TALLINNA ÜLIKOOL
INFORMAATIKA INSTITUUT

Kaisa Kaljuma

Ülevaade testimistehnikatest

Seminaritöö

Juhendaja: Marek Kusmin

Tallinn
2008

Sisukord

Sissejuhatus

Enamus tänapäevastest majandusharudest on olulises sõltuvuses arvutisüsteemidest, tarkvarast ning seetõttu ka nende arendusest. Arenduste kvaliteet on vastavate ettevõtete jaoks kriitilise, sageli elulise tähtsusega. Selleks, et olla kindel uute või muudetud süsteemide vastamises esitatud kvaliteedinõuetele pühendatakse aina enam tähelepanu testimisele.

Käesolevas töös antakse ülevaade olulisematest tarkvara testimise tehnikatest. Testimine on programmi vastavuse hindamine programmi kirjeldava dokumentatsiooniga. See tähendab, et testimisega tehakse kindlaks, kas tarkvara töötab õigesti. Testimise peamine eesmärk on leida vigu, lisaks mõõdetakse testimisega programmi kvaliteeti.

Kaasaegses testimisprotsessis rakendatakse erinevaid testimistehnikaid. Testimistehnika on universaalne ja standardiseeritud meetod, mille abil tuletatakse olemasolevast informatsioonist testitavad olukorrad. Testimistehnikate abil on näiteks suurem võimalus jälgida testide kvaliteeti ning katvust ja planeerida ning kontrollida testimisprotsessi.

Siinses seminaritöös antakse ülevaade põhilistest testimistehnikatest. Tehnikatele on võimalusel juurde toodud ka näited.

Teema aktuaalsus

Struktureeritud testimist tähtsustatakse aina rohkem. Kaootiliselt testides ei saada kindlat ülevaadet, kui põhjalikult on testitava süsteemi erinevad osad testidega kaetud ning millised osad on veel testimata. Lisaks ei ole teste võimalik täpselt korrata, kuna testide sooritamise tingimused ei ole kindlaks määratud.

Samuti pööratakse järjest enam suuremat rõhku testimise planeerimisele ja ettevalmistusele. Põhjaliku eeltööga on võimalik tagada sama kvaliteet vähema ajaga, kui kohe testima hakates. Seda seetõttu, et eeltöö käigus selgitatakse täpselt välja kõik testimist vajavad olukorrad, mille põhjal planeeritakse edasine tegevus. Nagu eespool mainitakse, nii tegutsedes ja testimist struktureerides on võimalik saavutada süsteemi väga hea katvus testidega, lisaks muutub testimise protsess läbipaistvamaks ja kontrollitavamaks ning on suurem tõenäosus, et vead leitakse varakult.

Teema valiku põhjendus

Huvi ja mõte sellel teemal kirjutamiseks tekkis autoril seoses oma ülesannetega tööl. Nimelt on autori töökohas üks eesmärkidest testimise teadlik struktureerimine ning seega peavad töötajad oskama rakendada muuhulgas ka testimistehnikaid. Kahjuks eesti keelne testimistehnikate-alane kirjandus on praktiliselt olematu. Ka võõrkeelset materjali ei ole palju ning lisaks võib võõrkeelse materjali järgi uue valdkonna õppimine osutuda liiga keeruliseks. Sellest tekkiski mõte koostada seminaritöö raames ülevaade testimistehnikatest. Sama teemat arendatakse edasi ka bakalaureuse töös, kus käsitletakse testimistehnikaid ja nende kasutust lähtudes vajadustest ja eripäradest autori töökohas.

Töö eesmärgid

Käesoleva seminaritöö eesmärgiks on erinevatest testimistehnikatest ülevaate andmine ning testimise planeerimise ja teostamise esmase abimaterjali pakkumine. Lisaks võib antud tööd käsitleda õppematerjalina ning samas sobib see lugemiseks kõigile testimise vastu huvi tundvatele isikutele.

1. Testimistehnikad

1.1. Algoritmi tehnika (ingl. k. *Algorithm test*)

Algoritmi tehnika eesmärk on testida programmi struktuuri. Tehnika testjuhud tuletatakse programmi algoritmist, mis on kujutatud skeemina. Skeemi läbimisel tuleks tähelepanu pöörata võimalikele teedele, mitte otsustus- ja tegevuspunktide sisusse. Iga testjuht sisaldab gruppi otsustuspunktide väärtusi, mis määravad tee läbi algoritmi. Olenevalt funktsionaalsuse kriitilisusest on võimalik testida erineva sügavusega sõltuvalt mitme otsustuspunktijagu tagasi vaadatakse. Kõige levinum on sügavus 1. See tähendab, et igas otsustuspunktis moodustatakse paarid, mis koosnevad ühest sisend- ja ühest väljundharust. Selliste unikaalsete paaride arv iga otsustuspunkti jaoks võrdub sisendharude arvu ja väljundharude arvu korrutisega. Seejärel sobitatakse harude paarid omavahel kokku ning moodustatakse teed. Teed läbivad skeemi otsast lõpuni ning iga tee on võrdeline ühe testjuhuga. [1, lk 207-211]

Testbaas (*Test basis*)

Algoritmi tehnika kasutatamise eelduseks on teadmine programmi struktuurist. Testbaasiks on funktsionaal- ja ärinõuded, mille põhjal koostatakse skeem ja tabel sooritatavatest tegevustest. [1, lk 207-211]

Kasutus

Algoritmi tehnikat kasutatakse erinevate moodulite testimisel ja integratsioonitestide puhul. Lisaks piisab vahel juba algoritmi skeemist, et avastada süsteemianalüüsi kitsaskohti ning vigu. [1, lk 207-211]

Katvus

Algoritmi tehnikat kasutades on võimalik saavutada järgmisi katvusi:
Otsustuspunktide katvus, sügavus 0 *(Decision coverage)* – iga otsustuspunkti otsus läbitakse vähemalt üks kord. See tähendab, et ka iga haru läbitakse vähemalt üks kord.
Tee^n katvus, sügavus >= 1 (*Path^n coverage*) – Sügavus 1 saavutatakse, kui skeemis ei ole tsükleid ning kõik võimalikud teed läbitakse vähemalt ühe korra. Suuremat

sügavust on mõtekas kasutada vaid tsüklite korral, mille puhul läbitakse teed vastavalt tsüklite sügavusele 2 või rohkem korda. [1, lk 207-211]

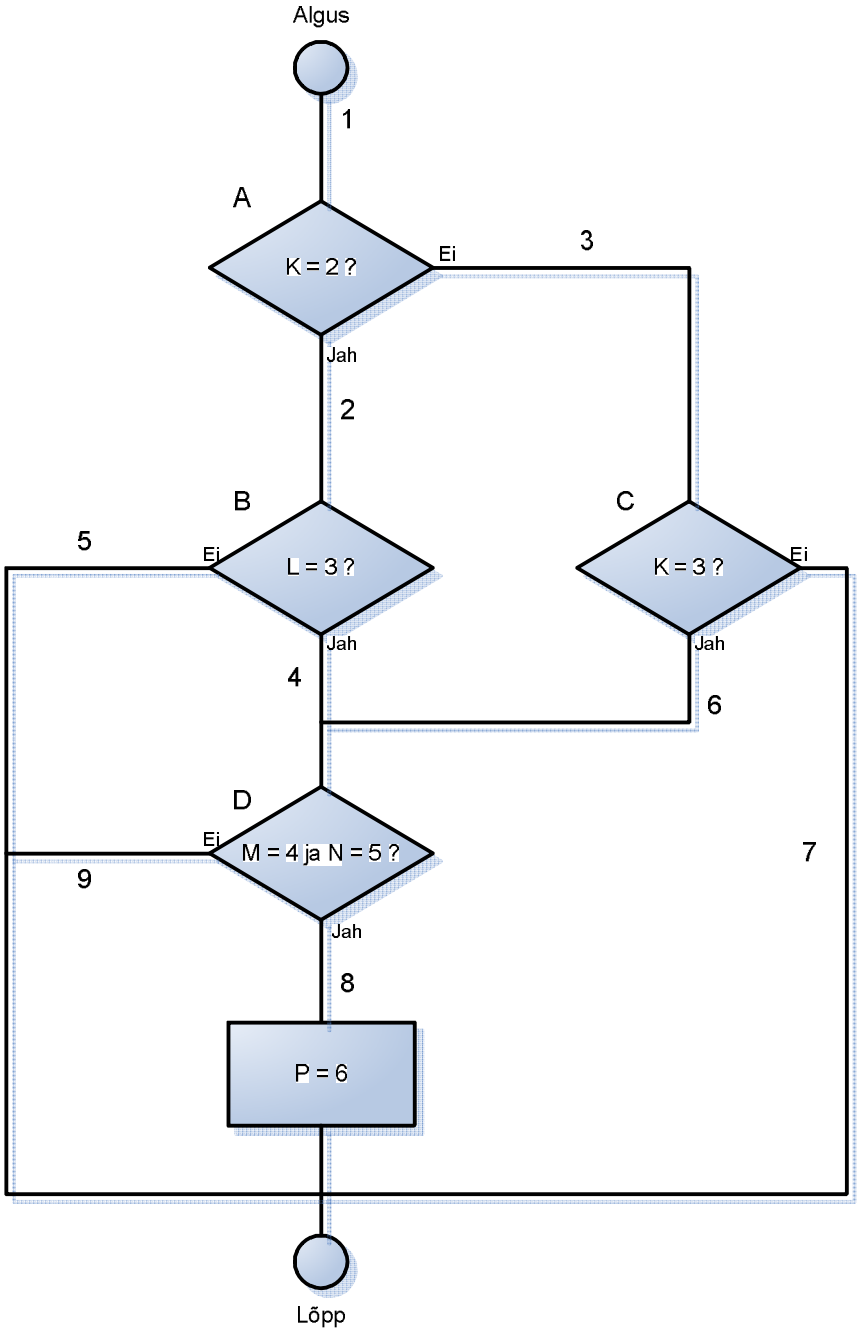
Näide

Struktuuri ülevaate koostamine

Rakendades algoritmi tehnikat tuleks kõigepealt uurida testbaasi ehk funktsionaal- ja ärinõudeid sisaldavaid dokumente ning nende põhjal koostada ülevaade algoritmi struktuurist. Ülevaade võib olla nii skeemina, diagrammina kui ka otsustustabelina. Samuti võib ülevaate koostamisel tugineda ka pseudo-koodile, nagu siin olevas näites on tehtud.

Pseudo-kood:

```
IF K=2
THEN
        IF L=3
        THEN
                IF M=4 AND N=5
                        THEN P=6
                ELSE END IF
        ELSE END IF
ELSE
        IF K=3
        THEN
                IF M=4 AND N=5
                        THEN P=6
                ELSE END IF
        ELSE END IF
```



Joonis 1. Ülevaade algoritmi testimistehnika rakendamise näite testbaasi struktuurist skeemina

Otsustuspunktide ja teede välja selgitamine

Ülevaate põhjal tähistatakse erinevad otsustuspunktid ning teed sümbolitega. Näiteks võiks otsustuspunktid märgistada tähtedega (A, B, C, …) ning teed numbritega (1, 2, 3, …).

Olenevalt soovitavast sügavusest kombineeritakse omavahel järjestikused teelõigud, mis moodustavad testitavad testjuhud ehk teed, mis läbivad skeemi algusest lõpuni. Näiteks, kui soovitud saavutatav sügavus = 1, siis koosneb üks teelõik ühest harust

enne otsustuspunkti ja ühest harust pärast otsustuspunkti. Seega uuritakse kahte järjestikust haru ehk teelõiku. Sellel põhimõttel kirjutatakse välja kõik kombinatsioonid kahest järjestikusest harust.

Otsustuspunktide tegevuste kombinatsioonid:

A:	1.2	1.3
B:	2.4	2.5
C:	3.6	3.7
D:	4.8	4.9
	6.8	6.9

Järgnevalt ühendatakse need kombinatsioonid teedeks, mis läbivad algoritmi algusest lõpuni. Eesmärk on teedes kasutada kõiki teelõikude kombinatsioone nii, et moodustuks võimalikult vähe testjuhte ja samas peab iga teelõik olema kasutatud vähemalt üks kord.

Testjuht 1:	1.2	2.4	4.8		Testjuht 1:	1.2.4.8
Testjuht 2:	1.3	3.6	6.8		Testjuht 2:	1.3.6.8
Testjuht 3:	1.2	2.5		→	Testjuht 3:	1.2.5
Testjuht 4:	1.3	3.7			Testjuht 4:	1.3.7
Testjuht 5:	1.2	2.4	4.9		Testjuht 5:	1.2.4.9
Testjuht 6:	1.3	3.6	6.9		Testjuht 6:	1.3.6.9

Kasutades algoritmi tehnikat on seda skeemi võimalik läbida kuut erinevat teed pidi, mis tähendab, et selle ülesande testimiseks tuleks sooritada kuus testjuhtu.

1.2. Andmete kombineerimise tehnika (*Data combination test*)

Andmete kombineerimise tehnikat kasutatakse nii programmi funktsionaalsuse detailseks testimiseks, kui ka süsteemi üldiseks testimiseks. Testbaasist tuletatakse muutujad ning neid kombineerides saadakse testjuhud. Testjuhu tulemuse määravad muutujate väärtused. Oluline on, et mistahes vaadeldava muutuja väärtuse muutumine kajastuks ka testi tulemuses. Vastasel juhul võivad süsteemivead jääda leidmata.

Testbaas

Andmete kombineerimise tehnika jaoks ei ole üksikasjalik testbaas vajalik. Kuna testida on võimalik ka tuginedes eriteadmistele, on seda tehnikat võimalik kasutada ka pooliku, aegunud või puuduva testbaasi korral. Tehnikat saab kasutada tuginedes ametlikule dokumentatsioonile, nagu näiteks funktsionaal- ja ärinõuded, kuid sobivad ka käsiraamatud, märkmed, ülevaated ja valdkonna eriteadmisted.
Samas, kuna testbaas võib olla poolik või põhineda ainult valdkonna eriteadmistel, siis sõltub testimise kvaliteet otseselt testija oskustest ning võib juhtuda, et testimise tulemused ei ole usaldusväärsed. [2, lk 648-649]

Kasutus

Andmete kombineerimise tehnikat sobib rakendada nii kriitiliste funktsioonide, kui ka lihtsalt kiiret testimist vajavate süsteemiosade testimiseks.
Kuna heade testjuhtude koostamine on võimalik vähese jõupingutusega, suudavad ka need, kes ei ole IT-spetsialistid, testjuhte kergesti koostada, tuginedes vaid enda arusaamale infosüsteemist. [1, lk 212-213]

Katvus

Kasutades andmete kombineerimise tehnikat, on võimalik saavutada järgnevaid katvusi:
Ekvivalentsiklassid (*equivalence classes*) – nõuded jaotatakse ekvivalentsiklassideks, millest valitakse muutujad. Kaetavus saavutatakse, kui testjuhtude sooritamisel valitakse igast ekvivalentsiklassist vähemalt üks muutuja.
Piirväärtused (*boundary value analysis*) – testjuhud koostatakse nõuetest tuletatud muutujate ja nende piirväärtuste põhjal. Katvus saavutatakse, kui kombineeritakse

kõiki saadud muutujaid ja piirväärtusi nii, et kõigi muutujate piirväärtused on läbi proovitud vähemalt üks kord.

Kõik tulemused – täielik katvus saavutatakse, kui testjuhud testitakse kõikide võimalike muutujatega. Täieliku katvuse saavutamiseks kulutatakse liiga palju ressurssi, mistõttu pole täieliku katvuse rakendamine tihti otstarbekas. [2, lk 648-649]

Näide

Järgnev näide illustreerib andmete kombineerimise tehnika kasutamist situatsioonis, kus klient soovib saada elukindlustust.

Testimisalus

Tingimused, mis peavad olema täidetud, et klient saaks elukindlustust:

- Lepingu alguskuupäev peab olema tänane või tulevikus
- Kindlustussumma saaja kindlustatu surma korral ei ole kohustuslik
- Minimaalne kindlustatu vanus = 18
- Maksimaalne kindlustatu vanus:
    - Kui kindlustussumma=6000, siis vanus kuni 120
    - Kui kindlustussumma=15 000, siis vanus kuni 45

  (ainult kaks kindlustussumma võimalust: 5000 ja 15000)
- Minimaalne kindlustusaeg = 2
- Maksimaalne kindlustusaeg = 40
- Minimaalne kuumakse:

Kindlustusperiood	<4 aastat	>=4 ja <7	>=7 ja <10	>=10
Minimaalne kuumakse	250	200	175	150

Tabel 1. Testimisalus andmete kombineerimise tehnika rakendamise näite jaoks.

Nõuete lahti kirjutamine erinevateks sisendväärtusteks:

Alguskuupäev	Sysdate-1; Sysdate; Sysdate+1;
Kindlustussumma saaja surma korral	Jah; Ei
Kindlustatud isiku vanus	17; 18; 46; 121;
Kindlustussumma	6000; 15 000
Kindlustuse periood	1; 5; 8; 40; 45
Minimaalne kuumakse	250; 200, 175, 150

Tabel 2. Andmete kombineerimise tehnika näite nõudete kirjeldus.

Märkus: Tabelis olev sysdate tähistab hetkel süsteemis olevat kuupäeva

Testjuhtude koostamine etteantud tingimuste põhjal

Erinevate tingimuste kombinatsioonid kirjutatakse välja nii, et igasse testjuhtu jääb maksimaalselt üks nõuetes mittelubatud sisendväärtus.

Testjuhud	Alguskpv	Kindlustussumma saaja	Kindlustatu vanus	Summa	Periood	Kuumakse	Tulemus
1.	**Sysdate-1**	Jah	**17**	6000	1	250	Väär
2.	Sysdate	Ei	18	15000	5	200	Tõene
3.	Sysdate+1	Jah	46	6000	8	175	Tõene
4.	Sysdate	Ei	**121**	15000	40	150	Tõene
5.	Sysdate	Jah	18	6000	**45**	150	Väär
1.1.	**Sysdate-1**	Jah	18	6000	1	250	Väär
1.2.	Sysdate	Jah	**17**	6000	1	250	Väär

Tabel 3. Andmete kombineerimise tehnika rakendamiseks vajalike testjuhtude kirjeldus.

Märkus: Tabelis poolpaksus kirjas on välja toodud mittesobivad sisendväärtused.

Selle näite testimiseks andmete kombineerimise tehnikaga on vajalik sooritada 7 testjuhtu.

1.3. Andmetsükli tehnika (*Data Cycle Test*)

Andmetsükli tehnikaga testitakse, kas erinevad (alam)süsteemid kasutavad ja töötlevad andmeid järjepidevalt. Tehnika peamine eesmärk on leida süsteemide ja nende funktsioonide integreerimise vigu ning see sobib süsteemide üldiseks funktsionaalsuse ja ühenduvuse testimiseks. Peamiselt keskendutaksegi funktsioonide ühendustele (liidestele) ning nende poolt ühiselt kasutatavate andmete töötlemise testimisele. [2, lk 670]

Andmetsükli tehnikat kasutatakse andmete elutsükli testimiseks. Tihti erinevates süsteemides andmed luuakse, kasutatakse, muudetakse ning lõpuks kustutatakse. Andmetsükli testimise käigus kontrollitakse, kas andmeid töödeldakse nendes süsteemides õigesti. [1, lk 219]

Andmetsükli tehnikat on võimalik kasutada kahte moodi. Esimene moodus on vaadelda põhiandmeid kontrollivaid funktsioone. Nende all mõeldakse funktsioone, millega sooritatakse andmetega peamised tegevused (andmete loomine, lugemine, muutmine, kustutamine). Tavaliselt on testijatel need tegevused lubatud, seetõttu on mõistlik neid funktsioone oma testides ka kasutada. Selle mooduse testimiseks ei ole CRUD (*Create, Read, Update, Delete*, vt allpool) maatriksi kasutamine kohustuslik. [1, lk 220]

Teine võimalus on andmetsükli testimine süsteemi tasemel, mille rakendamisel on CRUD maatriks kohustuslik. Testitavate andmetsüklite hulk sõltub mõjutatud süsteemide ja/või funktsioonide hulgast. [1, lk 220]

Testbaas

Andmetsükli tehnika rakendamisel on kõige sobivam testbaas CRUD maatriks ning rakendatavate tervikluse reeglite kirjeldus. Lisaks on vaja funktsionaalnõudeid või detailset valdkonnapõhist ülevaadet, et teada iga sooritatava testjuhu oodatud tulemust. [2, lk 670]

Kasutus

Andmetsükli tehnika kasutamine on kõige efektiivsem, kui süsteemide funktsionaalsus ja andmete kasutamine testitakse kõigepealt eraldi ning seejärel

testitakse süsteeme integreeritult. Seetõttu sobib andmetsükli tehnikat kasutada testimise hilises etapis. [2, lk 670]

Katvus

Andmetsükli tehnikaga on võimalik saavutada järgmine katvus:

- **CRUD** *(Create, Read, Update, Delete)* – kasutatakse andmete elutsükli katvuse saavutamiseks. CRUD printsiibi abil on võimalik uurida, kas funktsioonid toimivad rahuldavalt ja kas viitetervikluse ehk kokkusobivuse kontrollid (*consistency checks*) on täidetud.

1.4. Otsustustabeli tehnika (*Decision table test*)

Otsustustabeli tehnikaga keskendutakse süsteemi tingimuste testimisele. [2, lk 639] Tehnika rakendamisel kasutatavad testjuhud tuletatakse testbaasist paika pandud reeglite järgi.

Otsustustabeli tehnika sõltub kahest parameetrist: testimise ja detailsuse mõõtmisest. Testimise mõõtmine tähistab, kui suure ulatusega testitakse sõltuvusi erinevate otsustuspunktide vahel. Detailsuse mõõtmine näitab, kui põhjalikult testitakse otsustuspunktide sees olevaid sõltuvusi. Parameetrite valik määrab ära testjuhtude arvu ning testimisega saavutatava katvuse ulatuse.

Otsustustabeli tehnika rakendamisega keskendutakse funktsionaalsuse kvaliteediomadustele, nagu näiteks töötluse terviklusele ja õigsusele. [1, lk 223]

Testbaas

Rakendades otsustustabeli tehnikat peab testbaas sisaldama tingimus- või otsustustabeleid. Samas ei oma testbaasi tüüp ja struktuur suurt tähtsust. [2, lk 639]

Kasutus

Otsustustabeli tehnikat kasutatakse funktsionaalsuse põhjalikuks testimiseks. Tehnikat sobib rakendada väga tähtsate funktsioonide ja/või komplekssete arvutuste testimisel. [2, lk 639]

Katvus

Otsustustabeli tehnika kasutamisel on võimalik saavutada **tingimuse mitmekordne katvus** (*multiple condition coverage*). See tähendab, et kõik otsustuspunktide tingimuste tulemuste kombinatsioonid testitakse läbi vähemalt ühe korra.

Lisaks on võimalik rakendada **piirväärtuste katvust** suurema katvuse saavutamiseks ja **otsustuspunktide katvust** (*decision points coverage*) väiksema katvuse saavutamiseks. Otsustuspunktide katvus saavutatakse kui rakendada tingimuse (*condition coverage*), otsuse (*decision coverage*) või tingimuse/otsuse katvust (*condition/decision coverage*). [2, lk 639] Viimati mainitud katvused on lahti seletatud järgnevalt:

Tingimuse katvus - kõikide tingimuste tulemusi testitakse vähemalt ühe korra.

Otsuse katvuse - kõiki otsuste tulemusi testitakse vähemalt ühe korra.

Tingimuse ja otsuse katvuse - kõiki tingimuste ja otsuste tulemusi testitakse vähemalt ühe korra. [2, lk 605-607]

Näide

Oletame, et klient soovib autokindlustust. Sel juhul kontrollib süsteem, kas kliendi poolt sooritatavate õnnetuste arv on suurem kui A. Kui jah, siis kliendi autole kindlustust ei väljastata. Kui ei, siis kontrollib süsteem järgnevalt, kas auto tüüp on Opel või on auto vanus suurem kui B. Kui vähemalt üks nendest variantidest on tõene, siis kindlustus väljastatakse, kontrollitakse hinda ja määratakse erikoefitsent. Kui mõlemad nendest variantidest on valed, siis väljastatakse kindlustus, kontrollitakse hinda ning määratakse standardkoefitsent.

Pseudo-kood:

IF õnnetuste arv > A

THEN kindlustust ei väljastata

ELSE

IF auto tüüp = „Opel" OR auto vanus > B

THEN väljasta kindlustus, kontrolli hinda ning vali erikoefitsent

ELSE väljasta kindlustus, kontrolli hinda ning vali standardkoefitsent

Testimistulpade kirjeldamine

Testimistulpade kirjeldamine sisaldab endas kahte sammu:

- Sündmuste valimine

  Esimene alampunkt kirjeldab sündmusi. Protsessi võib alustada kasutaja, väline süsteem või sisemine funktsioon. Siinses näites on ainult üks sündmus: klient soovib kindlustust.

- Otsustustabeli loomine

  Teises alampunktis luuakse determinantide põhjal otsustustabel. Determinant on omadus, mis mõjutab protsessi töötlust. Iga

otsustustabeli tulp moodustab loogilise testimistulba, millest moodustatakse valim. Topeltjoonest üleval pool olev osa moodustab sündmuse ja situatsioonikirjelduse, samas kui topeltjoonest all pool on ära määratud tagajärjed ja/või tulemused.

Testimistulbad	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Õnnetuste arv	+	+	+	+	-	-	-	-
Auto tüüp	+	+	-	-	+	+	-	-
Auto vanus	+	-	+	-	+	-	+	-
Kindlustust ei väljastata	X	X	X	X				
Kindlustus väljastatakse, kontrollitakse hinda ning valitakse erikoefitsent								X
Kindlustus väljastatakse, kontrollitakse hinda ning valitakse standardkoefitsent					X	X	X	

Tabel 4. Otsustustabeli tehnika rakendamiseks vajalike testimistulpade kirjeldus.

Märkus: "+" tähistab pseudo-koodis oleva tingimuse kehtivust testitava tingimuse suhtes.

"-" tähistab pseudo-koodis oleva tingimuse mittekehtivust testitava tingimuse suhtes.

"X" tähistab tagajärgi erinevate tingimuste väärtuste korral.

Üleval pool topeltjoont on välja kirjutatud kõik determinantide võimalikud kombinatsioonid.

Otsustustabelid sisaldavad loogilisi testimistulpasid. Iga loogiline testimistulp tähistab, milline peaks olema tulemus tulpades kirjeldatud determinantide väärtustega. Testimise ajal tuleks otsustada, kas sellised tulemused ka tegelikult kirjeldatud olukordades eksisteeriksid.

Testjuhtude kirjeldamine

Testjuhud on valim testimistulpadest. Iga tagajärje või tulemuse kohta moodustatakse vähemalt üks testjuht.

	Testjuht 1	**Testjuht 2**	**Testjuht 3**
Õnnetuste arv	>A	<A	<A
Auto tüüp	(ei ole oluline)	Opel	Audi
Auto vanus	(ei ole oluline)	>B	<B
Oodatud tulemus	Kindlustust ei väljastata	Kindlustus väljastatakse, kontrollitakse hinda ning valitakse erikoefitsent	Kindlustus väljastatakse, kontrollitakse hinda ning valitakse standardkoefitsent
Tegelik tulemus			

Tabel 5. Otsustustabeli tehnika rakendamiseks vajalike testimisjuhtude kirjeldus

1.5. Elementaarvõrdlustehnika (*Elementary comparison test*)

Elementaarvõrdlustehnikat kasutatakse funktsionaalsuse põhjalikuks testimiseks. Läbi testitakse kõik programmi funktsionaalsed teed *(functional paths)*. Selleks uuritakse kõiki pseudo-koodis kirjeldatud tingimusi ning nende põhjal koostatakse testjuhud. [2, lk 654]

Testbaas

Elementaarvõrdlustehnika testbaasiks on pseudo-kood või samaväärne dokumentatsioon. Testbaas peab sisaldama üksikasjalikku ja struktureeritud ülevaadet otsustuspunktidest ja funktsionaalsetest teedest. [2, lk 654]

Kasutus

Elementaarvõrdlustehnikat kasutatakse kriitilise funktsionaalsuse testimiseks, kuna see on väga töömahukas ja aega nõudev. [1, lk 237]

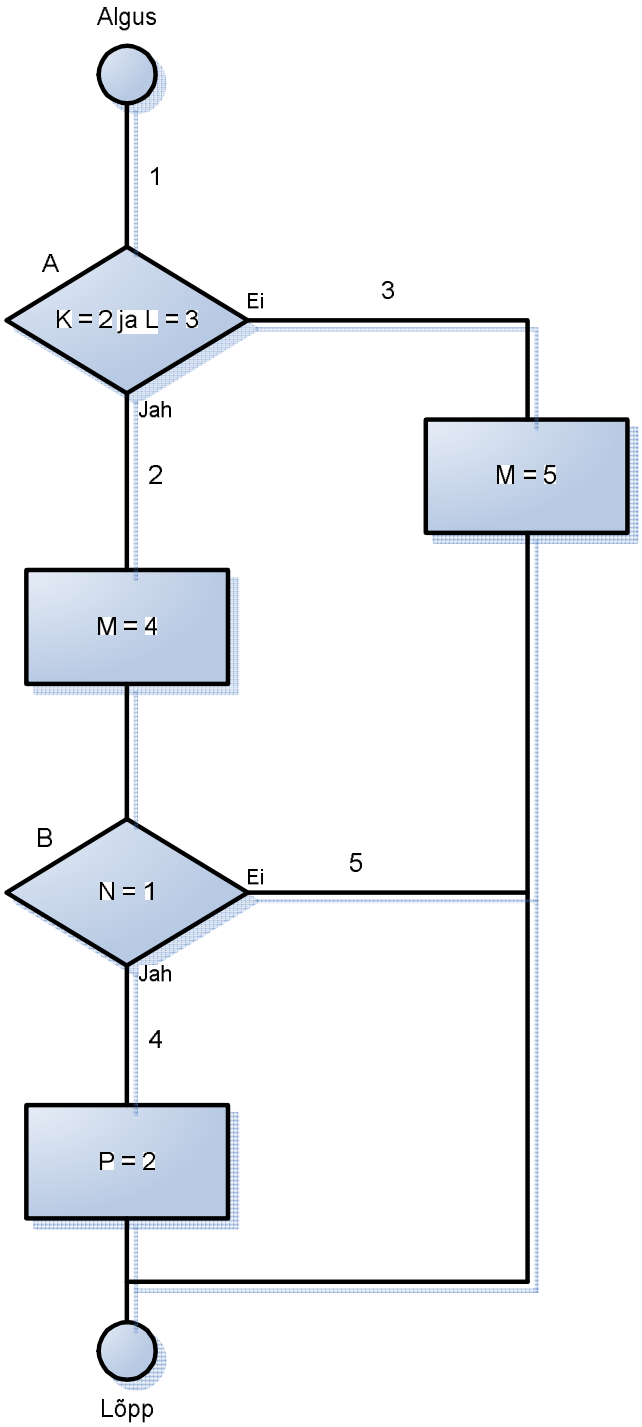
Katvus

Kasutades elementaarvõrdlustehnikat, on võimalik saavutada suhteliselt hea **otsustuspunktide katvus**. Lisaks võib kasutada **piirväärtuse analüüsi** või **paarikaupa testimist**. [2, lk 654]

Paarikaupa testimine (*pairwise testing*) – kahe faktori kõikide võimalike kombinatsioonide testimine. Elementaarvõrdlustehnika kasutamise raames testitakse funktsionaalsete teede võimalikke kombinatsioone. [2, lk 613]

Näide, tuginedes pseudo-koodile

```
IF (A == 2) AND (B == 3)
THEN C = 4
        IF (D == 1)
                THEN E = 2
        END-IF
        ELSE C = 5
END-IF
```



Joonis 2. Ülevaade elementaarvõrdlustehnika rakendamise näite testbaasi struktuurist skeemina

Elementaarvõrdlustehnika rakendamine sisaldab järgnevaid tegevusi:

Funktsiooni kirjelduse analüüsimine

Analüüsides funktsiooni kirjeldust, tuleks välja selgitada otsustuspunktid ja nende sisu. Selleks võib kasutada ka pseudo-koodi, mille analüüsimise käigus valitakse testitavad tingimused. Tingimused on tihti ära tuntavad terminite 'IF', 'DO' ja 'REPEAT' abil ning nad märgistatakse unikaalse identifikaatoriga (siin olevas näites on tingimuste identifikaatoriteks C1 ja C2).

```
IF (A == 1)                                    Tingimus 1 (C1)
THEN
        IF (B == 2) AND (C == 3)               Tingimus 2 (C2)
                THEN D = 5
        END-IF
        ELSE D = 6
ELSE B = 3 AND C = 4 AND D = 6
END-IF
```

Testolukordade loomine

Pärast tingimuste välja selgitamist määratakse iga tingimuse kohta testolukord.

Tingimusi on kahte tüüpi:

- Lihtne, harunemata tingimus – sisaldab ühte elementvõrdlust (*elementary comparision*). Näiteks on see tingimus **C1.**
- Keerukas tingimus – mitmekordne võrdlus, seotud 'AND' või 'OR' seostega. Näites tingimus **C2.**

Testitav tingimus	**C1.1**	**C1.2**
C1 = C1a	Tõene (1)	Väär (0)
C1a	A==1	A<>1

Tabel 6. Elementaarvõrdlustehnika rakendamiseks vajalike testimisolukordade kirjeldus

Testitav tingimus	**C2.1**	**C2.2**	**C2.3**
C2 = (C2b & C2c)	1(11)	0(10)	0(01)
C2b	B==2	B==2	B<>2
C2c	C==3	C<>3	C==3

Tabel 7. Elementaarvõrdlustehnika rakendamiseks vajalike testimisolukordade kirjeldus

Loogiliste testjuhtude kindlaksmääramine

Loogiliste testjuhtude kindlaks määramiseks leitakse funktsionaalsed teed, kus läbides testitavaid tingimusi läbides peaks jõudma vähemalt ühe korra lõppu (punkti Lõpp). Selle dokumenteerimiseks võib kasutada maatriksit, kus esimeses tulbas on kirjeldatud kõik testitavad tingimused, teises tulbas on välja toodud tingimuste sisendid ehk algsed väärtused ning kolmandas tulbas on loetletud väljundid ehk tingimuste väärtused, kuhu esimesest tingimusest edasi liigutakse.

Järgmistes tulpades on loetletud testjuhud. Testjuhud on koostatud õigesti, kui iga testitavat tingimust kasutatakse testimise käigus vähemalt korra.

Testitav tingimus	**Sisend**	**Väljund**	**1**	**2**	**3**	**4**
C1.1	1	C2	X	X	X	
C1.2	0	Lõpp				X
C2.1	1	Lõpp	X			
C2.2	0	Lõpp		X		
C2.3	0	Lõpp			X	

Tabel 8. Elementaarvõrdlustehnika rakendamiseks vajalike loogiliste testjuhtude kirjeldus

Füüsiliste testjuhtude kindlaksmääramine

Selles punktis teisendatakse loogilised testjuhud füüsilisteks testjuhtudeks.

Esimeses tulbas on loetletud testjuhud. Teises tulbas on kirjeldatud tee, mis iga testjuhu käigus läbitakse. Järgmistes tulpades on ära märgitud tingimuste elementide väärtused, millega testjuhud sooritatakse.

Testjuhud	**Tee**	**A**	**B**	**C**
1	C1.1, C2.1	1	2	3
2	C1.1, C2.2	1	2	8
3	C1.1, C2.2	1	7	3
4	C1.2	9	-	-

Tabel 9. Elementaarvõrdlustehnika rakendamiseks vajalike füüsiliste testjuhtude kirjeldus

Oodatavate tulemuste kindlaksmääramine

Oodatavate tulemuste kindlaks määramine on oluline, kuna siis on võimalik otsustada, kas testimistulemused on aktsepteeritavad või mitte.

Testjuhud	**Oodatav tulemus**
1	D = 5
2	D = 6
3	D = 6
4	B = 3, C = 4, D = 6

Tabel 10. Elementaarvõrdlustehnika rakendamisel oodatava tulemuse kirjeldus

Algandmete kindlaksmääramine

Testimise käigus tuleks otsustada, kas algandmete kirjeldamine on vajalik või mitte. Kui see on vajalik, siis tuleks kirjeldada kõik algandmed ning testjuhud, milles andmeid käsitletakse.

Testjuhud	**Kasutatavad algandmed**
1-4	Tabel K
1	Tabel L
2,3,4	Tabel M

Tabel 11. Elementaarvõrdlustehnika rakendamisel kasutatavate algandmete kirjeldus

Teststsenaariumi koostamine

Testjuhud koosnevad sooritamise järjekorras kirjeldatud testimistegevustest ja oodatavatest tulemustest. Nende testjuhtude põhjal koostatakse teststsenaarium, mis loetleb kõik testimiseks vajalikud eel- ning järeltingimused.

Näiteks võib teststsenaariumi X sooritamise vajalik eeltingimus olla, et süsteemikuupäev peab olema 25.november 2007 ning järeltingimus, et teststsenaariumi X täitmisele peab järgnema teststsenaariumi Y sooritus.

1.6. Vigade oletamise tehnika (*Error guessing test*)

Vigade oletamise tehnika põhineb mittestruktuursel testimisel. See tähendab, et muuhulgas testitakse olukordi, millele muidu tavaliselt rõhku ei pööratakse. Testimine põhineb testija kogemusel. Testija koostab testjuhud jooksvalt testimise käigus tuginedes enda intuitsioonile ning keskendudes tingimustele, mis tavaliselt vigu põhjustavad. Vigade oletamise tehnika kasutamise eeldus on, et testija tunneb hästi testitavat süsteemi. [1, lk 246]

Testbaas

Vigade oletamise tehnika rakendamisel ei ole põhjalik testbaasi olemasolu vajalik. Testija peab tundma testitavat süsteemi ning lisaks teadma eriolukordi, mida süsteem peab käsitlema korrektselt, kuid kust võib leida vigu. [1, lk 246]

Kasutus

Vigade oletamise tehnikat kasutatakse kõrg-tasemelise testimise osana. Seda tehnikat võib kasutada proffessionaalne testija, kellel on kogemusi ebaharilikest kohtadest vigade otsimisega, kui ka kasutaja, kes oskab aimata ebatavalisi olukordi ning soovib kontrollida, kas süsteem käsitleb neid õigesti.

Vigade oletamise tehnikat kasutatakse testimisprotsessi lõpus, kui enamus lihtsamaid vigu on juba struktuursete testimistehnikatega leitud. Siis on vigade oletamise tehnikaga võimalik keskenduda keerukate situatsioonide testimisele ja leida vigu, mida muidu võib-olla ei leitaks.

Lisaks võib vigade oletamise tehnikat kasutada ka testijas kindlustunde tekitamiseks, nagu näiteks “kui süsteem käsitleb need olukorrad rahuldavalt, siis on ka ülejäänud funktsionaalsusega suure tõenäosusega korras”. [2, lk 661-662]

Katvus

Vigade oletamise tehnikat üksi kasutades mõistlikku ega kindlat katvust ei saavutata, kuna keskendutakse vaid süsteemi teatud osadele ja kogu süsteemi üldiselt ei testita. Samas kasutades vigade oletamise tehnikat lisana mõnele struktuursele tehnikale, võib saavutada süsteemi väga hea töökindluse. [1, lk 246]

Kuna vigade oletamise tehnika rakendamiseks ei ole konkreetseid piiranguid või täpseid reegleid, ei ole käesolevale tehnikale näidet juurde lisatud.

1.7. Uuriv testimine (*Exploratory testing*)

Uuriv testimine on üheaegselt testimise õppimine, planeerimine ja sooritamine. See tähendab, et testija uurib testimise käigus mingit süsteemiosa samal ajal mõeldes, mida saaks ja võiks veel testida ning seejärel seda ka teeb. Nii kogub testija süsteemi kohta järjest uut informatsiooni ning tegevuste tsükkel hakkab korduma. Uuriva testimise käigus toimuvad testimise planeerimine ja sooritamine väga lähestikku. Testija kasutab kõige sobivamat baastehnikat, sõltuvalt testitavatest omadustest ja nende kohta saadavast informatsioonist. [2, lk 664-665]

Testbaas

Tihti arvatakse, et uurivat testimist kasutatakse ainult siis, kui testbaas on puudulik (puuduvad näiteks funktsionaalnõuded). Siiski ei vasta see alati tõele, kuna uuriv testimine on rakendatav ka hästi kirjeldatud süsteemide puhul. Sel juhul pannakse vähem rõhku juba kirjeldatud testbaasile ja rohkem moodustele, kuidas määrata testitava objekti vastavust testbaasile. [2, lk 664-665]

Kasutus

Kuna uurimuslik testimine on küllaltki reeglite vaba, on seda tehnikat võimalik kasutada väga laialdaselt. Uurimuslikku testimist on võimalik rakendada igasugust tüüpi testbaasidega kõikide kvaliteediomaduste testimiseks. Siiski on mõned olukorrad, millal selle tehnika kasutamine on parem ja halvem mõte.

Näiteks on uurimuslikku testimist otstarbekas rakendada, kui testijad on kogemustega, usaldusväärsed ning varasemate teadmistega testitavast valdkonnast. Vastasel juhul on näiteks suurem võimalus, et testide kvaliteet on ebapiisav. Uurimuslikku testimist on muuhulgas mõttekas kasutada ka siis, kui testbaas on puudulikult dokumenteeritud, testjuhtude koostamiseks ei ole piisavalt aega, oluline on testida võimalikult odavalt või rakendamiseks koos formaalsemate tehnikatega, et julgustada loovat testimist.

Uurimuslikku testimist ei ole mõtekas rakendada, kui testitakse kriitilist funktsionaalsust, kus leidmata viga võib põhjustada tõsiseid probleeme või kui testijal ei ole piisavalt kogemusi, kuna kindlapiirilise testimistehnika abita on testjuhtude koostamine suhteliselt keeruline.

Lisaks võimaldab uurimuslik tehnika saada süsteemist ja selle toimimisest kiire ülevaade. [2, lk 664-665]

Katvus

Kuna uurimuslik tehnika ei põhine ühelgi reeglil ning testija otsustab ise, kuidas ja mida testida, siis ei ole võimalik anda ka mingit katvuse garantiid, mis käesoleva tehnika abil oleks võimalik saavutada. [2, lk 664-665]

Kuna uuriva tehnika rakendamiseks ei ole mingeid piiranguid või täpseid reegleid, ei ole käesolevale tehnikale näidet juurde lisatud.

1.8. Protsessitsükli testimistehnika (*Process cycle test*)

Protsessitsükli testimistehnika eesmärk on kontrollida, kas süsteemi automatiseeritud osad sobivad administratiivse organisatsiooni protseduuride hulka. See tähendab, et kas vanale süsteemile juurde integreeritud osad töötavad koos vanade osadega ning samuti kas uued süsteemiosad töötavad omavahel. Eriti tuleks tähelepanu pöörata automatiseeritud ja manuaalsete toimingute vahelistele liidestele. Rakendades protsessitsükli tehnikat, eeldatakse, et automatiseeritud toimingud vastavad nõuetele. Seega keskendutakse testimise käigus hoopis automatiseeritud ja manuaalsete toimingute teineteisele vastavust. [1, lk 247]

Testbaas

Protsessitsükli testimiseks tuletatakse testjuhud protseduurivoo struktuurist mitte tehnilisest lahendusest. Testimise alus peaks sisaldama struktureeritud informatsiooni vajaliku süsteemi käitumisest teede ja otsustuskohtade vormis. [2, lk 675]

Kasutus

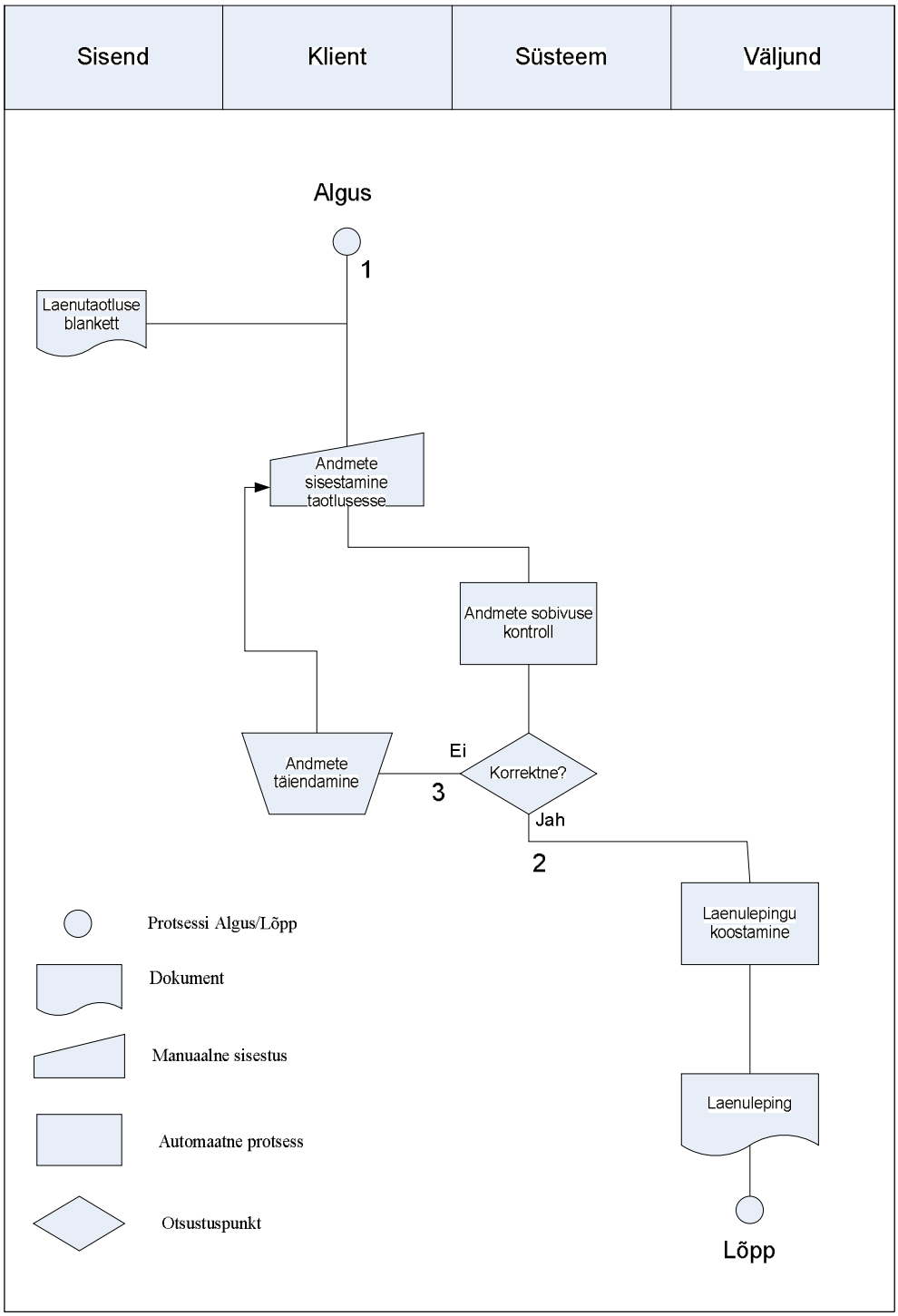
Protsessitsükli tehnikat kasutatakse peamiselt sobivuse kindlaks määramisel, lisaks on võimalik kasutada seda tehnikat ka kasutaja-sõbralikkuse ja turvalisuse testimisel. [1, lk 247]

Katvus

Kuna protsessitsükli testimistehnika rakendamise käigus kasutatakse ka natuke algoritmi testimistehnikat, siis on võimalik saavutada sama katvus, mis algoritmitehnika puhul – otsustuspunktide katvus või teede katvus. [2, lk 675]

Näide

Oletame, et klient soovib läbi süsteemi laenu taodelda. Selle jaoks täidab klient süsteemis oleva laenutaotluse blanketi. Süsteem kontrollib sisestatud andmed ning kui andmed on korreksed, koostab süsteem kliendile laenulepingu. Kui andmed ei ole korrektsed, saadab süsteem laenutaotluse kliendile parandamiseks tagasi, misjärel süsteem kontrollib uuesti, kas sisestatud andmed on korrektsed.



Joonis 3. Ülevaade protsessitsükli testimistehnika rakendamise näite testbaasi struktuurist skeemina

Kasutades testimiseks protsessitsükli tehnikat märgistatakse joonises lisaks algusele ja lõpule ka otsustuspunktid ning teed.

A: 1.2 1.3
3.2 3.3

Joonise abil koostatakse testjuhud algusest lõpuni nii, et iga tee läbitakse vähemalt ühe korra.

Testjuht 1: 1.2
Testjuht 2: 1.3.3.2

Testjuht	**Samm**	**Kirjeldus**	**Sooritaja**	**Tulemus**
1	1	Täida laenutaotlus	Klient	
	2	Kontrolli laenutaotlusse sisestatud andmete sobivust	Süsteem	
	3	Koosta laenuleping	Süsteem	
2	1	Täida laenutaotlus	Klient	
	2	Kontrolli laenutaotlusse sisestatud andmete sobivust	Süsteem	
	3	Koosta laenuleping	Süsteem	

Tabel 12. Protsessitsükli testimistehnika rakendamiseks vajalike testjuhtude kirjeldus

1.9. Programmi liidese testimistehnika (*Program interface test*)

Programmi liidese testimistehnikat kasutatakse programmide ja/või moodulite vaheliste liideste testimiseks. Kõigepealt testitakse programme eraldi ning kui nad töötavad õigesti, programmid integreeritakse ning järgnevalt on programmi liidese testimistehnika abil võimalik kontrollida, kas programmid töötavad endiselt korrektselt. [1, lk 255]

Testbaas

Testimiseks on vajalik informatsioon programmide vahelise ühenduste kohta. Näiteks peaks dokumentatsioon sisaldama infot, millise tegevuse tulemusena ja millisest programmist vajalikke andmeid laetakse.

Kasutus

Programmi liidese testimistehnikat kasutatakse peamiselt integratsioonitestide puhul erinevate programmide ja/või moodulite vaheliste liideste testimiseks. [1, lk 255]

Katvus

Programmi liidese testimistehnikaga saavutatav katvus sõltub programmide ja/või moodulite vaheliste liideste testimise hulgast. See tähendab, et katvus määratakse testitud liideste hulga ja kõikide võimalike liideste hulga suhtega.

Näide

Oletame, et testitakse valuutavahetuse programmi, kasutades programmi liidese testimistehnikat.

Teststsenaarium programmi testimiseks

Kasutaja sisestab väljadesse “Algvaluuta” ja “Lõppvaluuta” sobivad valuutad. Seejärel kutsub programm välja protseduuri “Kurss”, mis väljastab algvaluuta kursi lõppvaluuta suhtes. Kasutaja sisestab seejärel välja “Summa” vahetatava rahasumma ning vajutab “Vaheta”.

Kuvatakse uus leht, kus kuvatakse kasutaja poolt valitud valuutad ning kutsutakse uuesti välja protseduur “Kurss”, mis väljastab algvaluuta kursi lõppvaluuta suhtes. Kasutaja sisestab uuesti kinnituseks vahetatava rahasumma ning vajutab “Vaheta”.

Kirjeldatud süsteemis on kaks liidest, mida tuleks testida:
1. liides – programm kutsub protseduuri “Kurss” välja esimest korda, kui kasutaja on valinud alg- ning lõppvaluuta. Selle liidese testimiseks on testjuhud 1-3.
2. liides – programm kutsub protseduuri “Kurss” välja teist korda peale seda, kui kasutaja on valinud alg- ning lõppvaluuta, sisestanud summa, vajutanud “OK” ning programm kuvab järgmise lehe, kus väljastatakse uuesti algvaluuta kurss lõppvaluuta suhtes. Selle liidese testimiseks on testjuht 4.

	Testjuht 1	**Testjuht 2**	**Testjuht 3**	**Testjuht 4**
Testitav liides	1	1	1	2
Eeltingimused	-	Algvaluuta: EUR Lõppvaluuta: EEK	Algvaluuta: EUR Lõppvaluuta: EEK	Algvaluuta: EUR Lõppvaluuta: EEK
Tegevused	Algvaluuta: EUR Lõppvaluuta: EEK	Algvaluuta: USD Lõppvaluuta: EEK	Algvaluuta: EUR Lõppvaluuta: USD	Kasutaja vajutab “Vaheta”
Oodatav tulemus	Protseduur kuvab õige kursi	Protseduur kuvab õige kursi	Protseduur kuvab õige kursi	Protseduur kuvab õige kursi
Tulemus				

Tabel 13. Programmi liidese testimistehnika rakendamiseks vajalike testjuhtude kirjeldus

1.10. Reaaleluline testimine (*Real-life test*)

Reaalelulise testimise eesmärk on välja selgitada, kas süsteem suudab toime tulla selle realistliku kasutamisega. Selleks testitakse süsteemi nii, nagu seda reaalselt tulevikus kasutatama hakatakse. Süsteem ja selle testimine peavad sarnanema võimalikult palju selle kasutamisele toodangu keskkonnas. Testimise käigus püütakse leida mittefunktsionaalseid vigu, mis on seotud süsteemi lõplike suurustega, nagu näiteks kasutajate hulk, töötluses olevad tegevused ja süsteemi koormuse tase. Reaaleluline testimine aitab ennetada olukordi, kus süsteem toodangu keskkonda jõudes ei tööta.
Lisaks saab reaalelulise tehnika abil testida süsteemi seotust keskkonnaga ehk keskkonnas olevate erinevate liideste tegelikku töötamist. Reaalelulise tehnikaga on võimalik keskenduda süsteemi kvaliteediomadustele, nagu näiteks jõudlus, infrastruktuuriline sobivus või järjepidevus.
Kuna testimine sooritatakse toodangu keskkonnale võimalikult sarnastes tingimustes, aitab reaaleluline testimine ennetada olukorda, kus süsteem toodangu keskkonnas ei tööta. Samas ei pruugi reaalelulist tehnikat rakendavad testjuhud olla alati korratavad. Seetõttu võiks võimalusel uurida vigade põhjusi süsteemilogidest. [1, lk 259-682] [2, lk 681-682]

Testbaas

Reaalelulise tehnika kasutamise testbaasiks on testitava süsteemi tegeliku kasutuse ja koormuse kirjeldus. See tähendab, et kirjeldatud peavad olema kasutajate hulk ning sooritatavate tehingute järjestus ja hulk. [2, lk 681]

Kasutus

Reaalelulise tehnika kasutamine on tavaliselt keerulisem, kui on teiste tehnikate kasutamine, kuna komplekssete süsteemide puhul on kasutaja tegevuste juhuslikkust raske jäljendada. Tehnikat kasutatakse siis, kui süsteemi funktsionaalsus on juba testitud ning sooritatakse viimaseid teste otsustamaks, kas süsteem töötab korrektselt. [2, lk 682]

Katvus

Rakendades reaalelulist testimist, on võimalik saavutada süsteemikasutuse statistililise vastavuse katvus. See tähendab, et katvuse saavutamiseks jäljendatakse süsteemi kasutust, mitte ei testita süsteemi käitumist mingites olukordades. Katvuse saavutamiseks tuleks kirjeldada ning seejärel testida tingimused, kuidas erinevad kasutajatüübid süsteemi kasutavad ja kuidas süsteem käitub erineva koormuse puhul. [2, lk 681]

Järgnevalt on kirjeldatud reaalelulise testimise jaoks vajalikke tegevusi.

Reaalne süsteemikasutuse kirjeldus

Kuna reaalelulise tehnika eesmärk on süsteemi lõplikust versioonist vigade leidmine, peaksid testimise käigus sooritatavad tegevused olema võimalikult sarnased süsteemi tegelikule kasutusele. Seetõttu tuleks testitava süsteemi nõuetedokumentide uurimise käigus koostada kirjeldus, mis sisaldaks süsteemi aktiivsete kasutajate poolt sooritatavate tegevuste kirjeldusi. Mida põhjalikum ja detailsem on kirjeldus, seda parem pilt tegelikust süsteemikasutusest saadakse. [1, lk 260]

Testjuhtude kirjeldamine

Järgmine samm on detailselt kirjeldada testitavad süsteemiosad ning testjuhud. Ka siin tuleb tegelikku süsteemikasutust võimalikult täpselt jäljendada. Testitavate süsteemiosade täpsemaks uurimiseks tuleks välja selgitada nendega seotud kasutajarühmad ning tegevused, mida kasutajad antud süsteemiosaga seotult sooritavad. Seejärel tuleks kindlaks määrata tegevuste sagedus ja suhteline sagedus (konkreetse tegevuse arv jagatud kõikide tegevuste arvuga) ühes ajaühikus. [1, lk 261-262]

Testimise sooritamine

Testimine, rakendades reaalelulist tehnikat, on võrreldes teiste testimistehnikate rakendamisega tihti keerulisem. Näiteks, kui testitava süsteemi lõppkasutajate hulk on väga suur, peab süsteemikasutust erinevate vahenditega simuleerima. Kui aga lõppkasutajate hulk on väike, võib eelnevalt koostatud testjuhtude sooritamiseks kasutajatelt abi paluda. [1, lk 262]

1.11. Semantiline testimine (*Semantic test*)

Semantilise testimise rakendamise käigus kontrollitakse objektide vahelisi seoseid. Seosed võivad olla kõikvõimalike ekraanipõhiste andmete vahel, erinevate ekraanipiltide vahel või sisendandmete ja juba andmebaasis olevate andmete vahel.
Semantilise testimistehnika eesmärk on leida ühtivuses vigu ning seda sobib rakendada just andmeväljade ja nende vaheliste seoste ning kvaliteedi sobivuse testimiseks. [1, lk 263]

Testbaas

Semantilise testimistehnika rakendamiseks vajalik informatsioon on tavaliselt kirjeldatud funktsionaalnõuetes või ärireeglites. Testbaas sisaldab reegleid ja seoseid, millele andmed peavad vastama, et süsteem aksepteeriks nad korrektsete sisendandmena. [2, lk 687]

Kasutus

Semantilist testimist kasutakse peamiselt interaktiivsete süsteemide testimisel.
Semantilist testimist saab kasutada paralleelselt süntaktilise testimisega ning nende teststsenaariume saab tavaliselt kombineerida. Mõlemad tehnikad kuuluvad õigsuse kontrollimise testide hulka. See tähendab, et neid kasutatakse sisendandmete õigsuse ja nende vaheliste seoste testimiseks. [1, lk 263]

Katvus

Kuna semantilisi reegleid võib kirjeldada ka kui kombineeritud tingimusi sisaldavaid otsustuspunkte, siis võikski üheks testimise katvuse tüübiks valida **muudetud tingimuse/otsustuse katvuse**. Lisaks on võimalik saavutada veel kergema variandina **tingimuse/otsustuse katvus** ning **tingimuse mitmekordne katvus** põhjalikuma katvuse saavutamiseks.
Muudetud tingimuse/otsustuse katvus (*modified condition/decision coverage*) – iga tingimuse tulem määrab otsustuse tulemi vähemalt ühe korra. [2, lk 687]

Näide

Andmeväljade seoste testimine

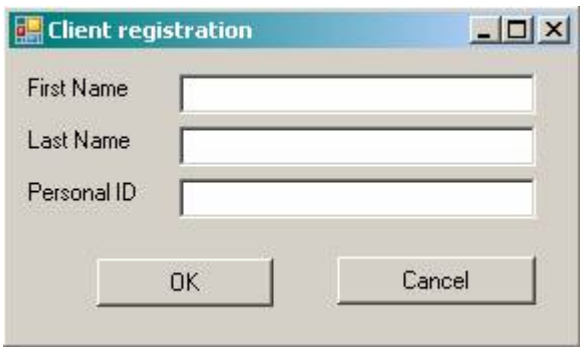
Oletame, et testida tuleb rakendusse sisselogimise akent. Sisselogimine õnnestub, kui kõik väljad on õigesti täidetud.

Sobivad väljade väärtused:

Eesnimi (*First Name*) – Juku

Perekonnanimi (*Last Name*) – Juurikas

Isikukood (*Personal ID*) - 123456



Joonis 4. Pilt semantilise testimistehnika kasutamise näites kirjeldatud rakenduse sisselogimisaknast

Andmeseoste kindlaks tegemine

Väljadel on omavahel järgnevad seosed:

First Name **AND** Last Name **AND** Personal ID

Pseudokood:

```
IF First Name
THEN
        IF Last Name
        THEN
                IF Personal ID
                THEN correct entry              (testjuhtum 1)
                ELSE error message              (testjuhtum 2)
        ELSE error message                      (testjuhtum 3)
ELSE error message                              (testjuhtum 4)
```

Teststsenaariumi koostamine

	Kirjeldus	**TJ 1**	**TJ 2**	**TJ 3**	**TJ 4**
Muutujad	Eesnimi	Juku	Juku	Juku	Mari
	Perekonnanimi	Juurikas	Juurikas	Maasikas	Juurikas
	Isikukood	123456	654321	654321	654321
Oodatud Käitumine	Veateade ilmub		X	X	X
	Veateade ei Ilmu	X			
Tulemus					

Tabel 14. Semantilise testimistehnika rakendamiseks vajaliku teststsenaariumi kirjeldus

1.12. Süntaktiline testimine (*Syntactic test*)

Lisaks semantilisele testimisele on kasutajaliidest võimalik testida ka süntaktilise testimise abil. Selle testimise eesmärgiks on vigade leidmine ekraanipiltidelt, aruannete vormingutest ning nendega seotud sisendikontrollidest. Sisendikontrollid kontrollivad vormingutesse ja aruannetesse sisestatud andmete õigsust ning nad ei tohiks vigaste andmete puhul segadusse minna. [2, lk 690]

Testbaas

Süntaktilise testimise testbaasiks võivad olla funktsionaalnõuded, kõiki andmetekirjeldusi sisaldavad dokumendid või stiilijuhendid, mis sisaldavad reegleid organisatsioonile, nagu näiteks värvi, kirjatüübi jne kasutus.
Lisaks peab testbaas sisaldama süntaktilisi reegleid, mis kirjeldavad, millistele omadustele peaksid andmed vastama, et nad aktsepteeritaks mingi süsteemi sisendina/väljundina. [2, lk 690-691]

Kasutus

Süntaktilist testimist, nagu ka semantilist testimist, kasutatakse sisend- ning väljundandmete õigsuse kontrollimiseks. Seda võib teha, testides nii interaktiivseid süsteeme kui ka aruandeid. [2, lk 690-691]

Katvus

Süntaktilised reeglid testitakse suvalises järjekorras üksteistest eraldi. Siin rakendatav katvuse tüüp on kontrollnimekiri (*checklist*), mille käigus iga omaduse või pildi juures rakendatavad kontrollid kirjeldatakse ja testitakse. [2, lk 690]

Näide

<u>Kontrollnimekirja loomine</u>

Kontrollnimekiri sisaldab näiteks järgmisi küsimusi:

- Mis tüüpi elemente see sisaldab: numbrilisi, kirjamärke, kuupäevi?
- Mis valikud on võimalikud igal ekraanil ja/või elemendil: küsimärgi valik, funktsiooniklahvid, abifunktsioonid?

- Elementide kontroll (sisend ja väljund, väljatüübid, kohustuslikud väljad, aruannete elementid)
- Paigutuse kontroll (pealkirjad, elementide nimed ja paigutus)

Oletame, et testitakse kliendi registreerimislehte, mis sisaldab järgmiseid väljasid:

Väli	**Kohustuslik?**	**Määratlus**	**Pikkus**
Kood (genereeritakse)	Ei	Arvuline	6
Eesnimi	Jah	Tekst	20
Perekonnanimi	Jah	Tekst	20
Tänav	Ei	Tekst	20
Linn	Ei	Tekst	20
Telefoninumber	Jah	Arvuline	15

Tabel 15. Süntaktilise testimistehnika kasutamise näite testbaasi kirjeldus

Lisaks on võimalik registreerimislehel kasutada lisafunktsiooniklahve:

Lisafunktsiooniklahv	**Tulemus**
F1	Abi
F8	Salvesta
F10	Lõpeta

Tabel 16. Süntaktilise testimistehnika kasutamise näite testbaasi kirjeldus

<u>Kontrollnimekiri testimiseks:</u>

Jäta kõik väljad tühjaks

Sisesta numbrilisi ja mittenumbrilisi väärtusi

Sisesta liiga palju sümboleid

Sisesta lubatud väärtusi

Vajutades kirjeldatud lisafunktsiooniklahve kuvatakse lubatud tulemus

Vajutades teisi kirjeldamata lisafunktsiooniklahve ei kuvata mingeid tulemusi

Testjuhtude koostamine

Testjuhtum	**Kirjeldus**	Kontroll	Tulemus	Märkused
1.	Vajutades lisafunktsiooniklahvi F1 avaneb korrektne abiaken	Lubatud		
2.	Ära sisesta eesnime	Lubamatu		
3.	Sisesta eesnime välja liiga palju sümboleid (>20)	Lubamatu		
4.	Sisesta eesnime välja korrektne eesnimi	Lubatud		
5.	Sisesta eesnime välja numbrid	Lubamatu		
6.	Vajutades lisafunktsiooniklahvi F1 avaneb korrektne abiaken	Lubatud		
7.	Ära sisesta perekonnanime	Lubamatu		
8.	Sisesta perekonnanime välja liiga palju sümboleid (>20)	Lubamatu		
9.	Sisesta perekonnanime välja korrektne perekonnanimi	Lubatud		
10.	Sisesta perekonnanime välja numbrid	Lubamatu		
11.	Vajutades lisafunktsiooniklahvi F1 avaneb korrektne abiaken	Lubatud		
12.	Ära sisesta tänavat	Lubatud		
13.	Ära sisesta linna	Lubatud		
14.	Sisesta tänava välja liiga palju sümboleid (>20)	Lubamatu		
15.	Sisesta linna välja liiga palju sümboleid (>20)	Lubamatu		
16.	Sisesta tänava ja linna väljadesse korrektsed väärtused	Lubatud		
17.	Vajutades lisafunktsiooniklahvi F1 avaneb korrektne abiaken	Lubatud		
18.	Ära sisesta telefoninumbrit	Lubamatu		

19.	Sisesta telefoninumbri välja liiga palju sümboleid (>15)	Lubamatu		
20.	Sisesta telefoninumbri välja korrektne telefoninumber	Lubatud		
21.	Sisesta telefoninumbri välja sümboliteks tähed	Lubamatu		
22.	Vajutades lisafunktsiooniklahvi F1 avaneb korrektne abiaken	Lubatud		
23.	Vajuta lisafunktsiooniklahvi F4	Lubamatu		
24.	Vajutades lisafunktsiooniklahvi F8 salvestatakse aknas tehtud muudatused	Lubatud		
25.	Vajutades lisafunktsiooniklahvi F10 suletakse avatud aken	Lubatud		

Tabel 17. Süntaktilise testimistehnika rakendamiseks vajalike testjuhtude kirjeldus

1.13. Kasutuslugude testimistehnika (*Use Case Test*)

Kasutuslugude testimistehnikat rakendatakse kvaliteediomaduste testimisel.
Kasutuslugu kujutab tüüpilist kasutaja ja süsteemi vahelist suhtlust ning sisaldab süsteemi poolt kasutajale pakutava funktsionaalsuse kirjeldust.
Kuna kasutuslugusid on võimalik kirjeldada erinevatel viisidel, tuleks enne testima asumist ja kasutusloo tehnika rakendamist uurida, kas kirjeldus sisaldab piisavalt vajalikku informatsiooni. Lihtsaim moodus on seda teha kontrollnimekirja abil. [2, lk 696]

Testbaas

Vajalik testbaas peab sisaldama kasutuslugusid ning soovitavalt ka kasutuslugude seoste diagrammi. [2, lk 696]

Kasutus

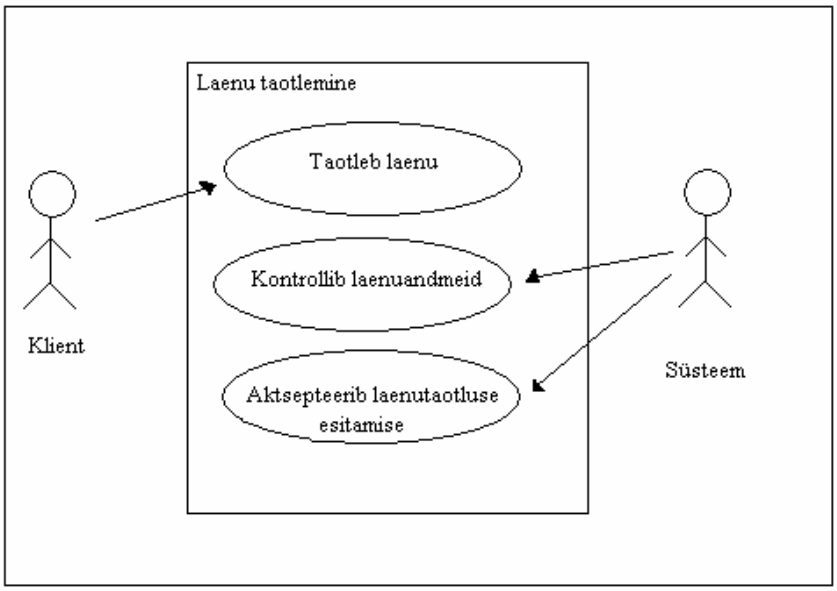
Kasutuslugude testimistehnikat rakendatakse testides kvaliteediomadusi, nagu näiteks efektiivsus ja kasutaja-sõbralikkus. [2, lk 696]

Katvus

Kasutuslugude tehnika rakendamisel keskendutakse kasutaja ja süsteemi vastastikude mõjude katvusele testidega. Katvus saavutatakse, kasutades kontrollnimekirja ning selle rakendamine on peaaegu alati võimalik. Lisaks on võimalik kasutada teede, otsustuspunktide või paarikaupa testimise katvusi, kuid nende miinuseks on, et nad on tugevalt sõltuvuses kasutuslugude kirjelduste sisust. [2, lk 696]

Näide

Oletame, et klient soovib taotleda laenu läbi veebikeskkonna. Selle kasutusloo kohta on järgnevalt välja toodud lihtsustatud skeem.



Joonis 5. Skeem laenu taotlemise kasutusloo kohta

Järgnevalt tuleks välja kirjutada põhi- ning lisastsenaariumid antud juhtumi kohta.

Põhistsenaarium **K: Klient** **S: Süsteem**	**Samm**	**Tegevus**
	1	K: Valib keskkonnas „Laenutaotlused”
	2	K: Valib laenu (näiteks väikelaen)
	3	S: Kuvab väikelaenu taotluse blanketi
	4	K: Täidab vajalikud väljad (laenusumma, tähtaeg)
	5	S: Kontrollib sisestatud väärtused
	6	S: Kuvab laenutaotluse valmis kujul
	7	K: Aktsepteerib laenutaotluse esitamise
	8	S: Lisab esitatud laenutaotluse veebikeskkonnas kliendi esitatud laenutaotluste nimekirja
Lisastsenaariumid	2a	Klient ei vasta laenusaajate tingimustele S: Kuvab teate ja väljub
	5a	Sisestatud väärtused ei ole korrektsed S: Kuvab teate ja väljub
	7a	Klient ei aktsepteeri laenutaotluse esitamist S: Kuvab teate ja väljub

Tabel 18. Kasutuslugude testimistehnika rakendamiseks vajalike stsenaariumite kirjeldus

Põhi- ning lisastsenaariumite põhjal on võimalik koostada testjuhud:

	TJ 1	**TJ 2**	**TJ 3**	**TJ 4**
Samm	1	1	1	1
	2	2	2	2
	3	2a	3	3
	4		4	4
	5		5	5
	6		5a	6
	7			7
	8			7a

Tabel 18. Kasutuslugude testimistehnika rakendamiseks vajalike testjuhtude kirjeldus

Selle juhtumi testimiseks on vajalik sooritada 4 testjuhtu. [3, lk 130-135]

Kokkuvõte

Testimistehnikate õige kasutamise abil on võimalik hõlpsasti ja vähema ajakuluga saavutada testitava süsteemi erinevate osade katvus vajalike testidega. Testimistehnikaid rakendades on võimalik leida erinevaid vigade tüüpe suurema tõenäosusega ja kiiremini, kui juhuslikke ning süstematiseerimata teste tehes. Oluline on ka näiteks, et testimistehnikate kasutamisel on testimisprotsess lihtsamini planeeritav ja kontrollitav, kuna testide kirjeldamise ja sooritamise protsess on konkreetselt paika pandud.

Käesoleva seminaritöö raames on autoril valminud ülevaade testimistehnikatest. Võimaluse korral on igale tehnikale juurde lisatud ka selle rakendamist tutvustav näide. Lisaks sobib antud töö abistava allikana testimise planeerimisel ja ettevalmistamisel, õppematerjalina kasutamiseks või huvi korral lugemiseks.

Kasutatud kirjandus

[1] Pol M., Teunissen R., Erik van Veenendaal
Software Testing, A Guide to the TMap Approach.
Biddles Ltd. 2002

[2] Koomen T., Leo van der Aalst, Broekman B., Vroon M.
Tmap Next for result-driven testing.
UTN Publishers. 2006

[3] Copeland L.
A Practitioner's Guide to Software Test Design.
Artech House Publishers. 2003