluvad kulud moodustavad 1/3 kuni poole süsteemi arendamise kogukulust [5]. Üks põhjus, miks tarkvara testimine nõuab suuri kulutusi on käsitöö rohkus testimise protsessis. Tarkvara testimise automatiseerimine vähendab vajalikku käsitööd. Tarkvara testimises mängivad suurt rolli testandmed, mida kasutatakse testide valideerimisel ja verifitseerimisel. Kuna süsteemi ei ole tulus testida lõpmata paljude sisenditega, siis on vaja valida sisendid, mis katavad enamiku suure prioriteediga testidest. Testandmete generaator automatiseerib eelnimetatud sisendite valiku. Testandmete generaatorit saab kasutada automaattestide skriptides, et korraga jooksutada sama testi erinevate sisendite korral. Skripti käivitamisel valitakse testandmete generaatori poolt genereeritud võimaliku sisendi koos vastava oodatava tulemusega. Vastava testandmete generaatori kirjutab autor ise programmeerimiskeeles ABAP.

Käesoleva töö eesmärk on realiseerida lahendus vähendamaks käsitööd ekvivalentsiklasside ja piirjuhtude meetodiga testide kirjutamisel. Selleks kirjutab autor testandmete generaatori, mis toetab ekvivalentsiklasside ning piirjuhtude meetodil graafilise kasutajaliidese testimiseks testide koostamist. Tulemusena võimaldab testandmete generaator automaattesti skriptides pärida sisendid JSON formaadis failist, mida testandmete generaator ise koostab.

Töö on jaotatud neljaks peatükiks. Esimeses peatükis tutvustatakse tarkvara kvaliteeti, selle tagamist, tarkvara testimise rolli tarkvara kvaliteedi tagamisel ning tarkvara testimist üldiselt. Teises peatükis tutvustatakse ja kirjeldatakse testandmete generaatori tähtsamad komponendid. Kolmandas peatükis kirjeldatakse, kuidas on testandmete genereerimise protsess realiseeritud. Neljandas peatükis kirjeldatakse testandmete generaatori väljundfail ning antakse juhiseid, kuidas nimetatud faili võib implementeerida automaattesti skriptides.

1. Tarkvara kvaliteet ja selle tagamine

Tarkvara kvaliteeditagamine on tarkvara loomise alamvaldkond, mis mõõdab kui hästi tarkvara on disainitud ning kui hästi on tarkvara kooskõlas selle disainiga [6]. Kvaliteedi all on mõeldud eelkõige taset, mis näitab komponendi, süsteemi või protsessi kooskõlalisust kindlalt paika pandud nõudmistega ja/või kasutaja/kliendi vajadustega ja ootustega [1]. Tarkvara kvaliteeti aitab tagada tarkvara testimise distsipliin. Järgnevates punktides tutvustab autor eeskätt antud tööd puudutavat tarkvara testimise komponente.

1.1 Tarkvara testimine

Tarkvara testimine on programmi käivitamise protsess kindla eesmärgiga leida vigu [3]. Kuid igal veal on oma riski tase. Mitu kirjaviga tekstis on oluliselt väiksema kaaluga või riskiga kui viga valuuta konverteerimises või viga õigete mõõtühikute valimisel. Testimisest saab enam kasu siis, kui peetakse silmas kindlaid printsiipe ja järgitakse tõestatud protsessi. ISTQB nimetab seitse testimise printsiipi, mis on ära toodud all olevas tabelis [1].

Tabel 1. Tarkvara testimise printsiibid

Testimise printsiip	Printsiibi lühikirjeldus
Testimine näitab vigade olemasolu	Testimine võib näidata vigade olemasolu, mitte aga nende puudumist.
Põhjalik testimine ei ole võimalik	Testida kõiki võimalike sisendite ja eeltingimuste kombinatsioone on teostatav ainult triviaalsetel juhtudel. Põhjaliku testimise asemel tuleks kasutada riski analüüsi ja prioriteete, et fokuseerida testimise vaeva.
Varajane testimine	Testimise tegevustega tuleks alustada võimalikult varakult tarkvara või süsteemi arenduse elutsüklis.
Defektide klasterdumine	Mida rohkem ühes moodulis vigu leitakse, seda tõenäolisem on selles moodulis veel vigu leida.
Pestitsiidi paradoks	Sama testide korduv käivitamine leiab aina vähem uusi defekte. Selleks tuleb testjuhud pidevalt üle vaadata ja korrastada ning kirjutada teistele funktsionaalsustele suunatud testid.
Testimine sõltub kontekstist	Erineva eesmärgiga tarkvara või süsteeme tuleb testida teistmoodi.
Vigade puudumise ekslikkus	Vigade leidmisest ning parandamisest ei ole kasu, kui ehitatud süsteem ei ole kasutatav ja ei vasta kasutaja vajaduste ja ootustele.

1.2 Tarkvara testimise protsess

ISTQB nimetab fundamentaalset testimise protsessi, mis on jaotatud viieks põhitegevuseks, mis on ära toodud allolevas tabelis [1].

Tabel 2. ISTQB põhimõtteline testimise protsess

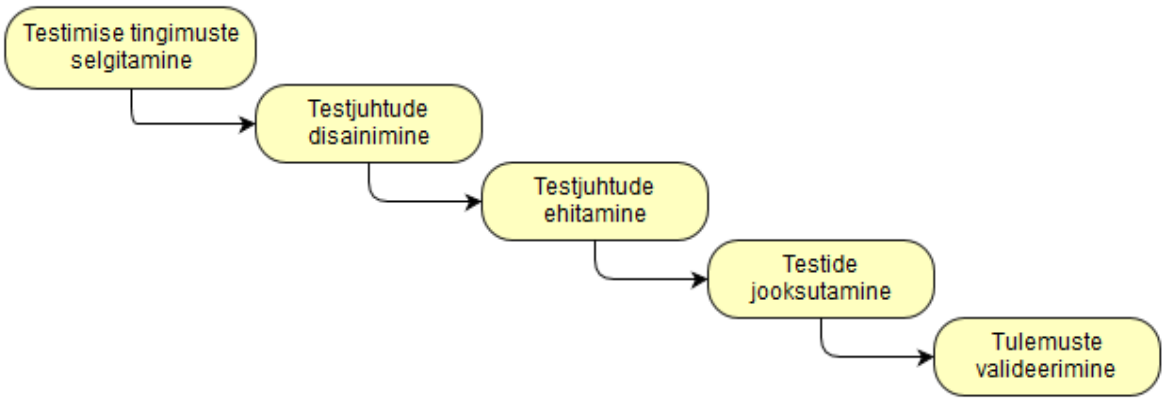
Tegevus	Tegevuse kirjeldus
Planeerimine ja järelevalve	Planeerimise käigus tehakse kindlaks, et kliendi ja projekti eesmärgid ning testimisega seotud riskid on selged.
Analüüs ja disain	Üldised testimise eesmärgid muudetakse mõõdetavateks testi tingimusteks ning disainideks.
Testi implementeerimine ja käivitamine	Testi tingimused muudetakse testjuhtudeks ja –varaks ning seatakse üles testkeskkond.
Lõpetamise kriteeriumide hindamine ja raporteerimine	Testide jooksutamine hinnatakse vastavalt defineeritud eesmärkidele ning koostatakse kokkuvõttev testraport huviisikutele.
Testimise lõpetamise tegevused	Lõpetatud testimise tegevuste kohta andmete kogumine kogemuste konsolideerimiseks.

Nimetatud tegevused on loogilises järjestuses, kuid sõltuvalt projektist võivad tabelis toodud tegevused seguneda, toimuda samaaegselt ja isegi korduda.

1.3 Tarkvara testimise automatiseerimine

Tarkvara testimine peab olema efektiivne vigade ja rikete tuvastamises, aga ka tõhus testide teostamises võimalikult kiiresti ja odavalt. Eespool sai öeldud, et tarkvara testimise väärtus seisneb selle võimes avastada testitavas tarkvaras olevaid vigu ja rikkeid. Seega testimise efektiivsuse üheks näitajaks võib nimetada testija poolt kirjutatud testide võime tuvastada tarkvaras seniavastamata vigu. Lisaks eelnimetatule saab määrata testimise efektiivsust testide läbiviimise kiiruse ja odavusega.

Tarkvara testimise automatiseerimine võimaldab vähendada testimiseks kulutatavat aega asendades testija käsitööd automaattesti skriptidega. Automaattesti skript on põhimõtteliselt testjuht, mis on kirjutatud arvutile arusaadavas keeles (programmeerimiskeeles). Skripti koosneb sammudest, sisendandmetest, sisendandmete valideerimisest ning testi tulemuste kuvamisest. Eelnimetatud protsess toetab üldlevinud testjuhu arendustsüklit [2], mis on näidatud joonisel 1.



Joonis 1. Testjuhu arendustsükkel

Testjuhtude disainimise tulemuseks on testide komplekt, mis koosneb testi sammudest, sisendandmetest ning oodatavatest tulemustest. Käesolev töö panustab eelkõige testjuhtude disainimisse kindlate reeglite alusel sisendandmete ning neile vastavate oodatavate tulemuste genereerimisega.

Testimises kasutatud sisendandmed on tõhusamad vigade avastamisel siis, kui sisendandmete valimine on toimunud mõne parimat praktikat toetava meetodi kasutamisel. Järgnevas peatükis tutvustatakse kahte enam levinud sisendandmete määramise meetodit, mida autor käesolevas töös kasutab.

1.4 Testimise meetodid

Testide disainimiseks on olemas erinevaid meetodeid. Antud töös kasutatakse musta kasti meetodeid. Musta kasti meetodid käsitlevad testitava tarkvara musta kastina koos sisendite ja väljunditega. Musta kasti meetodite rakendamisel ei ole testija teadlik testitava süsteemi sisemistest toimingutest, ehk milline on tarkvara sisemine struktuur.

1.4.1 Ekvivalentsklasside jaotamine

Testjuhtude disainimisel esimesena tavaliselt rakendatakse ekvivalentsiklasside jaotamise meetodid. Meetodi idee seisneb selles, et teatud testi tingimused jaotatakse partitsioonidesse või gruppidesse. Gruppidesse kuuluvad sisendid loetakse samaväärseteks, s.t. et süsteem või tarkvara komponent peaks ühte gruppi kuuluvate sisendite puhul käituma ühtmoodi [1].

Järgmise sammuna tuleb testida ainult ühte sisendit igast grupist, kuna eeldatakse, et ühte gruppi kuuluvad sisendid on omavahel võrdväärsed. Kui üks sisend töötab, siis eeldatataks, et ka teised, samasse gruppi kuuluvad sisendid töötavad. Vastupidiselt, kui üks sisend grupist ei tööta, siis eeldatakse, et ka teised sisendid ei tööta [1].

Järgnevalt on toodud näide, kuidas ekvivalentsiklassi jaotamise meetodit rakendada. Oletame, et üks rakenduse vormi väljadest aktsepteerib ainult täisarve ulatuses 100-999. Sellise kitsenduse järgi saab moodustada neli ekvivalentsiklassi, millest üks on kehtiv ja kolm kehtetud. Allpool on toodud tabel vastavate ekvivalentsiklassidega.

Tabel 3. Näide ekvivalentsiklassidesse jaotamisest

Kehtiv	Kehtetu	Kehtetu	Kehtetu
Täisarvud ulatuses 100-999	Täisarvud suuremad kui 999	Täisarvud väiksemad kui 100	Mitte täisarvud ja mitte numbrid

Oluline on testida, mitte ainult kehtivad, aga ka kehtetud grupid. Kehtetud grupid tähistavad sisendeid, mida rakendus ei oota kasutajalt. Sellegi poolest tuleb ka sellised sisendid testida, et veenduda rakenduse võimes toime tulla ka ootamatute sisenditega. Tavaliselt reageerib rakendus ootamatu sisendile informatiivse veateatega [1].

1.4.2 Piirjuhtude analüüs

Piirjuhtude analüüsi meetodi idee seisneb selles, et testitakse sisendandmete väärtusi, mis asuvad ekvivalentsiklassi piiridel. ISTQB definitsiooni järgi on piirjuhtude analüüs musta kasti testi disainimise meetod, mille järgi testjuhud koostatakse piirväärtustest [1].

Eelnevast peatükist näidet kasutades moodustame tabeli, mis koosneb piirväärtustest. Selleks tuleb võtta minimaalne ja maksimaalne väärtus kehtivast ekvivalentsiklassist ning esimese või viimase väärtuse igast kehtetust ekvivalentsiklassist. Näitlikkuse eemärgil jäetakse mittetäisarvuline ekvivalentsiklass arvestamata.

Tabel 4. Näide piirjuhtude analüüsi meetodist

Kehtetu	Kehtiv		Kehtetu
99	100	999	1000

Sellega on tuvastatud 4 testi, millel on suur tõenäosus tuvastada defekte. Programmeerimine on oma loomuse poolest kõige tundlikum vigade suhtes just piiraladel. Tarkvara on hästi

binaarne – midagi on kas tõene või väär. Kui operatsioon on teostatud teatud ulatusega numbritega, siis on tõenäolisem, et programmeerija tegi vea ulatuse piiride juures kui ulatuse keskel olevate väärtustega [4]. Levinud viga on massiivist esimese või viimase elemendi pärimine, kui ei arvestata, et massiivi esimene element asub kohal 0, mitte 1.

2. Testandmete generaator

Käesoleva töö autor kirjutab testandmete genereerimise programmi. Programmi eesmärk on vähendada käsitööd graafiliste kasutajaliideste testimisel. Testandmete all on mõeldud sisendit, mis kasutatakse testitava tarkvara funktsionaalsuse testimiseks. See tähendab, et sisendil on kindel otstarve. Üldjuhul on sisendi suurim otstarve veaohtlike kohtade avastamine testitavas tarkvaras. Selleks, et veaohtlikke kohti avastada aitab kindla formalismi järgi valitud sisend rohkem kui juhuslikult valitud sisend. Käesolevas töös kasutatud formalismiks on esimeses peatükis tutvustatud ekvivalentsiklassidesse jaotumise ja piirjuhtude analüüsi testimise meetodid.

Genereerimise all on mõeldud suhtlus testandmete generaatori ja testija vahel. Nimetatud suhtlus on jaotatud erinevateks etappideks, mis kõik koos moodustavad kogu testandmete genereerimise protsessi, millest räägitakse detailsemalt kolmandas peatükis.

Käesolevas peatükis antakse eelkõige ülevaade programmi arhitektuurilisest poolest. Seejärel antakse ülevaate programmi kujundavatest objektidest, milledeks on erinevate ülesannetega klassid. Käesoleva peatüki lõpus kirjeldatakse seos kahe kõige tähtsama klassi vahel.

Tehnoloogia

Autor realiseerib testandmete generaatori programmi ABAP programmeerimiskeeles. Programmi kasutajaliidese realiseerib autor kasutades tarkvara SAP WebDynpro for ABAP. ABAP on tarkvara firma SAP AG kõrg-taseme programmeerimiskeel ERP süsteemide programmeerimiseks. SAP WebDynpro for ABAP on SAP AG standard tehnoloogia kasutajaliideste programmeerimiseks.

ABAP on objekt-orienteeritud programmeerimiskeel. Testandmete generaatori objektideks on klassid, milledel kindlad parameetrid ja meetodid. Meetodid võivad olla privaatsed ja avalikud. Privaatseid meetodeid ei ole võimalik kutsuda väljaspool klassi, millele meetod kuulub. Avalike meetodeid on võimalik välja kutsuda ka teistes klassides.

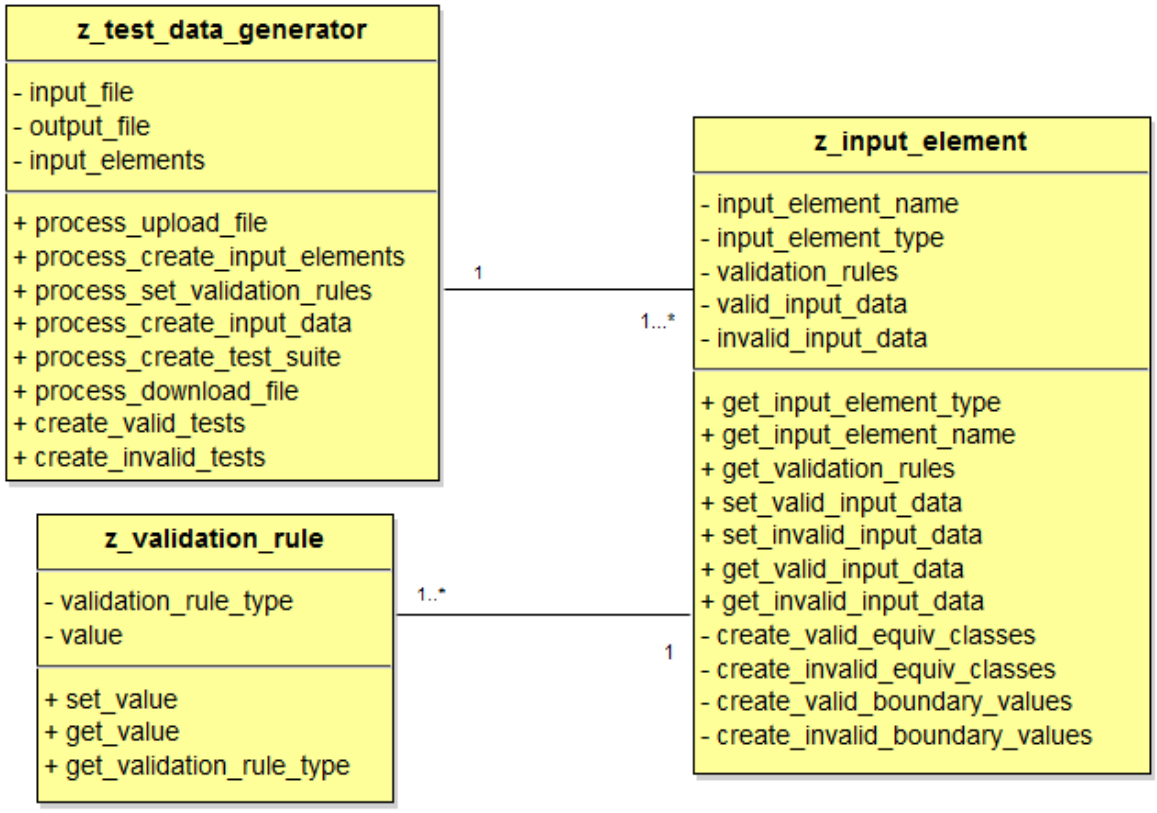
Lisaks parameetrite ja meetoditele, on klassidel ka massiivid, mida eelkõige kasutatakse suure hulga andmete hoidmiseks indekseeritud kujul (andmetel on kindel asukoht massiivis).

Programmeerimiskeeles ABAP on massiivideks sisemised tabelid (*internal table*), mis hoiavad kindla tüübiga (*data object*) andmete objekte. Lihtsuse mõttes kasutab autor sisemise tabeli nimetuse asemel lihtsalt massiiv. Käesolevas töös on massiivides kasutatud objektideks allolevad klassid. SAP standard näeb ette objektide nimede alguses tähtede *z* või *y* kasutamist.

- z_input_element – sisestuselement
- z_validation_rule – valideerimisreegel

Klassidiagramm

Autori poolt kirjutatud programmi tähtsamad klassid on toodud joonisel 2. toodud klassidiagrammil.



Joonis 2. Programmi klass ja nende suhteid kirjeldav klassidiagramm

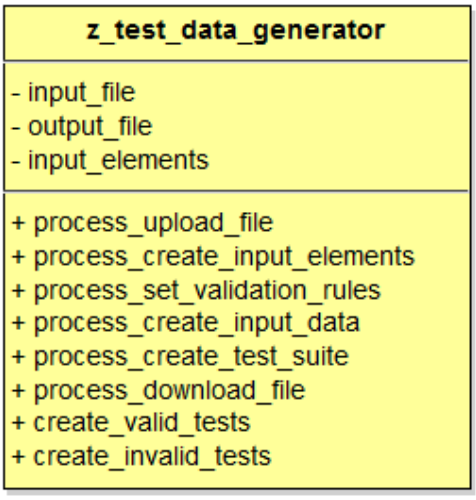
Järgnevas peatükis kirjeldatakse igat klassi detailsemalt.

2.1 Programmi klassid

Igal programmi klassil on muutujad ja meetodid. Muutujad on mõeldud kindla väärtuse hoidmiseks. Meetodid kasutatakse teatud funktsiooni realiseerimiseks. Joonistel on kasutatud "-" märki privaatse muutuja või meetodi defineerimiseks ning "+" märki avaliku muutuja või meetodi defineerimiseks.

2.1.1 Testandmete generaatori klass

Testandmete generaator on klass nimega *z_test_data_generator* (edaspidi lihtsalt testandmete generaator), mis on varustatud protsessimeetodite, loomismeetodite ning erinevate abimeetoditega. Protsessimeetoditega juhitakse kogu testandmete genereerimise protsessi. Igale testandmete genereerimise protsessi etapile vastab protsessimeetod. Allolev joonis illustreerib testandmete generaatorit koos tema protsessi- ja loomismeetoditega. Abimeetodite rohkuse tõttu puuduvad alloleval joonisel abimeetodid. Abimeetodid kirjeldatakse hilisemas peatükis.



Joonis 3. Testandmete generaatori klass

Protsessimeetodid

Meetodiga *process_upload_file* käivitatakse kogu testandmete genereerimise protsess. Selle meetodi käivitamisel kuvatakse testijale kasutajaliidese, kus testija saab HTML faili üleslaadida.

Kui faili üleslaadimine õnnestus, siis hoitakse nüüdsest see HTML faili sisu muutujas *input_file* ja käivitub protsessimeetod *process_create_input_elements*. Selle protsessimeetodi eesmärk on muutuja *input_file*-i seest sisestuselementide tuvastamine ja loomine. Loodud sisestuselemendid lisatakse massiivi *input_elements.* Enne järgmist etappi kuvatakse kasutajaliideses kõik loodud sisestuselemendid, et võimaldada testijal veenduda, kas kõik soovitud sisestuselemendid on loodud. Sellest sõltuvalt on testijal siin kohal võimalik, kas minna tagasi esimesse etapi ja fail uuesti üleslaadida, vajadusel eelnevalt faili korrigeerides, või minna edasi järgmisse etappi, milleks on valideerimisreeglite määramine.

Valideerimisreeglite määramise etapis käivitatakse protsessimeetod *process_set_validation_rules*. Selle protsessimeetodi eesmärk on võimaldada testijal käsitsi määrata iga sisestuselemendi puhul tema valideerimisreeglid. Peale protsessimeetodi käivitamist kuvatakse testijale kasutajaliides, kus on kuvatud kõik sisestuselemendid testandmete generaatori massiivist *input_elements* ning valideerimisreeglid iga sisestuselemendi massiivist *validation_rules*. Testija sisestatud valideerimisreegli väärtus salvestatakse iga valideerimisreegli muutujasse *value*. Nüüd on testandmete genereerimiseks vajalikud muutujad ja massiivid väärtustatud ning minnakse järgmise etapi juurde, milleks on testandmete loomine.

Testandmete loomise etapp algab protsessimeetodi *process_create_input_data* käivitamisega. Kuna kogu töö teostatakse tagataustal, siis kasutajaliidest ei ole. Testandmete loomise protsessimeetod on oma töö lõpetanud kui iga sisestuselemendi massiividesse *valid_input_data* ja *invalid_input_data* on testandmed lisatud ja salvestatud. Järgmises etapis koostatakse sisestuselementide testandmetest testikomplekt.

Testikomplekti (*test_suite*) loomise etapp algab meetodi *process_create_test_suite* käivitamisega. Testikomplekti loomise etapi tulemuseks kuvatakse testijale kasutajaliideses tabel testandmete generaatori poolt koostatud testidega. Tabelis on kuvatud kõik testandmete genereerimise protsessi jooksul loodud sisestuselemendid ja sisestuselementide valideerimisreeglite alusel loodud testandmed.

Lisaks komplekteeritud testidele kuvatakse testijale kasutajaliidese faili allalaadimiseks. Allalaadimisprotsess algab protsessimeetodi *process_download_file* käivitamisega. Allalaaditavas failis on täpselt samad andmed nagu eelmises etapis testijale kuvatud testikomplekti tabelis ainult, et fail on JSON formaadis.

Abimeetodid

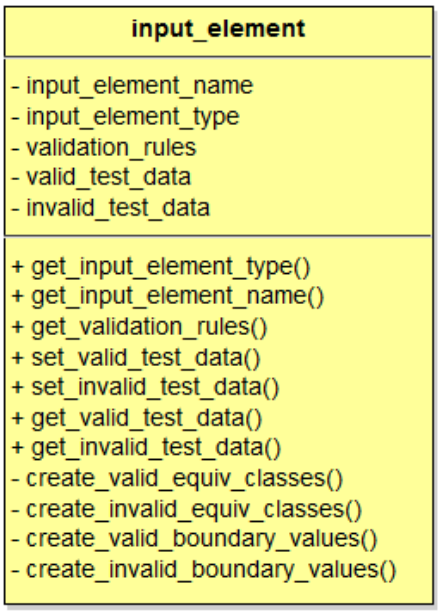
Testandmete generaatori abimeetodid on avalikud meetodid, mida kasutab eelkõige sisestuselement testandmete loomisel. Järgnevas tabelis on toodud mõned testandmete generaatori abimeetodid koos sisend- ja väljundparameetritega. Lisaks on toodud iga abimeetodi juures selle konkreetse abimeetodi kirjeldus. Sidekriips "-" tähendab, et meetodil sisend parameetrid puuduvad. Testandmete generaatori kõik abimeetodid on toodud lisas 1.

Tabel 5. Testandmete generaatori abimeetodid ja nende kirjeldused

Abimeetodi nimi	Sisendid	Väljundid	Kirjeldus
get_string	integer *string_length*	string *output_string*	Tagastab väärtuse, mis koostatakse suvalisi tähti kombineerides, kuni saadakse *string_length* pikkune tekstiline väärtus
get_number	integer *length*, integer *precision*	float *output_number*	Tagastab etteantud pikkuse (*length*) ning komakohtade arvuga (*precision*) numbrilise väärtuse
get_upper_case	-	char *upper_case_letter*	Tagastab ühe suvalise suurtähe suurtähtedega täidetud massiivist
get_special_char	-	char *special_char*	Tagastab ühe suvalise erimärgi massiivist, kus hoitakse erinevaid erimärki
get_valid_month	-	integer *valid_month*	Tagastab väärtuse ulatuses 1 kuni 12
get_invalid_month	-	integer *invalid_month*	Tagastatakse väärtus 13, mis ei vasta kehtivale kuu väärtusele

2.1.2 Sisestuselemendi klass

Sisestuselement on klass, millel on muutujad ja meetodid. Sisestuselement, koos temale omaste muutujate ja meetoditega, on illustreeritud alloleval joonisel.



Joonis 4. Testandmete generaatori objekt sisestuselement

Muutujad *input_element_name* ja *input_element_type* on vastavalt sisestuselemendi nimi ja tüüp, mida testandmete generaator HTML faili parsimise tulemusel tuvastas. Uue sisestuselemendi loomise eeltingimuseks ongi nimi ja tüüp. Meetod *get_input_element_type* tagastab muutuja *input_element_type* ja meetod *get_input_element_name* tagastab muutuja *input_element_name*, mida kasutatakse testandmete genereerimise protsessi viimases etapis, testikomplektide loomisel.

Massiiv *validation_rules* hoiab valideerimisreegli klasse, mille tagastamiseks on mõeldud meetod *get_validation_rules.* Klassist *z_validation_rule* räägitakse detailsemalt hilisemas peatükis.

Massiivid *valid_input_data* ja *invalid_input_data* hoiavad vastavalt kehtivaid ja kehtetuid testandmeid. Meetodid *set_valid_input_data* ja *set_invalid_input_data* kasutatakse testandmete generaatori poolt, et käivitada testandmete loomise protsessi vastavas sisestuselemendis. Selle tulemusel väärtustatakse massiivid *valid_input_data* ja *invalid_input_data*. Seda teostatakse testandmete genereerimise protsessi etapis testandmete loomine.

Meetodeid *get_valid_input_data* ja *get_invalid_input_data* kasutab testandmete generaator, et pääseda ligi massiivides *valid_input_data* ja *invalid_input_data* olevatele kehtivatele ja kehtetutele testandmetele testikomplekti loomise etapis.

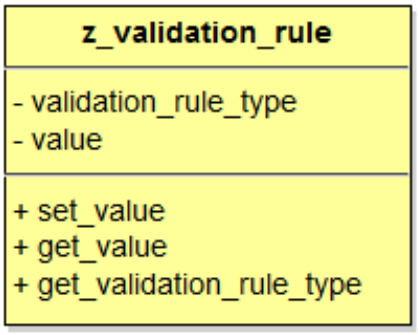
Toetatud sisestuselemendid

Kasutaja sisestuse vastuvõtmiseks graafilises kasutajaliideses näeb HTML standard ette erinevaid sisestuselementide (*input*) tüüpe. Vabasisestuselemendid on sisestuselemendid, mis lubavad kasutajale kõige enam vabadust andmete sisestamisel. Käesolevas töös realiseeritakse järgmised HTML ja HTML5 standarditele vastavad vabasisestuselemendid:

1) Tekstiväli (<*input type =“text“ />*) – kasutatakse lühemat sorti tekstilise väärtuse sisestamiseks
2) Paroolivälji (<*input type =“password“ />*) – kasutatakse sisestatud andmete visuaalseks peitmiseks nagu paroolide puhul on tavaks. Sarnane tekstiväljale ainult sisestatud tähemärgid kuvatakse kasutajale tavaliselt asteriskide (*) näol
3) Kuupäevaväli (<*input type =“date“ />*) – kasutatakse kuupäeva formaadis andmete sisestamiseks
4) E-maili väli (<*input type =“email“ />*) – kasutatakse emaili sisestamiseks
5) Numbriväli (<*input type =“number“ />*) – kasutatakse numbrilise sisendi vastuvõtmiseks
6) Kellaajaväli (<input type =“*time*“ />) – kasutatakse ajatsooni märkideta kellaaja sisestamiseks
7) Telefoniväli (<input type =“*tel*“ />) – kasutatakse telefoninumbri sisestamiseks

2.1.3 Valideerimisreegli klass

Testandmete generaatori mõistes on valideerimisreegel klass, millel on muutujad ja meetodid. Allolev joonis illustreerib valideerimisreeglit koos temale omaste muutujate ja meetoditega.



Joonis 5. Testandmete generaatori klass valideerimisreegel

Testija määratud valideerimisreegli väärtus salvestatakse meetodiga *set_value* ning väärtus hoitakse muutujas *value*. Valideerimisreegli tüüpi määrab muutuja *validation_rule_type*.

Toetatud valideerimisreeglid

Allolevas tabelis kirjeldatakse testandmete generaatori poolt toetatud valideerimisreeglid.

Tabel 6. Programmi poolt toetatud valideerimisreeglid

Valideerimisreegli tüüp	Võimalikud väärtused	Kirjeldus
min_length	Numbrilised sisendid	Minimaalne tähemärkide arv väärtuses
max_length	Numbrilised sisendid	Maksimaalne tähemärkide arv väärtuses
min_value	Numbrilised sisendid	Väikseim oodatav väärtus. Siin on võimalik panna paika kas negatiivsete väärtuste sisestamine on lubatud
max_value	Numbrilised sisendid	Suurim oodatav väärtus
required	*Required, Optional*	Kas väli on lubatud tühjaks jätta?
valid_length	Numbrilised sisendid	Parooli sisendi pikkus, mis peab valideeruma
upper_case	*Allowed, Forbidden*	Kas suurtähed on lubatud?
numbers	*Allowed, Forbidded*	Kas numbrid on lubatud?
precision	Numbrilised sisendid	Mitu komakohta on lubatud sisestada? Vaikimisi on kuvatud väärtus 0, mis tähendab, et sisend peab olema täisarv
special_chars	*Allowed, Forbidden*	Saab määrata, millised erimärgid on lubatud kasutada. Kui valitakse *Allowed*, siis erimärkide suhtes kitsendusi ei ole, kui sisestatakse *Forbidden*, siis erimärgid ei ole lubatud
date_format	*dd.mm.yyyy, dd/mm/yyyy*	Määratakse, milline on õige kuupäeva formaat
time_format	*hh:mm, hh:mm:ss, h:mm AM/PM, h:mm:ss AM/PM*	Määratakse, milline on õige kellaaja formaat

3. Testandmete genereerimise protsess

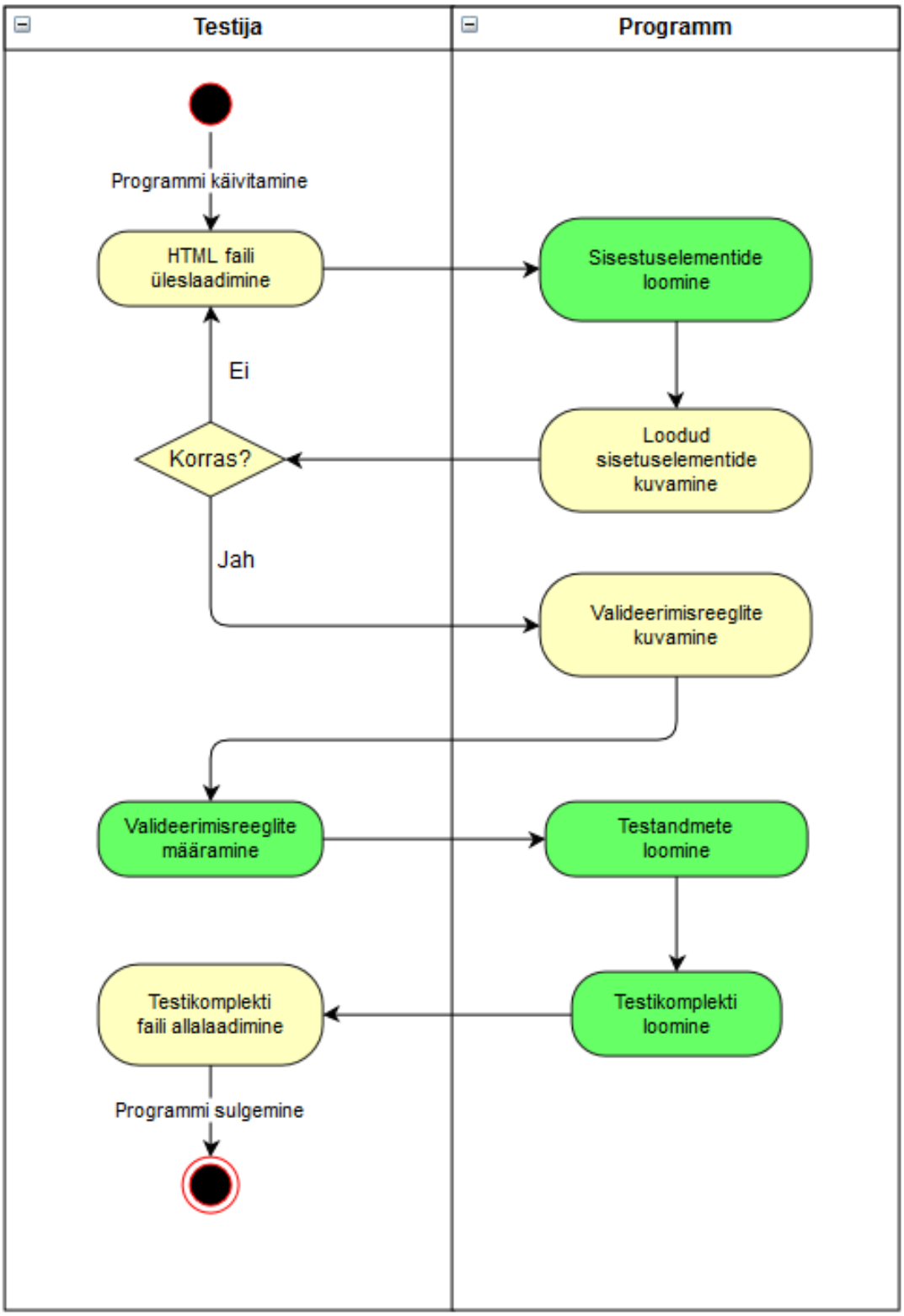
Testimise automatiseerimise üks raskustest on testandmete genereerimine. Testandmete genereerimine on protsess, mille tulemusel luuakse andmete komplekt sisendandmetena kasutamiseks rakenduse testimisel [7].

Esimeses peatükis sai mainitud, et testimise tõhususe tõstmiseks tuleb osata valida sisendid, millel on suurem tõenäosus tuvastada programmis olevaid vigu. Kaks meetodit, kuidas sellised sisendandmed kindlaks määrata, on ekvivalentsiklassidesse jaotamine ja piirjuhtude analüüs, mida samuti kirjeldati eelmises peatükis. Käesolevas töös kasutab autor nimetatud meetodid testandmete genereerimise realiseerimises.

Testandmete generaatori realiseerimiseks on võetud aluseks testandmete genereerimise protsess [5], mis koosneb järgmistest etappidest:

1) Oodatavate sisendandmete tüüpide määramine testija poolt. Teostatakse testitava kasutajaliidese analüüs, et määrata iga vormi sisestuselemendi andmetüüp. Programm võimaldab valida eeldefineeritud andmetüüpide hulgast, mis vastavad standardsetele HTML elementidele
2) Valideerimisreeglite määramine testija poolt. Igal andmetüübil on oma vastav tabel, mis koosneb testija poolt sisestatud testitava kasutajaliidese elementidest, mis vastavad sellele andmetüübile, ning valideerimisreeglitest sisestavatele andmetele
3) Testija poolt kirjeldatud valideerimisreeglitele vastavate testandmete genereerimine. Programm määrab testija poolt sisestatud reeglitele ekvivalentsiklassid ja piirjuhud ning loob sisendid.

Käesoleva töö autori poolt realiseeritav testandmete generaator toetub allpool oleval joonisel kirjeldatud testandmete genereerimise protsessile.

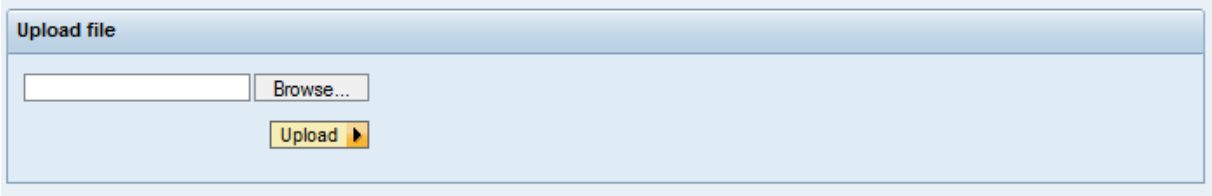


Joonis 6. Testandmete genereerimise protsessi tegevusdiagramm

Rohelise taustaga on märgitud testandmete genereerimise protsessi põhietapid, mida kirjeldatakse detailsemalt järgnevates peatükkides.

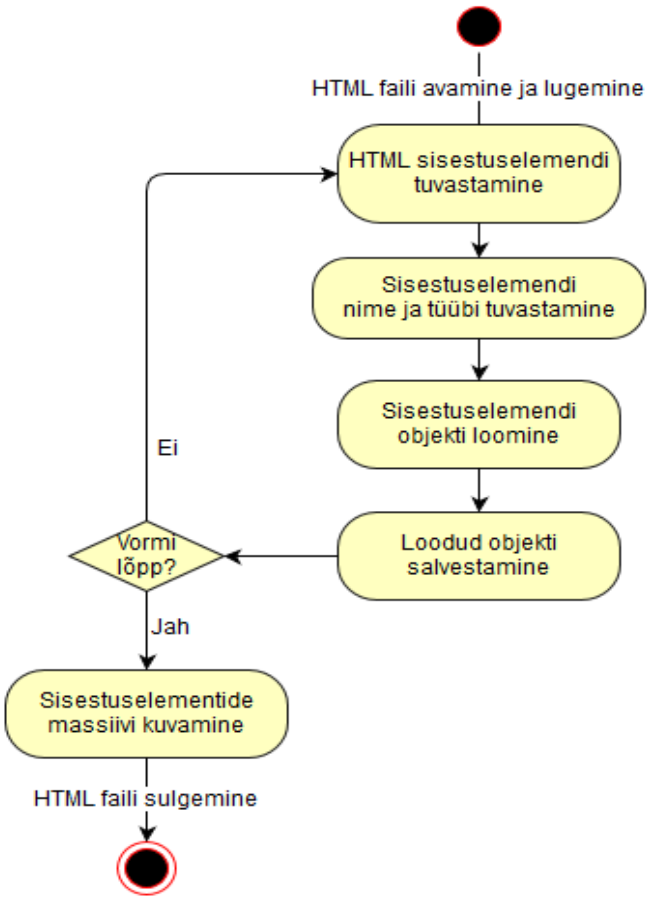
3.1 Sisestuselementide loomine

Käesolevas töös teostab sisestuselementide loomise testandmete generaator. Testijalt nõutakse testitava veebivormi HTML formaadis faili, mis on ühtlasi ka testandmete genereerimise eeltingimus. Faili sisestamise kasutajaliides on toodud allpool oleval joonisel.



Joonis 7. Faili üleslaadimise kasutajaliides

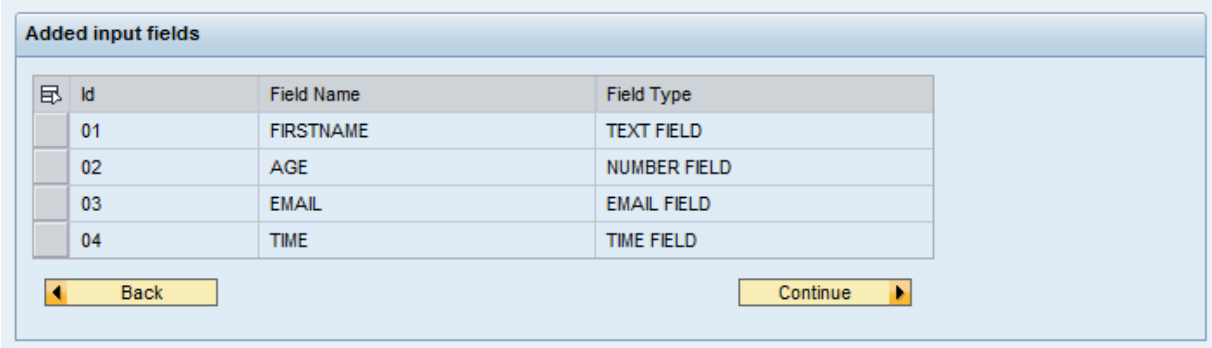
Allolev joonis annab ülevaate tegevustest, mida testandmete generaatori meetod *process_create_input_elements* (sisuliselt HTML faili parser) teostab sisestuselementide loomise protsessi vältel.



Joonis 8. Sisestuselementide loomist kirjeldav tegevusdiagramm

HTML input elementide tuvastamiseks otsitakse testija sisestatud failist kõik *input* nimega harud (*tag*), mis asuvad veebivormi ehk *form* haru sees. Igas *input* harus otsitakse omakorda *type* ja *name* harud, millede väärtused (*value*) kasutatakse, et luua uus sisestuselemendi objekt. Loodud objekt lisatakse massiivi, mis hoiab kõik loodud sisestuselemendid *input_elements*. Kirjeldatud protsessi teostatakse nii kaua kuni jõutakse HTML vormi lõppu.

Kui programm jõuab HTML vormi lõppu, käib testandmete generaator massiivi *input_elements* kõik sisestuselemendid, et kuvada testija etteantud faili alusel leitud sisestuselemendid koos nende nimede ja tüübiga, kasutades vastava sisestuselemendi objekti meetodid *get_input_element_name* ja *get_input_element_type*. Leitud sisestuselemendid on kuvatud kasutajaliideses tabeli kujul ning võimaldab testijal kontrollida, et kõik soovitud sisestuselemendid on süsteemi salvestatud.

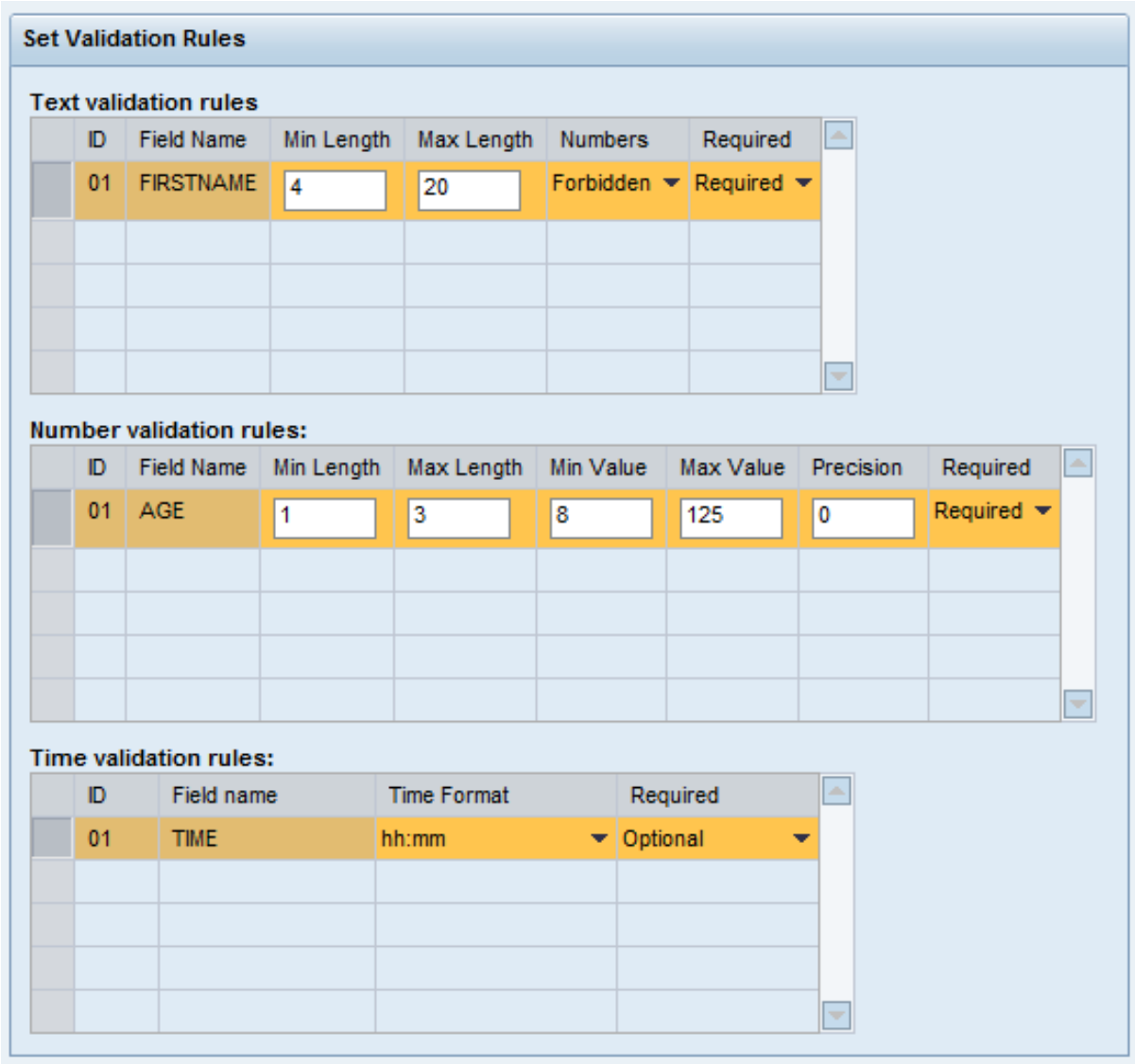


Joonis 9. Kasutajaliides lisatud sisestuselementide kontrollimiseks

Igal sisestuselemendi massiivis *validation_rules* on sellele sisestuselemendile iseloomulikud valideerimisreeglid, mis tuleb testijal käsitsi määrata. Selleks kuvatakse testijale kasutajaliides, kus on sisestuselemendi nimi ja selle sisestuselemendi valideerimisreeglid.

3.2 Sisestuselementide valideerimisreeglite määramine

Peale sisestuselementide loomist võimaldatakse testijal määrata iga sisestuselemendi kohta täiendavad sisestuselemendi valideerimisreeglid. Valideerimisreeglite määramise kasutajaliides on toodud allpool oleval joonisel.



Joonis 10. Valideerimisreeglite määramise kasutajaliides

Allpool olevas tabelis on toodud iga sisestuselemendi juures sätestatavad valideerimisreeglid, mis hoitakse sisestuselemendi massiivis *validation_rules*. Tabeli esimeses reas on sisestuselemendid ja tabeli esimeses veerus on valideerimisreeglid. Ristumiskohad, mis on märgitud tähega X, panevad paika iga sisestuselemendi testija poolt käsitsi sätestatavad valideerimisreeglid.

Tabel 7. Sisestuselementide ja valideerimisreeglite seoste tabel

	Tekst	Parool	Kuupäev	Email	Number	Kellaaeg	Telefon
min_length	X				X		X
max_length	X				X		X
min_value					X		
max_value					X		
[1]required	X	X	X	X	X	X	X
valid_length		X					
[1]upper_case		X					
[1]numbers	X	X					
[1]special_chars		X	X				X
[1]date_format			X				
precision					X		
[1]time_format						X	

1 Valideerimisreeglite *required, upper_case, numbers, special_chars, date_format* ja *time_format* puhul määratakse väärtused rippmenüüst, et vähendada veaohtlikke olukordi ja lihtsustada testandmete loomist. Vastavate valideerimisreeglite rippmenüü valikuelemendid olid toodud peatükis 2.1.3.

3.3 Testandmete loomine

Peale valideerimisreeglite määramist, teostab testandmete generaator testandmete loomist lähtudes käesoleva töö eespool kirjeldatud ekvivalentsiklassidesse jaotamise ja piirjuhtude analüüsimise meetoditest. Testandmete loomist alustab testandmete generaator protsessimeetodiga *process_create_input_data.*

Sisestuselementide massiivis *validation_rules* olevate valideerimisreeglite muutuja *value* väärtused kasutatakse ekvivalentsiklasside ja piirjuhtude loomiseks. Ekvivalentsiklass ja piirjuht ongi tegelik sisend, mis vastab kindlale tingimusele. Tingimuseks on üldjuhul valideerimisreeglile vastamine või mittevastamine. Kuid tingimuseks võib olla ka erinevate valideerimisreeglite kombinatsioon. Näiteks valideerimisreeglid *min_length* ja *max_length* on võimalik kombineerida, et jõustada tingimus, mis nõuab sisendilt, et selle pikkus oleks minimaalse ja maksimaalse pikkuse vahel.

Vastavalt sellele, kas sisend peab või ei pea vastama tingimustele, pannakse paika kehtivad ja kehtetud ekvivalentsiklassid ja piirjuhud. Järgmises peatükis kirjeldatakse testandmete generaatori protsessimeetodi *process_create_input_data* töötamise põhimõtted.

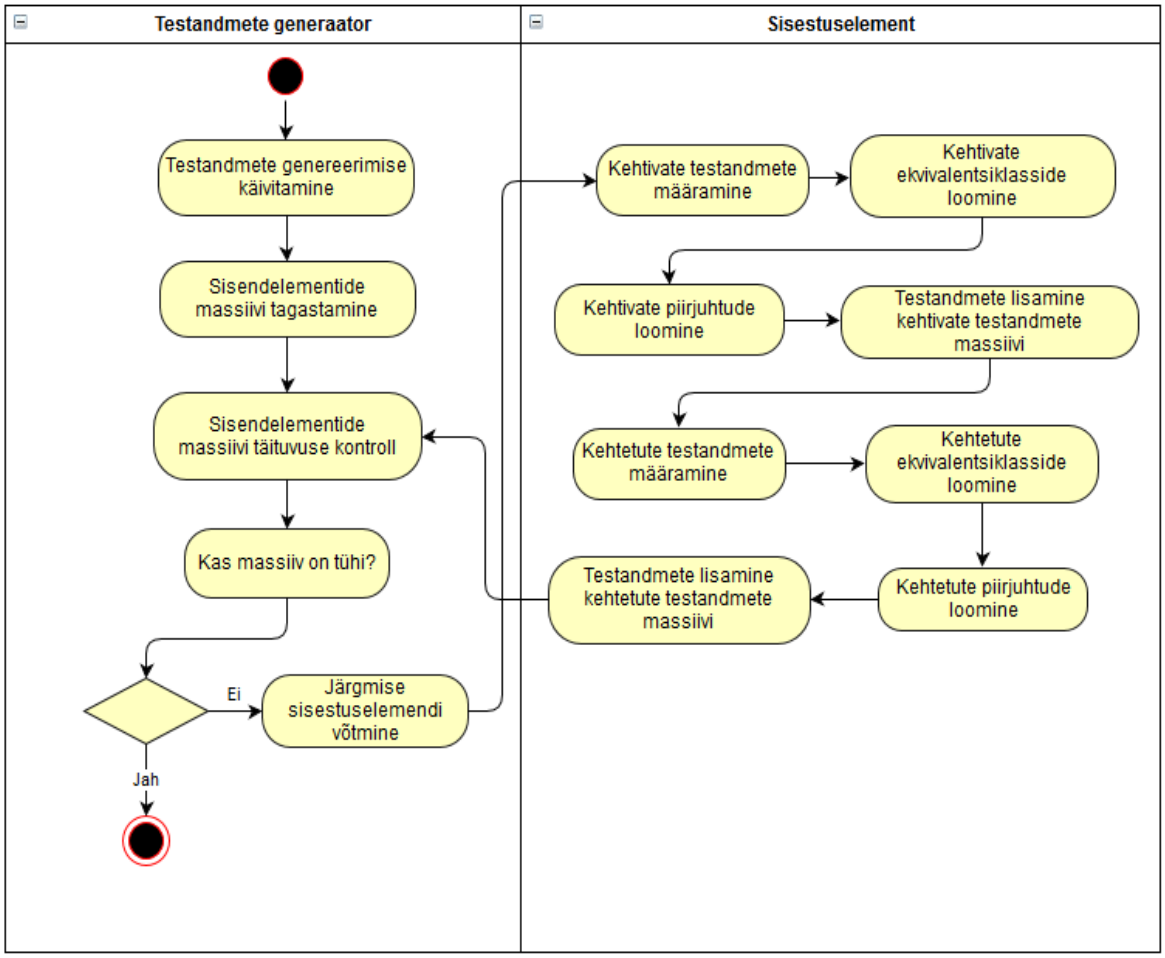
3.3.1 Testandmete loomise protsess

Testandmete loomiseks on vajalikud järgmiste eeltingimuste täitmine:

- Testandmete generaatori massiivis *input_elements* on vähemalt 1 sisestuselemendi objekt
- Sisestuselemendi massiivis *validation_rules* valideerimisreeglite muutuja *value* on väärtustatud

Testandmete generaator käib kõik sisestuselemendid läbi massiivis *input_fields* ning käivitab iga sisestuselemendi puhul meetodid *set_valid_input_data* ja *set_invalid_input_data*.

Protsess, kuidas iga sisestuselement loob testandmed, sõltub sisestuselementide valideerimisreeglitest, kuid suhtlust klasside *z_test_data_generator* ja *z_input_element* abstraktselt illustreerib allpool olev joonis.



Joonis 11. Testandmete generaatori ja sisestuselemendi suhtlus testandmete loomise protsessi jooksul

Meetod *set_valid_input_data* käivitab sisestuselemendi privaatsed meetodid *create_valid_equiv_classes* ja *create_valid_boundary_values*, mis loovad kehtivad testandmed ja lisavad need sisestuselemendi massiivi *valid_input_data*. Meetod *set_invalid_input_data* käivitab meetodid *create_invalid_equiv_classes* ja *create_invalid_boundary_values*, mis loovad kehtetud testandmed ja lisavad need sisestuselemendi massiivi *invalid_input_data*.

3.3.2 Valideerimisreeglite ekvivalentsiklassid ja piirjuhud

Järgnevalt kirjeldatakse iga sisestuselemendi kohta tema valideerimisreeglitest koostatavad ekvivalentsiklassid ja piirjuhud ning põhimõtted, mille järgi testandmeid luuakse. Ekvivalentsiklasside ja piirjuhtude koostamisel on autor võtnud aluseks [5].

Tekstivälja ekvivalentsiklassid ja piirjuhud

Allolevas tabelis on toodud mõned tekstivälja kehtivad ja kehtetud ekvivalentsiklassid ning piirjuhud. Lisaks on tabelis toodud testandmete loomise põhimõtted. Terviklik tabel on toodud lisas 2.

Tabel 8. Tekstivälja testandmete loomise põhimõtted

Ekvivalentsiklassi/piirjuhu kirjeldus	**Testandmete loomise põhimõte**	**Tüüp**
Testandmed, mis koosnevad tähemärkide arvust, mis on minimaalse ja maksimaalse tähemärkide arvude vahel	Abimeetodiga *get_string* saadakse tekst, mille tähemärkide pikkus on *min_length* ja *max_length* väärtuste keskmine, mida arvutatakse abimeetoditega *average* ja *floor*	Kehtiv
Testandmed, mille tähemärkide arv on võrdne minimaalsega	Abimeetodiga *get_string* saadakse tekst, mille tähemärkide pikkus on võrdne *min_length* väärtusega	Kehtiv
Testandmed, mille tähemärkide arv on poole võrra väiksem kui minimaalne	Abimeetodiga *get_string* saadakse tekst pikkusega, mis saadakse kui *min_length* jagatakse kahega ja tulemus ümardatakse abimeetodiga *floor*	Kehtetu
Testandmed, mille tähemärkide arv on poole võrra suurem kui maksimaalne	Abimeetodiga *get_string* saadakse tekst pikkusega, mis saadakse kui *max_length* korrutatakse kahega	Kehtetu

Paroolivälja ekvivalentsiklassid ja piirjuhud

Allolevas tabelis on toodud mõned paroolivälja kehtivad ja kehtetud ekvivalentsiklassid ning piirjuhud. Lisaks on tabelis toodud testandmete loomise põhimõtted. Terviklik tabel on toodud lisas 3.

Tabel 9. Paroolivälja testandmete loomise põhimõtted

Ekvivalentsiklassi/piirjuhu kirjeldus	**Testandmete loomise põhimõte**	**Tüüp**
Testandmed, mis sisaldavad määratud valideerimisreegleid	Abimeetodiga *get_string* luuakse etteantud pikkusega testandmed, mille lõpp asendatakse lubatud valideerimisreeglitega. St, kui lubatud on *upper_case*, *special_chars* ja *numbers*, siis *get_string* väärtuse viimased kolm tähemärki asendatakse abimeetodite *get_upper_case*, *get_special_char* ja *get_number* väärtustega	Kehtiv
Kui väli ei ole kohustuslik, siis tühiväärtus on lubatud	Kui *required* muutuja väärtus on *false*, siis lisatakse *null* ehk tühiväärtus massiivi *valid_input_data*	Kehtiv
Testandmed, mille tähemärkide pikkus on pool kehtivast pikkusest	Abimeetodiga *get_string* luuakse etteantud pikkusega testandmed, kusjuures pikkus saadakse jagades kehtivat pikkust kahega ja ümardades abimeetodiga *floor*	Kehtetu
Testandmed, mille tähemärkide pikkus on ühe võrra väiksem kui kehtiv pikkus	Abimeetodiga *get_string* luuakse testandmed pikkusega *valid_length – 1*	Kehtetu

Kuupäevavälja ekvivalentsiklassid ja piirjuhud

Tabel 10. Kuupäevavälja testandmete loomise põhimõtted

Ekvivalentsiklassi/piirjuhu kirjeldus	**Testandmete loomise põhimõte**	**Tüüp**
Testandmed, mis on õige kuupäeva formaadiga	Abimeetoditega *get_valid_day*, *get_valid_month* ja *get_valid_year* saadakse kehtiv kuupäev, kusjuures *get_valid_day* sõltub *get_valid_month* väärtusest. Kui formaat on eestipärane, siis eraldajana kasutatakse punkti, teisel juhul kaldkriipsi.	Kehtiv
Kui väli ei ole kohustuslik, siis tühiväärtus on lubatud	Kui *required* muutuja väärtus on *false*, siis lisatakse *null* ehk tühiväärtus massiivi *valid_input_data*	Kehtiv
Testandmed, mis on vale kuupäeva formaadiga	Abimeetoditega *get_valid_day*, *get_valid_month* ja *get_valid_year* saadakse kehtiv kuupäev, kusjuures *get_valid_day* sõltub *get_valid_month* väärtusest. Kui formaat on eestipärane, siis eraldajana kasutatakse kaldkriipsu, teisel juhul punkti.	Kehtetu
Testandmed, mis on vigase päevaga	Põhimõte on sarnane õige formaadiga kuupäeva formuleerimisel, ainult päev saadakse meetodiga *get_invalid_day*.	Kehtetu
Testandmed, mis on vigase kuuga	Põhimõte on sarnane õige formaadiga kuupäeva formuleerimisel, ainult päev saadakse meetodiga *get_invalid_month*.	Kehtetu
Kui väli on kohustuslik, siis tühiväärtus ei ole lubatud	Kui *required* muutuja väärtus on *true*, siis lisatakse *null* ehk tühiväärtus massiivi *invalid_input_data*	Kehtetu

E-maili välja ekvivalentsiklassid ja piirjuhud

Tabel 11. E-maili välja testandmete loomise põhimõtted

Ekvivalentsiklassi/piirjuhu kirjeldus	**Testandmete loomise põhimõte**	**Tüüp**
Testandmed, mis on õige e-maili formaadiga	Abimeetoditega *get_local_part* ja *get_domain* saadakse kehtiv e-maili formaad lisades lokaalse osa ja domeeni vahele "@" märk	Kehtiv
Kui väli ei ole kohustuslik, siis tühiväärtus on lubatud	Kui *required* muutuja väärtus on *false*, siis lisatakse *null* ehk tühiväärtus massiivi *valid_input_data*	Kehtiv
Testandmed, mis on vale e-maili formaadiga	Sarnane õige e-maili formaadiga testandmetele, ainult lokaalse osa ja domeeni vahele lisatakse "@" märgi asemel tühik	Kehtetu
Testandmed, mis on vale lokaalse osaga	Sarnane õige e-maili formaadiga testandmetele, ainult et lokaalne osa jäetakse tühjeks, st. e-mail koosneb "@" märgist ja domeenist	Kehtetu
Testandmed, mis on vale domeeniga	Sarnane õige e-maili formaadiga testandmetele, ainult et domeen jäetakse tühjeks, st. e-mail koosneb lokaalsest osast ja "@"	Kehtetu
Kui väli on kohustuslik, siis tühiväärtus ei ole lubatud	Kui *required* muutuja väärtus on *true*, siis lisatakse *null* ehk tühiväärtus massiivi *invalid_input_data*	Kehtetu

Numbrivälja ekvivalentsiklassid ja piirjuhud

Allolevas tabelis on toodud mõned numbrivälja kehtivad ja kehtetud ekvivalentsiklassid ning piirjuhud. Lisaks on tabelis toodud testandmete loomise põhimõtted. Terviklik tabel on toodud lisas 4.

Tabel 12. Numbrivälja testandmete loomise põhimõtted

Ekvivalentsiklassi/piirjuhu kirjeldus	**Testandmete loomise põhimõte**	**Tüüp**
Testandmed, mille pikkus on minimaalse ja maksimaalse pikkuse vahel	Abimeetodiga *get_number* saadakse ette antud pikkuse ja komakohtade arvuga numbrilise sisendi, kusjuures pikkus saadakse abimeetodiga *average*, mis omakorda ümardatakse abimeetodiga *floor*	Kehtiv
Testandmed, mis vastavad lubatud valideerimisreegli *precision* väärtusele	Abimeetoditega *average* ja *floor* ja valideerimisreeglite *min_length* ja *max_length* ja *max_value* saadakse keskmine pikkus. Kui *precision* väärtus on 0, siis abimeetodiga *get_number* tagastatakse täisarvuline sisend. Kui *precision* väärtus on 1, siis abimeetodiga *get_number* tagastatakse ratsionaalarvuline sisend. Kui *precision* väärtus on 2, siis saadakse nii täisarvuline kui ka ratsionaalarvuline sisend	Kehtiv
Testandmed, mille väärtus on poole võrra väiksem kui minimaalne väärtus	Saadakse kui *min_value* väärtust jagatakse kahega ja tulemus ümardatakse abimeetodiga *floor*	Kehtetu
Testandmed, mis ei vasta lubatud valideerimisreegli *precision* väärtusele	Abimeetoditega *average* ja *floor* ja valideerimisreeglite *min_length* ja *max_length* ja *max_value* saadakse keskmine pikkus. Kui *precision* väärtus on 0, siis abimeetodiga *get_number* saadakse ratsionaalarvuline sisend. Kui *precision* väärtus on 1, siis saadakse täisarvuline sisend	Kehtetu

Kellaajavälja ekvivalentsiklassid ja piirjuhud

Tabel 13. Kellaajavälja testandmete loomise põhimõtted

Ekvivalentsiklassi/piirjuhu kirjeldus	Testandmete loomise põhimõte	Tüüp
[1]Testandmed, mis on õige kellaaja formaadiga	24-tunnise formaadi puhul kasutatakse abimeetodid *get_valid_hours*, *get_valid_minutes* ja *get_valid_seconds* ning väärtuste eraldaks on koolon. Kui formaat on 12-tunnine, siis tunnid saadakse abimeetodiga *get_valid_us_hours* ja abimeetodiga *get_period* pannakse paika periood	Kehtiv
Kui väli ei ole kohustuslik, siis tühiväärtus on lubatud	Kui *required* muutuja väärtus on *false*, siis lisatakse *null* ehk tühiväärtus massiivi *valid_input_data*	Kehtiv
Testandmed, mis on vale kellaaja formaadiga	Kui kehtiv formaat on 24-tunnine, siis luuakse 12-tunnise formaadiga kellaaeg ja vastupidi	Kehtetu
Testandmed, mis on vigase tunni osaga	Kellaaja formaat luuakse sarnaselt õige kellaaja formaadiga testandmetele, selle vahega, et tunnid saadakse meetoditega *get_invalid_hours* või *get_invalid_us_hours*	Kehtetu
Testandmed, mis on vigase minuti osaga	Kellaaja formaat luuakse sarnaselt õige kellaaja formaadiga testandmetele, selle vahega, et minutid saadakse meetoditega *get_invalid_minutes*	Kehtetu
[2]Testandmed, mis on vigase sekundi osaga	Kellaaja formaat luuakse sarnaselt õige kellaaja formaadiga testandmetele, selle vahega, et minutid saadakse meetoditega *get_invalid_seconds*	Kehtetu
Kui väli on kohustuslik, siis tühiväärtus ei ole lubatud	Kui *required* muutuja väärtus on *true*, siis lisatakse *null* ehk tühiväärtus massiivi *invalid_input_data*	Kehtetu

1 Abimeetod *get_valid_seconds* kasutatakse siis, kui kehtiv formaat sisaldab sekundi osa.

2 Sisend kehtetu sekundi osaga luuakse siis, kui kehtiv formaat sisaldab sekundi osa.

Telefonivälja ekvivalentsiklassid ja piirjuhud

Allolevas tabelis on toodud mõned telefonivälja kehtivad ja kehtetud ekvivalentsiklassid ning piirjuhud. Lisaks on tabelis toodud testandmete loomise põhimõtted. Terviklik tabel on toodud lisas 5.

Tabel 14. Telefonivälja testandmete loomise põhimõtted

Ekvivalentsiklassi/piirjuhu kirjeldus	Testandmete loomise põhimõte	Tüüp
Testandmed, mis vastavad määratud telefoninumbri formaadile	Vastavalt määratud telefoninumbri formaadile, lisatakse abimeetodi *get_zone* väärtuse ette "+" märk või pannakse väärtus sulgudesse. Seejärel koostatakse abimeetodiga *get_number* etteantud pikkusega number, mis jaotatakse pooleks etteantud eraldusmärgiga abimeetodiga *split*	Kehtiv
Testandmed, mille pikkus on minimaalse ja maksimaalse pikkuse vahel	Kasutatakse määratud telefoninumbri formaadis testandmed	Kehtiv
Testandmed, mille formaat ei vasta määratud telefoninumbri formaadile	Testandmete loomiseks kasutatakse formaati, mis ei vasta määratud formaadile	Kehtetu
Testandmed, mille pikkus on pool minimaalsest pikkusest	Sarnaselt määratud telefoninumbri formaadiga testandmetele, ainult pikkus saadakse kui *min_length* jagatakse kahega ja ümardatakse abimeetodiga *floor*	Kehtetu

3.4 Testikomplekti koostamine

Testandmete genereerimise protsessi viimases etapis luuakse testandmete generaatori protsessimeetodiga *process_create_test_suite* testikomplekt iga sisestuselemendi valideerimisreeglite testimiseks. Selleks, et igat valideerimisreeglit põhjalikult testida, on vaja luua eraldi testid kehtivate ja kehtetute testandmetega. Kusjuures kehtetute testandmetega testi koostamisel, valitakse ühe sisestuselemendi üks kehtetu ekvivalentsiklass või piirjuht ning kõikidelt teistelt sisestuselementidelt valitakse kehtiv ekvivalentsiklass või piirjuht. Seega testikomplekti koostamise eeldused on järgmised:

- Testandmete generaatori massiivis *input_elements* on vähemalt üks sisestuselement

- Sisestuselemendi massiivis *validation_rules* olevad valideerimisreeglid on väärtustatud
- Sisestuselemendi kehtivate testandmete massiiv *valid_input_data* ja kehtetute testandmete massiiv *invalid_input_data* ei ole tühjad

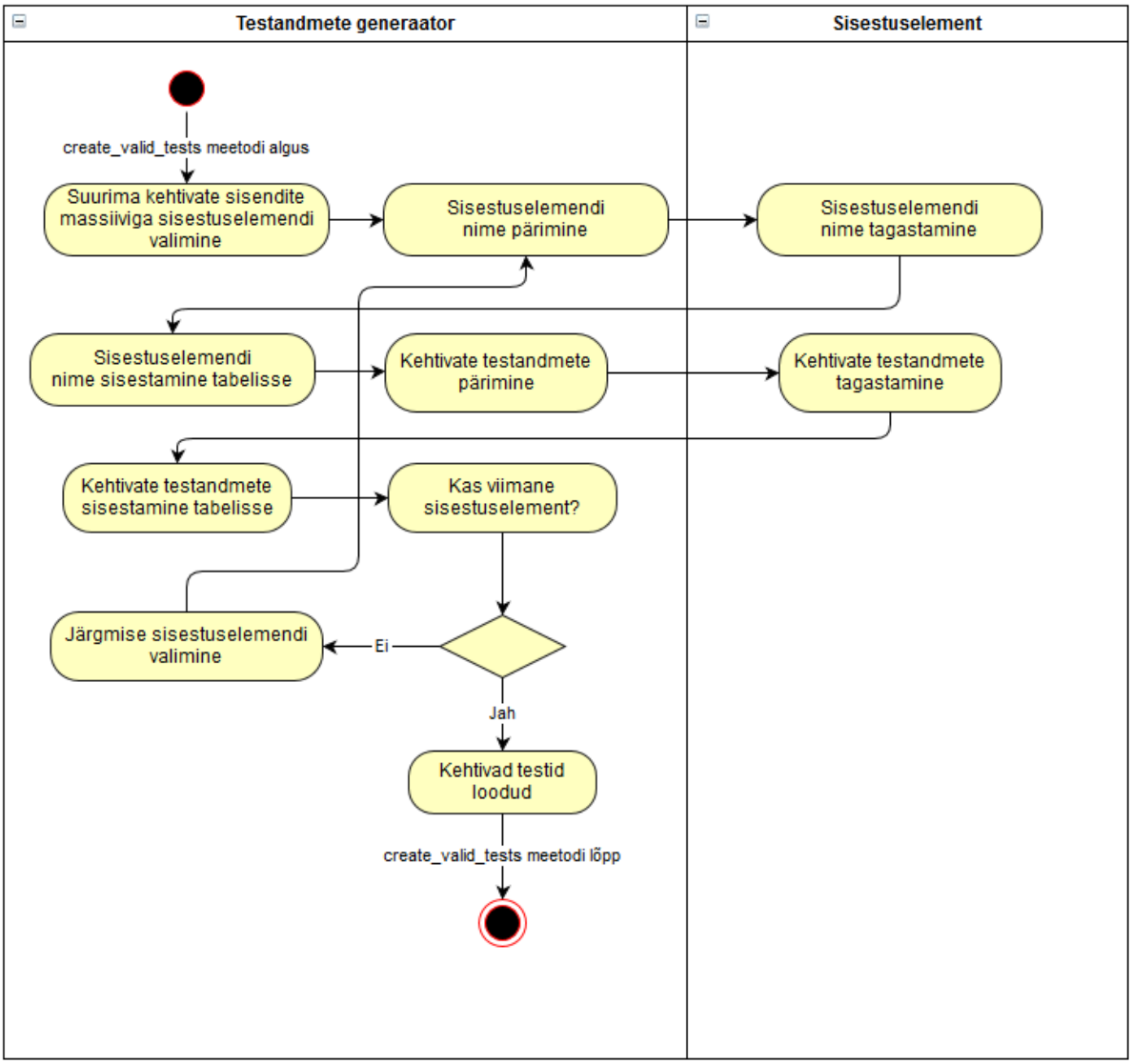
Testikomplekt koosneb kehtivatest ja kehtetutest testidest. Kehtivate testide koostamiseks on testandmete generaatoril meetod *create_valid_tests* ja kehtetute testide koostamise jaoks on meetod *create_invalid_tests.* Testikomplekti loomise käivitamiseks on testandmete generaatoril meetod *create_test_suite*, mis omakorda käivitab meetodid *create_valid_tests* ja *create_invalid_tests*. Meetodite käivitamise tulemuseks on tabel *test_suite*, mille iga rida vastab ühele testile.

Testikomplekti koostamise protsess koosneb kahest etapist. Esimeses etapis luuakse kehtivad testid meetodiga *create_valid_tests* ning teises etapis luuakse kehtetud testid meetodi *create_invalid_tests* abil. Järgnevates peatükkides kirjeldan igat etapi detailsemalt.

3.4.1 Kehtivate testide loomine

Kehtivad testid on sellised testid, mis koosnevad sisenditest, mis peavad valideeruma. See tähendab, et testitav programm aktsepteerib sisendid ning testi jooksul ei teki ühtegi viga või veateadet testitava programmi poolt. Kui mõni veateade või ootamatu viga on tekkinud, tähendab see seda, et testitavas programmis on leitud viga või programmi nõuded, mille järgi sisendelementide valideerimisreeglid määrati, ei lähe kokku programmi realisatsiooniga.

Kehtivate testide loomise protsessi kirjeldab alloleval joonisel toodud tegevusdiagramm.



Joonis 12. Kehtivate testide loomise protsessi kirjeldav tegevusdiagramm

Kehtivate testide loomise meetod *create_valid_tests* leiab esimesena suurima kehtivate sisendite massiiviga sisestuselemendi. See on vajalik selleks, et iga sisendelemendi sisendid saaksid testitud. Tabeli esimesele reale sisestatakse sisestuselemendi nimi, mida saadakse meetodiga *get_input_element_name*. Sisestuselemendi nime alla sisestatakse sisestuselemendi kehtivad sisendid, mida saadakse sisestuselemendi meetodiga *get_valid_input_data*.

Kui suurima kehtivate sisendite massiiviga sisestuselement koos kehtivate sisenditega on sisestatud, kontrollitakse kas massiivi *input_elements* on tühi. Kui ei ole, siis võetakse massiivist uus sisestuselement ning lisatakse tabelisse sisestuselemendi nimi ning tema kehtivad sisendid nii kaua kuni tabelis olevate sisestuselementide kehtivate sisendite arv on ühesugune. Selle tagajärjel mõne sisestuselemendi puhul võib sama kehtiv sisend esineda

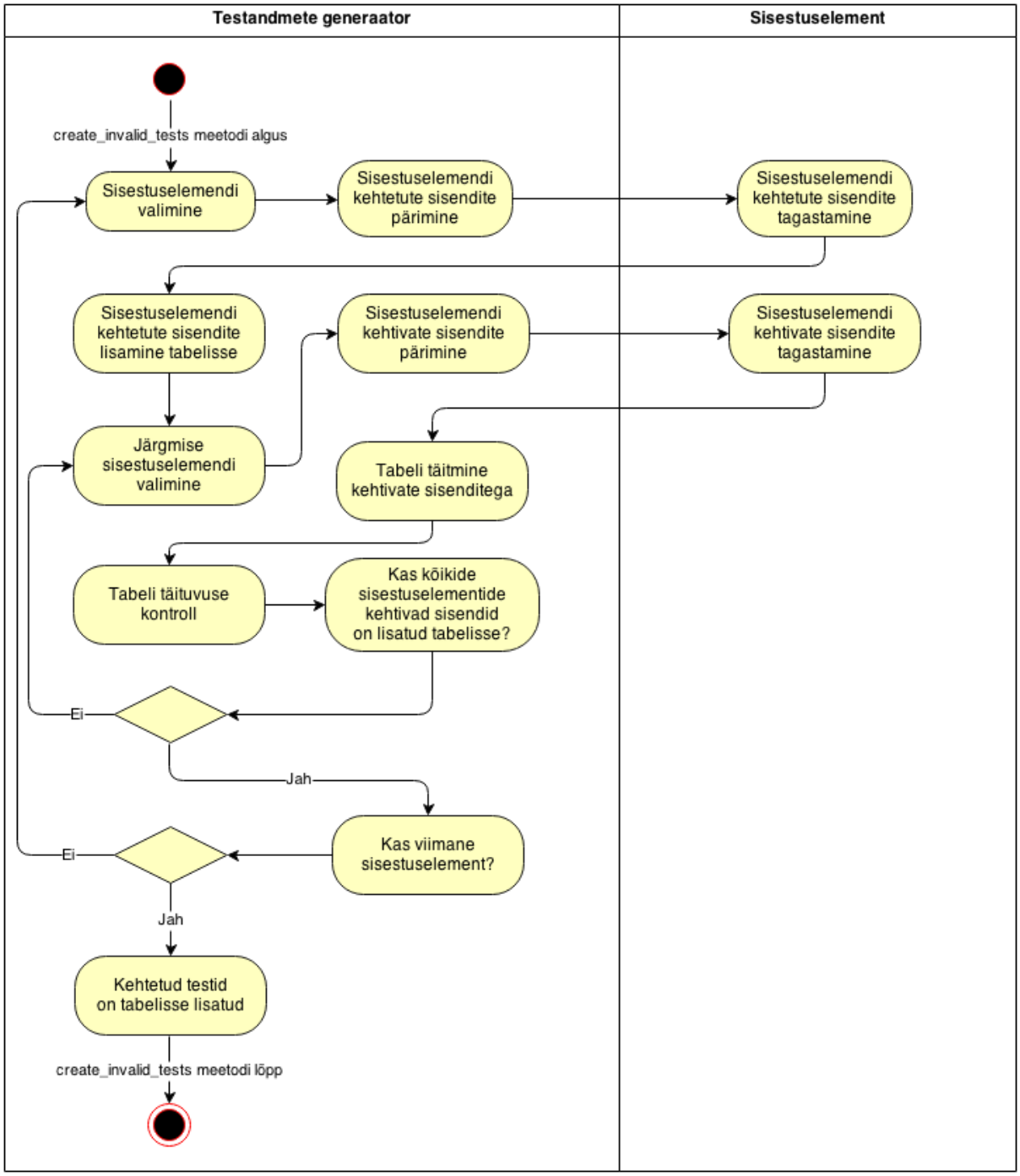
mitu korda. Sellisel põhimõttel sisestuselementide kehtivate sisendite tabelisse sisestamine jätkub, kuni kõik massiivi *input_elements* sisestuselemendid on tabelisse sisestatud.

Meetodi *create_valid_tests* lõpus on tabel täidetud sisendelementide nimedega ning kõikide sisestuselementide kõik sisendid on kaetud vastavate kehtivate testidega.

3.4.2 Kehtetute testide loomine

Kehtetu testi all on mõeldud testi, mis peab kindlasti läbi kukkuma. See tähendab, et testis on selline sisend (kehtetu sisend), mille peale testitav tarkvara peab vastama veateatega. Kui kehtetu test ei kukku läbi, siis on tegemist veaga testitavas programmis või testitava programmi nõuded, mille järgi valideerimisreeglid määrati, ei lähe kokku programmi realisatsiooniga.

Kehtetute testida loomise protsessi kirjeldab allpool toodud tegevusdiagramm.



Joonis 13. Kehtetute testide loomise protsessi kirjeldav tegevusdiagramm

Kehtivate testide koostamisel lähtutakse põhimõttest, et iga testi eesmärk on testida ainult ühte kehtetut sisendit korraga. See tähendab, et iga test koosneb ühe sisestuselemendi kehtetust sisendist ja kõikide teiste sisestuselementide kehtivatest sisenditest.

Meetodi *create_invalid_tests* käivitamisel valitakse massiivist *input_elements* esimese sisestuselemendi. Kuna kehtivate testide loomisel said sisestuselementide nimed

testikomplekti tabelisse sisestatud, siis nüüd jäetakse see samm vahele ja päritakse kohe valitud sisestuselemendi kehtetute sisendite massiivi *invalid_input_data*.

Testandmete generaatori lisab kõik kehtetud sisendid massiivist *invalid_input_data* testikomplekti tabelisse vastava sisestuselemendi alla. Kuna igas kehtetus testis võib olla ainult üks kehtetu sisend, siis nüüd valitakse järjest alles jäänud sisestuselemendid, päritakse nende kehtivate sisendite massiiv *valid_input_data* ja lisatakse testikomplekti tabelisse vastava sisestuselemendi all. Kehtivaid sisendeid lisatakse iga sisestuselemendi alla sama palju kui eelnevas sammus lisati kehtetuid sisendeid. See tähendab, et mõne sisestuselemendi all võib olla korduvaid kehtivaid sisendeid või mõnda kehtivat sisendit ei pruugi üldse kasutada.

Peale iga sisestuselemendi kehtivate sisendite lisamist tabelisse, kontrollitakse kas kõik tabeli sisestuselemendi kehtivad sisendid on lisatud tabelisse. Kui tingimus ei vasta tõele, siis võetakse järgmine sisestuselement, millelt päritakse taas kehtivaid sisendeid. Kui tingimus vastab tõele, siis kontrollitakse kas sisestuselement, mille kehtetud sisendid tabellise lisati, oli massiivi *input_elements* viimane. Kui tingimus vastab tõele, siis kõikide sisestuselementide kehtetud testid on lisatud testikomplekti tabelisse. Kui tingimus ei vasta tõele, siis võetakse järjekorras järgmine sisestuselement, mille kehtetuid sisendeid hakatakse tabelisse lisama ning seejärel algab uuesti kehtivate sisendite lisamine.

Meetodi *create_invalid_tests* käivitamise tulemusel on testikomplekti tabel täidetud nüüd ka kehtetute testidega. Sellega on kõik kehtivad ja kehtetud ekvivalentsiklassid ja piirjuhud kaetud vastavate testidega.

Kui testikomplekti tabel on koostatud, siis konverteerib testandmete generaator tabelit JSON formaadis failiks, mida testijale võimaldatakse allalaadida. Tabel konverteeritakse JSON formaadis failiks kasutades ABAP standard klassi *CL_TREX_JSON_SERIALIZER*.

Järgmises peatükis kirjeldatakse, kuidas testandmete generaatori JSON formaadis väljundfaili on võimalik implementeerida automaattesti skripti failis.

4. Testandmete generaatori väljundfail

Testandmete generaatori protsessimeetodi *process_create_test_suite* tulemusel konverteeritakse ABAP standard klassi *CL_TREX_JSON_SERIALIZER* abil ABAP-i sisemise tabeli (*InternalTable*), kus hoitakse testid koos testandmetega, JSON formaadis failiks. Kuigi JSON formaadiks konverteerimiseks kasutati programmeerimiskeele ABAP spetsiifilist klassi, on ka teistes programmeerimiskeeltes vastavad JSON failide lugemiseks ja kirjutamiseks võimalused olemas (nt. *Gson*, *JSONObject*, *JSONArray* Java programmeerimiskeeles).

Selleks, et testandmete generaatori väljundfaili kasutada on vaja teada, kuidas hoitakse ABAP testikomplektid JSON failis. Allpool on toodud lühendatud kujul (kaks testi) testikomplekt peale JSON formaati konverteerimist.

```
[
	{
		"firstname" : "plkdfaAdlv",
		"age" : 21,
		"email" : "sopd.2.2s@yahoo.co.uk",
		"time" : "7:51 AM"
	},
	{
		"firstname" : "asdad",
		"age" : 23,
		"email" : "german.mumma@gmail.com",
		"time" : "21:23:12"
	}
]
```

Iga ABAP sisemise tabeli test hoitakse loogelistes sulgudes ja testid on eraldatud komaga. Igat testi kirjeldavad atribuudid ja nende väärtused. Atribuudid on ABAP sisemise tabeli päises olevad sisestuselementide nimetused (*firstname, age, jne.*) ning atribuudi väärtus on iga sisestuselemendile vastavad testandmed.

Testandmete generaatori väljund faili implementeerimiseks on igal programmeerimiskeelel olemas oma vastavad teegid JSON failide lugemiseks ja kirjutamiseks. Järgnevalt

kirjeldatakse JSON faili implementeerimist *Selenium WebDriver* raamistikus programmeerimiskeeles Java.

Selenium WebDriver on veebilehitseja automatiseerija. *WebDriver*-it kasutatakse eelkõige kasutaja tüüpiliste käitumismustrite nagu teksti sisestamine lahtritesse, erinevate valikelementide kasutamine, nuppude vajutamine ja muu sellise automatiseerimiseks. Kui nimetatud raamistikuga automatiseerida mõne registreerimisvormi testimist, siis käesolevas töös realiseeritud testandmete generaatori JSON formaadis väljund faili implementeerimine näeks välja järgmiselt. Käesoleva näite jaoks on kasutatud Java teeki *Gson* ja eespool toodud lihtsustatud JSON faili sisu.

1) Eelnevalt tuleb luua Java objekt:

```
public class TestCase {

  String firstname;
  int age;
  String email;
  String time;
}
```

2) Kui teste sisaldav faili nimi on *test_suite.json*, siis testide objektide loomine näeks välja järgmiselt:

```
Gson gson = new Gson();
TestCase[] test_cases = gson.fromJson(new
FileReader("test_suite.json"), TestCase[].class);
```

3) Nüüd on kõik testikomplekti failis olevad testid muudetud objektideks *TestCase* ja lisatud massiivi *test_cases*. Seejärel tuleb iga sisestusväljale teha *WebElement* tüüpi objekt:

```
WebElement firstname =
driver.findElement(By.name("firstname"));
WebElement age = driver.findElement(By.name("age"));
WebElement email = driver.findElement(By.name("email"));
WebElement time = driver.findElement(By.name("time"));
```

4) *WebElement* objektid võimaldavad *WebDriver* raamistikul suhelda testitava vormi tegelike sisestuselementidega kasutades meetodit *sendKeys(String text)*, kus *text* on sisend, mida soovitakse sisestuselementi sisestada. Sisendiks kasutatakse nüüd eespool

loodud *test_cases* massiivis olevate testide atribuutide väärtused. Järgnevas koodi näites on tehtud tsükkel, mis käib kõik massiivis *test_suite* olevad *TestCase* objektid ja iga sisestusemendi puhul sisestatakse vastav atribuudi väärtus:

```
for (int test_case_index = 0; test_case_index <
test_suite.length; test_case_index++) {

     firstname.sendKeys(test_suite[test_case_index].first
name);
     age.sendKeys(test_suite[test_case_index].age);
     email.sendKeys(test_suite[test_case_index].email);
     time.sendKeys(test_suite[test_case_index].email);

     /** Edasised sammud */
}
```

Kasutatud näide on küllaltki lihtsustatud, kuid samal ajal piisavalt detailne ja ülevaatlik, et kirjeldada testandmete generaatori JSON formaadis väljund faili implementeerimist automaattesti skriptides. Kirjeldatud protsessiga on vähendatud testija vaeva iseseisvalt sobilike testandmete loomisel ja sisestamisel. Taolist protsessi on võimalik kasutada erinevates programmeerimiskeeltes kirjutatud automaattesti skriptides. Eeltingimusteks on JSON faili parser, tsüklid, massiivid ja *WebDriver*, et automatiseerida suhtlust veebilehitsejaga (Java puhul *Selenium Webdriver*, Ruby puhul *Watir WebDriver*, kuid veebilehitseja suhtluse automatiseerimiseks ei pea ilmtingimata kasutama *WebDriver* raamistikku.

Kokkuvõte

Käesoleva töö eesmärgiks oli käsitöö vähendamine graafiliste kasutajaliideste testimisel. Eesmärgi saavutamiseks realiseerisin testandmete generaatori, mis toetab ekvivalentsiklassidesse jaotumise ja piirjuhtude analüüsi testimise meetoditel põhinevale testandmete genereerimise protsessi.

Töö tulemusena leian, et töö käigus realiseeritud testandmete generaator vähendab käsitööd ekvivalentsiklasside jaotumise ja piirjuhtude analüüsi testide koostamisel. Samuti suurendab testandmete generaatori poolt loodud testandmete kasutamine testide läbiviimisel vea avastamise tõenäosust, sest testandmete generaatori loob testandmeid toetudes testija poolt määratud valideerimisreeglitele.

Lisaks, näen töö tulemusena käesolevat tööd juhendina, kuidas on võimalik sarnast eesmärki saavutada sõltumata programmeerimiskeelest. Realisatsioonis kasutatud muutuja tüübid ja massiivid on iga programmeerimiskeele põhikomponendid ning meetodid on küllaltki lihtsa loogikaga. Realiseerimist teeb lihtsamaks programmeerimiskeele objekt-orienteeritus ja polümorfismi tugi.

Samuti on realiseeritud testandmete generaator võimalik kerge vaevaga rikastada täiendavate sisestuselementide või valideerimisreeglite tüüpidega. Mainitud muudatuste implementeerimine ei nõua olemasoleva loogika muutmist.

Käesoleva töö eesmärk saavutati programmeerimiskeele ABAP ja kasutajaliideste programmeerimisvahendiga SAP WebDynpro ABAP. Nimetatud raamistiku kasutamine lihtsustas vajaliku kasutajaliidese realiseerimist. Programmeerimiskeele ABAP standard klasside kasutamine lihtsustas sisestuselementide loomist ning testandmete genereerimise viimases etapis loodud testikomplekti konverteerimist JSON formaadis failiks.

Summary

The goal of this thesis was to decrease manual workload when testing graphical user interfaces. The goal was to be achieved by realizing a test data generator which supports test data creation based on equivalence partitioning and boundary value analysis testing techniques.

Testing graphical user interfaces can be a tedious task. Designing tests with high probability of defect detection is also a hard task. The author realized test data generator supports test data creation by using tester defined validation rules in order to create test data that has high chance of finding defects.

The results of this thesis can be used to create test data generators independent of the programming language as long as object-oriented programming and polymorphism are supported. The JSON files created by the test data generator can easily be implemented in automated test scripts to decrease manual labor even further while preserving high probability of defect detection.

Kasutatud kirjandus

1. Black, R., Evans, I., Graham, D., Van Veenendaal, E. Foundations of Software Testing. ISTQB Certification. Revised Edition.Cengage Learn EMEA, 2008.

2. Fewster M., Graham D. Software Test Automation. Effective use of test execution tools. Addison-Wesley, 1999.

3. Myers, J. G. The Art of Software Testing. Second Edition. John Wiley & Sons, Inc., 2004.

4. Patton, R. Software Testing. Sams Publishing, 2001.

5. Markvardt, M. Vain J. Testiandmete genereerimine sisendi valideerimiseks. TTÜ, Tallinn 2007.

6. Wikipedia. Software Quality. [WWW] http://en.wikipedia.org/wiki/Software_quality (16.04.2014).

7. Wikipedia. Test Data Generation. [WWW] http://en.wikipedia.org/wiki/Test_data_generation (17.04.2014).

Lisa 1. Testandmete generaatori abimeetodid ja nende kirjeldus

Abimeetodi nimi	Sisendid	Väljundid	Kirjeldus
get_string	integer *string_length*	string *output_string*	Tagastab väärtuse, mis koostatakse suvalisi tähti kombineerides, kuni saadakse *string_length* pikkune tekstiline väärtus
average	float *value1*, float *value2*	float *average_value*	Tagastab keskmise väärtuse, jagades summa *value1* + *value2* kahega
floor	float *input_value*	integer *output_value*	Tagastab väärtuse, mis saadakse kui *input_value* ümardatakse alla. St kui *input_value* = 45.95, siis tagastatakse 45. Ümardatakse väiksema täisarvuni
get_number	integer *length*, integer *precision*	float *output_number*	Tagastab etteantud pikkuse (*length*) ning komakohtade arvuga (*precision*) numbrilise väärtuse
get_upper_case	-	char *upper_case_letter*	Tagastab ühe suvalise suurtähe suurtähtedega täidetud massiivist
get_special_char	-	char *special_char*	Tagastab ühe suvalise erimärgi massiivist, kus hoitakse erinevaid erimärki
get_valid_day	-	integer *valid_day*	Sõltuvalt meetodi *get_valid_month* väärtusest tagastatakse number ulatuses 1 kuni *valid_month_days*, kus *valid_month_days* on kehtiva kuu päevade arv
get_valid_month	-	integer *valid_month*	Tagastab väärtuse ulatuses 1 kuni 12
get_valid_year	-	integer *valid_year*	Tagastab neljakohalise numbri

get_invalid_day	-	integer *invalid_day*	Sõltuvalt meetodi *get_valid_month* väärtusest tagastatakse number, mille väärtus on *valid_month_days* + 1, kus *valid_month_days* on kehtiva kuu päevade arv
get_invalid_month	-	integer *invalid_month*	Tagastatakse väärtus 13, mis ei vasta kehtivale kuu väärtusele
get_local_part	-	string *local_part*	Koostatakse tekstiline väärtus, mis sisaldab ühekordseid punkte. Nt *asdf.fhsksasd*
get_domain	-	string *domain*	Domeeni võitakse olemasolevast massiivist, mis sisaldab enamlevinud domeene nagu *google.com*, *ttu.ede.ee* ja *yahoo.co.uk*
get_valid_hours	-	integer *valid_hours*	Tagastatakse täisarvuline väärtus ulatuses 1 kuni 24
get_valid_minutes	-	integer *valid_minutes*	Tagastatakse täisarvuline väärtus ulatuses 1 kuni 60
get_valid_seconds	-	integer *valid_minutes*	Tagastatakse täisarvuline väärtus ulatuses 1 kuni 60
get_valid_us_hours	-	integer *valid_us_hours*	Tagastatakse täisarvuline väärtus ulatuses 1 kuni 12
get_period	-	string *period*	Tagastatakse kas *AM* või *PM*
get_invalid_hours	-	integer *invalid_hours*	Tagastatakse täisarvuline väärtus 25
get_invalid_minutes	-	integer *invalid_minutes*	Tagastatakse täisarvuline väärtus 61
get_invalid_seconds	-	integer *invalid_seconds*	Tagastatakse täisarvuline väärtus 61
get_invalid_us_hours	-	integer *invalid_us_hours*	Tagastatakse täisarvuline väärtus 13
get_zone	-	integer *zone*	Tagastatakse täisarvuline väärtus pikkusega 1 kuni 3 numbri märki
split	string *input_string*, char *separator*	string *output_string*	Asendab sisendis *input_string* üht elementi ette antud *separator* väärtusega

Lisa 2. Tekstivälja ekvivalentsiklassid ja piirjuhud

Ekvivalentsiklassi/piirjuhu kirjeldus	**Testandmete loomise põhimõte**	**Tüüp**
Testandmed, mis koosnevad tähemärkide arvust, mis on minimaalse ja maksimaalse tähemärkide arvude vahel	Abimeetodiga *get_string* saadakse tekst, mille tähemärkide pikkus on *min_length* ja *max_length* väärtuste keskmine, mida arvutatakse abimeetoditega *average* ja *floor*	Kehtiv
Testandmed, mille tähemärkide arv on võrdne minimaalsega	Abimeetodiga *get_string* saadakse tekst, mille tähemärkide pikkus on võrdne *min_length* väärtusega	Kehtiv
Testandmed, mille tähemärkide arv on võrdne maksimaalsega	Abimeetodiga *get_string* saadakse tekst, mille tähemärkide pikkus on võrdne *max_length* väärtusega	Kehtiv
Testandmed, mille tähemärkide arv on ühe võrra suurem kui minimaalne	Abimeetodiga *get_string* saadakse tekst, mille tähemärkide pikkus on võrdne *min_length* + 1 väärtusega	Kehtiv
Testandmed, mille tähemärkide arv on ühe võrra väiksem kui maksimaalne	Abimeetodiga *get_string* saadakse tekst, mille tähemärkide pikkus on võrdne *max_length* – 1 väärtusega	Kehtiv
Kui valideerimisreegli *numbers* väärtus on *true*, siis saadakse sisend, mis sisaldab nii tähti kui numbreid	Abimeetodiga *get_string* saadakse sisend, mille lõppu lisatakse üks number abimeetodiga *get_numbers*	Kehtiv
Kui väli ei ole kohustuslik, siis tühiväärtus on lubatud	Kui *required* muutuja väärtus on *false*, siis lisatakse *null* ehk tühiväärtus massiivi *valid_input_data*	Kehtiv
Testandmed, mille tähemärkide arv on poole võrra väiksem kui minimaalne	Abimeetodiga *get_string* saadakse tekst pikkusega, mis saadakse kui *min_length* jagatakse kahega ja tulemus ümardatakse abimeetodiga *floor*	Kehtetu
Testandmed, mille tähemärkide arv on poole võrra suurem kui maksimaalne	Abimeetodiga *get_string* saadakse tekst pikkusega, mis saadakse kui *max_length* korrutatakse kahega	Kehtetu
Kui väli on kohustuslik, siis tühiväärtus ei ole lubatud	Kui *required* muutuja väärtus on *true*, siis lisatakse *null* ehk tühiväärtus massiivi *invalid_input_data*	Kehtetu
Testandmed, mille tähemärkide arv on ühe võrra väiksem kui minimaalne	Abimeetodiga *get_string* saadakse tekst, mille tähemärkide pikkus võrdub *min_length* – 1 väärtusega	Kehtetu

Testandmed, mille tähemärkide arv on ühe võrra suurem kui maksimaalne	Abimeetodiga *get_string* saadakse tekst, mille tähemärkide pikkus võrdub *max_length* + 1 väärtusega	Kehtetu

Lisa 3. Paroolivälja ekvivalentsiklassid ja piirjuhud

Ekvivalentsiklassi/piirjuhu kirjeldus	**Testandmete loomise põhimõte**	**Tüüp**
Testandmed, mis sisaldavad määratud valideerimisreegleid	Abimeetodiga *get_string* luuakse etteantud pikkusega testandmed, mille lõpp asendatakse lubatud valideerimisreeglitega. St, kui lubatud on *upper_case*, *special_chars* ja *numbers*, siis *get_string* väärtuse viimased kolm tähemärki asendatakse abimeetodite *get_upper_case*, *get_special_char* ja *get_number* väärtustega	Kehtiv
Kui väli ei ole kohustuslik, siis tühiväärtus on lubatud	Kui *required* muutuja väärtus on *false*, siis lisatakse *null* ehk tühiväärtus massiivi *valid_input_data*	Kehtiv
Testandmed, mille tähemärkide pikkus on võrdne kehtiva pikkusega	Kasutatakse eelnevalt loodud testandmeid, mis sisaldavad määratud valideerimisreegleid	Kehtiv
Testandmed, mis ei sisalda ühtegi lubatud valideerimisreeglit	Abimeetodiga *get_string* luuakse etteantud pikkusega testandmed	Kehtetu
Testandmed, mille tähemärkide pikkus on pool kehtivast pikkusest	Abimeetodiga *get_string* luuakse etteantud pikkusega testandmed, kusjuures pikkus saadakse jagades kehtivat pikkust kahega ja ümardades abimeetodiga *floor*	Kehtetu
Testandmed, mille tähemärkide pikkus on kaks korda suurem kui kehtiv pikkus	Abimeetodiga *get_string* luuakse etteantud pikkusega testandmed, kusjuures pikkus saadakse korrutades kehtivat pikkust kahega	Kehtetu
Kui väli on kohustuslik, siis tühiväärtus ei ole lubatud	Kui *required* muutuja väärtus on *true* siis lisatakse *null* ehk tühiväärtus massiivi *invalid_input_data*	Kehtetu
[1]Testandmed, mis ei sisalda ühte lubatud valideerimisreeglit	Abimeetodiga *get_string* luuakse etteantud pikkusega testandmed, mille lõpp asendatakse ainult kahe valideerimisreegliga. St, kui lubatud on *upper_case*, *special_chars* ja *numbers*, siis *get_string* väärtuse asendatakse abimeetodite *get_upper_case* ja *get_special_char* väärtustega. Ehk testandmeid ei vasta ühele konkreetsele valideerimisreeglile	Kehtetu

Testandmed, mille tähemärkide pikkus on ühe võrra väiksem kui kehtiv pikkus	Abimeetodiga *get_string* luuakse testandmed pikkusega *valid_length – 1*	Kehtetu
Testandmed, mille tähemärkide pikkus on ühe võrra suurem kui kehtiv pikkus	Abimeetodiga *get_string* luuakse testandmed pikkusega *valid_length + 1*	Kehtetu

1 Kui lubatud on kõik kolm valideerimisreeglit, siis luuakse kolm sisendit, üks iga valideerimisreegli kohta. Testandmeid luuakse iga lubatud valideerimisreegli kohta, et testida iga valideerimisreegli kehtivust.

Lisa 4. Numbrivälja ekvivalentsiklassid ja piirjuhud

Ekvivalentsiklassi/piirjuhu kirjeldus	**Testandmete loomise põhimõte**	**Tüüp**
Testandmed, mille väärtus on minimaalse ja maksimaalse väärtuse vahel	Abimeetoditega *average* ja *floor* ja valideerimisreeglite *min_value* ja *max_value* väärtustega saadakse keskmine väärtus	Kehtiv
Testandmed, mille pikkus on minimaalse ja maksimaalse pikkuse vahel	Abimeetodiga *get_number* saadakse ette antud pikkuse ja komakohtade arvuga numbrilise sisendi, kusjuures pikkus saadakse abimeetodiga *average*, mis omakorda ümardatakse abimeetodiga *floor*	Kehtiv
Testandmed, mis vastavad lubatud valideerimisreegli *precision* väärtusele	Abimeetoditega *average* ja *floor* ja valideerimisreeglite *min_length* ja *max_length* ja *max_value* saadakse keskmine pikkus. Kui *precision* väärtus on *Allowed*, siis abimeetodiga *get_number* tagastatakse täisarvuline sisend. Kui *precision* väärtus on 1, siis abimeetodiga *get_number* tagastatakse ratsionaalarvuline sisend.	Kehtiv
Kui väli ei ole kohustuslik, siis luuakse tühiväärtus	Kui *required* muutuja väärtus on *false*, siis lisatakse *null* ehk tühiväärtus massiivi *valid_input_data*	Kehtiv
Testandmed, mille väärtus on võrdne minimaalse väärtusega	Kasutatakse valideerimisreegli *min_value* väärtust	Kehtiv
Testandmed, mille väärtus on võrdne maksimaalse väärtusega	Kasutatakse valideerimisreegli *max_value* väärtust	Kehtiv
Testandmed, mille väärtus on ühe võrra suurem kui minimaalne väärtus	Kasutatakse väärtust, mis on võrdne *min_value* +1	Kehtiv
Testandmed, mille väärtus on ühe võrra väiksem kui maksimaalne väärtus	Kasutatakse väärtust, mis on võrdne *max_value* – 1	Kehtiv
Testandmed, mille pikkus on võrdne minimaalse pikkusega	Kasutades abimeetodit *get_number* saadakse numbriline sisend pikkusega, mis on võrdne *min_length* väärtusega	Kehtiv
Testandmed, mille pikkus on võrdne maksimaalse pikkusega	Kasutades abimeetodit *get_number* saadakse numbriline sisend pikkusega, mis on võrdne *max_length* väärtusega	Kehtiv
Testandmed, mille pikkus on ühe võrra suurem kui minimaalne pikkus	Kasutades abimeetodit *get_number* saadakse numbriline sisend pikkusega, mis on võrdne *min_length* + 1 väärtusega	Kehtiv

Testandmed, mille pikkus on ühe võrra väiksem kui maksimaalne pikkus	Kasutades abimeetodit *get_number* saadakse numbriline sisend pikkusega, mis on võrdne *max_length* - 1 väärtusega	Kehtiv
Testandmed, mille väärtus on poole võrra väiksem kui minimaalne väärtus	Saadakse kui *min_value* väärtust jagatakse kahega ja tulemus ümardatakse abimeetodiga *floor*	Kehtetu
Testandmed, mille väärtus on poole võrra suurem kui maksimaalne väärtus	Saadakse kui *max_value* väärtust korrutatakse kahega	Kehtetu
Testandmed, mille pikkus on poole võrra väiksem kui minimaalne pikkus	Kasutades abimeetodit *get_number* ja pikkust, mis saadakse *min_length* jagades kahega ja tulemus ümardades abimeetodiga *floor*, saadakse numbriline sisend	Kehtetu
Testandmed, mille pikkus on poole võrra suurem kui maksimaalne pikkus	Kasutades abimeetodit *get_number* ja pikkust, mis saadakse korrutades *max_length* kahega, saadakse numbriline sisend	Kehtetu
Kui väli on kohustuslik, siis luuakse tühiväärtus	Kui *required* muutuja väärtus on *true*, siis lisatakse *null* ehk tühiväärtus massiivi *invalid_input_data*	Kehtetu
Testandmed, mis ei ole numbrilise väärtusega	Kasutades abimeetodeid *get_string*, *average* ja *floor* ja valideerimisreegleid *min_length* ja *max_length* saadakse keskmise pikkusega tekstiline sisend	Kehtetu
Testandmed, mille väärtus on ühe võrra väiksem kui minimaalne väärtus	Kasutatakse väärtust, mis on võrdne *min_value* – 1	Kehtetu
Testandmed, mille väärtus on ühe võrra suurem kui maksimaalne väärtus	Kasutatakse väärtust, mis on võrdne *max_value* + 1	Kehtetu
Testandmed, mille pikkus on ühe võrra väiksem kui minimaalne pikkus	Kasutades abimeetodit *get_number* saadakse numbriline sisend pikkusega, mis on võrdne *min_length* – 1	Kehtetu
Testandmed, mille pikkus on ühe võrra suurem kui maksimaalne pikkus	Kasutades abimeetodit *get_number* saadakse numbriline sisend pikkusega, mis on võrdne *max_length* + 1	Kehtetu
Testandmed, mis ei vasta lubatud valideerimisreegli *precision* väärtusele	Abimeetoditega *average* ja *floor* ja valideerimisreeglite *min_length* ja *max_length* ja *max_value* saadakse keskmine pikkus. Kui *precision* väärtus on 0, siis abimeetodiga *get_number* saadakse ratsionaalarvuline sisend. Kui *precision* väärtus on 1, siis saadakse täisarvuline sisend	Kehtetu

Lisa 5. Telefonivälja ekvivalentsiklassid ja piirjuhud

Ekvivalentsiklassi/piirjuhu kirjeldus	Testandmete loomise põhimõte	Tüüp
Testandmed, mis vastavad määratud telefoninumbri formaadile	Vastavalt määratud telefoninumbri formaadile, lisatakse abimeetodi *get_zone* väärtuse ette "+" märk või pannakse väärtus sulgudesse. Seejärel koostatakse abimeetodiga *get_number* etteantud pikkusega number, mis jaotatakse pooleks etteantud eraldusmärgiga abimeetodiga *split*	Kehtiv
Testandmed, mille pikkus on minimaalse ja maksimaalse pikkuse vahel	Kasutatakse määratud telefoninumbri formaadis testandmed	Kehtiv
Kui väli ei ole kohustuslik, siis tühiväärtus on lubatud	Kui *required* muutuja väärtus on *false*, siis lisatakse *null* ehk tühiväärtus massiivi *valid_input_data*	Kehtiv
Testandmed, mille pikkus on võrdne minimaalse pikkusega	Sarnaselt määratud telefoninumbri formaadiga testandmetele, ainult pikkus on võrdne *min_length* väärtusega	Kehtiv
Testandmed, mille pikkus on võrdne maksimaalse pikkusega	Sarnaselt määratud telefoninumbri formaadiga testandmetele, ainult pikkus on võrdne *max_length* väärtusega	Kehtiv
Testandmed, mille pikkus on ühe võrra suurem kui minimaalne pikkus	Sarnaselt määratud telefoninumbri formaadiga testandmetele, ainult pikkus on võrdne *min_length* + 1 väärtusega	Kehtiv
Testandmed, mille pikkus on ühe võrra väikse kui maksimaalne pikkus	Sarnaselt määratud telefoninumbri formaadiga testandmetele, ainult pikkus on võrdne *max_length* - 1 väärtusega	Kehtiv
Testandmed, mille formaat ei vasta määratud telefoninumbri formaadile	Testandmete loomiseks kasutatakse formaati, mis ei vasta määratud formaadile	Kehtetu
Testandmed, mille pikkus on pool minimaalsest pikkusest	Sarnaselt määratud telefoninumbri formaadiga testandmetele, ainult pikkus saadakse kui *min_length* jagatakse kahega ja ümardatakse abimeetodiga *floor*	Kehtetu
Testandmed, mille pikkus on kaks korda maksimaalset pikkust	Sarnaselt määratud telefoninumbri formaadiga testandmetele, ainult pikkus saadakse kui *max_length* korrutatakse kahega	Kehtetu

Kui väli on kohustuslik, siis tühiväärtus ei ole lubatud	Kui *required* muutuja väärtus on *true*, siis lisatakse *null* ehk tühiväärtus massiivi *invalid_input_data*	Kehtetu
Testandmed, mille pikkus on ühe võrra väiksem kui minimaalne pikkus	Sarnaselt määratud telefoninumbri formaadiga testandmetele, ainult pikkus on võrdne *min_length* - 1	Kehtetu
Testandmed, mille pikkus on ühe võrra suurem kui maksimaalne pikkus	Sarnaselt määratud telefoninumbri formaadiga testandmetele, ainult pikkus on võrdne *max_length* + 1	Kehtetu