TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Johannes Olaf Kurss

Aspekt-orienteeritud testimise efektiivsusanalüüs

Bakalaureusetöö

Juhendaja: Jüri Vain
Doktorikraad

Tallinn 2021

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.

Autor: Johannes Olaf Kurss

18.05.2021

Annotatsioon

Bakalaureusetöö eesmärgiks on näidata eksperimentaalselt aspekt-orienteeritud mudelite eelis mitte aspekt-orienteeritud mudelite ees mudeli-põhiste testide genereerimisel. Vaatluse all olid erinevad aspekt-orienteeritud testikatte kriteeriumid ja testide automaatseks genereerimiseks kasutati mudelkontrolli tehnikat. Eksperimentides näidati, et testide genereerimine aspekt-orienteeritud mudelitest, mis rahuldavad aspekt-orienteeritud testi kattekriteeriume skaleerub oluliselt paremini kui genereerimine samade kattekriteeriumite korral mitte aspekt-orienteeritud mudelitest.

Töös alguses tutvustati metoodikat, seletades täpsemalt aspekt-orienteeritusega kaasnevaid termineid ning tutvustati uuringu läbiviimiseks kasutatud tarkvara. Esitati testides kasutatud mudelite detailne tutvustus ning testide kirjeldus. Viidi läbi kolm katseseeriat, mille käigus suurendati aspektide, nende põimpunktide ja aspektidega seotud teede arvu mudelites. Testijadad esimese kahe katseseeria korral genereeriti kolme mudelkontrolli päringuga ja viimane katseseeria nelja päringuga. Iga päringu läbimisel kasutas mudelkontrollil sügavuti otsingu algoritmi, mis leidis päringut rahuldava lühima tee. Töös on toodud välja katsetes saadud tulemused ning nende põhjal graafiline analüüs. Analüüsi põhjal tehtud järeldused kinnitavad analüütilistest keerukusmudelitest saadud hinnanguid. Eksperimentidest tulenevad praktilised järeldused ja nende võimalik mõju tarkvara arendustehnikatele seoti üldiste arengutendentsidega tarkvaramaailmas.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 29 leheküljel, 5 peatükki, 13 joonist.

Abstract

Aspect oriented testing efficiency analysis

The aim of the bachelor's thesis is to demonstrate experimentally the advantages of aspect-oriented models ahead of non-aspect-oriented models in model-based test generation. Different aspect-oriented test coverage criteria were under consideration while tests were automatically generated using model checking technique. It was show in experiments that generating tests from aspect-oriented models where tests satisfy aspect-oriented coverage criteria, scales inherently better than generating same tests from non-aspect-oriented models.

A detailed introduction of the terms of aspect orientation and the software used to conduct the study is presented in the methodology part of the work. A detailed introduction of models used in the tests was provided and the results obtained in the tests were highlighted. Three series of experiments were performed. The first two consisted of three model checking queries that generated test sequences and the last one of four queries. For each query, UPPAAL model checker used a depth first search algorithm that found the shortest test paths. At the end of thesis, the results were summarized in diagrammic form. The analysis led to a conclusion that confirm the predictions of analytical complexity estimates on TCTL model checking presented in research literature. The practical implications of the work done have been outlined and put into the context of general tendencies of software world.

The thesis is in Estonian and contains 29 pages of text, 5 chapters, 13 figures.

Lühendite ja mõistete sõnastik

AA	Ajaga automaadid
AK	Aspekti kattuvus
AO	Aspekt-orienteeritud
AOM	Aspekt-orienteeritud modelleerimine
AOP	Aspekt-orienteeritud programmeerimine
AOTA	Aspekt-orienteeritud tarkvaraarendus
ATK	Aspekti tee kattuvus
GB	Gigabait
KB	Kilobait
MPT	Mudelipõhine testimine
NEK	Nõuandemudeli elemendi kattuvus
OOP	Objekt-orienteeritud programmeerimine
PP	Protseduuriline programmeerimine
PPK	Põimpunkti kattuvus
S	Sekund
TA	Testikatte avaldis
UAA	UPPAALi ajaga automaadid

Sisukord

1 Sissejuhatus 8
2 Metoodika 10
2.1 Aspekti mõiste süsteemi arenduses 10
2.2 UPPAALi ajaga automaadid 12
2.3 Aspekt-orienteeritud modelleerimine 15
2.4 Aspekt-orienteeritud testi katte kriteeriumid 16
3 Peamised tulemused 19
3.1 Päringu täitmine aspekti teede lisamisel, läbides kõiki aspekte 23
3.2 Päringu täitmine aspektide lisamisel, läbides sama hulka aspekti teesid 24
3.3 Päringu täitmine aspekti teede lisamisel, läbides üht aspekti 25
4 Analüüs ja järeldused 27
4.1 Esimese katseseeria järeldused 27
4.2 Teise katseseeria järeldused 29
4.3 Kolmanda katseseeria järeldused 32
4.4 Järelduste sidumine tarkvaraarenduse maailmaga 35
5 Kokkuvõte 36
Kasutatud kirjandus 37
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 39
Lisa 2 – Nõuandemudeli tagastav kaar 40
Lisa 3 – Baasmudelite lõppasur 41
Lisa 4 – Aspekt mudel ja tasapinnaline mudel lisatud teedega 42
Lisa 5 – Kahe aspektiga AO mudelile lisatud aspekt 43
Lisa 6 – Kolme aspektiga AO mudelile lisatud aspekt 44
Lisa 7 – Veateade tasapinnalise mudeli läbimisel 45
Lisa 8 – Tasapinnalise mudeli esimese nõuandemudeli kõikide teede läbimise päringu jooksutamise protsessi näitajad 46

Jooniste loetelu

Joonis 1. Programmi aspektid süsteemi kihtides 11
Joonis 2. Rakenduse kihtide suhe logimisega 12
Joonis 3. Aspektmudel täitmise jälgedega. 20
Joonis 5. Kahe aspektiga AO mudel koos täitmise jälgedega. 21
Joonis 4. Ühe aspektiga AO mudel koos täitmise jälgedega. 21
Joonis 6. Kolme aspektiga AO mudel koos täitmise jälgedega. 22
Joonis 7. Viie nõuandemudeliga tasapinnaline mudel koos täitmise jälgedega. 22
Joonis 8. Graafik kujutab mudelkontrollile kuluva koguaja muutust konstantse hulga aspektide/ nõuandemudelite läbimisel ja mudelite teede arvu suurendamisel. 28
Joonis 9. Graafik kujutab virtuaalmälu kasutuse tulemusi konstantse hulga aspektide/ nõuandemudelite läbimisel ja mudelite teede arvu suurendamisel. 29
Joonis 10. Graafik kujutab mudelkontrollile kuluva koguaja muutust konstantse arvu mudelite teede läbimisel ja aspektide/ nõuandemudelite hulga suurendamisel. 31
Joonis 11. Graafik kujutab virtuaalmälu kasutuse tulemusi konstantse arvu mudelite teede läbimisel ja aspekti/ nõuandemudelite hulga suurendamisel. 32
Joonis 12. Graafik kujutab mudelkontrollile kuluva koguaja muutust konstantselt ühe aspekti/ nõuandemudeli läbimisel ja mudelite teede arvu suurendamisel. 34
Joonis 13. Graafikuna virtuaalmälu kasutus tulemused konstantse ühe aspekti/ nõuandemudeli läbimisel mudelite teede arvu suurendades..... 35

1 Sissejuhatus

Laialt tuntud programmeerimisviisi objekt-orienteeritud programmeerimine (OOP), nähakse arenduses kõikeparandava vahendina [1]. Aastakümnetega on tehnika ennast tõestanud, lihtsustades mahukate koodide arendust ja haldamist, lubades infot peita ja simuleerides pärismaailma probleeme [2]. Mitme positiivse küljega objekt-orienteeritud programmeerimine ei suuda siiski kõigile probleemidele lahendust leida. G. Kiczales kaasautoritega kirjutas [1], et tähtsaid disainiga kaasnevaid lahendusi, mida programmis peab rakendama, ei suudeta katta ei OOP ega tema eelkäija protseduurilise programmeerimisega (PP) [1].

Üks suur probleem, mille taha traditsioonilised programmeerimisviisid kinni jäävad, on võimetus kompaktselt esitada disaini läbivaid aspekte ("*cross-cutting concerns*") [1]. Enamasti on rakendustes üldiseid funktsionaalsusi, mis ei ole otseselt seotud äriloogikaga [3]. Taolised põhifunktsionaalsust toetavad funktsionaalsused ehk aspektid võivad olla jagatud ka mitme rakenduse vahel [4]. Aspekt võib olla näiteks logimine, turvalisus, veakäsitlus, vahemällu salvestamine, jõudluse monitoorimine ja andmete edastamine [3] [4].

Antud kitsaskohta silmas pidades arendati uus programmeerimisviis: aspekt-orienteeritud programmeerimine (AOP). AOP kodeerimistehnika arvestab disaine läbivaid aspekte [1]. Põhiliselt seostatakse aspekt-orienteeritust programmeerimisega, aga mainitud terminit on hakatud kasutusele võtma ka mudelpõhises tarkvaraarenduses. Aspekt-orienteeritud programmeerimise ja aspekt-orienteeritud modelleerimise (AOM) põhimõtted on samad. Mõlemad tagavad disaini läbivate aspektide lahususe, kerge hooldatavuse ja soodustavad paindlikku arendamist [5].

Autoril tekkis huvi aspekt-orienteeritust uurida, kuna siiani oli ta kokku puutunud peamiselt OO-ga ja osaliselt PO-ga. Tekkis soov AO paremini tundma õppida ning välja selgitada, mis hetkel peaks hakkama AO metoodikat eelistama. Otsustati uurida seda mudelite abil. Ühed mudelid on aspekt-orienteeritud ja teised on mitte aspekt-orienteeritud ehk tasapinnalised mudelid. AO mudelid kujutavad süsteeme, kus on rakendatud aspekt-orienteeritud arendust. Tasapinnalised mudelid kujutavad süsteeme,

kus puudub aspekt-orienteeritud arendus. Hüpoteesiks kujunes, et AO mudelite eelis suureneb mitte-AO mudelite ees, kui suureneb süsteemides aspektid arv.

Autor keskendub oma töös aspekt-orienteeritud modelleerimise eeliste väljatoomisele tavalise modelleerimise ees Uppaali ajaga automaatide (UAA) näitel, keskendudes AO mudelite kasutusele mudelipõhises testimises. Aspekt-orienteeritud mudeli käsitsi kirjeldamine ja analüüs on aeganõudev ja selle käigus on lihtne teha vigu. Teisest küljest on UAA AOM mudelteisenduste semantika hästi defineeritud ja kergesti automatiseeritav. Käesoleva lõputöö käigus toob autor esile näitajad, mis tõestavad AOM eeliseid ja tugevusi tavamudelite ees nii testimudelite spetsifitseerimisel, kui ka AO testi kattekriteeriumite esitamisel ja testide genereerimisel.

Siinkohal tänan oma juhendajat Jüri Vainu, kes avas ukse aspekt-orienteeritud arendusele, õpetas esitama olulisi küsimusi, ning toetas töö valmimist ammendamatu kannatlikkuse ja suunava asjatundlikkusega.

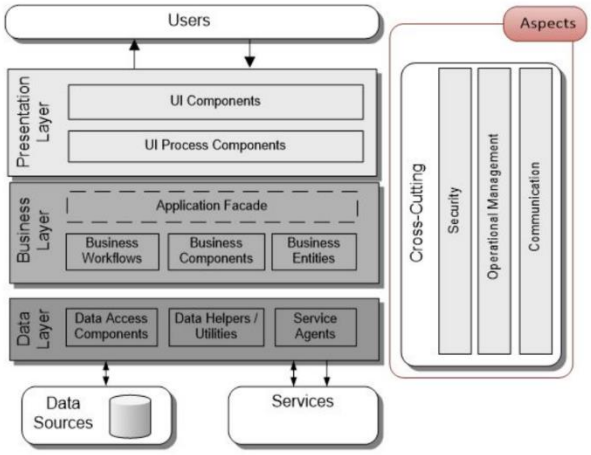
2 Metoodika

Käesolevas peatükis tutvustatakse aspekti mõistet, selgitades tema rolli süsteemi disainis ja arenduses. Edasi räägitakse lähemalt UPPAALi ajaga automaadist. UPPAAL tarkvara kasutati töös aspekt-orienteeritud mudelite modelleerimiseks, verifitseerimiseks ja testide genereerimiseks. Järgnevalt tutvustataksegi aspekt-orienteeritud modelleerimist ja selle ülesannet tänapäeva tarkvaraarenduses. Mudelipõhiste testide genereerimise puhul jälgiti testi katte kriteeriumeid, mille mõisted seletatakse peatüki lõpus.

2.1 Aspekti mõiste süsteemi arenduses

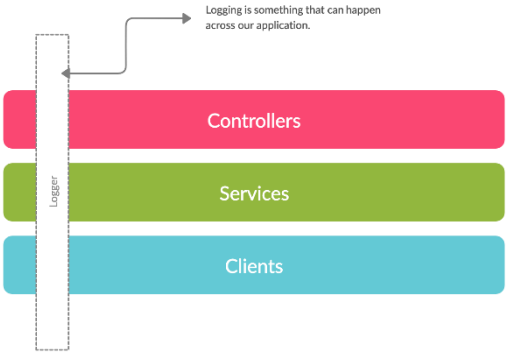
Mitmeid süsteeminõudeid ei saa kapseldada ühe tervikuna, kuna süsteemiga on seotud mitmed moodulid ja alammoodulid, milles nõudeid tuleb jälgida. Tavaliselt on taolisteks nõueteks logimine, turvalisus, püsivus, mälu haldamine, jälgimine (*tracing*) jne. Nende nõuete rakendamine võib süsteemi muuta raskesti hallatavaks ja süsteemi on raske edasi arendada. [6]

Programmi aspekt on tavaliselt seotud mitme erineva programmielemendiga [7]. Disaini läbivad aspektid on osa projekti igast olemasolevast või tulevast infrastruktuurist. Aspektid on vajalik osa tarkvararakendusest, mis esinevad rakenduse erinevates kihtides. Lisaks levinud programmiaspektidele (logimine, veakäsitlus jne), on igal rakendusel ka omad operatsioonid, mis on lahutamatu osa tarkvarast ja mille funktsionaalsused on vajalikud igas kihis. Tavaliselt hallatakse andmekihti, ärikihti ja kasutajaliidest eraldi projektides, nagu joonisel näidatud (Joonis 1). Joonisel on ka kujutatud kuidas programmi aspektid on jaotunud ühtlaselt igasse kihti. [8]



Joonis 1. Programmi aspektid süsteemi kihtides

Disaini piires on taolist arhitektuurilist struktuuri kerge kujutada, aga tõelised raskused algavad siis, kui algab päris tarkvaraarendus. Tugev disain ja õige teostus on väga tähtsad, muidu on suur oht, et terve rakenduse haldamine muutub väga keeruliseks ja kurnavaks. Näide nõrgast disainist oleks see, kui rakenduse valmides ilmnevad probleemid logimisega. Taoline olukord võib tekkida juhul, kui disain varajastes arendusfaasides ei toeta piisavalt infrastruktuuri. [8] Joonisel 2 on hästi ja lihtsalt näidatud, kuidas logimine ei ole seotud ainult ühe kihiga, vaid läbib kõik rakenduse kihte. Pidades logimist üldiseks funktsionaalsuseks, mis ei ole otseselt seotud süsteemi põhifunktsionaalsustega, on see siiski vajalik osa disainist, millega tuleb arvestada. Maja ehitamise puhul saaks antud näidet võrrelda olukorraga, kus hoone projektis ei ole kujutatud elektrijuhtmete asukohta. Professionaalid arvestavad juba varakult rakenduse jaoks vajalike disaini läbivate aspektidega. [8] Kui ei pöörata piisavalt tähelepanu programmi aspektidele, luuakse oht, et programmi sisemised sõltuvused omavahel n-ö takerduvad. Koodi tasemel tähendab see näiteks koodi hajali (*scattered*) olekut või koodi dubleerimist mitmes erinevas asukohas, mis omakorda kaotab süsteemi modulaarsuse. [1]



Joonis 2. Rakenduse kihtide suhe logimisega

Pahatihti ei peeta rakenduses disaini läbivaid aspekte prioriteediks. A. Choudhary väidab, et disaini tasemel nendega veel arvestatakse, kuid kodeerides omistatakse neile juba vähem tähtsust. Head tarkvararakendused põhinevad infrastruktuuril, mis eraldab disaini läbivaid aspekte tuumfunktsionaalsusest [7]. A. Choudhary usub, et programmisisesed aspektid väärivad sama suurt tähelepanu kui tarkvararakenduse teised arhitektuurikihid. [8]

Disaini läbivate aspektide eraldamiseks on loodud aspekt-orienteeritud arendusmudel [7]. Aspekt-orienteeritud tarkvaraarendust (AOTA) kasutatakse tarkvarasüsteemide modulariseerimiseks, et isoleerida teisejärgulised ja tarkvara toetavad funktsionaalsused äriloogikast. [1]

Järgnevas peatükis tutvustatakse lähemalt tarkvara, mida uurimisel kasutati.

2.2 UPPAALi ajaga automaadid

Mudelipõhine testimine (MPT) on saanud mugavaks lahenduseks tarkvara testimise juures, vähendades testimisele kuluva inimtöö mahtu. MPT abstraktsete mudelitega spetsifitseeritakse testitava süsteemi oodatavaid käitumisi ja sellest tuletatakse automaatselt testid. Utting et al. viitab oma töös, et MPT-l on kolm konkreetselt eristatavat faasi: testi spetsifitseerimine, testi genereerimine ning testi täitmine. [9]

Võrreldes traditsiooniliste testimismeetoditega, mis pelgalt määravad testi eesmärgi, modelleerib mudelipõhine testimine testi nõuded, millele testimisel olev süsteem peaks vastama. Mudel-kontroll on üks meetoditest, mida kasutatakse testi genereerimisel. Andes testitava süsteemi mudelile ette sobivad väljad, genereerib mudel-kontroll täitmisjäljed (*traces*), mis tõestavad antud käitumisnõuete paikapidavust. Saadud jälgede

abil genereeritakse testitaval süsteemil täidetavad testid. UPPAALi ajaga automaadid on levinud modelleerimisformalism, mis omab vahendeid mudel-kontrolliks ja testimiseks. [9]

UPPAAL-i integreeritud tööriista keskkonda (*integrated tool environment*) kasutatakse reaalaja süsteemide modelleerimiseks, valideerimiseks ja kontrollimiseks. UPPAAL-i mudeli editor võimaldab kujutada reaalajasüsteemide mudeleid ajaga automaatide võrguna, millele lisanduvad kõik enamlevinud andmetüübid ning lisaks spetsiaalne andmetüüp kellad. [10] UPPAALi ajaga automaate kasutatakse testitava süsteemi omaduste ajastustingimuste spetsifitseerimisesks ja analüüsiks ja seetõttu kasutatakse UPPAALi tööriista reaalajasüsteemide disainimisel. [9]

Paralleelselt komponeeritud UPPAALi ajaga automaadid moodustavad laiendatud ajaga automaatide suletud võrgustiku. Neid automaate, mille parameetrid on väärtustatud, nimetatakse omakorda protsessideks. Protsessid agregeeritakse üheks süsteemiks kasutades sünkroonset paralleelkompositsiooni. Automaadi graafilises esituses kutsutakse sõlmi asuriteks (*locations*) ja asuri vahelisi kaari servadeks (*edges*). [9] Ajaga automaat koosneb lõplikust hulgast asuritest (L), mis sisaldab algasurit (l0), kaarte hulgast (E) ja invariantide hulgast (I), mis on omistatud asuritele. [11] Automaadi olek koosneb tema jooksvat juhtimisolekut näitavast asurist ja kõikide muutujate, kaasa arvatud kellad (*clocks*) väärtustusest [9]. Kell on andmetüüp UPPAAL-is, millele saab täisarvulisi väärtusi omistada ning kasutada mudeli tingimustes. Enamasti tähistatakse kella andmetüübiga muutujaid x, z või y-ga [11]. Sünkroonne suhtlus protsesside vahel luuakse sünkroniseerimislinkide abil. Neid nimetatakse kanaliteks (*channels*). Kanaliga sünkroniseeritavad servad defineeritakse mõlemasse protsessi, mille vahel soovitakse tekitada sünkroonne link. Mõlemal protsessil märgitakse soovitud kaared vastava kanali sümboliga, näiteks „ch“. Kanali sisendi tähiseks on kanali ch puhul ch? ja väljundi tähiseks on ch!. Asünkroonset suhtlust protsesside vahel modelleeritakse kasutades globaalseid muutujaid, mida saavad väärtustada ja lugeda kõik mudeli protsessid. [9]

Aspekt-orienteeritus annab meetodi funktsionaalsuse kapseldamiseks, mis võimaldab modelleerida erinevaid disaini läbivaid aspekte suhteliselt sõltumatult. Neid aspekte modelleeritakse baasmudeli (*base model*) ja nõuandemudeli (*advice model*) kompositsioonina. Komposiitmudel koosneb baasmudelist ja nõuandemudelist, mis on teineteisega põimitud (*woven*). Baasmudel on komplekt UAA protsesse, mis modelleerivad süsteemi baasfunktsionaalsust ehk põhifunktsionaalsust. Nõuandemudel

ehk aspektmudel on UPPAALi AA protsess või komplekt paralleelsetest protsessidest, mis modelleerivad süsteemi disaini läbivaid aspekte. Põimpunkt (*join point*) on mudeli fragment baasmudelis, millega saab põimida aspekte. Põimitud mudeleid nimetatakse augmenteeritud mudeliks. Augmenteeritud mudel on UAA, kus on baasmudel põimitud kõikide vajalike aspektmudelitega. Põimimine (*weaving*) on protsess, mille käigus lisatakse baasmudelile aspektid. Põimurid (*weaving adapters*) on mudeli fragmendid, mis tagavad, et põimisel säilivad nii baasmudeli kui aspektmudeli algsed omadused. [9]

Autor lähtus oma töös samadest eeldustest, mis on esitatud publikatsioonis [9]. Eeldused on järgnevad:

- Baasmudeli põimispunkti külge ei põimita üle ühe nõuandemudeli [9].
- Jõudes põimpunktini, antakse juhtimine üle baasmudelilt aspektmudelile, mis on põimitud vastava põimpunktiga. Baasmudeli täitmine põimimispunktist edasi ei jätku enne, kui sellega põimitud nõuandemudel on lõpetanud täitmise. Baasmudeli täitmine jätkub, kui aspekt lõpetab oma töö ja juhtimine on tagastatud samasse põimpunkti. Ühe nõuande mudeli täitmine ei takista teiste põimpuntide täitmist teistes protsessides. Erinevate protsesside aspekte võidakse täita samaaegselt. [9]
- Nõuandemudelil on ainult üks algpunkt, aga tal võib-olla üks kuni mitu lõpp-punkti, mille puhul juhtimine antakse tagasi samasse põimpunkti [9].
- Baasmudelit ja nõuandemudelit põimitakse kasutades UAA sünkroniseerimiskanaleid [9].
- Põimuri enda elemendid ei saa olla põimpunktid, aga põimitud nõuandemudeli elemendid saavad olla omakorda põimpunktid järgmise nõuandemudeli põimimisel. Sellega garanteeritakse baasmudeli algsete omaduste säilitamise. [9]

Kirjeldatud põimimistehnika puhul on näidatud, et rakendades aspekt-orienteeritud modelleerimise põhimõtteid UPPAALi ajaga automaadile, paraneb mudelipõhise valideerimise ja testimise modulaarsus ning testide genereerimise efektiivsus [9].

Töös kasutatud põimimistehnika täpsemaks kirjeldamiseks anti UPPAALi ajaga automaatidele formaalse definitsiooni. Edasises peatükis räägitakse lähemalt aspekt-orienteeritud modelleerimisest.

2.3 Aspekt-orienteeritud modelleerimine

Tarkvaratehnika arengut on juhtinud vajadus leida parem viis, kuidas eraldada erinevaid aspekte ning kapseldada funktsioone eraldi üksustesse, et lokaliseerida muutusi ja tegeleda korraga ühe probleemiga. A. Rashidi väitel oli sellise lahenduse saamise esimeseks vajalikuks sammuks arendada välja kõrgetasemeline keel, mis lubab eraldada programmi kontseptuaalse struktuuri tema mehaanilisest esitusest. Kõrgetasemeliste keelte loomisele järgnes funktsioonide lisamine, mis võimaldavad integreerida arendusprobleemid (*concerns*) sidusateks käitumuslikeks ja struktuurseteks abstraktsioonideks, näiteks protseduurideks või dokumentatsiooniks. [7]

Üheks niisuguseks programmeerimistehnikaks kujunes aspekt-orienteeritud programmeerimine (AOP). AOP arendamine põhines samamoodi vajadusel täiustada aspektide eraldamist üksteisest. Eesmärgiks on samuti kapseldada aspekte, mis on tihti hajutatud erinevate programmi moodulite vahel. Disaini läbivate aspektide sissetoomine, muutmine või arendamine on kulukas ülesanne, kuna neid aspekte ei saa kapseldada ühteainsasse moodulisse ning taolised ülesanded üldjuhul mõjutavad süsteemi tervikuna. Kui tuua veel mõned näited disaini läbivatest aspektidest nagu mälu haldamine, vea käsitlemine, reaalaja piirangud, ressursi jagamine, sünkroonimine jt, on näha, et kuigi pole tegu põhifunktsionaalsustega, läheb neid aspekte siiski iga süsteemi puhul vaja. [7]

Algselt võeti aspekt-orienteeritus kasutusele programmeerimiskeeles. Nüüdseks kasutatakse AO ka mudelipõhises tarkvaraarenduses, millega tagatakse parem aspektide esitamise modulaarsus mudelites. [5] Aspekt-orienteeritus on suunatud disaini läbivate aspektide modelleerimisele, et vältida programmi aspektide laiali valgumist üle terve süsteemi [12]. Aspektide eraldamine on vajalik, et saavutada modulaarsed ja kergesti hallatavad spetsifikatsioonid. Aspekt-orienteeritud modelleerimise kontekstis käsitletakse aspekti kui süsteemi omaduste ja käitumiste kogu, mis iseloomustab põhifunktsioone toetavat funktsionaalsust. AOM võimaldab arendajal keskenduda süsteemi tugifunktsionaalsustele individuaalselt, isoleerides selle süsteemi ülejäänud funktsionaalsusest. Samasugused eelised ilmnevad testimisel, võimaldades muuta testimist eesmärgipärasemaks ja lokaliseerida valesti disainitud ja implementeeritud aspektidest tekkinud vigade algpõhjusi. [5]

Aspekt-orienteeritusega kaasneb vastav terminoloogia, mida seletati täpsemalt peatükis 2.2 “UPPAALi ajaga automaadid”.

2.4 Aspekt-orienteeritud testi katte kriteeriumid

Aspekt-orienteeritud modelleerimisega kaasnevad testi katte kriteeriumid, mis on aspekt-orienteeritud testimise aluseks. AO kattuvuse piirangute semantika ja ulatuse saab kompaktsel kujul kindlaks määrata, jälgides aspekt-orienteeritud mudeli elementide hierarhiat ja alamtüüpe. Katte kriteeriumite täpsustamisel lähtub autor samast formaalse esituse vormist nagu [5], kus kattekriteeriumid esitatakse ajaga temporaalloogika TCTL valemitena. Neid valemeid nimetatakse testikatte avaldiseks (TA) (*coverage expression*). Testikatte avaldiste struktuuris on seatud aspekt-orienteeritud kattuvuse alamvalemid laiemat skoopi omavatesse valemitesse, mis määrab omakorda alamvalemi skoobi. Katvusskoopide hierarhia laiemalt kitsamale on järgmine: aspekti kattuvus (*aspect coverage*) (AK), põimpunkti kattuvus (*join point coverage*) (PPK), aspekti tee kattuvus (*aspect path coverage*) (ATK) ja nõuandemudeli elemendi kattuvus (*advice model element coverage*) (NEK). [5]

Aspekt-orienteeritud testikatte avaldiste põhjal genereerib Uppaali mudelkontrollija (*model checker*) sümbolkujul testi jäljed (*traces*), mille parameetrite väärtustamisel saab testitava süsteemi poolt otseselt täidetavad testijadad. Niisugune testide genereerimise tehnika võimaldab hinnata aspekt-orienteeritud testigenereerimise efektiivsuse mõõdikuid, milleks on virtuaalmälu kasutus, mudelkontrollile kuluv koguaeg ja genereeritud jälje pikkus. Lühidalt nimetatud mõõdikutest, mida uurimise käigus jälgiti:

- Virtuaalmälu maksimaalne kasutus näitab maksimaalset teoreetilist mälu kasutust, mida läheb jooksval protsessil vaja minna [13].
- Mudelkontrollile kuluv koguaeg sisaldab puhtalt protsessoriaeg kui ka vahetulemuste mällu kirjutamist ja sealt lugemise aega.
- Mudeli täitmise jälg on sisend- ja väljundsümolite jada. UAA mõistes on need testiportidel ilmnevad sisend- väljundsündmused ja nende ajatemplid.

TA-d lahti seletatuna on järgnevad:

- Aspekti kattuvus, mis näitab baasmudeliga põimitutud aspekti täitmist testi käigus.
- Põimpunkti kattuvus, mis näitab ühe või mitme põimpunkti läbimist testi käigus.

- Aspekti tee kattuvus, mis näitab nõuandemudelis olevate võimalike teede hulga läbimist testi käigus.

- Nõuandemudeli elemendi kattuvus, mis näitab nõuandemudelis eraldi spetsifitseeritud üksikute elementide läbimist testi käigus.

AO mudelite puhul saavad testikatte avaldised olla nõrgalt või tugevalt kaetud. Kui aspekti mudelil on tugev aspekti kattuvus, tähendab see, et mudelis peavad olema kõik testikatte avaldisega määratud elemendid kaetud testidega. Tugeva AK rakendamiseks kasutatakse parameetritega UPPAALi AA malle, kus parameetri p_i indeksi ulatus on ühest n-ni, mis identifitseerivad mudeli aspekte. P(i) tähistagu predikaati, millele omistatakse tõene väärtus, kui i-s aspekt on täidetud. Mudeli täitmise jäljed, mis testivad tugevat AK-d, peaksid rahuldama järgnevat päringut (päring 1): [5]

$$E\Diamond\ forall\ (i{:}\,int\ [1,n])\ P(i). \qquad \text{(päring 1)}$$

Nõrga aspekti kattuvuse puhul on vähemalt üks aspekti nõuandemudel läbinud testi. Nõrga AK päringul (päring 2) peaks ainult üks aspekt olema täidetud. [5]

$$E\Diamond\ exists\ (i{:}\,int\ [1,n])\ P(i) \qquad \text{(päring 2)}$$

Tugeva põimpunkti kattuvuse peaks rahuldama päring (päring 3), kus j vahemik katab kõik põimpunkide indeksid, mille aspektidele viidati aspekti kattuvuse osas. R(j) on tõeväärtus-muutuja, mille väärtus muutub tõeseks iga läbitud põimpunkti juures. [5]

$$E\Diamond\ forall\ (j{:}\,int\ [1,m], i{:}\,int\ [1,n])\ P(i)\ \&\&\ R(j) \qquad \text{(päring 3)}$$

Nõrk põimpunkti kattuvuse päring (päring 4) on rahuldatud, kui vähemalt üks põimpunkti täitmise jälg on läbitud. [5]

$$E\Diamond\ exists\ (j{:}\,int\ [1,m], i{:}\,int\ [1,n])\ P(i)\ \&\&\ R(j) \qquad \text{(päring 4)}$$

Tugeva aspekti tee kattuvuse saavutamiseks peaks nõuandemudelis läbima kõik teekonnad nõuandemudeli algasurist nõuandemudeli kõikidesse lõppasuritesse. Olgu aspektmudeli sisend- ja väljundkaared dekoreeritud entry(i, j) ja exit(i, j) predikaatidega, siis tugeva ATK päringu (päring 5) rahuldamiseks on vaja läbita testis kõik jäljed, mis sisaldavad sisend- ja väljundkaarte täitmist. [5]

$E\Diamond\ forall\ (Path(entry(i,j), exit(i,j)))$ (päring 5)

Nõrga ATK puhul peaks vähemalt üks aspekti teekond olema läbitud. See tähendab, et nõrga ATK päring (päring 6) vajab, et ainult üks sisend- ja väljundkaar oleks testijäljes esindatud. [5]

$E\Diamond\ exists\ (Path(entry(i,j), exit(i,j)))$ (päring 6)

Nõuandemudeli elemendi katte kriteeriumid kehtestavad piirangud UAA elemendi tüüpidele, mida kaetakse nõuandemudelis. Tugeva nõuandemudeli elemendi kattuvuse korral peab päring (päring 7) läbima kõik nõuandemudeli elemendid.

$E\Diamond\ forall\ (Elements)\ \&\&\ Epilogue$ (päring 7)

Nõrga nõuandemudeli elemendi kattuvuse korral peaks päring (päring 8) läbima vähemalt ühe elemendi nõuandemudelis.

$E\Diamond\ exists\ (Elements)\ \&\&\ Epilogue$ (päring 8)

3 Peamised tulemused

Peatükk käsitleb mudelite ja mudeli täitmise jälgede genereerimist erinevate testikatte kriteeriumide alusel ning toob välja uuringu käigus saadud tulemused. Võrreldavateks mudeliteks on aspekt-orienteeritud mudelid ja nendega sama funktsionaalsusega mitte-aspekt-orienteeritud mudelid nn tasapinnalised mudelid. Iga aspekt-orienteeritud mudeli ja sellega bisimulatsiooni relatsiooni mõttes ekvivalentse mitte-aspekt-orienteeritud mudeli võrdlemise on kasutatud samu mudelkontrolli päringuid, mis genereerivad mudelitel sümbolkujul oleva täitmisjälje. Samuti on kasutatud selguse mõttes võrreldavate mudelite korral samu konstruktsiooni elemente ja ühesuguseis asurite nimesid.

Tasapinnalise mudeli ja kõigi AO mudelite viimase asuri nimeks on pandud *final*, nagu on näha Lisa 3. Nimetus võimaldab, et kõikidel mudelitel läbimisel on sama algus punkt ja lõpu punkt. Testide jaoks on see vajalik, et saada tulemusi võrrelda.

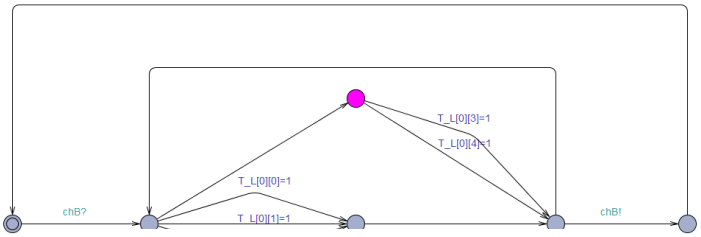
Selgitamaks kuidas mõjutavad aspektidega seotud parameetrid testide genereerimise efektiivsust on katsetes varieeritud aspektide arvu ja teiseks iga aspekti korral põimpunktide arvu. Katsetes on kasutatud kokku kolm erinevat aspekt-orienteeritud mudelit, mis erinevad aspektide arvu poolest. Uuringu käigus loodi ühe, kahe ja kolme põimpunktiga baasmudelid.

Bisimulaarse tasapinnalise mudeli konstrueerimiseks on rakendatud põimpunktides nõuandemudelite superponeerimist. See tähendab, et baasmudelis on põimpunktid asendatud (ilma põimurita) nõuandemudelitega.

AO baasmudelis võib nõuandemudel olla põimitud mitme põimpunktiga koos teda ümbritseva põimuriga (joonis 3). Katsetulemuste võrreldavuse tagamiseks on katseteks konstrueeritud nõuandemudel läbi kõikide katsete ühesugune. Nõuandemudel koosneb kuuest asurist ja neid ühendab üksteist kaart, mis moodustavad aspekti teed. Kaarte küljes olevad Boole'i tüüpi katvusmuutujate või Boole'i massiivi elementide omistamised võimaldavad päringus identifitseerida kaari, mille läbimine antud testikatte kriteeriumi järgi on nõutav. Näiteks täitmise jäljed, mille läbimise tulemusena väärtustatakse massiivi T_L[k][i] elemendid väärtusega true spetsifitseerivad tee kattuvust, kus indeks k viitab

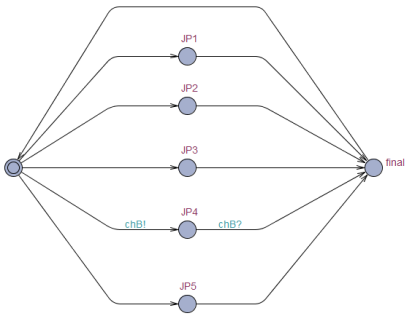
aspektile ja indeks i identifitseerib aspekti tee. Katsetes kasutatus mudeli struktuuri korral on nõuandemudeli teid võimalik läbida viiel erineval viisil, seetõttu on indeks i ulatus nullist neljani. Ainsaks indekseerimata aspekti teeks jääb nõuandemudelis tagastav kaar, mida on näha Lisas 2. Tagasi algasurisse suunav kaar on n-ö abiks, et nõuandemudeli läbimisel oleks võimalik jõuda ühe päringuga kõikide aspekti teedeni. See tagab mudeli sidususe, et tugev aspekti tee kattuvus oleks saavutatav.

Mudeli ümber olev põimur koosneb sisend- ja väljundkaartest. Sisendkaar koosneb omakorda algasurist ja kaarest, mille küljes on sünkroniseerimis tingimus (chB?). Sisendkanali chB? eesmärgiks on luua sünkroniseerimislink aspektmudeli ja baasmudeli vahel. Väljundkaar koosneb lõppasurist ja sünkroniseerimistingimusega (chB!) kaarest. Väljundkanal chA! käivitab baasmudeli põimpunktist väljuva kaare, st baasmudel saab jätkata täitmist põimpunktist edasi.



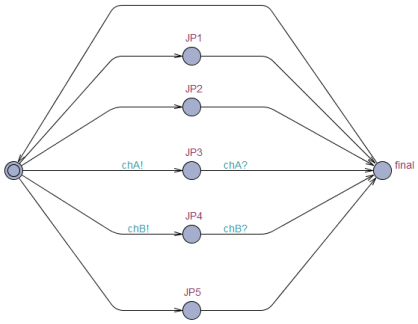
Joonis 3. Aspektmudel täitmise jälgedega.

Baasmudel koosneb seitsmest asurist ja üheteistkümnest kaarest (joonis 4). Baasmudelil on algasur ja lõppasur. Kõigi keskmiste asurite külge on võimalik põimida aspekt, mille tulemusena neid asureid on katsetes kasutatud põimpunktidena. Ned põimpunktid on tähistusteks nimedga JP1 kuni JP5. Joonisel 4 toodud AO mudelil on ainult üks aspekt, mida on kujutatud joonisel 3. ChB kanalite funktsioon on samasugune nagu põimuritel sisend- ja väljundühenduste puhul, kus juhtimine antakse nõuandemudelile ja nõuantemudelist väljudes jätkub baasmudeli täitmine. Baasmudel jääb niikauaks ootama olekusse JP3, kuni nõuandemudel oma protsessid lõpetab. Sünkroniseerimispunktide chB! ja aspektmudeli sünktoroniseerimispunkti chB? vahel on sünkroonlink, samamoodi on baasmudeli sisend sünkroniseerimispunkti chB? ja aspekt mudeli väljund spnkroniseerimispunkti chB! vahel kanal. Kanalite järgi oskab UPPAALi interpretaator sünkroniseerida baasmudeli ja nõuandemudeli koostäitmist.



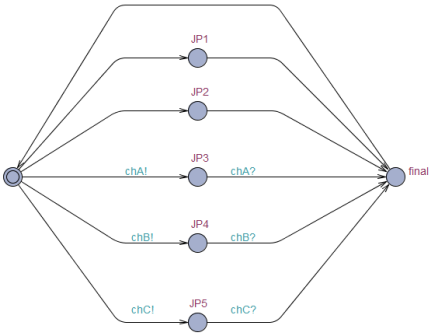
Joonis 5. Ühe aspektiga AO mudel koos täitmise jälgedega.

Teisel AO mudelil on kaks aspekti (joonis 5). Skaleerimissammude sarnasuse tagamiseks on mõlemad aspektmudelid oma ülesehituselt identsed. Teise aspekti kanalite tähiseks on chA (Lisa 5). Nende funktsionaalsus on sama, mis joonise 3 kanalitel. Kanali erinev nimetus viitab sellele, et tegu on kahe erineva aspektiga. AO mudel on identne joonise 4 mudeliga. Erinevus on ainult kanalite arvus. Baasmudelil on kaks põimpunkti, sest kahele kaarele on lisatud chA kanalid, et põimida baasmudel lisatud aspektiga.



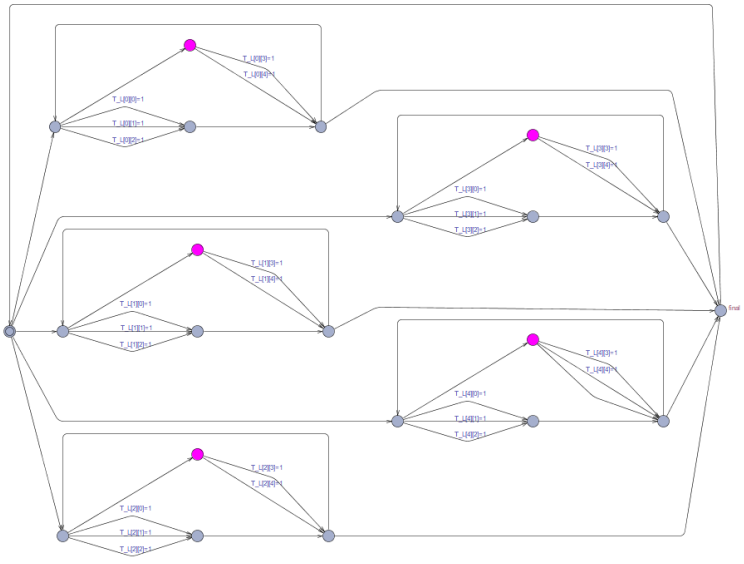
Joonis 4. Kahe aspektiga AO mudel koos täitmise jälgedega.

Kolmandal mudelil on kolm aspekti, mis on põimitud baasmudeli asuritega (joonis 6). Baasmudel on taas peaaegu identne varasemate baasmudelitega. Keerukuse inkrementaalseks suurendamiseks on lisatud ainult kanalid chC, millega tekkis mudelile juurde uus põimpunkt. Kanalite teistsugune nimetus annab teada, et tegu on erineva aspektiga. Nõuandemudel on ülesehituselt identne varasemate nõuandemudelitega, ainult kanali nimetus chC eristab seda teistest (Lisa 6).



Joonis 6. Kolme aspektiga AO mudel koos täitmise jälgedega.

Tasapinnalisel mudelil on kattekriteeriumi kirjeldamiseks massiiv T_L[k][i], kus k tähistab nõuandemudelit ja i tähistab nõuandemudeli teid (joonis 7). Tasapinnalisel mudelil on viis nõuandemudelit. Nõuandemudelid on identsed eelmiste aspektide nõuandemudeliga (joonis 3). Tegemist on mitte-AO mudeliga, sellepärast puuduvad nõuandemudelitel põimurid ja kanalid. Tasapinnalisel mudelil on 22 asurit ja 51 kaart. Täitmise jälgede indeks i ulatus on samaväärne nõuandemudeli teede hulgaga (joonis 3), ja indeks k ulatub nullist neljani.



Joonis 7. Viie nõuandemudeliga tasapinnaline mudel koos täitmise jälgedega.

Järgmisena esitatakse testijälgede genereerimise päringud eelkirjeldatud mudelitel ja nende genereerimisele kulunud arvutusressursi (aeg, mälu) hinnangud.

3.1 Päringu täitmine aspekti teede lisamisel, läbides kõiki aspekte

Aspekt-orienteeritud mudelil ja sellele vastava mitte aspekt-orienteeritud mudelil kasutati sümboljälgede genereerimiseks kolme päringut. Esimeses katseseerias kasutati kolme aspektiga AO mudelit ja tasapinnalist mudelit, millel läbiti ainult kolm esimest nõuandemudelit. Päringu indeks k ulatus oli nullist kaheni, aga indeks i muutus. Iga päring käis läbi kõik aspektid. Iga päringuga suurendati aspekti teede arvu läbimist. Kõikide päringute puhul oli tugev aspekti kattuvus ja nõrk aspekti teede kattuvus, sest ükski päring ei läbinud kõiki aspekti teesid.

Esimese päringuga (päring 9) läbiti kõigi aspektide teed kohal i, kus katvusmuutujate massiivi indeks i võrdus nulliga. Tasapinnalise mudeli puhul läbis päring esimese kolme nõuandemudeli tee sama indeksiga katvuselemendi korral. Läbimise pikkus oli mõlemal mudelil 20 olekut. Kolme aspektiga AO mudeli mudelkontrollile kuluv koguaeg oli 0,282 sekundit ja virtuaalmälu kasutus oli 49 096 KB. Tasapinnalise mudelil mudelkontrollile kuluv koguaeg oli 0,165 s ja virtuaalmälu kasutus oli 67 304 KB.

$E\Diamond\ forall(k{:}\,K)\ T[k][0]\ \&\&\ Model.final$ (päring 9)

Teine päring (päring 10) läbis kõigi aspektide teed katvusmassiivi indeksi i väärtustega vahemikus nullist üheni. Tasapinnalise mudeli puhul läbis päring esimese kolme nõuandemudeli esimest kaht teed. Mõlema mudelil täideti 47 mudeli täitmise jälge. Aspekt-orienteeritud mudeli mudelkontrollile kuluv koguaeg oli 1,406 sekundit ja virtuaalmälu kasutus oli 87 544 KB. Tasapinnalisel mudelil kulus päringu täitmiseks 3,907 sekundit ja see kasutas 298 424 KB virtuaalmälu.

$E\Diamond\ forall(k{:}\,K)\ \ T[k][0]$ && T[k][1]&& M$odel.final$ (päring 10)

Kolmandal päringul (päring 11) varieerus indeksi i väärtus nullist kaheni ehk läbiti kõigi aspektide teed , mis vastasid indeksi i väärtusele. Siin indeksi i vahemik on nullist kaheni ning tasapinnalise mudeli puhul läbiti kolme esimese nõuandemudeli kolm esimest teed. Mudelist genereeriti jälg pikkusega 65 olekut. Kolme aspektiga AO mudelil kulus päringu täitmiseks 10,651 s ja tasapinnalisel mudelil kulus 92,324 s. Virtuaalmälu kasutus oli AO mudelil 259 992 KB ja tasapinnalisel mudelil 4 162 696 KB.

$E\Diamond\ forall(k{:}\,K)\ forall(i{:}\,I)\ T[k][i]\ \&\&\ Model.final$ (päring 11)

3.2 Päringu täitmine aspektide lisamisel, läbides sama hulka aspekti teesid

Aspekt-orienteeritud mudelitest ja neile vastavatest tasapinnalistest mudelitest jälgede genereerimisel kasutati kolme päringut. Teise testi puhul olid AO mudeliteks ühe, kahe ja kolme aspektiga AO mudel. Tasapinnalisel mudelil varieerus nõuandemudelite läbimise hulk päringust päringusse. Iga päringu puhul oli indeks i ulatus konstantselt nullist kaheni. Indeks k vahemik varieerus. Iga aspekti juures läbiti alati kolm esimest aspekti teed. Iga päringuga suurendati aspektide hulka.

Päring (päring 12) läbis ühe aspektiga AO mudeli kolme esimest aspekti teed ja tasapinnalise mudeli esimese nõuandemudeli kolme esimest teed. Päringu puhul oli tegemist nõrga aspekti kattuvusega ja nõrga aspekti tee kattuvusega. Läbimisel genereeriti jälg pikkusega 21. Ühe aspektiga AO mudeli mudelkontrollile kuluv koguaeg oli 0,002 sekundit ja tasapinnalisel mudelil kulus 0,006 sekundit. Virtuaalmälu kasutus oli AO mudelil 42 373 KB ja tasapinnalisel mudelil 57 672 KB.

$$E\Diamond\ forall(i{:}\,I)\ T[0][i]\ \&\&\ Model.final \qquad \text{(päring 12)}$$

päring (päring 13) läbis kahe aspektiga AO mudeli kõikide aspektide kolme esimest teed. Tasapinnalisel mudelil läbiti kahe esimese nõuandemudeli kolme esimest teed. Päring puhul oli tegemist nõrga aspekti kattuvusega ja nõrga aspekti tee kattuvusega. Päring leidis mõlemal mudelil lühima tee pikkusega 43. Kahe aspektiga AO mudeli päringul kulus 0,134 s ja see kasutas 45 872 KB virtuaalmälu, tasapinnaline mudeli päringul kulus 1,646 s ja kasutati 155 972 KB mälu.

$$E\Diamond\ forall(i{:}\,I)\ T[0][i]\ \&\&\ T[1][i]\ \&\&\ Model.final \qquad \text{(päring 13)}$$

päring (päring 11) oli sama, mis esimese testi kolmas päring ja ka vastavad jälgede pikkused oli samad. Päringu puhul oli tegemist tugeva aspekti kattuvusega ja nõrga aspekti tee kattuvusega. Kolme aspektiga AO mudeli päringul kulus 10,536 sekundit ja tasapinnalisel mudelil 93,05 sekundit. AO mudeli päring kasutas 260 784 KB ja tasapinnalise mudeli päring 4 182 924 KB. Andmed erinesid minimaalselt esimese testi kolmandast päringust.

3.3 Päringu täitmine aspekti teede lisamisel, läbides üht aspekti

Kolmanda katseseeria juures kasutati nelja uut päringut. Kasutati ühe aspektiga AO mudelit ja tasapinnalist mudelit. Aspekti indeks k võrdus alati nulliga, aga iga päringuga tõsteti põimpunkte indekseeriva i väärtust (suurendati põimpunktide arvu). AO mudelis läbiti alati esimest aspekti ja tasapinnalisel mudelil alati esimest nõuandemudelit. Kiire testide läbimise tõttu mõlema mudeli puhul ei olnud arvutusressursside kasvutrend täheldatav. Seepärast lisasime selle testi puhul nii tasapinnalise mudeli igasse nõuandemudelisse (joonis 6) kui ka AO mudeli aspektmudelisse (joonis 3) viis teed juurde (Lisa 4). Lisatud teedega tuli kordades selgemalt välja tasapinnalise ja AO mudeli mõõtmistulemuste trend.

Esimese päringuga (päring 14) läbiti AO mudeli ühe aspekti esimene tee ja tasapinnalise mudeli esimese nõuandemudeli esimene tee. Mudeleid läbides täideti üheksa jälge. Päringu puhul on tegemist nõrga aspekti kattuvusega ja nõrga aspekti tee kattuvusega. AO mudeli mudelkontrollile kuluv koguaeg oli 0,001 sekundit ja tasapinnalise mudeli koguaeg oli 0,009 s. Virtuaalmälu kasutus oli AO mudelil 40 580 KB ja tasapinnalisel mudelil 57 236 KB.

$$E\diamondsuit\ T[0][0]\ \&\&\ Model.final \qquad \text{(päring 14)}$$

Teise päringuga (päring 12) läbiti AO mudeli ühe aspekti esimest nelja teed ja tasapinnalise mudeli esimese nõuandemudeli esimest nelja teed. Mõlemal puhul läbiti 27 mudeli täitmise jälge. Päringu puhul on tegemist taas nõrga aspekti kattuvusega ja nõrga aspekti tee kattuvusega. AO mudeli päringul kulus 0,021 sekundit ja tasapinnalise mudeli päringul 0,488 sekundit. AO mudel kasutas uuesti 41 056 KB virtuaalmälu ja tasapinnaline mudel 63 720 KB.

Katses kolmanda päringuga (päring 12) läbiti AO mudelil ühe aspekti seitset teed ja tasapinnalise mudeli esimese nõuandemudeli seitset teed. Lühima tee pikkus mõlemal puhul oli 45. Päringu juures jäid kattuvused samaks eelnevate päringutega. AO mudeli päringul mudelkontrollile kuluv koguaeg oli 0,077 sekundit ja tasapinnalise mudeli päringul 98,871 sekundit. AO mudeli virtuaalmälu kasutus oli 41 765 KB ja tasapinnaline mudelil 3 508 6684 KB.

Viimase päringuga (päring 12) läbiti AO mudeli aspekti kõik teed. Tasapinnalise mudeli läbimisel katkestati päring ära, sest mudeli virtuaalmälu kasutus ületati. AO mudelis leiti

jälg pikkusega 57. Mudelkontrollile kuluv koguaeg oli 0,102 s ja vituaalmälu kasutus oli 42 176 KB.

Järgnevalt analüüsitakse katsetes saadud tulemusi, et teha järeldused AO mudelite eeliste kohta tasapinnaliste mudelite ees.

4 Analüüs ja järeldused

Peatükk tutvustab uuringu käigus läbi viidud kolme testi tulemuste analüüsi, mis esitatakse graafiku kujul. Kõik testid viidi läbi kasutades sama metoodikat. Igas testis kasutatud mudelit läbiti sügavuti otsingu algoritmiga, et leida alati mudeli kõige lühem tee algpunktist lõpp-punkti. Lühema tee leidmine võimaldas AO mudeli tulemusi võrrelda mitte AO mudeli omadega. Järgevalt tuuaksegi esile saadud analüüs ja selle pealt tehtud järeldused.

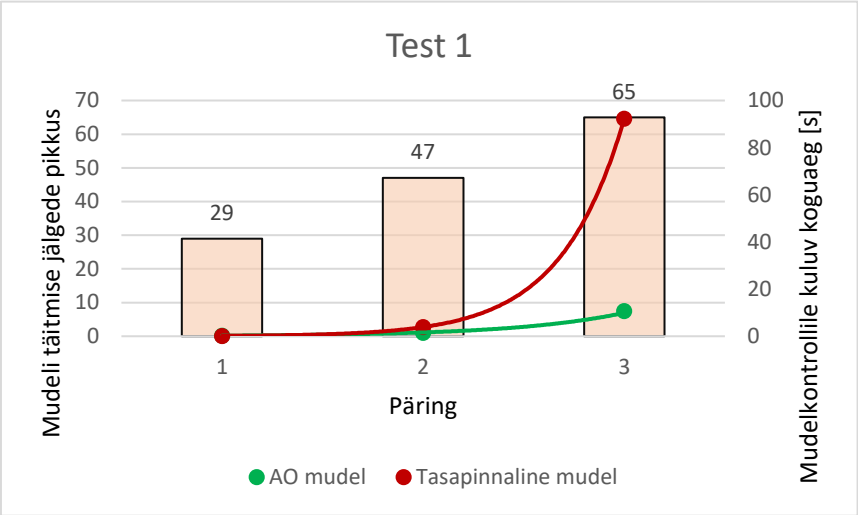
4.1 Esimese katseseeria järeldused

Esimeses katseseerias võrreldi AO mudeli ja tasapinnalise mudeli mudelkontrollile kuluva koguaja ja virtuaalmälu kasutuse muutumist, suurendades AO mudeli kõikide aspektide läbimisel ja tasapinnalise mudeli esimese kolme nõuandemudeli läbimisel teede arvu. Testi jaoks kasutati kolme päringut, kus iga järgmise päringuga läbiti suurem arv teid ja alati sama arv aspekte ja nõuandemudeleid. Hüpotees oli, et iga päringuga AO mudeli edukus suureneb tasapinnalise mudeli ees.

Andmete analüüsiks loodi kombineeritud diagramm, mis koosneb joon- ja tulpdiagrammist (joonis 8). Graafiku x-teljel on kujutatud päring üks kuni kolm. Päringud on täpsemalt seletatud kolmanda peatükki teises alapeatükkis. Graafikul on kaks y-telge, kus üks näitab mudeli täitmise jälgede pikkust ja teine mudeli läbimisel mudelkontrollile kuluvat koguaega. Punane joon näitab tasapinnalise mudeli mudelkontrollile kuluva koguaja muutust päringu kaupa. Roheline joon kujutab AO mudeli koguaja muutust päringu kaupa. Punktide põhjal loodi eksponentsiaalne trendijoon. Iga päringu juures on näidatud tulpdiagrammi kujul mudeli täitmise jälgede pikkus. Graafiku põhjal on näha, et nõuandemudelite ja aspekti teede lisamisega kasvab AO mudeli ja tasapinnalise mudeli mudekontrollile kuluva koguaja erinevus mitmekordselt.

Esimese päringuga läbiti tasapinnaline mudel kiiremini kui AO mudel. Teise päringuga läbiti AO mudelit kaks korda kiiremini kui tasapinnaline mudel ja kolmanda päringuga oli erinevus juba üheksakordne. Tähtis on mõõtmistulemuste juures, et iga päring läbis

nii AO mudelis kui tasapinnalises mudelis sama pikkusega jäljed, mida näitab ka tulpdiagramm. Mudeli läbimisel kasutas päring alati lühimat teed. See tähendab, et mudelite läbimise kiiruse erinevus ei tulnud läbitud tee pikkusest, sest see oli nii AO mudelil kui ka tasapinnalisel mudelil sama. Vahe tuli aspekt-orienteerid modelleerimise eelisest tavalise modelleerimise ees. Graafik tõestab, et kui läbides iga päringuga mitut aspekti ja nõuandemudelit, jättes nende hulga samaks ning tõestes iga läbimisega ainult teede arvu, kasvab AO mudeli eelis tasapinnalise mudeli ees mitmekordselt.

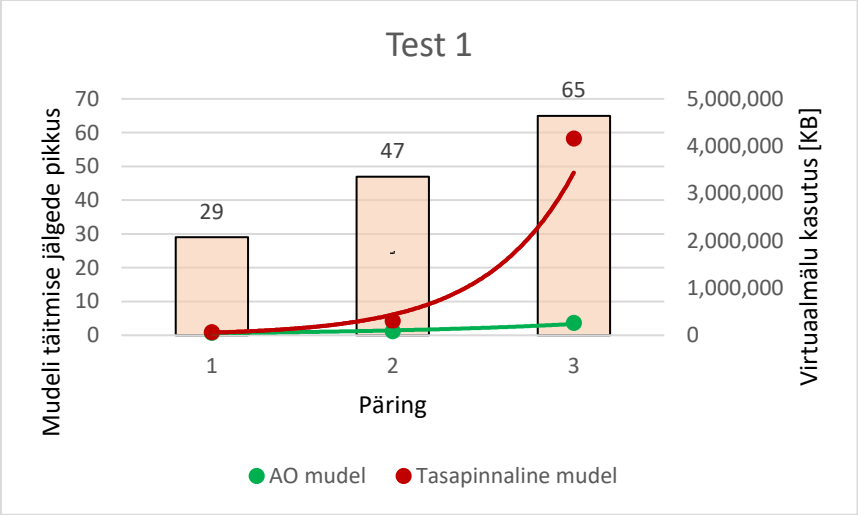


Joonis 8. Graafik kujutab mudelkontrollile kuluva koguaja muutust konstantse hulga aspektide/ nõuandemudelite läbimisel ja mudelite teede arvu suurendamisel.

Tegemist on kombineeritud diagrammiga, kus esinevad koos tulpdiagramm ja joondiagrammid (joonis 9). Tegemist on sama testiga, aga graafiku ühel y-teljel on mudelkontrollile kuluva koguaja asemel esitatud teine mõõdetud näitaja, milleks on virtuaalmälu kasutus. Iga päringuga kasvab AO mudeli ja tasapinnalise mudeli ressursikasutuse vahe.

Kui algul kasutab tasapinnaline mudel umbes 18 000 KB virtuaalmälu rohkem kui AO mudel, siis teise päringuga kasvab vahe peaaegu viiekordseks. Teise päringu indeks i ulatust suurendati ainult ühe võrra ja isegi ainult sellest piisas, et näha AO mudeli eelist. Kolmanda päringuga suurendati indeks i ulatust veel ühe võrra. Graafikul on näha, et kui AO mudeli virtuaalmälu kasutus läks umbes 4 korda suuremaks, siis tasapinnaline mudel kasutas peaaegu 14 korda rohkem virtuaalmälu kui teises päringus. AO mudeli ja tasapinnalise mudeli vahe joonisel on 16-kordne. AO mudeli kolmanda päringu tulemused oli paremad kui tasapinnalise mudeli teise päringu omad. AO mudelil jäi umbes 40 000 KB puudu, et ületada tasapinnalise mudeli teise päringu virtuaalmälu

kasutust. See tähendab, et isegi pikemat jälgede täitmise teed läbides kasutas AO mudel vähem virtuaalmälu kui tasapinnaline mudel lühema tee puhul. Graafik tõestab AO mudeli tugevat eelist ka virtuaalmälu kasutuse osas, kui tõsta iga läbimisega mudelite teede arvu ning jätta aspektide ja nõuandemudelite hulk samaks.



Joonis 9. Graafik kujutab virtuaalmälu kasutuse tulemusi konstantse hulga aspektide/ nõuandemudelite läbimisel ja mudelite teede arvu suurendamisel.

Esimese testi puhul pidas oodatu paika. Mõlema näitaja osas on näha AO mudeli selget eelist tasapinnalise mudeli ees.

4.2 Teise katseseeria järeldused

Teise katseseeria päringute puhul uuriti AO mudeli ja tasapinnalise mudeli tulemuste muutust, suurendades iga päringuga aspektide ja nõuandemudelite hulka, aga läbides alati sama hulk mudeli teid. Testimisel jooksutati kolme päringut, kus aspektide katvusindeks i vahemik oli nullist kaheni, aga põimpunktide indeks k vahemikku suurendati iga päringuga ühe võrra. Esimese päringu puhul oli k null (üks põimpunkt). Teise katse hüpoteesiks oli, et põimpunktide arvu suurenedes kasvab AO mudeli edu tasapinnalise mudeliga võrreldes. Katses suurendati AO mudeli aspektide arvu ja tasapinnalise mudeli nõuandemudelite arvu.

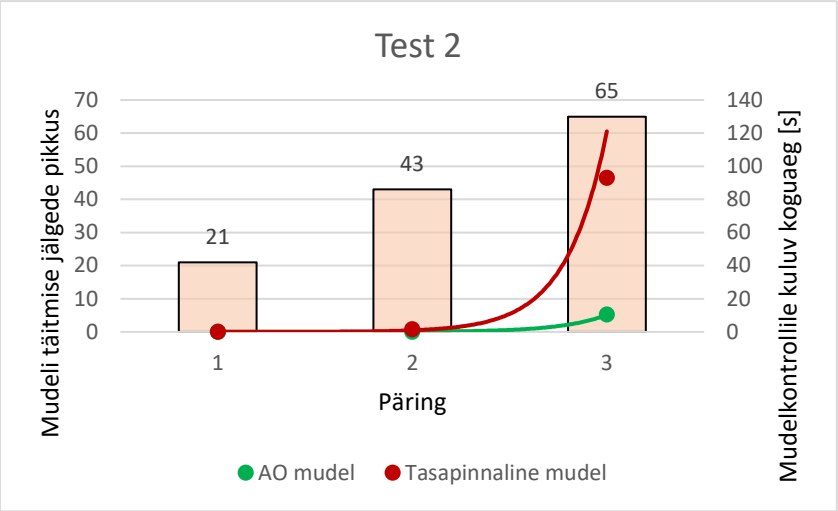
Tulemused on esitatud kombineeritud diagrammi kujul (joonis 10). X-telg näitab, mitmenda päringuga on tegemist, vasakpoolsel y-teljel on mudeli täitmise jälgede pikkus ja parempoolsel teljel on mudelkontrollile kuluv koguaeg. Rohelised punktid kujutavad taas AO mudeli läbimiseks kulutatud aja muutust päringu kaupa. Punased punktid

näitavad tasapinnalise mudeli läbimiseks kulunud aja muutust päringute haaval. Mõlema katse korral oli lühim jälg ühepikkune. Seda kujutab graafikus olev tulpdiagramm.

Esimese päringuga läbiti AO mudeli ühe aspekti kolm esimest teed ja tasapinnalise mudeli esimese nõuandemudeli kolm esimest teed. AO mudel läbiti 0,002 sekundiga ja tasapinnaline mudel 0,004 sekundi võrra aeglasemalt. Esimese päringu puhul suurt ajalist vahet märgata polnud. Mõlemad mudelid läbiti väga kiirelt.

Teise päringu juures läbiti AO mudelis kaks aspekti ja tasapinnalises mudelis kaks nõuandemudelit. Mõlema mudeli mudelkontrollile kuluv koguaeg suurenes, aga AO mudeli aeg suurenes märksa vähem kui tasapinnalises mudelis. AO mudel oli 12 korda kiirem.

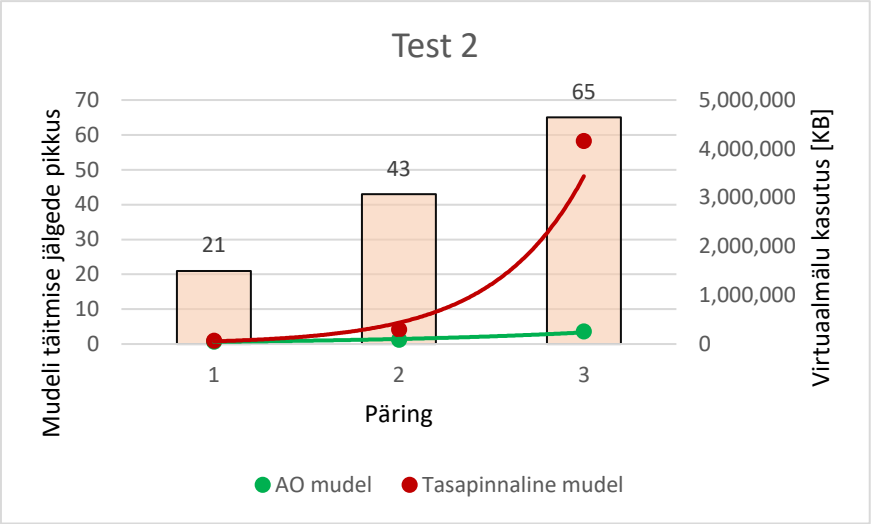
Viimasel päringul kulus taas mõlema mudeli läbimiseks kauem aega, aga ajaline vahe vähenes. Tasapinnaline mudel oli 12 korra asemel üheksa korda aeglasem. Tasapinnalise mudeli läbimiseks kulus umbes 1 minut ja 30 sekundit, kui AO mudeli läbimiseks kulus umbes 10 sekundit. Mõlema mudeli punktide põhjal loodi eksponentsiaalne trendijoon, et paremini iseloomustada mudelite läbimise kiirust aspektide lisamisel. Taaskord on tähtis esile tuua, et iga päringu juures oli mudelite täitmise jälgede pikkus sama ehk ajaline vahe ei saanud tulla tee pikkuse erisusest, sest mudelid läbisid alati sama pika tee. Graafiku põhjal saab öelda, et suurendades üheaegselt samavõrra aspektide hulka AO mudelis ja nõuandemudelite hulka tasapinnalises mudelis, on AO mudeli läbimisel tugev ajaline eelis tasapinnalises mudeli ees. Peale teist päringut jällegi viitab trend AO mudeli eelise vähenemisele, kui samaaegselt põimitud aspektide arv suureneb. Selle seose täpsemaks kvantitatiivseks analüüsiks oleks vaja edasisi katseid. Uurimisel piirduti selle testi puhul kolme päringuga, sest edasine teede hulga suurendamine katkestas mõlemas mudelis päringu ära virtuaalmälu ületamise tõttu (Lisa 7).



Joonis 10. Graafik kujutab mudelkontrollile kuluva koguaja muutust konstantse arvu mudelite teede läbimisel ja aspektide/ nõuandemudelite hulga suurendamisel.

Teise kombinatsiooni graafiku (joonis 11) kaks telge näitavad samu väärtusi, mis eelmises graafikus. Parempoolne y-telg näitab virtuaalmälu kasutuse väärtuste vahemikku. Päringud olid samad mis eelmise graafiku puhul.

Esimese päringu tulemusena kasutasid mõlemad mudelid enam-vähem sama palju virtuaalmälu: AO mudeli kasutus oli 42 373 KB ja tasapinnalise mudeli kasutus umbes 15 000 KB võrra suurem. Teise päringuga tõusis AO mudeli virtuaalmälu kasutus umbes 3 500 KB võrra, mis on ikka veel väiksem kui tasapinnalise mudeli esimese päringu kasutus. Tasapinnalises mudelis tõusis teise päringuga kasutus peaaegu kolmekordselt ning võrreldes AO mudeliga oli kasutuse vahe rohkem kui kolm korda. Kolmanda päringuga oli ka AO mudelil näha suuremat kasutus tõusu. Näitaja suurenes 45 872 KB-lt 260 748 KB-le, mis on umbes viiekordne tõus, kui suurendati aspekti hulka 1 võrra. Võrreldes seda tasapinnalise mudeli virtuaalmälu kasutus tõusuga on AO mudeli kasutus jällegi suhteliselt väike, sest tasapinnalise mudeli kasutus tõusis kolmanda päringuga peaaegu 27 korda. Kolmanda päringu AO mudeli ja tasapinnalise mudeli kasutuse vahe oli 16-kordne, kui teise päringu puhul oli see kolmekordne. Nii need andmed kui ka graafiku punktide põhjal genereeritud trendijoon tõestavad, et aspektide ja nõuandemudelite hulga samaaegsel suurendamisel kasvab AO mudeli virtuaalmälu kasutuse tõhusus eksponentsiaalselt.



Joonis 11. Graafik kujutab virtuaalmälu kasutuse tulemusi konstantse arvu mudelite teede läbimisel ja aspekti/ nõuandemudelite hulga suurendamisel.

Test kahe põhjal saab väita, et testieelne hüpotees leidis katselise kinnituse. Suurendades aspektide ja nõuandemudelite hulka ning jättes mudelite läbimisel teede arvu samaks, suureneb AO mudeli eelis virtuaalmälu kasutuse osas tasapinnalise mudeli ees. Mudelkontrollile kuluva koguaja osas oli AO mudelil samuti kindel eelis, kuid testis tehtud päringute hulk ei võimaldanud öelda, kas aspektide ja nõuandemudelite hulga edasisel suurendamisel kasvab või kahaneb AO mudeli eelis tasapinnalise mudeli ees.

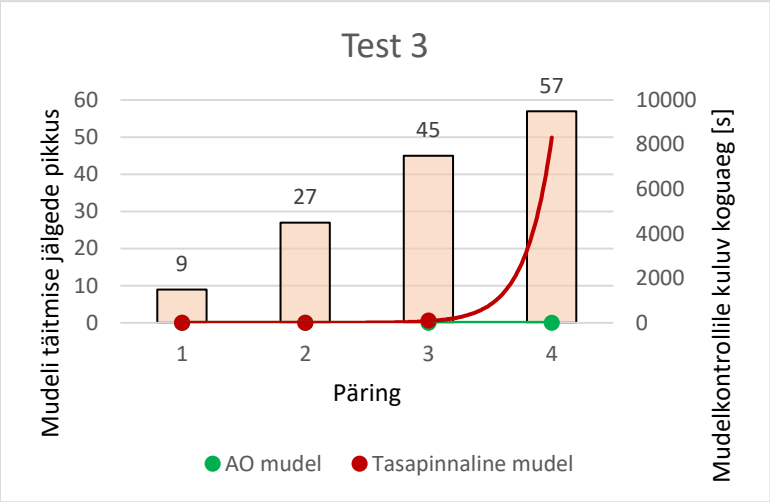
4.3 Kolmanda katseseeria järeldused

Kolmanda testiga mõõdeti kahe näitaja muutumist läbides pidevalt AO mudelil üht aspekti ja tasapinnalises mudelis esimest nõuandemudelit ning suurendades iga päringuga läbitavate teede arvu. Katse kolm sarnaneb mõtte poolest esimese testiga. Erinevuseks on see, et esimese testi puhul läbiti pidevalt mitut aspekti ja nõuandemudelit. Kolmas test erines päringute arvu poolest. Testides jooksutati neli päringut. Neljanda päringu vajadus tekkis, sest esimesed kolm päringut täideti nii kiiresti, et tulemustes ei olnud märgatavaid muutusi. Neljanda testi tulemuste põhjal loodeti kindlamaid järeldusi teha. Indeks k võrdus iga päringu puhul nulliga, indeks i ulatuspiiri tõsteti iga päringuga. Ootused olid, et AO mudeli edu tasapinnalise mudeli ees suureneb teede arvu tõstmisel.

Taas on graafik esitatud kombineeritud diagrammina (joonis 12). X-telg näitab, mitmenda päringuga on tegemist. Vasakpoolne y-telg näitab mudeli täitmise jälgede pikkuse väärtuste vahemikku ja parempoolne y-telg mudelkontrollile kuluva koguaja väärtuste

vahemikku. Rohelised punktid kujutavad taas AO mudeli läbimiseks kulutatud aja muutust päringu kaupa. Punased punktid näitavad tasapinnalise mudeli läbimiseks kulunud aja muutust päringute haaval. Mõlemad mudelid läbisid iga päringuga sama pikad jäljed. Jälje pikkust kujutab graafikus olev tulpdiagramm.

Esimesel päringul võrdus indeks i nulliga. AO mudelil läbiti ühe aspekti esimene tee ja tasapinnalises mudelis esimese nõuandemudeli esimene tee. AO mudeli läbimiseks kulus 0,001 s. Tasapinnalise mudeli läbimiseks läks 0,008 s kauem. Vahe on minimaalne ja AO mudeli eelis peaaegu olematu. Teise päringuga lisati mudelite läbimismarsruudile kolm teed juurde. Mõlemad mudelid läbisid esimest nelja teed. AO mudeli läbimise aeg oli 0,021 s ja tasapinnaline mudel läbiti 0,488 s. Tasapinnalises mudelis läks teise päringuga oluliselt kauem aega kui esimese päringuga. AO mudeli läbimisele kuluv koguaeg oli 23 korda kiirem kui tasapinnalises mudelis. Juba kolme tee lisamisega võib selgelt märgata, et AO mudeli läbimisele kuluv aeg kasvab aeglasemalt kui tasapinnalises mudelis. Päringu kolm puhul tõsteti läbitavate teede arvu taas kolme võrra. Kolmanda päringuga tõusis AO mudeli läbimise koguaeg kolmekordselt, mis on märgatavalt väiksem kui 21-kordne tõus, mis tekkis teise päringuga. Tasapinnalise mudeli hüppas 54-kordselt tõusult 202-kordsele tõusule, sest kolmanda päringuga kulus tema läbimiseks üle 1,5 minuti. AO mudeli ja tasapinnalise mudeli läbimise kiiruse vahe on rohkem kui 1000-kordne. Neljanda päringuga lisati marsruudile viimased kaks teed juurde. See päring pidi läbima AO mudeli ühe aspekti ja tasapinnalise mudeli nõuandemudeli kõik teed. Päringuga neli läbiti AO mudel kiiremini kui päringuga kaksa tasapinnaline mudel. AO mudelis läbiti kõik ehk üheksa teed kiiremini kui tasapinnalise mudeli neli teed, mis on alla poole. AO mudeli läbimiseks kuluv koguaeg tõusis 0,077 s-lt 0,102 s-le. Tasapinnalise mudeli puhul päring katkestati, sest virtuaalmälu kasutus ületati (Lisa 7). Päringu jooksutamise protsessi ajal oli näha, et mudeli läbimisele oli juba kulunud 138 s (Lisa 8). Selle põhjal on näha, et AO mudeli ja tasapinnalise mudeli vahe oli veelgi suurem kui eelmise päringu puhul. Protsess polnud lõppenud, nii et vahe oleks veelgi kasvanud. Tasapinnalises mudelis kuluva koguaja punktide põhjal genereeritud trendijoon on taaskord eksponentsiaalne. Trendijoon viitab, et koguaja kulu oleks olnud 8000 sekundi kandis ehk üle kahe tunni. Selle kindlaks tegemiseks oleks vaja teha edasisi katseid. Graafikul on näha AO mudeli tugevat ajalist edu tasapinnalise mudeli ees.

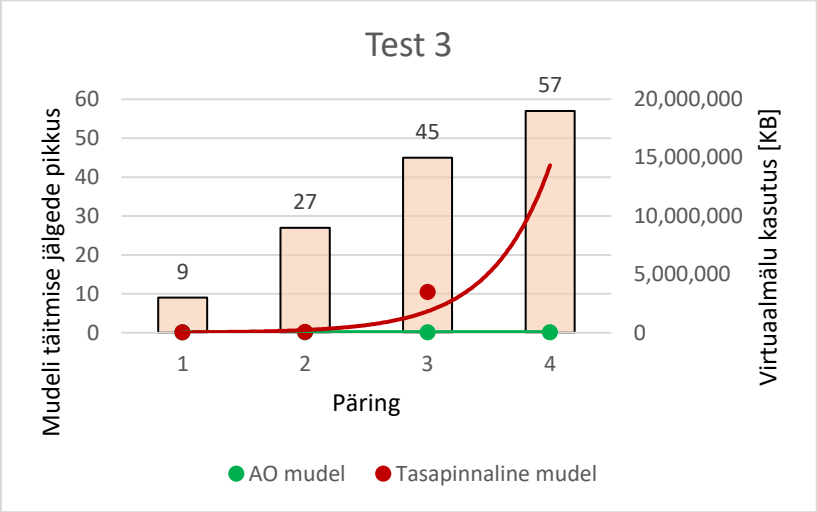


Joonis 12. Graafik kujutab mudelkontrollile kuluva koguaja muutust konstantselt ühe aspekti/ nõuandemudeli läbimisel ja mudelite teede arvu suurendamisel.

Graafik (joonis 13) on koostatud sarnasel põhimõttel ja samas stiilis kui eelnev graafik. Parempoolel y-teljel on näidatud virtuaalmälu kasutuse väärtuste vahemikku, teised teljed kujutatavad samade näitajate väärtuste vahemikku kui eelnev graafik.

Esimese päringuga kasutas tasapinnaline mudel 57 236 KB ja AO mudel umbes 17 000 KB võrra vähem. Väike vahe kahel mudelil oli, aga suures plaanis kasutasid mõlemad efektiivselt virtuaalmälu. Teise päringuga mõlema mudeli virtuaalmälu kasutus suurenes. Tasapinnalise mudeli läbimise mälukasutus suurened rohkem kui AO mudelil. Vahe on 22 000 KB, mis näitab, et AO mudel kasutus on natuke efektiivsem, aga jällegi pole näha märgatavat eelist tasapinnalise mudeli ees. Pärast kolmanda päringu jooksutamist jäi AO mudeli mälukasutus põhimõtteliselt samaks, aga tasapinnalise mudeli virtuaalmälu kasutus tõusis 55-kordselt. AO mudeliga suudeti esimese kolme päringuga virtuaalmälu kasutust suhteliselt samana hoida. Tasapinnalisel mudelil tekkis suur vahe võrreldes AO mudeliga, kui mõlemad mudelid pidid läbima seitse teed. AO mudeli ja tasapinnalise mudeli virtuaalmälu kasutuse vahe kolmanda päringu puhul oli 84-kordne. Neljanda päringuga samuti ei tõusnud AO mudeli mälu kasutus väga palju. Selle päringu puhul on AO mudeli mälukasutus parem kui tasapinnalisel mudelil esimese päringu puhul. See tähendab, et AO mudeliga suudetakse ühe aspekti kõik teed efektiivsema mälukasutusega läbida kui tasapinnaline mudeliga esimese nõuandemudeli üks tee. Tasapinnalise mudeliga neljas päring ei õnnestunud (Lisa 7). Trendijoon näitab, et neljanda päringuga oleks mälukasutus olnud 15 000 000 KB ehk 15 GB kandis, mis oleks tähendanud 355-kordselt efektiivsemat kasutust AO mudeli puhul. Jällegi oleks selle kindlaks tegemiseks

vaja teha edasisi katseid. Graafikul on näha kindlat virtuaalmälu kasutus eelist tasapinnalise mudeli ees teede arvu suurendamisel.



Joonis 13. Graafikuna virtuaalmälu kasutus tulemused konstantse ühe aspekti/ nõuandemudeli läbimisel mudelite teede arvu suurendades.

Test kolm tõestas oodatut. AO mudelil oli tugev eelis tasapinnalise mudeli ees. Testi kolm näitajate põhjal tuli teiste katsetega võrreldes eriti hästi esile AO mudeli ajaline kiirus ja mälu kasutuse efektiivsus.

4.4 Järelduste sidumine tarkvaraarenduse maailmaga

Kõikide katsete analüüside järeldused tõestasid üldist AO mudeli eelist tasapinnalise mudeli ees. Aspekt-orienteeritud modelleerimine suudab kiiremini ja tõhusamalt aspektidega tegeleda kui tasapinnaline mudel. Sama loogika käib üldiselt aspekt-orienteeritud arendamise kohta. Aspekt-orienteeritus suudab paremini aspektidega arvestada. Rohkemate aspektidega süsteemis tuleb AO eelis tugevamalt esile. AO meetodiga on võimalik kiiremini ja efektiivsemalt paljude aspektidega süsteeme käsitleda, võrreldes objekt-orienteeritud ja protseduurilise meetodiga. Hästi tuli mudelitest testide genereerimisel esile aspekt-orienteeritud modelleerimise võimekus tagada disaini läbivate aspektide lahusus, mis soodustab kerget hooldatavust ja paindlikku arendamist. Veel üheks eeliseks, mis testidega tõestati, on AO mudelite ajalised ja efektiivse mälu kasutuse eelised tavamudelite ees nii testimudelite spetsifitseerimisel, kui ka AO testi kattekriteeriumite esitamisel ja testide genereerimisel. See võimaldab genereerida teste suurematele süsteemidele, säilitades testimise efektiivust.

5 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli tõestada aspekt-orienteeritud mudelite eelise mitte aspekt-orienteeritud mudelite ees ja testide genereerimise keerukuse sõltuvus erinevatest aspekt-orienteeritud testi kattekriteeriumidest.

Esimeses peatükis tutvustati teemat ning esitati bakalaureusetöö eesmärk. Järgnes metoodika, kus selgitati aspekt-orienteeritusega kaasnevaid termineid ning tutvustati uuringu läbiviimiseks kasutatud tarkvara. Kolmandas osas kirjeldati detailselt kõiki katsetes kasutatud mudeleid ja toodi esile katsetulemused. Viidi läbi kolm katseseeriat. Esimeses kahes kasutati testijadade genereerimiseks kolme mudelkontrolli päringut ja viimases nelja päringut. Iga päringu täitmisel kasutas mudeli olekuruumi sügavuti otsingu strateegiat, mis leidis alati lühima kriteeriume rahuldava testijada.

Põhiosa lõpus tehti saadud tulemuste põhjal analüüs, mida esitati kombineeritud diagrammide kujul. Iga testi puhul kirjeldas üks graafik mudelite mudelkontrollile kuluva koguaja sõltuvust kattekriteeriumist ja teine graafik kirjeldas mudelite virtuaalmälu kasutuse muutust päringute kaupa. Analüüsi põhjal jõuti järeldusele, mis lõpuks seoti üldiste arengutega tarkvaramaailmas.

Lõputöö tulemusena tõestati hüpoteesi, et suurema hulga aspektidega süsteemis on AO mudelil suurem eelis virtuaalmälu kasutuses ja mudeli läbimise kiiruses tasapinnalise mudeli ees. Enamus teste tõestas AO mudeli eelise suurenemist mitte-AO mudeli ees, kui suurendada aspektide arvu. Üks test näitas eelise vähenemist aspektide arvu suurenedes. Selle seose täpsemaks kvantitatiivseks analüüsiks oleks vaja edasisi katseid. Üldise järeldusena saab öelda, et aspekt-orienteeritud modelleerimine suudab kiiremini ja tõhusamalt aspektidega tegeleda kui tasapinnaline mudel. Sama järeldust saab laiendada aspekt-orienteeritusele üldises tarkvaraarenduse kontekstis, kus aspekt-orienteeritus soodustab kergemat hooldatavust ja paindlikumat arendamist.

Kasutatud kirjandus

[1] Kiczales, et al., Aspect-Oriented Programming, [Võrgumaterjal]. 1997. Available: https://www.cs.ubc.ca/~gregor/papers/kiczales-ECOOP1997-AOP.pdf, [Kasutatud 27.03.2021]

[2] Tallinna Tehnikaülikool, Java objektorienteeritud programmeerimise kontsept, [Võrgumaterjal]. Available: https://ained.ttu.ee/javadoc/oop/oop-general.html, [Kasutatud 27.03.2021]

[3] Packt, What are cross-cutting concerns?, [Võrgumaterjal]. Available: https://subscription.packtpub.com/book/application_development/9781788299459/6/ch06lvl1sec34/what-are-cross-cutting-concern, [Kasutatud 29.03.2021]

[4] Mishar, D., Cross-cutting Concern, [Võrgumaterjal], 2019. Available: https://medium.com/anatta-design/cross-cutting-concern-aadf4f51a5c1, [Kasutataud 29.03.2021]

[5] Vain, et al., Aspect-Oriented Model-Based Testing with UPPAAL Timed Automata, 2018.

[6] Iqbal, Allen, Representing Aspects in Design, [Võrgumaterjal], 2009. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5198527, [Kasutatud: 04.04.2021]

[7] Rashid, Aspect-Oriented Database Systems, [Võrgumaterjal], 2004. Available: https://archive.org/details/springer_10.1007-978-3-662-05851-0/page/n19/mode/2up, [Kasutatud: 07.04.2021]

[8] Choudhary, Managing Cross Cutting Concerns - The Logger and Logging, [Võrgumaterjal], 2020. https://www.c-sharpcorner.com/UploadFile/vendettamit/managing-cross-cutting-concerns-the-logger-and-logging/, [Kasutatud: 07.04.2021]

[9] Vain, et al., On the Benefits of Using Aspect-Orientation in UPPAAL Timed Automata, 2017.

[10] Uppsala Ülikool, Aalborgi Ülikool, [Võrgumaterjal]. Available: https://uppaal.org/, [Kasutatud: 10.04.2021]

[11] Sarna, Aspect-Oriented Model-Based Testing, 2018.

[12] Memon, Advances in Computers, Volume 91, [Võrgumaterjal], 2013. Available: https://www.sciencedirect.com/topics/computer-science/cross-cutting-concern, [Kasutatud: 11.04.2021]

[13] Intel, Peak virtual memory, [Võrgumaterjal]. Available https://www.intel.com/content/www/us/en/programmable/quartushelp/13.0/mergedProjects/msgs/msgs.htm#msgs/iqexe_end_peak_vsize_memory.htm [Kasutatud: 15.04.2021]

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

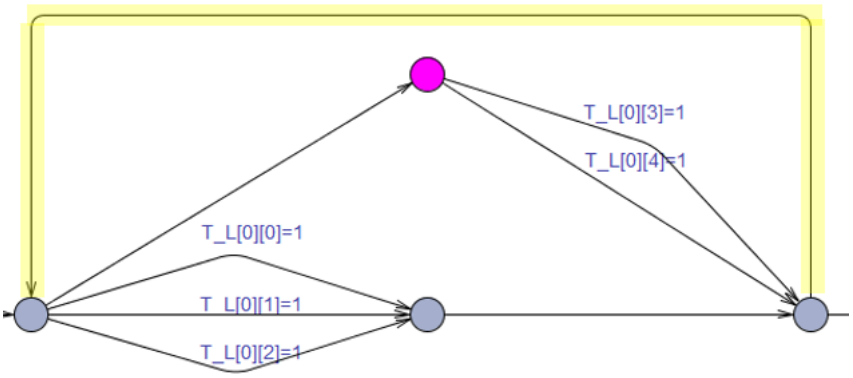
Mina, Johannes Olaf Kurss

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Aspekt-orienteeritud testimise efektiivsusanalüüs“, mille juhendaja on Jüri Vain
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

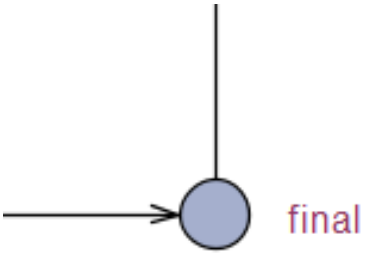
18.05.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

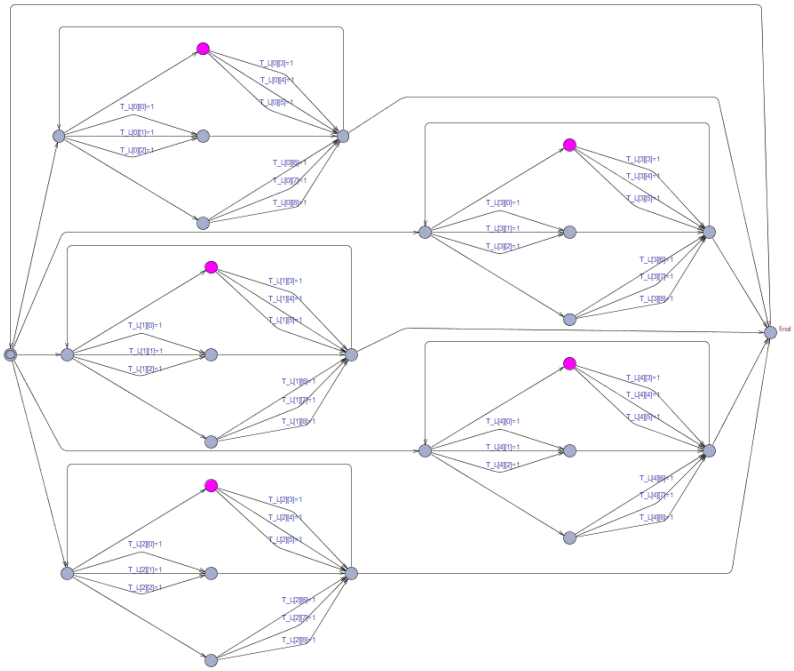
Lisa 2 – Nõuandemudeli tagastav kaar

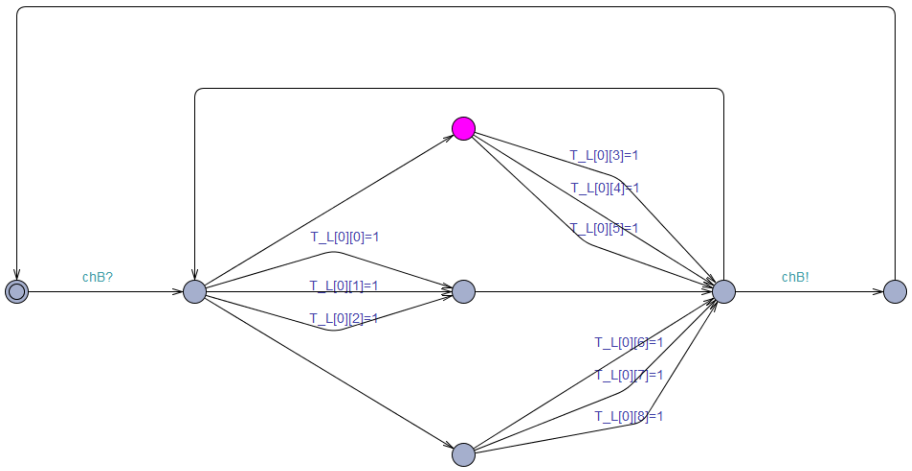


Lisa 3 – Baasmudelite lõppasur

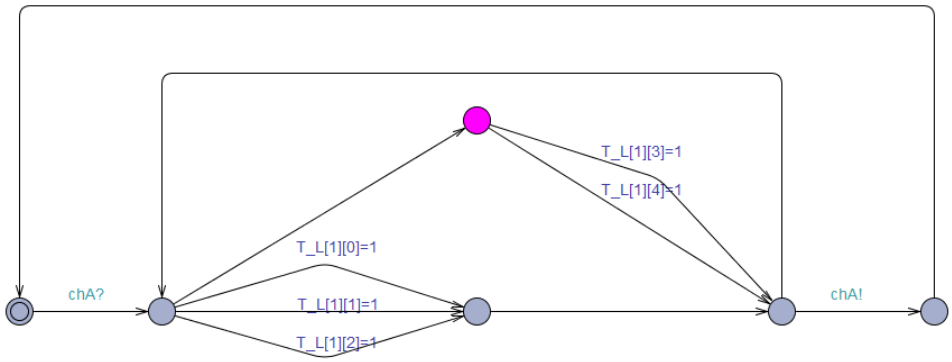


Lisa 4 – Aspekt mudel ja tasapinnaline mudel lisatud teedega

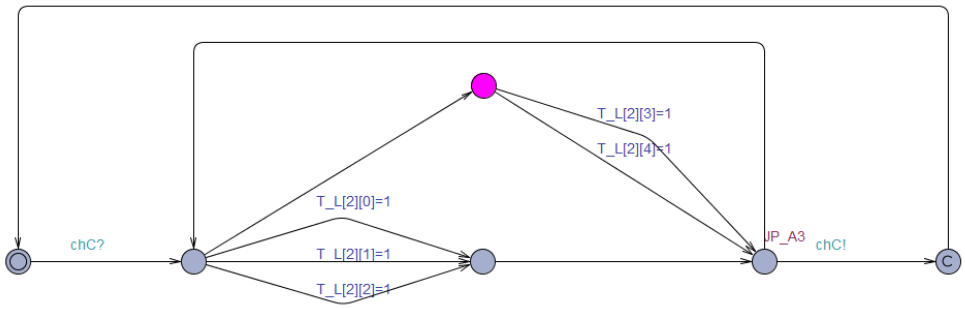




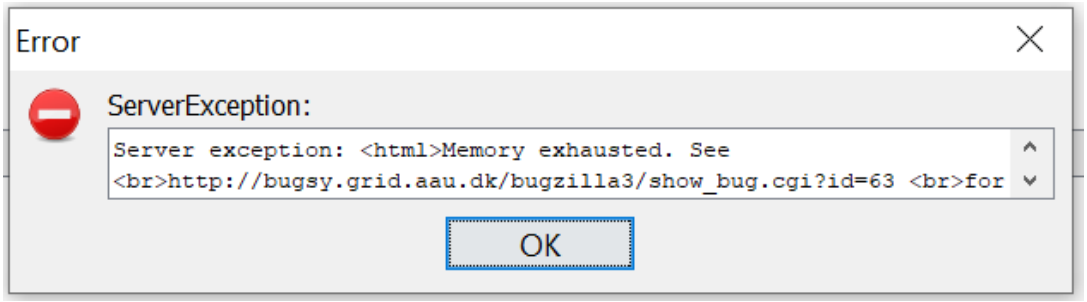
Lisa 5 – Kahe aspektiga AO mudelile lisatud aspekt



Lisa 6 – Kolme aspektiga AO mudelile lisatud aspekt



Lisa 7 – Veateade tasapinnalise mudeli läbimisel



Lisa 8 – Tasapinnalise mudeli esimese nõuandemudeli kõikide teede läbimise päringu jooksutamise protsessi näitajad