TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Ege Käosaar 142269IABB

LÄHTEKOODI HALBADE LÕHNADE ÜLDISTAMINE SÜSTEEMIANALÜÜSI MUDELITELE

Bakalaureusetöö

Juhendaja: Erki Eessaar
Doktor

Tallinn 2017

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: Ege Käosaar

22.05.2017

Annotatsioon

Puhta koodi põhimõtted keskenduvad lähtekoodi võimalikult kerge loetavuse ja muudetavuse tagamisele. Kuna lähtekoodi peavad suure tõenäosusega lugema ja muutma inimesed, kes pole selle autorid või koodi unustanud autor ise, tuleb puhas kood tarkvara elutsüklis kasuks lähtekoodi pidevale arengule ja täiustamisele. Kõrvalekalletele puhtast koodist ehk potensiaalsetele probleemidele viitavad lähtekoodi halvad lõhnad. Halva lõhna põhjuseks olevate probleemide refaktoreerimine võimaldab vähendada tehnilist võlga tulevikus.

Sarnaselt lähtekoodiga peaksid ka süsteemiarenduse käigus tekkivad mudelid olema lihtsalt loetavad ja muudetavad. Halvasti loetavad mudelid võivad pärssida infosüsteemide loomist, viia vigase tarkvara või infosüsteemi halva toimimiseni. Raskelt muudetavad mudelid võivad kiiresti aeguda ning nendest pole seljuhul enam tarkvara hooldusel kasu. Kuna töö kirjutamise aja (2017. aasta kevad) seisuga on puhaste mudelite teemat ebapiisavalt uuritud, siis selgitatakse seda mõistet käesolevas lõputöös. Lisaks üldistatakse raamatus [35] väljatoodud lähtekoodi halvad lõhnad infosüsteemide analüüsi mudelitele ja modelleerimisprotsessile, antakse juhiseid mudelite refaktoreerimiseks ning tuuakse konkreetseid näiteid, mis lihtsustab mudeli halbade lõhnade tuvastamist. Lõpuks analüüsitakse juhuslikult valitud 30-t süsteemianalüüsi dokumenti, et saada teada, kas ja kui palju leidub neis mudelite halbu lõhnu. Need dokumendid on loodud üliõpilaste poolt TTÜ õppeaines „Andmebaasid I“ ühel ja samal õppeaastal ning on saanud arvestuse.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 47 leheküljel, 6 peatükki, 15 joonist, 4 tabelit.

Abstract

Generalizing Code Smells to System Analysis Models

Clean code is a popular set of principles that aim to make source code as easy to read and modify as possible. There are usually multiple people involved in the software development process. Source code will probably be read and modified by people other than the original author or by the original author who no longer remembers the code. Thus, easily readable and modifiable source code will become an invaluable asset in ensuring software's continuous improvement. The easier it is to modify source code, the longer it will be usable and relevant. Deviation from clean code aka potential problems can be detected based on code smells. Being able to successfully detect code smells and refactor out their underlying problems, helps minimize technical debt in the future.

Similarly to source code, the models created during the systems development life cycle should be easy to read and modify. Models that are difficult to read and comprehend can suppress software development or lead to faulty bug-filled systems. Models that are difficult to modify can quickly become obsolete and therefore will not be of any help during software maintenance.

Since the notion of clean models has currently (as of spring 2017) not been sufficiently explored in the context of software development, this thesis aims to do so. Model smells have been described in previous research papers. However, those descriptions are mostly limited to class diagrams (with some exceptions). Hence, this thesis will also analyze code smells written in the book [35] and try to generalize them into model smells that can apply to many commonly used system analysis models. Every model smell will be accompanied with ways to refactor the model that contains them and specific examples (textual and/or visual), which will help detect them. In addition, there will be a review based on 30 randomly chosen system analysis documents of students from the TUT course „Databases I“ to find out if and how many model smells they contain. The documents are course projects of the same year and have all been accepted.

The thesis is in Estonian and contains 47 pages of text, 6 chapters, 15 figures, 4 tables.

Lühendite ja mõistete sõnastik

Agiilne arendus	Hulk põhimõtteid, mis keskenduvad infosüsteemide kiirele arendamisele ja muutustega kohanemisele. Põhimõtted pärinevad „*Manifesto for Agile Software Development*“ manifestist. [34] Agiilsuse põhimõtteid jälgivaid arendusmetoodikaid kutsutakse eesti keeles ka paindmetoodikateks.
Antimuster	Antimuster on levinud halb lahendus mingisugusele probleemile, mis on kirja pandud arusaamist ja teiste lahendustega võrdlemist soodustavas struktureeritud formaadis [3].
CASE vahend	*Computer-Aided Software Engineering tool* Automatiseeritud vahend, mis toetab arendusprotsessi, näiteks süsteemi disainimisel [46].
Ebapuhas mudel/kood	Lähtekood/mudel, mis ei järgi täiuslikult puhta koodi/puhaste mudelite põhimõtteid. Teiste sõnadega, neis esineb vähemalt üks halb lõhn, mis osutab mingile lähtekoodi/mudeli probleemile.
Heuristika	Kiire ja loov lahendus probleemile, millele tavameetodid ei suuda lahendust pakkuda [26]. Lahendus ei pruugi olla täiuslik, kuid selle abil saab vähemasti enamikel juhtudel probleem lahendatud.
Mudel	Mudel on (tarkvara)süsteemi abstraktne ja lihtsustatud kujutis [40].
Mudelielement	Mudelielement on mudeli alamosa, näiteks diagramm või element diagrammil.
Mõõdik	Ka meetrika. Mõõdetav näitaja koodis või mudelites, mille põhjal on võimalik seda mingites aspektides – näiteks jõudlus, otstarbekus, korrektsus, keerukus – hinnata [25].
OCL	*Object Constraint Language*, objektikitsenduse keel Osa UML standardist, mille abil saab täiendada UML mudeleid [23].
Refaktoreerimine	Tehise (näiteks lähtekood/mudel) kvaliteedi parandamine, ilma selle funktsionaalsust (mudelite puhul edastatavat infot) muutmata.
TTÜ	*TUT, Tallinn University of Technology*. Tallinna Tehnikaülikool
UML	*Unified Modeling Language*, unifitseeritud modelleerimiskeel

Sisukord

1 Sissejuhatus 9
1.1 Taust ja probleem 9
1.2 Ülesande püstitus 10
1.3 Metoodika 10
1.4 Ülevaade tööst 11
2 Mudelite kvaliteet 12
2.1 Süntaks, semantika ja pragmaatika 12
2.2 *6C* kvaliteedieesmärgid 13
2.3 Tehniline võlg 14
3 Puhtad mudelid 15
3.1 Mudeli halb lõhn 16
3.2 Mudelite valideerimine 17
3.2.1 Käsitsi valideerimine 17
3.2.2 Automaatne valideerimine 18
3.3 Mudelite refaktoreerimine 19
4 Kataloog 21
4.1 Mudelite üldised halvad lõhnad 22
4.2 Stiili halvad lõhnad 37
4.3 Nimede halvad lõhnad 43
4.4 Mudelite valideerimiste halvad lõhnad 45
4.5 Kataloogi loomise protsess 48
4.6 Tulemuste analüüs 49
4.7 Visioon mudelite halbade lõhnade otsimise ja refaktoreerimise automatiseerimisest 52
5 Andmebaasid I aineprojektide analüüs 53
6 Kokkuvõte 55
Kasutatud kirjandus 56
Lisa 1 – Mudelite halbade lõhnade esinemine aineprojektides 59

Jooniste loetelu

Joonis 1. Üleliigsed kommentaarid [28]. 23
Joonis 2. Üleliigsed kommentaarid on eemaldatud. 23
Joonis 3. Pikk ja lohisev lause [27]. 24
Joonis 4. Kergemini arusaadavad laused. 24
Joonis 5. Kordused kasutusjuhtudes [45]. 26
Joonis 6. Kordused on eemaldatud. 26
Joonis 7. Kasutusjuhtude ebaloogiline järjekord [2]. 33
Joonis 8. Kasutusjuhud on järjestatud kellaosuti suunas. 34
Joonis 9. Liiga mahukas diagramm [11]. 38
Joonis 10. Paremini hoomatav tükk algsest diagrammist, mis kujutab põhiolemitüüpi ja selle seoseid. 39
Joonis 11. Liiga pikad jooned [30]. 40
Joonis 12. Parandatud joonepikkused. 40
Joonis 13. Üldistusseos ei vasta modelleerimistavadele [10]. 42
Joonis 14. Üldistusseos vastab modelleerimistavadele. 42
Joonis 15. Erinevat tüüpi mudelite halbade lõhnade arv igas projektis. 60

Tabelite loetelu

Tabel 1. Spetsiifilisemate mudelite halbade lõhnade esinemisvõimalikkus..... 50
Tabel 2. Kataloogi tulemite võrdlemine varasemate uuringutega. 51
Tabel 3. Mudeli halbade lõhnade esinemissagedus. 53
Tabel 4. Kõikide mudelite halbade lõhnade esinemissagedus. 59

1 Sissejuhatus

Infosüsteemide arenduses ja selle alamosaks olevas tarkvaraarenduses on üheks oluliseks etapiks süsteemianalüüs, mille eesmärgiks on võimalikult täpselt aru saada planeeritava süsteemi olemusest ja kasutajale nähtavast oodatavast käitumisest. Põhjalikult läbimõeldud nõuded ja nende esitus mudelitena annavad suure panuse infosüsteemi tõrgeteta toimimisse, mistõttu on oluline, et need oleksid lihtsalt ja üheselt arusaadavad.

Puhas kood (*clean code*) [35] on tarkvaraarenduses kasutatav levinud põhimõtete kogum. Lühidalt tähendab see, et lähtekood on lihtsalt ja üheselt arusaadav ning seetõttu kergemini täiendatav ja muudetav. Modelleerimise efektiivsuse ja viljakuse tagamiseks peaksid ka mudelid järgima sarnaseid põhimõtteid.

Käesoleva bakalaraureusetöö lahendatavaks probleemiks on poolikud juhised puhaste mudelite loomiseks ja mudelite üleüldine halb loetavus. Mudelid, millest ainult autor ise aru saab, kui sedagi, on paremal juhul lihtsalt kasutud ja halvemal juhul takistuseks töötava tarkvara loomisele.

1.1 Taust ja probleem

Puhas kood on tuntud tarkvaraarenduse põhimõtete kogum ning teemale vastavat informatsiooni leidub palju – Google Scholar keskkonnas 20. mail 2017 tehtud otsing „*clean code*“, „*software*“ tõi välja 2600 tulemust. Puhta koodi põhimõtete järgimine aitab kirjutada kergesti loetavat ja muudetavat lähtekoodi, millega suudavad töötada paljud programmeerijad peale autori. Tänu sellele on lähtekood jätkusuutlik ning püsib kauem töökorras.

Modelleerimises ei ole tänase seisuga sarnast ühtset puhaste mudelite loomise juhendit, mis oleks orienteeritud mudelite kergele loetavusele ja muudetavusele ning hõlmaks sealjuures mitmeid erinevaid mudelitüüpe. Google Scholar keskkonnas 20. mail 2017 tehtud otsing „*clean model*“, „*system analysis*“ tõi välja 27 tulemust. Kuna mudelid mängivad tarkvaraarenduses suurt rolli ning nendega peavad tegelema paljud inimesed

peale autori(te), peaksid need samuti olema võimalikult lihtsasti ja üheselt mõistetavad. Ka autor ise ei pruugi aja möödudes mäletada, miks ja mida ta oma kunagi loodud mudelites esitas ning vajab mälu värskendamiseks hästi loetavaid mudeleid.

Mida hilisemas faasis on tarkvaraarendusprotsess, seda keerulisem, kulukam ja aeganõudvam on vigade parandamine [43]. Seega on äärmiselt oluline tegeleda juba varakult võimalike tõrgete ennetamise ja avastamisega. Korrektsed ja hästi loetavad mudelid aitavad vigu kergemini avastada, mis omakorda võib tähendada suurt ressursside (raha, aeg, inimesed) kokkuhoidu.

Eelnev osutab vajadusele puhaste mudelite loomise juhendi järele, mis aitaks kaasa modelleerimisprotsessile ning annaks nõuandeid ebapuhaste mudelite refaktoreerimiseks.

1.2 Ülesande püstitus

Bakalaureusetöö peamisteks eesmärkideks on

- koostada kataloog analüüsimudelite halbadest lõhnadest, mis aitaks neid mudelites identifitseerida ja modelleerimisel vältida;
- tuletada juhised, mille abil ebapuhtaid mudeleid refaktoreerida ja tuua konkreetseid näiteid;
- leida ühe õppeaasta juhuslikult valitud 30-s TTÜ õppeaine „Andmebaasid I“ aineprojektis leiduvate halbade lõhnade esinemissagedus.

1.3 Metoodika

Püstitatud eesmärkide saavutamiseks on aluseks võetud Robert C. Martini raamatus [35] välja toodud koodi halvad lõhnad ehk võimalikud kõrvalekalded puhtast koodist, mille põhjal üldistatakse ja tuletatakse vastavad halvad lõhnad järgnevatele autori poolt ülikoolis õpitud mudelitüüpidele:

- kontseptuaalne andmemudel,
- valdkonnamudel ehk domeenimudel ehk kontseptuaalmudel,
- allsüsteeme ja nende seoseid kirjeldav mudel,
- jadadiagramm,

- kasutusjuhtude mudel,
- andmebaasioperatsioonide lepingud,
- tegevusdiagramm,
- olekudiagramm ehk seisundidiagramm ehk olekumasina mudel.

Samuti leitakse halvad lõhnad modelleerimisprotsessile üldiselt, näiteks mudelite valideerimistele.

Iga koodi halva lõhna korral üritatatakse leida selle lõhna tuumprobleem ning hinnata, kas see tuumprobleem võib mingil kujul esineda ka eelnevalt nimetatud mudelitüüpides või üldiselt modelleerimisprotsessis. Halbade lõhnade üldistamine baseerub autori ülikoolis õpitud teadmistel tarkvaraarenduses ja modelleerimises.

Lisaks mainitud raamatule [35] aitavad eesmärke saavutada ja nende õigsuses veenduda varem sarnastel teemadel tehtud uurimused ning levinud tavad ja harjumused modelleerimises. Metoodikat ja kataloogi tekkimisprotsessi on veel täpsemalt kirjeldatud töö praktilises osas.

1.4 Ülevaade tööst

Töö esimene osa andis kiire ülevaate lõputöö lahendatavatest probleemidest, taustast ja eesmärkidest. Töö teises osas kirjeldatakse mudelite olemasolevaid kvaliteedinõudeid, mis annavad sisendi mõiste “puhtad mudelid” selgitamiseks. Töö kolmandas osas kirjeldatakse puhtaid mudeleid (sealhulgas kirjeldatakse mõistet “mudeli halb lõhn”) ja teatud praktikaid, mida tuleks nende saavutamiseks rakendada. Töö neljandas osas on mudeli halbade lõhnade kataloog ning tabelid, mis kirjeldavad nende esinemist erinevates mudelitüüpides ja kokkulangevusi juba kirjanduses eelnevalt välja pakutud mudelite halbade lõhnadega. Töö viiendas osas esitatakse 30 „Andmebaasid I“ aineprojekti uurimise tulemusi. Uuring otsis vastust küsimusele, kas ja kui palju mudelite halbu lõhnu need sisaldavad.

2 Mudelite kvaliteet

Mudelipõhise tarkvaraarenduse kasvav populaarsus on suurendanud vajadust pöörata erilist tähelepanu mudelite kvaliteedile. Mudeleid kasutatakse üha enam lisaks süsteemi kirjeldusele ka süsteemi simuleerimisel, sellele testjuhtude ning osa või kogu lähtekoodi genereerimisel, seega on eriti oluline nende korrektsus. Kvaliteet mudelipõhises tarkvaraarenduses tähendab rohkemat kui ainult süntaksireeglite järgimist. See hõlmab laiemalt mudelite, modelleerimiskeelte, modelleerimisvahendite, modelleerimisprotsesside ja mudelite refaktoreerimise kvaliteeti. [40] Seetõttu keskendub ka käesolev lõputöö mudelite halbadele lõhnadele laiemas ja üldisemas tähenduses.

Mudelite kvaliteeti on tulenevalt mudelite erinevatest eesmärkidest defineeritud ja kirjeldatud mitmel erineval moel. Mida suuremat rolli mängivad mudelid süsteemi lähtekoodi loomisel, seda olulisem on mudeli kvaliteedinõuete järgimine. Kui lähtekood ja mudelid eksisteerivad koos, peavad mudelid olema korrektsed ja täielikud (mingi teatud piirini) ning neid peaks olema lihtne muuta, et tagada pidev vastavus koodiga. [40]

Järgnevalt kirjeldatakse kahte levinud mudelite kvaliteediraamistikku. Samuti antakse lühike ülevaade ebakvaliteetsete mudelite kasutamise võimalikust tagajärjest pikemas perspektiivis.

2.1 Süntaks, semantika ja pragmaatika

Üheks mudelite kvaliteediraamistikuks on mudelite hindamine süntaksi, semantika ja pragmaatika aspektidest lähtuvalt [31]. Raamistiku elemente saab kirjeldada järgnevalt:

- **Süntaks.** Eesmärgiks on tagada modelleerimiskeele süntaktiline korrektsus ehk vastavus keele süntaksireeglitele. Igal modelleerimiskeelel on omad süntaksireeglid, mis määravad keele elemendid ja kuidas neist elementidest panna kokku maailma kohta tõeseid väiteid esitavad laused. Süntaktiline viga on näiteks olukord, kui mudel sisaldab elementi, mida pole modelleerimiskeeles

defineeritud. [41] Süntaksivigu aitavad vältida CASE vahendid, mis ei luba koostada ebakorrektse süntaksiga mudeleid.

- **Semantika ehk tähendus.** Eesmärgiks on tagada mudelite korrektsus ja täielikkus. Korrektsus tähendab seda, et kõik mudeli esitatavad väited on tõesed ja asjakohased. Täielikkus tähendab seda, et mudel esitab süsteemi kohta kõiki olulisi tõeseid ja asjakohaseid väiteid. Raamistiku autorid möönavad sealjuures, et täielikku korrektsust ja täielikkust on raske saavutada, seega tuleks semantikale tähelepanu pöörata mõistlikkuse piirides. [41]
- **Pragmaatika.** Eesmärgiks on tagada mudelite arusaadavus ehk tagada mudeli õigesti tõlgendamine. Arusaadavus ei tähenda, et igaüks peaks aru saama mahuka mudeli igast väiksemast alamosast. Pragmaatika panustab sellesse, et mudelid oleksid lihtsamini mõistetavad, kasutades sealjuures näiteks mudelite inspekteerimist, visualiseerimist (graafilised mudelid tekstiliste mudelite asemel jne) ja filtreerimist (detailide peitmine jne). [41]

2.2 *6C* kvaliteedieesmärgid

Mudelite kvaliteeti on kirjeldatud ka *6C* mudeli kvaliteedieesmärkide (*Correctness, Completeness, Consistency, Comprehensibility, Confinement, Changeability*) [40] raamistiku järgi, mis käsitleb kvaliteeti laiemas mõistes. See on defineeritud järgmiselt.

- ***Correctness* ehk korrektsus.** Korrektsus tähendab, et mudel esitab süsteemi kohta korrektseid väiteid, sisaldades korrektseid elemente ja seoseid nende vahel; mudel järgib levinud reegleid ja tavasid, näiteks stiili- ja nimetamistavasid. [41]
- ***Completeness* ehk täielikkus.** Täielikkus tähendab, et mudel esitab kogu vajalikku informatsiooni, mis on asjakohane ja vastava mudeli eesmärkide seisukohast piisavalt detailne. [41]
- ***Consistency* ehk järjepidevus.** Järjepidevus tähendab, et mudelis puuduvad vasturääkivused. Järjepidevus jaguneb omakorda horisontaalseks ja vertikaalseks järjepidevuseks. Horisontaalne järjepidevus tagab järjepidevuse samal abstraktsioonitasemel või arendusfaasis olevate erinevate süsteemi aspektide kirjelduste vahel. Vertikaalne järjepidevus tagab järjepidevuse mudelite vahel, mis kujutavad sama süsteemi aspekti, mõistet või protsessi

erinevatel abstraktsioonitasemetel või erinevates arendusfaasides. Järjepidevus tagab ka semantilise järjepidevuse ehk ühel ja samal elemendil poleks erinevates mudelites erinev tähendus. [41]

- ***Comprehensibility* ehk arusaadavus.** Arusaadavus tähendab, et arendusprotsessi või kirjeldatava süsteemiga seotud kasutajad (inimkasutajad, CASE vahendid jne) saavad mudeli esitatavast informatsioonist aru. [41]
- ***Confinement* ehk kitsendatus.** Kitsendatus tähendab, et mudel on fokusseeritud mingile kindlale teemale, sisaldades näiteks asjakohaseid diagramme ja olles õigel abstraktsioonitasemel. Mudel on süsteemi kirjeldus, millest osa detaile on meelega välja jäetud. Kitsendatud mudel ei esita üleliigset informatsiooni ning pole liialt keeruline ega detailne. [41]
- ***Changeability* ehk muudetavus.** Muudetavus tähendab, et mudelit on lihtne muuta ja parandada. See toetab mudelite kiiret ja pidevat arenemist ning sobib seega kokku agiilse arenduse põhimõtetega. [41]

2.3 Tehniline võlg

Kui mudelid ei vasta eelnevalt kirjeldatud kvaliteedinõuetele, võib neid lugeda ebakvaliteetseteks. Ebakvaliteetsed mudelid viivad tulevikus suure tõenäosusega tehnilise võlani, mis on defineeritud kui minevikus tehtud otsused, mis mõjutavad negatiivselt tarkvarasüsteemi tulevikku. Tehnilist võlga on tihtipeale raske automaatsete vahenditega avastada, eriti kui vigu on tehtud süsteemi struktuuris või arhitektuuris, mille loomisel ja kujutamisel kasutatakse tihtipeale just mudeleid. Tehnilise võla peamisteks tekkepõhjusteks on tähtaegadega seotud pinged, hooletus, vähesed oskused, halvastitoimivad protsessid, vähene kvaliteedikontroll ja ebakompetentsus. Püüdes olla pidevalt kursis süsteemis või selle kirjelduses esinevate vigade ja nende tekkepõhjustega, on võimalik tehnilist võlga vähemalt mõneti vältida. [29]

Üheks tehnilise võla tunnuseks, millele ka käesolevas töös keskendutakse, on mudelite halb loetavus, mis võib olla tingitud kõikidest eelnimetatud tehnilise võla tekkepõhjustest. Kuna mudelite loetavus ja arusaadavus on üheks olulisemaks eelduseks tarkvarasüsteemide efektiivsel hooldamisel [15], tuleks nende tagamisele kindlasti panustada. Mudelite kvaliteetsuse tagamisel, kulutamata nende loomisele liialt aega ja energiat, võiksid abiks olla mudelitele üldistatud puhta koodi põhimõtted.

3 Puhtad mudelid

Puhaste mudelite mõistest on antud teema kontekstis praeguse seisuga vähe kirjutatud. Puhta koodi erinevaid definitsioone on aga mitmeid ning järgnevalt esitatakse neist mõned levinumad põhimõtted.

- Puhas kood on fokusseeritud ning ei ürita korraga liiga paljude ülesannetega tegeleda [35].
- Puhas kood on lihtne ja otsekohene ning seda on kerge lugeda [35].
- Puhast koodi suudavad lugeda ja parandada ka teised arendajad peale autori [35].
- Puhas kood ja testimine käivad käsikäes; testimine peaks olema regulaarne [35].
- Puhtas koodis on võimalikult vähe kordusi, rohkelt väljendusrikkust ja palju lihtsaid abstraktsioone [35].
- Puhas kood täidab korrektselt süsteemile püstitatud nõudeid [35].
- Skaudireegel (*The Boy Scout Rule*) – matkaplats tuleb endast maha jätta puhtamana kui see leiti. Seega ei piisa vaid puhta koodi kirjutamisest, vaid puhtust tuleb ka hoida [35].

Kõiki eelnevaid põhimõtteid on võimalik rakendada ka mudelitele ning need kattuvad suures osas eespool kirjeldatud *6C* kvaliteedinõuetega. Seega võiks puhtaid mudeleid defineerida näiteks *6C+V* raamistiku abil, mis lisab algsele kvaliteediraamistikule ka testitavuse ehk valideeritavuse nõude:

- **Valideeritavus.** Valideeritavus tähendab, et mudeleid on võimalik kiirelt ja efektiivselt kas käsitsi või automaatselt valideerida, sõltuvalt valitud valideerimismeetoditest.

Olles selgitanud puhta mudeli mõistet antud teema kontekstis, tuleks mudelitele üldistada ka koodi „halva lõhna“ mõiste ning kirjeldada mudelite valideerimist ja refaktoreerimist. Need on kõik mudelite puhtuse tagamise olulised tegurid. Järgnevalt on nendest lähemalt kirjutatud.

3.1 Mudeli halb lõhn

Koodi halb lõhn (*code smell*) on mõiste kirjeldamaks mingisuguseid väljapoole nähtavaid sümptomeid koodis, mis tavaliselt viitavad sügavamale veale koodis, selle loomise protsessis või selle realiseeritavas süsteemis. Taolisi halbu lõhnu on üldiselt koodis kerge märgata – näiteks suure koodiridade arvuga meetodid. Samas ei pruugi halb lõhn alati probleemi tähendada ehk nende otsimine võib anda valepositiivseid tulemusi. Probleemi olemasolus veendumiseks peab koodi sügavamalt uurima. Halvad lõhnad ei tähenda seega iseenesest kohe probleemi, kuid võivad sellele viidata. [18]

Koodi halbu lõhnu on konkreetselt kirjeldatud näiteks juba varem mainitud ja käesolevas töös aluseks võetud raamatus [35] ning Martin Fowler'i ja Kent Beck'i raamatus [16].

Mudeli halb lõhn võib nagu koodigi halb lõhn viidata mingisugusele sügavamale probleemile mudelis, modelleeritavas süsteemis või modelleerimisprotsessis. Ka siin ei pruugi halb lõhn otseselt probleemi tähendada, kuid annab modelleerijatele teada, mida oleks vaja täpsemalt uurida ja potensiaalselt parandada. Halb lõhn võib samas viidata potensiaalsetele probleemidele tulevikus, seega võiks neid juba varakult parandada [5].

Juba kirjeldatud konkreetseid koodi halbu lõhnu saaks otse ümber sõnastada tarkvara klassidiagrammidele, mille abil saab kirjeldada tarkvaraklasside struktuuri objektorienteeritud programmeerimiskeeltes nagu C++ ja Java. Kahjuks pole vastavused muude mudelitüüpide ja lähtekoodi vahel nii selged, seega ei saa neile otse rakendada konkreetseid koodi halbu lõhnu. [5] Seetõttu on käesolev töö keskendunud analüüsimudelite halbadele lõhnadele, jättes skoobist välja tarkvara klassidiagrammid.

Mudelite halbu lõhnu on võimalik kirjeldada ja täpsustada mõõdikute ning antimustrite abil. Kui mudelite halvad lõhnad on täpsustatud, saab neid arvesse võtta mudelite kvaliteedi tagamise protsessis, mille käigus mudeleid refaktoreeritakse. Sealjuures tuleb ka tähele panna, et refaktoreerimisel võivad tekkida uued vead ja neile viitavad halvad lõhnad, seega peaks kvaliteedi tagamise protsess toimuma kordustena vähemalt nii kaua, kuni on saavutatud soovitav mudeli kvaliteedi tase. [5]

Mudeli halbu lõhnu on konkreetselt varem kirjeldatud uuringus [6], mis keskendus kasutusjuhtude, klassi- ja olekudiagrammidele. Sarnaseid põhimõtteid esitab uuring

[12], mis kirjeldab kasutusjuhtude mudelite kvaliteedi tõstmist antimustrite abil. Uuringu [5] raames valmis automatiseeritud vahend mudeli halbade lõhnade avastamiseks, millest kirjutatakse veidi lähemalt mudelite valideerimise osas.

3.2 Mudelite valideerimine

Mudelite korrektsuse tagamine on modeelleerimise üks olulisemaid tegevusi, eriti suurte ja keerukate süsteemide puhul [42]. Mudelite valideerimine on protsess, mille käigus selgitatakse välja, kuivõrd korrektselt esitab mudel mingit reaalset maailma mõistet või protsessi [32]. Mudelite valideerimiseks on palju erinevaid meetodeid, mis jaotuvad üldjoontes kaheks – käsitsi (manuaalseks) ja automaatseks valideerimiseks.

3.2.1 Käsitsi valideerimine

Mudelite käsitsi valideerimisel viivad kogu valideerimisprotsessi läbi inimesed. *6C* kvaliteedinõuetest lähtudes saab käsitsi valideerimise abil veenduda mudelite järjepidevuses, täielikkuses ja kitsendatuses. Käsitsi valideerimistel peaksid osalema nii modelleerijad kui ka teiste valdkondade esindajad (sealhulgas inimesed, kes tunnevad modelleeritavat valdkonda), eriti mudelite arusaadavuse ja kitsendatuse hindamisel. [40] Järgnevalt kirjeldatakse mõnda käsitsi valideerimise meetodit.

Üheks käsitsi valideerimise meetodiks on juhitud inspekteerimine, mis üritab tuvastada mudeli puudujääke. Juhitud inspekteerimine lähtub testjuhtudest, mille abil uuritakse mudeli semantikat. Testjuhud tuletatakse näiteks kasutusjuhtudest. Inspekteerimisel jälgitakse ka kattuvust, ehk peetakse arvestust mudeli juba inspekteeritud osade üle. Protsessis osaleb mitmete rollide esindajaid – näiteks testjuhtude kirjutaja, moderaator (kes kontrollib sessiooni) ja salvestaja (kes teeb mudelites vajalikke muudatusi). [33]

Teiseks käsitsi valideerimise meetodiks on mudelite ülevaatused. Mudelite ülevaatuste puhul koguneb mudeleid hindama grupp kvalifitseeritud inimesi. Ülevaatuse jooksul määratakse kindlaks, kas mudel vastab tellija ootustele, aga ka seda, kui lihtne või raske oleks mudeli pealt lähtekoodi kirjutada ning selle põhjal valminud tarkvara hooldada ja edasi arendada. [38]

Juhitud inspekteerimise ja ülevaatuste peamine kriitika on nende aeganõudvus ja liigne mahukus. Et mudelipõhist arendust saaks integreerida agiilse arendusega, peab kõik

mudelitega seonduv, sealhulgas valideerimine, olema võimalikult kiire ja efektiivne. Samuti pole võimalik täie kindlusega öelda, et käsitsi valideeritud mudel on korrektne, sest see sõltub valideerimisel osalenud inimeste subjektiivsetest arvamustest. Seega tuleks eelnevalt mainitud meetodeid agiilses arenduses kasutada pigem harvem. [38]

Agiilse arenduse seisukohalt oleks paremaks käsitsi valideerimise meetodiks mudelite rünnakud (*model storming*). Kui mudelis avastatakse viga, kutsub vea leidnud modelleerija enda juurde teisi meeskonnaliikmeid, kellel on valdkonnast piisavad teadmised. Mudelit uuritakse koos ja üritatakse leida probleemile lahendus. Protsess võiks ajaliselt kesta kuni kümme minutit. Kui probleem on lahendatud, jätkavad kõik oma pooleli jäänud toimetusi. [39]

Seega soovitab käesolev lõputöö mudelite käsitsi valideerimisel enamasti kasutada mudelite rünnaku meetodit. Sealjuures ei pea protsessis osalema ainult arendusmeeskonna liikmed, vaid vajadusel võiks suhelda ka kliendiga. Kui mudelite rünnaku jooksul ei suudeta lahendust leida või selgub, et tegemist on sügavama probleemiga, võiks läbi viia mudelite ülevaatuse.

3.2.2 Automaatne valideerimine

Mudelite automaatsel valideerimisel viib valideerimisprotsessi läbi mingisugune automatiseeritud vahend. *6C* kvaliteedinõuetest lähtudes suudab automaatne valideerimine tuvastada ebajärjepidevust, ebatäielikkust ja ebakorrektsust, näiteks nimekonflikte, kaotsiläinud seoseid ning valesti defineeritud liideseid [40]. Automaatset valideerimist teostavaid vahendeid on mitmeid Vahendeid, mis teostavad automaatset valideerimist, on mitmeid ning need töötavad erinevatel põhimõtetel. Järgnevalt kirjeldatakse mõnda neist.

- Üheks automaatse valideerimise vahendiks on eelnevalt mainitud uuringu [5] raames valminud *Eclipse*'i plugin „EMF *Smell*“, mis suudab tuvastada EML-i (*Eclipse Modeling Framework*) baasil loodud mudelitest halbu lõhnu. Leitud lõhnadega tegelemine jääb modelleerija käsitsi läbiviidavaks ülesandeks.
- Teiseks automaatse valideerimise vahendiks on USE (UML-*based Specification Environment*), mis toetab OCL kitsendustega UML mudelite loomist. USE vahendis saab luua klassi-, objekti-, jada-, oleku- ja

kommunikatsioonidiagramme. Selle abil saab modelleerija mudeleid valideerida ehitades testistsenaariume. [23]

- Kolmanda automatiseeritud vahendiga saab valideerida UML'i klassidiagramme, millele on lisatud OCL kitsendused. Meetod võrdleb olemasolevat klassidiagrammi hulga eelnevalt defineeritud tingimustega ja otsustab nende põhjal, kas diagramm on korrektne või ei. Kuna tingimuste hulk on piiratud, võib korrektseks tunnistatud diagramm siiski mingeid vigu sisaldada. [8]

Kuigi mudelite automaatne valideerimine on võrreldes käsitsi valideerimisega tunduvalt kiirem protsess ning selle kasutamisel on omad eelised, ei tohiks käsitsi valideerimist täiesti kõrvale jätta. Praegused valideerimisvahendid ei ole veel piisavalt arenenud, et kõiki mudelite kvaliteediga seonduvaid aspekte korrektselt hinnata; selleks oleks siiski vaja teatud inimteadmisi ja –kogemusi. Seega on hea variant käsitsi ja automaatset valideerimist kombineerida.

3.3 Mudelite refaktoreerimine

Üha enam muutuvate ja kasvavate ärinõuetega kohanemiseks peavad info- ja tarkvarasüsteemid pidevalt arenema [22]. Tarkvara hooldamist loetakse tarkvara elutsüklis kõige kulukamaks tegevuseks, mis võib tarkvara kogukulust moodustada kuni 75% [20]. Mida vanem tarkvara on, seda kiiremini hakkab selle kvaliteet langema, sest hoolduse käigus tehtud parandused kipuvad olema ebaefektiivsed ja tekitama juurde uusi vigu [44]. Üks võimalus selliste probleemide leevendamiseks on järjepidev tarkvarasüsteemide struktuuri ja disaini täiustamine ehk refaktoreerimine [22].

Lähtekoodi refaktoreerimise põhieesmärk on parendada tarkvarasüsteemi sisemist struktuuri, muutmata seejuures süsteemi käitumist. Refaktoreerimine on distsiplineeritud meetod juba valmis kirjutatud koodi puhastamiseks, mis vähendab vigade tekkimise tõenäosust ja seega ka tehnilist võlga. [16]

Mudelite refaktoreerimine on osa mudelite teisendamisest, mis tähendab lähtemudelite muutmist sihtmudeliteks. Mudelite teisendamine võib olla vertikaalne või horistontaalne: kui lähte- ja sihtmudel kuuluvad erinevatele abstraktsioonitasemetele, on tegemist vertikaalse teisendusega; kui lähte- ja sihtmudel kuuluvad samale

abstraktsioonitasemele, on tegemist horisontaalse teisendusega. Vertikaalse teisenduse näide on andmebaasi tabelite disainimudeli loomine kontseptuaalsest andmemudelist. Horisontaalse teisenduse näide on kasutusjuhtude leidmine põhiobjekti seisundidiagrammi põhjal. Kui lisaks samale abstraktsioonitasemele on lähte- ja sihtmudel sama tüüpi ning samas keeles, on tegemist refaktoreerimisega. Selle puhul on oluline, et muudatus jätaks lähtemudeli põhiolemuse samaks. Mudelite refaktoreerimine aitab täiustada mudelite struktuuri, säilitades või parandades sealjuures selle sisemisi kvaliteediomadusi. [37]

Nii nagu lähtekoodi refaktoreerimisel, on ka mudelite refaktoreerimiseks üritatud välja arendada sobivaid vahendeid, mis teeksid sellest võimalikult efektiivse ja odava protsessi. Taoliste vahendite arendamisel mudelitele leidub aga teatud takistusi. Koodi puhul annavad refaktoreerimisvajadusest märku koodi halvad lõhnad, mida pole võimalik enamikele mudelitüüpidele otseselt rakendada. Mudelitele on halbu lõhnu ja neile sobivaid refaktoreerimismeetodeid defineeritud aga tunduvalt vähem, mistõttu on raskem arendada vahendit, mis neid tuvastada oskaks. Enamik eksisteerivaid refaktoreerimisvahendid on seega poolautomaatsed, kuna osa refaktoreerimiseks vajalikest sisenditest (teadmised, intuitsioon ja kogemus) on olemas vaid modelleerijatel endil. [22] Järgnevalt kirjeldatakse paari mudelite refaktoreerimise vahendit.

- Uuringus [4] kirjeldatud jõudlusepõhine refaktoreerimine, mis tuvastab mudelitest jõudluse antimustrite esinemisi ja tagastab parandatud mudeli, kust leitud probleemid on eemaldatud. Sealjuures säilitab refaktoreerimine mudeli algselt kirjeldatud funktsionaalsuse.
- Uuringu [22] raames loodud näidetepõhine refaktoreerimine, mis kasutab sisenditena algset refaktoreeritavat mudelit; komplekti struktuurimõõdikutest, mille väärtused on arvutatavad algse mudeli ja näitemudelite pealt; näiteid refaktoreeringutest, mis on väljavõetud erinevatest tarkvarasüsteemidest. Väljundiks on jada refaktoreerimisi, mida saab algsele mudelile rakendada.

Ka käesolevas töös on iga mudeli halva lõhna juures välja toodud refaktoreerimise juhiseid. Kuna eksisteerivad refaktoreerimisvahendid ei ole veel laialdast kasutust leidnud, siis keskendutakse töös mudeli halbade lõhnade tuvastamisele ja nende refaktoreerimisele pigem läbi mudelite käsitsi valideerimiste, milleks antakse juhiseid mudelite valideerimise peatükis.

4 Kataloog

Käesolev mudelite halbade lõhnade kataloog on mõeldud kasutamiseks süsteemi- ja ärianalüütikutele, äriinfotehnoloogia tudengitele ja kõigile teistele, kes soovivad halbu lõhnu oma mudelites tuvastada ja nende põhjust mudelitest välja refaktoreerida. Kataloog põhineb suures osas Robert C. Martini raamatus [35] peatükis 17 kirjeldatud lähtekoodi halbadel lõhnadel ja heuristikatel, kuid lisaks on toodud näiteid ja juhiseid ka muudest seotud allikatest. Kataloog on jaotatud neljaks osaks:

- Üldised mudelite halvad lõhnad (nimetähis „H“).
- Stiiliga seotud mudelite halvad lõhnad (tähis „S“).
- Mudelite ja nende elementidega nimetamisega seotud halvad lõhnad (tähis „N“).
- Mudelite valideerimistega seotud halvad lõhnad (tähis „V“).

Kategooriates on lõhnu omakorda grupeeritud nii, et üldisemad lõhnad asuvad koos eespool ning spetsiifilisemad lõhnad koos allpool.

Iga mudeli halva lõhna kirjeldus koosneb järgmistest osadest:

- Eestikeelne nimi.
- Koodi halva lõhna või heuristika nimi, millel see põhineb (parema seose tektitamiseks on need originaalkeeles ehk inglise keeles).
- Mudeli halva lõhna üldine seletav idee. Seal võib sisalduda ka konkreetseid näiteid.
- Mudeli refaktoreerimise seletus. Seal võib sisalduda ka konkreetseid näiteid.
- Sõnaline näide halvast lõhnast. Mõnikord on sealjuures ka kommentaare, kuidas seda parandada.
- Valikuliselt on mõnede lõhnade juures ka pildiline näide. Pildilised näited pärinevad „Andmebaasid I“ aineprojektidest, mida käesolevas töös analüüsiti (vt peatükk 5).

4.1 Mudelite üldised halvad lõhnad

H1: Sobimatu informatsioon

Koodi halb lõhn/heuristika: C1: Inappropriate Information

Üldine idee: Visuaalses/tekstilises mudelielemendis ei tohiks kuvada informatsiooni, mis sobiks paremini tekstilisse/visuaalsesse mudelielementi. Metaandmed nagu autorid ja viimane muutmise aeg ei tohiks olla visuaalsetes mudelielementides nähtaval (metaandmete sidumine visuaalse mudelielemendiga võib olla toetatud CASE vahendi poolt). Samamoodi ei tohiks visuaalsetes mudelielementides esitada detailseid tekstilisi kirjeldusi, sealhulgas nõudeid. Visuaalsed mudelielemendid peaksid olema süsteemi visuaalseks kirjeldamiseks. Tekstilised mudelielemendid peaksid olema süsteemi tekstiliseks kirjeldamiseks.

Refaktoreerimine: Sobimatu informatsioon visuaalses mudelielemendis tuleks tõsta ümber tekstilisse mudelielementi ja vastupidi. Kui avastatakse, et informatsioon on üleüldse kohatu või üleliigne, tuleks mudelielement kustutada.

Näited: Kasutusjuhtude diagrammis on nähtaval kasutusjuhtude tekstikirjeldused või selle osad. Olemi-suhte diagrammile on välja toodud kitsendused üksikute atribuutide väärtustele, mis oleks parem kirjutada atribuutide tekstilistesse kirjeldustesse.

H2: Aegunud mudelielement

Koodi halb lõhn/heuristika: C2: Obsolete Comment; C5: Commented-Out Code; G9: Dead Code

Üldine idee: Mudelielemendid, mis esitavad vananenud informatsiooni, on aegunud ning muutuvad ajas üha kasutumaks. Need võivad kirjeldada näiteks protsesse, mida enam ei eksisteeri, atribuute, mille nimed on muutunud ning tavasid, mida enam ei järgita. Need võivad asuda erinevates mudeli osades, kuid ei eksisteeri enam ühelgi mudelit kujutaval diagrammil. Aegunud mudelielemendid esitavad süsteemi või selle kohta käivate nõuete kohta valeinformatsiooni – seega on neid kohatu mudelis hoida.

Refaktoreerimine: Mudelielemendid võivad kiiresti vananeda, seega tuleks aegunud elemendi leidmisel see kas uuendada või kustutada.

Näited: Kasutusjuhu kirjeldus dokumentatsioonis, mis on välja kommenteeritud. Tegevusdiagramm, mis kujutab protsessi, mida kirjeldatavas süsteemis enam ei eksisteeri. Vananenud informatsiooniga kommentaar, mis kirjeldab vahepeal uuendatud mudelielementi. Kitsendus atribuutide definitsioonis, mis enam ei kehti.

H3: Üleliigne mudelielement

Koodi halb lõhn/heuristika: C3: Redundant Comment

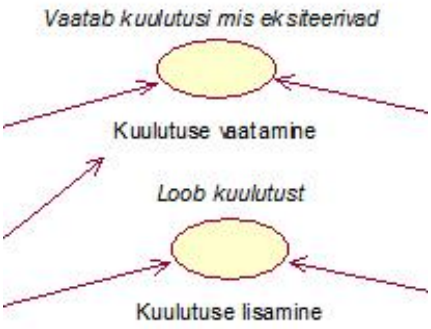
Üldine idee: Mudelielement on üleliigne, kui see kirjeldab midagi, mis kirjeldab end ise piisavalt hästi või on niigi iseenesest mõistetav. Näiteks kommentaarid (mis on samuti mudelielemendid) ei peaks esitama aksioome [1], vaid asju, mida mudel või mudelielement, mille kohta see kommentaar käib, ise kirjeldada ei suuda.

Refaktoreerimine: Üleliigne mudelielement tuleks mudelist kustutada.

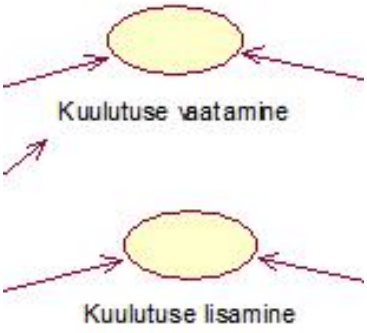
Näited: Kahe elemendi vahel on agregatsioon, mida sümboliseerivale joonele on lisatud selgitav tekst „agregatsiooniseos". Kahe elemendi vahel on assotsiatsioon, mida sümboliseerivale joonele on lisatud selgitav tekst „on seotud". Olemitüübi „Füüsiline isik" juures on märgitud, et ta kuulub liiki *Homo sapiens*, mis on üldiselt ilmselge.

Pildiline näide: esitatud alljärgnevatel joonistel (Joonis 1 ja Joonis 2).

Halvem: Parem:



Joonis 1. Üleliigsed kommentaarid [28].



Joonis 2. Üleliigsed kommentaarid on eemaldatud.

H4: Halvasti kirjutatud mudelielement

Koodi halb lõhn/heuristika: C4: Poorly Written Comment

Üldine idee: Tekstilisi mudelielemente tuleks kirjutada korrektselt. Näiteks õigekirjavead, pikad ja lohisevad laused ning ebaloogiliselt struktureeritud tekstiosad tähendavad halvasti kirjutatud tekstilist mudelielementi. Kirjutades tekstilisi mudelielemente, tuleks seda teha nii hästi kui võimalik.

Refaktoreerimine: Halvasti kirjutatud mudelielemente tuleks parandada, võttes arvesse korrektset süntaksi, semantikat ja pragmaatikat [9]. Sõnu tuleks valida hoolikalt, kasutades korrektset grammatikat ja kirjavahemärke; tekst peaks olema struktureeritud loogiliselt ning vormistatud loetavalt. Teemast ei tohiks kõrvale kalduda ega kirjeldada

iseenesest mõistetavat; rääkida tuleks vaid olulisest. Informatsiooni lugejal ei tohiks peale lugemise lõppu olla kahju asjata kulutatud ajast.

Näited: Vigases keeles kirjutatud kasutusjuhtude tekstilised kirjeldused, olemitüüpide ja nende atribuutide definitsioonid.

Pildiline näide: esitatud alljärgnevatel joonistel (Joonis 3 ja Joonis 4).

Halvem:

> Tarnijad ja tellijad töötavad vastavalt oma ettevõtete eeskirjadele kas ettevõtte kontoris neile eraldatud arvutitel või mõnes muus asukohas, eeldades et ta omab seal ligipääsu seadmele, mis on internetiga ühendatud ning kui ta ei kasuta Telema brauseripõhist teenust, peab seadmesse olema installeeritud ka Telema tarkvara.

Joonis 3. Pikk ja lohisev lause [27].

Parem:

> Tarnijad ja tellijad töötavad vastavalt oma ettevõtete eeskirjadele kas ettevõtte kontoris neile eraldatud arvutitel või mõnes muus asukohas. Muus asukohas töötamise eelduseks on ligipääs internetiga ühendatud seadmele. Kui ei kasutata Telema brauseripõhist teenust, peab seadmesse olema installeeritud ka Telema tarkvara.

Joonis 4. Kergemini arusaadavad laused.

H5: Mudelielemendid, mida ei lähe vaja teistel mudelielementidel

Koodi halb lõhn/heuristika: F4: Dead Function

Üldine idee: Igal mudelisse lisatud elemendil peab olema põhjus, milleks see seal on. See peab looma selgust ja konteksti teiste mudelielementide ning selle kaudu modelleeritava süsteemi mõistmiseks.

Refaktoreerimine: Mudelielement, mida teistel mudelielementidel vaja ei lähe, tuleks kustutada.

Näited: Mudelielement, mida ei lähe vaja teistel mudelielementidel, võib olla näiteks

- kasutusjuht, mille poole ei pöördu ükski tegutseja ja mis ei ole mingi teise kasutusjuhu alamkasutusjuht;
- toiming tegevusdiagrammil, kuhu ei sisene ükski töövoog;
- olemitüüp või atribuut kontseptuaalses andmemudelis, mille kirjeldatavaid andmeid kunagi ei kasutata;
- andmebaasioperatsioonid, millele ei viidata üheski kasutusjuhus.

H6: Kordused

Koodi halb lõhn/heuristika: G5: Duplication; G25: Replace Magic Numbers with Named Constants; G28: Encapsulate Conditionals; G35: Keep Configurable Data at High Levels

Üldine idee: Üks olulisemaid printsiipe tarkvaraarenduses on korduste vältimine ning sama kehtib ka modelleerimise kohta. Kordused mudelites viitavad üldistamise võimalusele. Muutes korduse abstraktsiooniks, suureneb mudelites kasutatav sõnavara, mida saavad kasutada ka teised modelleerijad. Mudelite loomine muutub kiiremaks ning väheneb vigade tegemise tõenäosus, sest abstraktsioonitase on kõrgem ja paljusid elemente, mis on juba valideeritud, on võimalik taaskasutada. Duplikatsioonil on kolm levinumat vormi:

1. **Samade mudelielementide kogumid, mida on mudelis igale poole kopeeritud**

   Üldine idee: Korduseid on lihtne ära tunda, nähes mudelites täpselt samade mudelielementide kogumeid, mis on tekkinud näiteks kopeerimise teel.

   Refaktoreerimine: Kõik sellised elemendid tuleks kustutada ning jätta alles vaid üks eksemplar. Sealjuures tuleb eelnevalt kontrollida nende elementide küljes olevaid seoseid ning ühendada need allesjäetava elemendiga, et midagi kaduma ei läheks.

2. **Samade otsustuspunktide esitamine erinevates mudelielementides**

   Üldine idee: Otsustuspunktide kordamisel on erinevates mudelielementides kirjas samad kontrollimist vajavad tingimused.

   Refaktoreerimine: Samade otsustuspunktide esitamisel erinevates kasutusjuhtude stsenaariumites tuleks luua uus kasutusjuht, mis sisaldaks tingimuste kontrollimist ja millele viidatakse *include* seose abil või viia need tingimused mittefunktsionaalsetesse nõuetesse. Kui erinevates andmeid muutvates kasutusjuhtudes kontrollitakse andmemuudatuste vastavust samadele ärireeglitele, tuleks need reeglid eraldi (näiteks kontseptuaalsesse andmemudelisse) kirja panna ja kasutusjuhtude kirjeldusest viited nendele eemaldada. Samade otsustuspunktide esitamisel erinevates tegevusdiagrammides võiks otsustusprotsessi kirjeldamiseks luua eraldi tegevusdiagrammi, millele saaks teistest diagrammidest viidata.

   Näited: Kasutusjuhtude stsenaariumites korduvalt esinevad otsustuspunktid, mis konrollivad ühtesid ja samu tingimusi.

3. **Sarnase käitumisega mudelid**

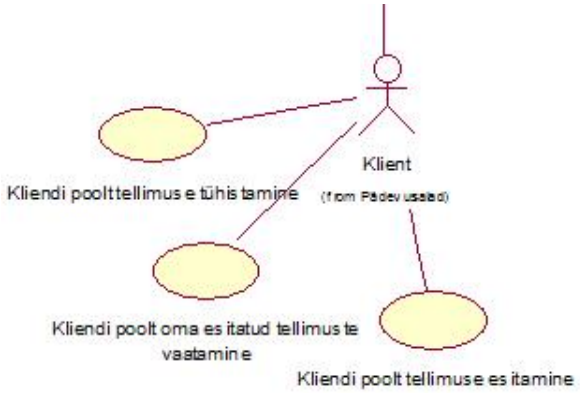
Üldine idee: Üks raskestimärgatav korduse vorm on mudelid, millel on sarnane käitumine või mis esitavad sarnaseid mõisteid, kuid millel on osaliselt erinevad elemendid.

Refaktoreerimine: Sarnane käitumine kasutusjuhtude stsenaariumites tuleks koondada eraldi kasutusjuhuks. Sellise kasutusjuhu oodatavat tulemust võib kasutada teiste kasutusjuhtude eeltingimusena. Sarnast käitumist kirjeldavad tegevusdiagrammid saaks võimalusel kokku liita üheks tegevusdiagrammiks.

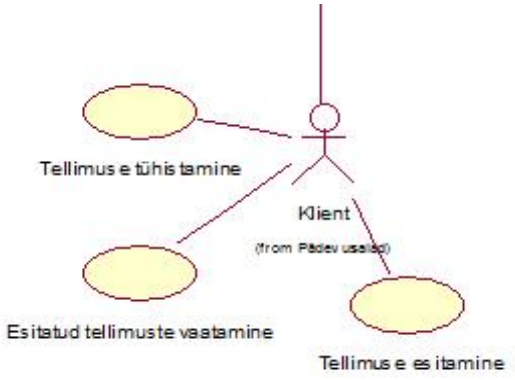
Näited: Kasutusjuhtude stsenaariumid, mis kõik algavad sisse logimisega. Parem oleks see koondada eraldi kasutusjuhuks „Sisselogimine“ ning kasutada eeltingimusena.

Pildiline näide: esitatud alljärgnevatel joonistel (Joonis 5 ja Joonis 6).

Halvem: Parem:



Joonis 5. Kordused kasutusjuhtudes [45].



Joonis 6. Kordused on eemaldatud.

H7: Valel abstraktsioonitasemel olevad mudelielemendid

Koodi halb lõhn/heuristika: G6: Code at Wrong Level of Abstraction; G7: Base Classes Depending on Their Derivatives; G34: Functions Should Descend Only One Level of Abstraction

Üldine idee: Modelleerimisel tekivad tavaliselt erinevate arendusetappide käigus erineva detailsusastmetega mudelid, millel on ka erinev lugejaskond. Sellised mudelid tuleks hoida üksteisest eraldi, mitte esitada läbisegi. Kasutusjuhud kasutusjuhtude diagrammil ja tegevused tegevusdiagrammil peaksid kõik olema ühel abstraktsioonitasemel, mis on ühe taseme võrra madalam, kui see protsess, mida need kirjeldavad (millele viitab kasutusjuhtude või tegevusdiagrammi nimi).

Refaktoreerimine: Leides mudelist valel abstraktsioonitasemel oleva elemendi, tuleks see viia üle mudelisse, mis kirjeldab elemendi esitatavat abstraktsioonitaset või muuta elemendi abstraktsioonitaset vastavalt sellele mudelile, milles see sisaldub.

Näited:

1. Valel abstraktsioonitasemel on liiga detailne mudelielement, mis asub kõrgtaseme mudelis. See võib segadusse ajada lugejat, kes loodab näha üldisemat mudelit ning ei saa aru taolisest detailsest elemendist.
2. Kasutusjuhtude diagrammis „Toodete haldamine“, kus on kasutusjuhud nagu „Kustuta toode“ ja „Loo uus toode“, on tunduvalt spetsiifilisem eraldi kasutusjuht „Loo uus toidutoode“ (välja arvatud juhul, kui see on lisatud *include* või *extend* seosega). Taolise ja sarnaste kasutusjuhtude jaoks tuleks luua uus diagramm, eelneva näite põhjal näiteks „Uue toote loomine“, kuhu läheksid veel kasutusjuhud nagu näiteks „Loo uus joogitoode“ ja „Loo uus puhastustoode“.

H8: Ebakõlad

Koodi halb lõhn/heuristika: G11: Inconsistency

Üldine idee: Kui teha midagi ühte kindlat viisi, tuleks kõiki sarnaseid tegevusi teha samamoodi. Erinevaid tavasid tuleks järgida hoolikalt ja kasutada neid mudelites läbivalt. Kasutades mingit elemendi nime ühes mudelis, siis tuleks selle sama elemendi kirjeldamiseks sama nime kasutada ka teistes mudelites. Samuti on oluline kasutada läbivalt kas ainsuse või mitmuse vormi. Kasutades järjepidevalt selliseid tavasid, muutuvad mudelid kergemini loetavaks ja muudetavaks.

Refaktoreerimine: Ebakõlade leidmisel tuleks valida üks mingisugune tava (soovitatavalt levinuim) ning seda järgides muuta elemente, mis seda tava ei järgi. Kasulik oleks need tavad kõigile arendusmeeskonna liikmetele teatavaks teha.

Näited:

1. Kasutusjuhu nimi tekstikirjelduses on „Vaata ostukorvi“, aga kasutusjuhtude diagrammis on sama element nimega „Kiika ostukorvi“.
2. Olemi-suhte diagrammis kasutatakse olemitüüpide nimedes mitmuse vormi („Isikud“ ja „Teenused“), olemitüüpide tekstikirjelduses ainsuse vormi („Isik“ ja „Teenus“).
3. Olemitüübi „Inimene“ kirjeldamiseks kasutatakse läbisegi nimesid „Inimene“, „Isik“ ja „Persoon“. Nendest nimedest tuleks valida üks ning kasutada mudelites

läbivalt ainult seda terminit. Olemitüüpide definitsioonide hulgas võib kirja panna ka alternatiivsed nimed.

H9: Üleliigne informatsioon mudelielementides

Koodi halb lõhn/heuristika: G12: Clutter; G13: Artificial Coupling; G30: Functions Should Do One Thing; G36: Avoid Transitive Navigation

Üldine idee: Kõik mudelielemendid peaksid olema fokusseeritud ning esitama informatsiooni vaid ühe kindla mõiste või protsessi kohta. Üleliigse informatsiooni korral ei saa lugeja kindel olla, mis on mudelielemendis kõige olulisem. See esitub näiteks

- elementidena, mis esitavad mingit mõistet või protsessi, mis ei ole seotud diagrammi üldise läbiva teemaga;
- elementidena, mis ei kirjelda midagi kasulikku ega huvipakkuvat;
- elementidena, mis kirjeldavad mitut mõistet või protsessi;
- ühele diagrammile kuhjatud üksteisest sõltumatute elementidena;
- dokumentatsiooni osadena, mis on kirjutatud liialt süvitsi ja millest keegi pole huvitatud.

Refaktoreerimine: Kõik eelnevad näited tuleks mudelitest eemaldada ning alles jätta vaid peamist informatsiooni edasiandvad elemendid. Kui element on mudelis vajalik, et aidata luua mõttelist seost teiste mudelitega, kuid samas ei puutu see element otseselt mudeli teemasse, siis tuleb elemendi esituse detailsus viia miinimumini (element kui seoselooja). Selle elemendi detailide vaatamiseks on mõni teine mudel.

Näited:

1. Teenuse registri olemi-suhte diagrammis on kujutatud teenuse registri endaga otseselt seotud registreid ja teisi registreid, millega otsene seos puudub. Diagramm kujutab liiga palju informatsiooni ja keskne idee võib jääda segaseks.
2. Registri olemi-suhte diagrammis esitatakse ka teiste registrite, kui diagrammi teemaks oleva registri, olemitüüpide atribuute ja kitsendusi.
3. Kasutusjuhtude mudelis kirjeldatakse detailselt andmebaasis andmete salvestamist, mis koormab niigi tihedat teksti.

H10: Kõrge sõltuvus

Koodi halb lõhn/heuristika: G14: Feature Envy

Üldine idee: Süsteeme tuleks modelleerida nii, et süsteemi alamosad ja neid alamosasid kirjeldavad mudelid oleksid üksteisest võimalikult vähe sõltuvad. Kuigi täielikku

sõltumatust pole võimalik saavutada, tuleks töötada selle nimel, et sõltuvus oleks võimalikult madal.

Refaktoreerimine: Kõrge sõltuvuse puhul tuleks läbi mõelda, kas kõik hetkel üksteisest sõltuvuses olevad süsteemi alamosad peavad seda ikkagi olema. Kui selgub, et mõni nendest sõltuvustest on üleliigne, tuleks sellele vastavalt süsteemi struktuuri uuendada.

Näited: Kontseptuaalses andmemudelis on näha, et kõik kirjeldatava süsteemi registrid on omavahel seotud, kuigi mõned nendest seostest on ebavajalikud. Näiteks olemitüüpide „Klient“ ja „Kaup“ vahele on modelleeritud seos „tellib“, samas kui tellimuste jaoks on olemas eraldi tellimuste register.

H11: Mudelielemendid vales asukohas

Koodi halb lõhn/heuristika: G17: Misplaced Responsibility

Üldine idee: Modelleerimisel on oluline selgelt läbi mõelda, kuhu liigitada erinevad mudelielemendid. Vastus ei pruugi olla kohe ilmselge, kuid otsustamisel tuleb appi vähima üllatuse printsiip – mudelielemendid peaksid asuma seal, kus neid eeldatakse asuvat.

Refaktoreerimine: Vales asukohas olev mudelielement tuleks tõsta sinna, kus teda kõige suurema tõenäosusega eeldatakse olevat.

Näited:

1. Kus peaks asuma kasutusjuht „Loo uus toode“? Kas Toodete, Tellimuste, või Tarnijate funktsionaalses allsüsteemis? Ilmselt kuulub kasutusjuht „Loo uus toode“ toodete allsüsteemi.
2. Kasutusjuhtude diagramm, mis on osaks registrite kirjeldusest.
3. Olemi-suhte diagramm, mis on osaks pädevusalade kirjeldusest.
4. Andmebaasioperatsiooni leping, mis asub kontseptuaalses andmemudelis.

H12: Ebaselge kirjeldus

Koodi halb lõhn/heuristika: G19: Use Explanatory Variables

Üldine idee: Keerulisest mõistest või protsessist arusaamine on veel raskem siis, kui seda on kirjeldatud liiga üldiselt või lühidalt, mis ei ava mõiste või protsessi sisu piisavalt. Samuti muudavad arusaamist raskemaks liiga pikad ja valdkonnaspetsiifilist sõnavara kasutavad laused.

Refaktoreerimine: Lihtne ja mõjuv viis mingisuguse keerulise mõiste või protsessi arusaadavamaks muutmiseks on selle väikesteks ja seletavateks osadeks tükeldamine.

Erinevate protsesside sõnalisel kirjeldamisel võiks kasutada lühikesi (liht)lauseid, mida on kerge lugeda ja mõista. Tegevusdiagrammides võiks liialt üldised ja seega segased tegevused lahutada väiksemateks ja täpsemateks tegevusteks.

Näited: Kauba saatmist kirjeldavas detailses tegevusdiagrammis on liiga üldine toiming „Halda kaupa“. See tuleks lahutada väiksemateks toiminguteks, nagu näiteks „Paki kaup“ ja „Kirjuta pakile saatja aadress“.

H13: Modelleeritava protsessi mittemõistmine

Koodi halb lõhn/heuristika: G21: Understand the Algorithm

Üldine idee: Protsesside modelleerimisel on tihtipeale probleemiks nende soovitavast käitumisest valesti või puudulikult arusaamine. Seetõttu ei mõista mudeli lugejad modelleeritava süsteemi käitumist. Selline olukord võib juhtuda näiteks väga spetsiifiliste valdkondadega seotud süsteemide modelleerimisel, millest selle valdkonna väline inimene ei suuda täpselt aru saada.

Refaktoreerimine: Modelleeritava protsessi mõistmiseks tuleks teha pidevat koostööd kliendiga ning vajadusel täpsustada üle segaseks jäänud nõuded.

Näited: Keemialabor on tellinud infosüsteemi, mille loomiseks on vajalik teatud keemiaalaste teadmiste omamine, kuid sellegipoolest üritavad modelleerijad enda teadmistega hakkama saada. Selle asemel tuleks tagada pidev suhtlus ja koostöö kliendiga, et vähendada üksteisest möödarääkimise võimalust.

H14: Sõltuvussuhted mudelites ei ole nähtavad igal pool

Koodi halb lõhn/heuristika: G22: Make Logical Dependencies Physical

Üldine idee: Kui üks mudelielement sõltub teisest, peaks see sõltuvussuhe olema mudelis esitatud nii visuaalsetes kui ka tekstilistes mudelielementides.

Refaktoreerimine: Kui tekstis on kirjeldatud kahe elemendi sõltuvussuhet, siis peaks selle nähtavale tooma ka diagrammides ja vastupidi.

Näited: Kasutusjuhtude *include* seosed on kirjeldatud ainult diagrammil, kuigi peaksid olema kirjeldatud nii diagrammil kui ka kasutusjuhtude tekstikirjeldustes.

H15: Ebatäpsused

Koodi halb lõhn/heuristika: G26: Be Precise

Üldine idee: Tehes modelleerimisel mingeid otsuseid, tuleb sealjuures olla täpne, et vähendada millegi mitmeti mõistmise tõenäosust. Ebatäpsused mudelites viivad vigase

tarkvarani, mida on juba raskem parandada, seega on täpsus modelleerimisel väga oluline.

Refaktoreerimine: Leides mingisuguse mudelielemendi, mida on võimalik mitut moodi tõlgendada, tuleks konsulteerida kellegagi, kes teab täpselt, mida selle all on mõeldud. Seejärel tuleks küsimuse all olev mudeli osa ümber teha nii, et mitmeti mõistmise tõenäosus oleks võimalikult väike.

Näited:

1. NASA kosmosesond „*Mars Climate Orbiter*“ ebaõnnestus oma ülesandes jääda Marsi orbiidile, sest selle navigeerimissüsteem kasutas inglise mõõtühikuid, kuid sond ise ootas SI-mõõtesüsteemi ühikuid [24]. Taoline nõuete ebatäpsus läks maksma 125 miljonit dollarit [52].
2. Kontseptuaalses andmemudelis on atribuudid, mis peaksid esitama mingite mõõtmiste tulemusi esitavaid väärtuseid, kuid atribuutide kirjelduses pole mõõtühikut ära märgitud või on kirjas vale mõõtühik.

H16: Omavolitsemine

Koodi halb lõhn/heuristika: G32: Don’t Be Arbitrary

Üldine idee: Mudeleid tuleks ühtemoodi struktureerida. Kui mudelis on läbivalt kasutatud kindlat struktuuri, siis järgivad ja kasutavad seda üldiselt ka kõik teised modelleerijad. Kui kindlalt struktureeritud mudelisse teeb keegi omavoliliselt midagi struktuurivälist, siis võib see julgustada ka teisi modelleerijaid struktuuri mitte jälgima, mille tulemusel muutub mudel väga raskesti loetavaks. Sellist käitumist kirjeldab katkise akna teooria [7], mille kohaselt kutsub maja üks katkine aken esile rohkem vandaalitsemist. Samas ei tohiks struktuuri järgimine olla liiga range ja loovuse kasutamise võimalus ei tohiks olla pärsitud, kuna loomingulisusest võivad tihtipeale tekkida hoopis paremad lahendused.

Refaktoreerimine: Leides mudelite struktuuris mingi suurema vea või avastades parema mooduse millegi lahendamiseks, tuleks enne suuremate muudatuste tegemist ka teisi meeskonnaliikmeid informeerida.

H17: Liigne otsustuspunktide arv

Koodi halb lõhn/heuristika: G15: Selector Arguments; G23: Prefer Polymorphism to If/Else or Switch/Case

Üldine idee: Liigne otsustuspunktide kasutamine kasutusjuhtude stsenaariumites ja tegevusdiagrammides on enamasti tingitud sellest, et see on lihtne ja mugav valik.

Suurtes kogustes ei mõju need aga kahjuks mudelite loetavusele hästi. Mudelite loetavuse seisukohast oleks ideaalne, kui otsustuspunktide arv kasutusjuhtude stsenaariumites ja tegevusdiagrammides ei oleks liiga suur (*a la* otsustuspunkt iga toimingu ees/järel).

Refaktoreerimine: Avastades rohkete otsustuspunktidega kasutusjuhtude stsenaariumi või tegevusdiagrammi, võiks uurida, kas sel kirjeldatakse rohkem kui ühte protsessi või kas kõik otsustuspunktid on protsessi kirjeldamisel vajalikud. Kui kirjeldatakse rohkem kui ühte protsessi, tuleks iga protsessi kohta luua uus kasutusjuhtude stsenaarium või tegevusdiagramm. Kui leidub ebavajalikke protsesse, mille eemaldamisel jääb protsessi põhimõte samaks, tuleks need loetavuse parandamiseks kustutada.

Näited: Tegevusdiagrammis „Kinopileti ostmine“ on iga toimingu järel otsustuspunkt „Kas klient soovib piletiostuga jätkata?“. Ilmselt pole selline otsustuspunkt vajalik näiteks toimingu „Piletimüügi keskkonda sisse logimine“ või „Valitud pileti kinnitamine“ järel ning selle võiks diagrammilt kustutada.

H18: Sobimatud alamkasutusjuhud

Koodi halb lõhn/heuristika: G18: Inappropriate Static

Üldine idee: Kasutades kasutusjuhtude diagrammides alamkasutusjuhte, tuleks veenduda, et see on olukorrale parim lahendus. Mida vähem alamkasutusjuhte, seda parem, kuna need võivad rohke kasutuse korral diagrammi üle koormata. Mõni allikas soovitab näiteks *extend* suhet kasutusjuhtude vahel mitte kunagi kasutada [19].

Refaktoreerimine: Sobimatud alamkasutusjuhud tuleks diagrammist eemaldada või viia need diagrammi, kuhu sobiksid paremini.

Näited: Kasutusjuhtude diagramm, mille iga põhikasutusjuht sisaldab mitut alamkasutusjuhtu, on liiga kirju ja muudab lugemise raskeks. Ilmselt esitavad mõned neist alamkasutusjuhtudest ebavajalikku või liiga detailset informatsiooni ning tuleks seega eemaldada.

H19: Kasutusjuhtude ebaloogiline järjekord

Koodi halb lõhn/heuristika: G31: Hidden Temporal Couplings

Üldine idee: Kasutusjuhtude mudelis ei modelleerita ilmutatult kasutusjuhtude läbiviimise järjekorda. Siiski on kasutusjuhud üksteisega eel- ja järeltingimuste kaudu seotud, mis paneb paika teatud loogilise kasutusjuhtude läbimise järjekorra. Sellegipoolest esitatakse kasutusjuhud tihtipeale ebaloogilises järjekorras. Samuti peaks

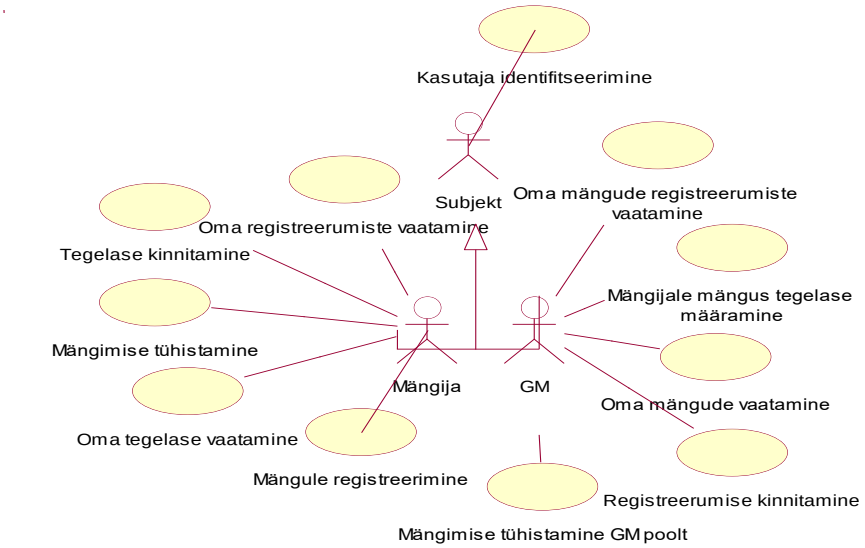
kasutusjuhtude tekstikirjelduste alamosadel olema loogiline järjekord – näiteks peaks eeltingimused esitama enne järeltingimusi.

Refaktoreerimine: Kasutusjuhtude esitamise järjekorra määramisel tulevad appi hästi defineeritud eeltingimused. Esimesena peaks esitama kasutusjuhu, millel eeltingimused puuduvad. Teisena peaks esitama kasutusjuhu, mille eeltingimuseks on esimese kasutusjuhu oodatavate tulemuste saavutamine. Kolmandana peaks esitama kasutusjuhu, mille eeltingimusteks on esimese ja teise kasutusjuhu oodatavate tulemuste saavutamine jne. [51] Kui kasutusjuhtude leidmiseks kasutatakse olekudiagramme, annab olekudiagramm informatsiooni kasutusjuhtude õige järjekorra leidmiseks.

Näited: Kasutusjuht „Kustuta toode“ ilmub dokumendis enne/diagrammis ülalpool kasutusjuhtu „Loo uus toode“.

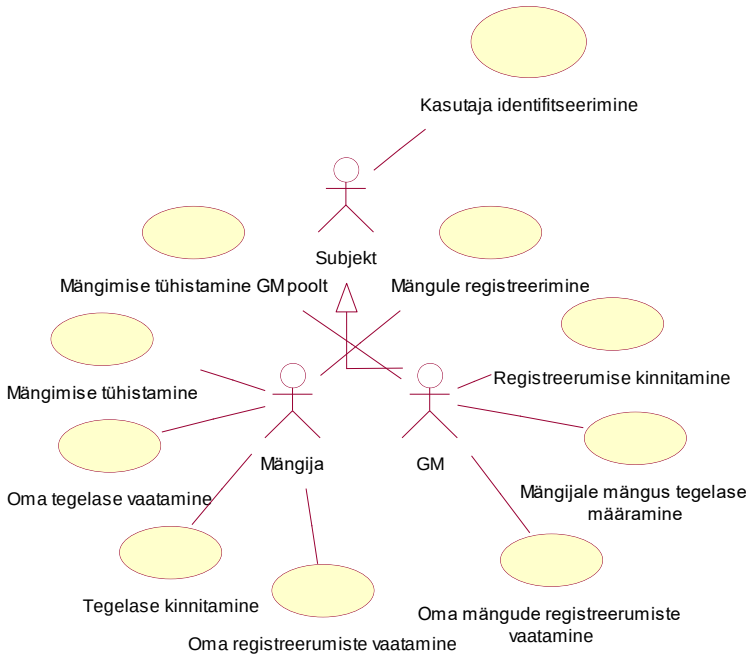
Pildiline näide: esitatud alljärgnevatel joonistel (Joonis 7 ja Joonis 8).

Halvem:



Joonis 7. Kasutusjuhtude ebaloogiline järjekord [2].

Parem:



Joonis 8. Kasutusjuhud on järjestatud kellaosuti suunas.

Märkus. Kuigi kasutusjuhud said loogilisse järjekorda seatud, tekkisid diagrammil ristuvad jooned ehk ühe mudeli halva lõhna parandamisel tekkis uus mudeli halb lõhn.

H20: Väär käitumine piirjuhtudel

Koodi halb lõhn/heuristika: G3: Incorrect Behavior at the Boundaries; G33: Encapsulate Boundary Conditions

Üldine idee: Süsteemid peaksid käituma korrektselt, kuid kahjuks ei pruugi arusaam korrektsest käitumisest alati vastata tegelikkusele. Tihtipeale luuakse mudeleid, mida arvatakse kirjeldavat süsteemi piisavalt õigesti, kuid ei kontrollita mudelite esitatava käitumise korrektsust piirjuhtudel. Kuna iga võimalikku piirjuhtu pole võimalik fikseerida ning liialt äärmustesse pole vaja samuti laskuda (näiteks mis saab tuumasõja puhul jne), tuleks rõhku panna piirjuhtudele, mille esinemistõenäosus on suurem.

Refaktoreerimine: Tuleks analüüsida, mis on konkreetse kirjeldatava süsteemi juures kõige tõenäolisemad piirjuhud ning selgitada välja, mida oleks nende esinemise juures vaja teha. Seejärel tuleks süsteem modelleerida nii, et see võtaks neid piirjuhte arvesse.

Näited: Tellimuse menetluse protsessi juures oleks suurema tõenäosusega piirjuht näiteks postipandud paki kaduma minek ja vähemtõenäoline piirjuht tarnija pankrotti minek, kuid analüüsis on tähelepanu pööratud rohkem tarnija pankrotile või mitte kummalegi piirjuhule.

H21: Ilmselgena tunduvat käitumist ei kirjeldata

Koodi halb lõhn/heuristika: G2: Obvious Behavior Is Unimplemented

Üldine idee: Järgides vähima üllatuse printsiipi, peaks iga kasutusjuht või tegevusdiagramm kirjeldama käitumist, mida teised mudeli lugejad sellelt eeldavad ja ootavad. Kui kasutusjuhud ja tegevusdiagrammid ei kirjelda oodatavat käitumist, ei saa mudeli lugejad loota oma intuitsioonile, vaid peavad hakkama lugema mudeli detaile.

Refaktoreerimine: Kasutusjuht või tegevusdiagramm tuleks ümber teha nii, et see kirjeldaks oodatavat käitumist. Võimalik lahendus on ka kasutusjuhu või tegevusdiagrammi nime muutmine, mis sobiks kirjeldatava käitumise nimetamiseks.

Näited: Kasutusjuht nimega „Otsi kaupu“, mille lühikirjelduses pole kaupade otsimisest midagi kirjutatud.

H22: Mitu keelt ühes mudelifailis

Koodi halb lõhn/heuristika: G1: Multiple Languages in One Source File

Üldine idee: Oleks ideaalne, kui iga mudel oleks loodud kasutades ühte modelleerimiskeelt, kuid ilmselt pole see alati võimalik. Siiski tuleks pingutada, et erinevate keelte arv oleks võimalikult väike. Samuti ei tohiks mudelis olla läbisegi mitmeid erinevaid inimkeeli. Mida rohkem erinevaid keeli mudel kasutab, seda raskem on seda lugeda, sest inimene peab mõttes pidevalt erinevate keelte vahel ümber lülituma ja neid kõiki keeli ka valdama.

Refaktoreerimine: Võimalusel tuleks erinevates modelleerimiskeeltes loodud mudelid tõlkida ümber ühte neist, eelistatavalt kõige levinumasse või kõige paremasse. Erinevate inimkeelte puhul tuleks samuti valida üks neist ning tõlkida teistes keeltes olevad mudelielemendid sellesse keelde.

Näited: Mudelites on läbisegi kasutusel eesti ja inglise keel.

H23: Hoiatuste väljalülitamine

Koodi halb lõhn/heuristika: G4: Overridden Safeties

Üldine idee: Hoiatuste väljalülitamine on ohtlik. Mudelitest uute tehiste (teine mudel, lähtekood, dokumentatsioon, testid) genereerimisel CASE vahendi väljastatavate hoiatuste andmise väljalülitamine võib genereerimise küll esmapilgul lihtsamaks teha, kuid tulemus võib olla vigane. Kuna kasutaja hoiatust ei näe, siis ei oska ta viga oodata.

Refaktoreerimine: Kuna erinevate hoiatuste ja vigade ignoreerimine võib hiljem valusalt kätte maksta, oleks targem seda mitte teha ja parandada vead, mida CASE vahend esile toob.

H24: Mudelist uue tehise genereerimine võtab kaua aega

Koodi halb lõhn/heuristika: E1: Build Requires More Than One Step

Üldine idee: Mudelitest uute tehiste (teine mudel, lähtekood, dokumentatsioon, testid) genereerimine peaks olema kiire tegevus. Seega peaksid kõik mudelid võimaluse korral asuma ühes projektifailis. Sellisel juhul poleks näiteks vaja läbi käia mitmeid faile, et anda üha uuesti ja uuesti sama käsku. Samuti puuduks erinevate failide otsimise, avamise ja sulgemisega kaasnev ajaraiskamine ning poleks tarvis alatasa veenduda, et käske antakse õiges järjekorras. Mudelitest uue tehise genereerimine peaks toimuma ühe käsuga.

Refaktoreerimine: Kõik mudelid tuleks võimalusel tõsta ühte projektifaili.

Näited: Kui infosüsteemi iga registrit realiseeriva SQL-andmebaasi alamosa kirjeldus on eraldi CASE vahendi failis, tuleks SQL lähtekoodi genereerimiseks anda ühte ja sama käsku korduvalt. Lisaks on genereeritud koodis ilmselt ülekate (kuna registrid on omavahel seotud, siis korduvad samade tabelite kirjeldused erinevates mudelites), mida tuleks hakata käsitsi kõrvaldama.

H25: Läbimõtlemata atribuutide tüübid

Koodi halb lõhn/heuristika: J3: Constants versus Enums

Üldine idee: Olemitüüpide atribuutide tüübid tuleks hoolega läbi mõelda, sest tüüp aitab mõista atribuudi tähendust ja viitab operatsioonidele, mida selle väärtusega tegema hakatakse.

Refaktoreerimine: Vali atribuudile parem tüüp. Kui CASE vahend ei paku eeldefineeritud tüüpide nimekirjas soovitud tüüpi, siis uuri võimalust CASE vahendis uusi tüüpe kirjeldada.

Näited: Isikukood on arvutüüpi, mis ei arvesta võimalusega, et isikukood võib sisaldada tähti, alata nullidega, olla pikem kui vastavasse arvutüüpi kuuluv suurim arv ning kõigele lisaks jätab mulje, et isikukoodidega tehakse aritmeetilisi operatsioone.

H26: Tõeväärtusparameetrid

Koodi halb lõhn/heuristika: F3: Flag Arguments

Üldine idee: Tõeväärtusparameetrid andmebaasioperatsioonide lepingutes võivad viidata sellele, et operatsioon täidab mitut ülesannet ja läheb sellega vastuollu otstarbe lahususe põhimõttega. Samas võib see lõhn anda valepositiivseid [14] ehk alati ei pruugi tõeväärtusparameetrid mitme ülesande täitmist tähendada. See võib juhtuda siis, kui operatsioon peab tõepoolest registreerima etteantud tõeväärtustüüpi väärtuse.

Refaktoreerimine: Juhul kui operatsioon täidab mitut ülesannet, tuleks see jaotada väiksemateks operatsioonideks, millest igaüks täidab ühte ülesannet.

4.2 Stiili halvad lõhnad

S1: Liiga mahukad diagrammid

Koodi halb lõhn/heuristika: F1: Too Many Arguments; G8: Too Much Information

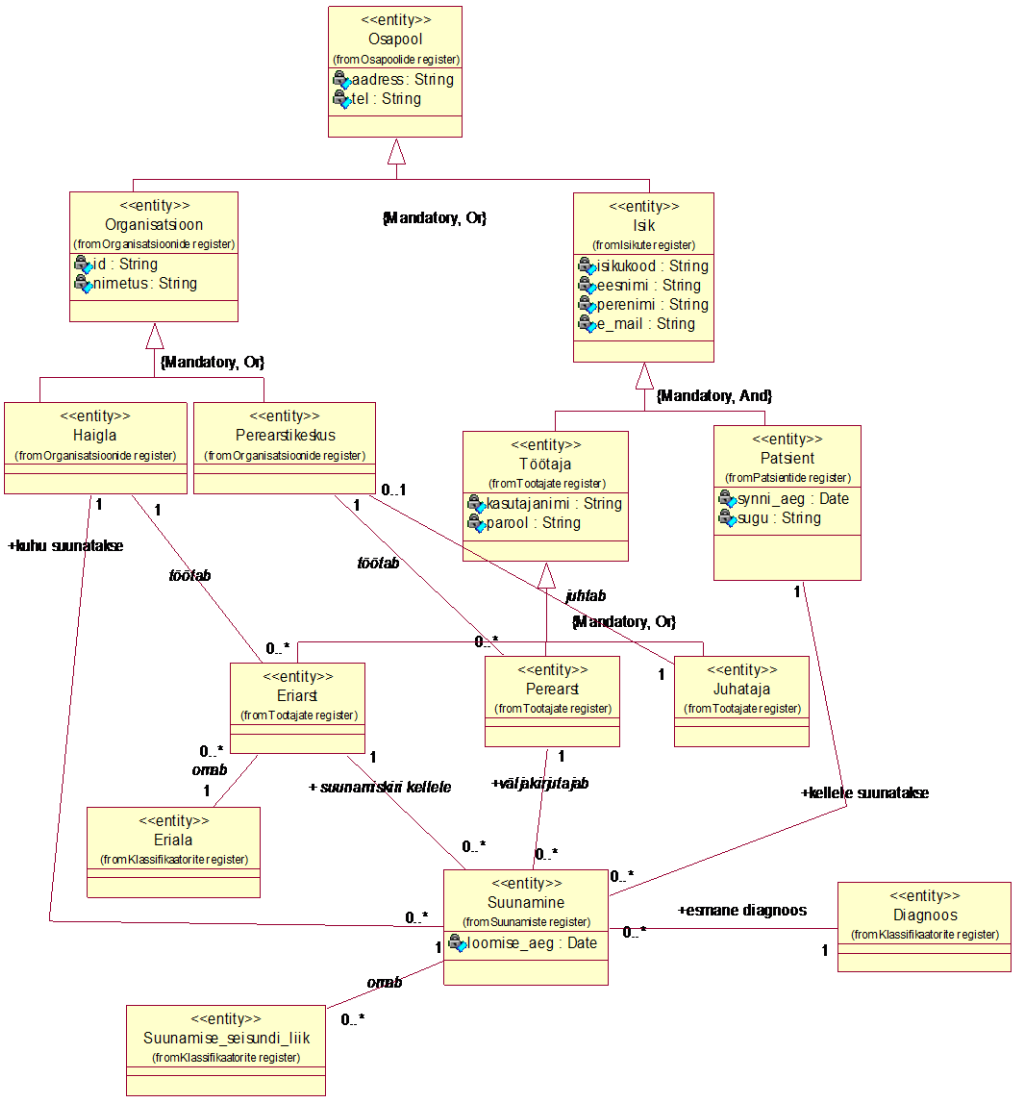
Üldine idee: Head diagrammid on nii lihtsad kui võimalik ja ei grammigi lihtsamad [13]. Halvasti loodud diagrammid on pindalalt suured, mis sunnib lugejat arvutis kerimise ning suurendamise/vähendamise operatsioone tegema. Väljatrükituna on need A4 lehel liiga väikesed, kuid suuremale lehele trükkides puudub neil fookus ja need on raskesti hoomatavad. Hästi defineeritud diagrammid ei esita liiga palju informatsiooni, seega on nende sisemine sidusus kõrge (*high cohesion*). Halvasti defineeritud diagrammid kujutavad koos väga palju informatsiooni, seega on nende sisemine sidusus madal (*low cohesion*). Diagrammide liigset mahtu võib piirata kasutades näiteks 7+/-2 mustrit [13], mis soovitab ühel diagrammil kujutada orienteeruvalt seitset elementi.

Refaktoreerimine: Kui diagramm sisaldab liiga palju elemente, mis on kõik olulised, tuleks diagramm lõigata väiksemateks ja hoomatavamateks tükkideks. Kui diagrammil on liiga palju elemente, millest kõik ei ole olulised, võiks kaaluda ebaoluliste elementide kustutamist või vähemalt ära peitmist (näiteks atribuudid olemitüüpides, kui need ei lisa midagi olulist diagrammile).

Näited: Olemi-tüübi diagrammis on 20 olemitüüpi, mis ei mahu loetavas suuruses korraga arvutiekraanile.

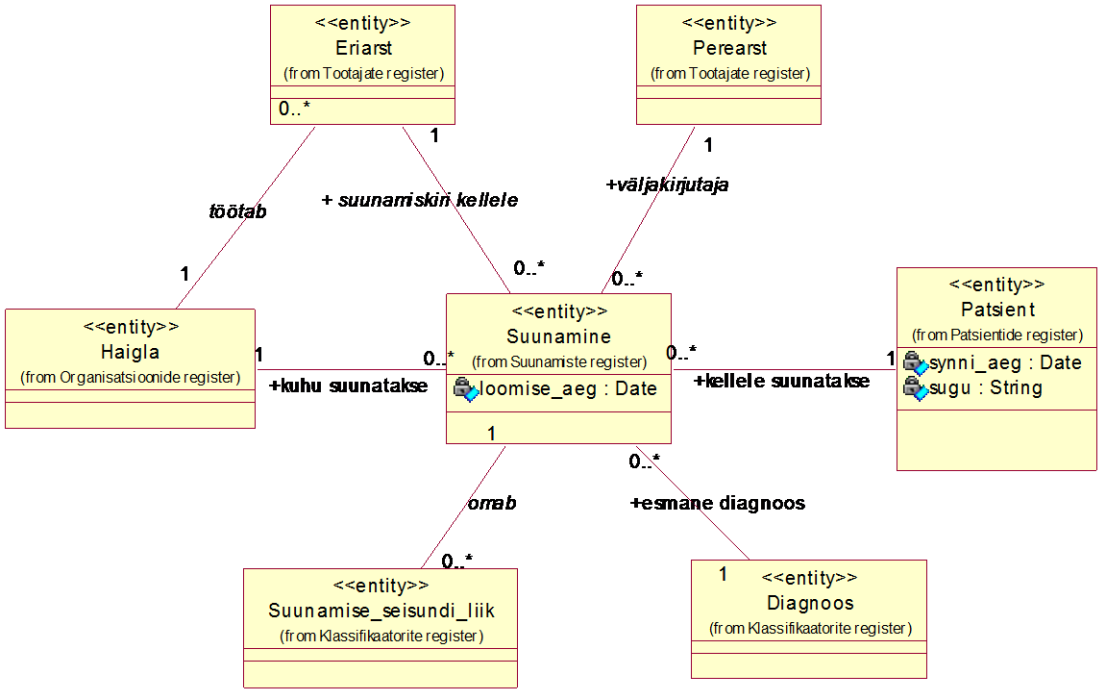
Pildiline näide: esitatud alljärgnevatel joonistel (Joonis 9 ja Joonis 10).

Halvem:



Joonis 9. Liiga mahukas diagramm [11].

Parem:



Joonis 10. Paremini hoomatav tükk algsest diagrammist, mis kujutab põhiolemitüüpi ja selle seoseid.

Märkus. Eelnev parandatud diagramm on vaid üks osa algsest diagrammist. Infokao vältimiseks tuleks algne diagramm asendada mitme ülekattega diagrammiga.

S2: Pikad jooned diagrammis

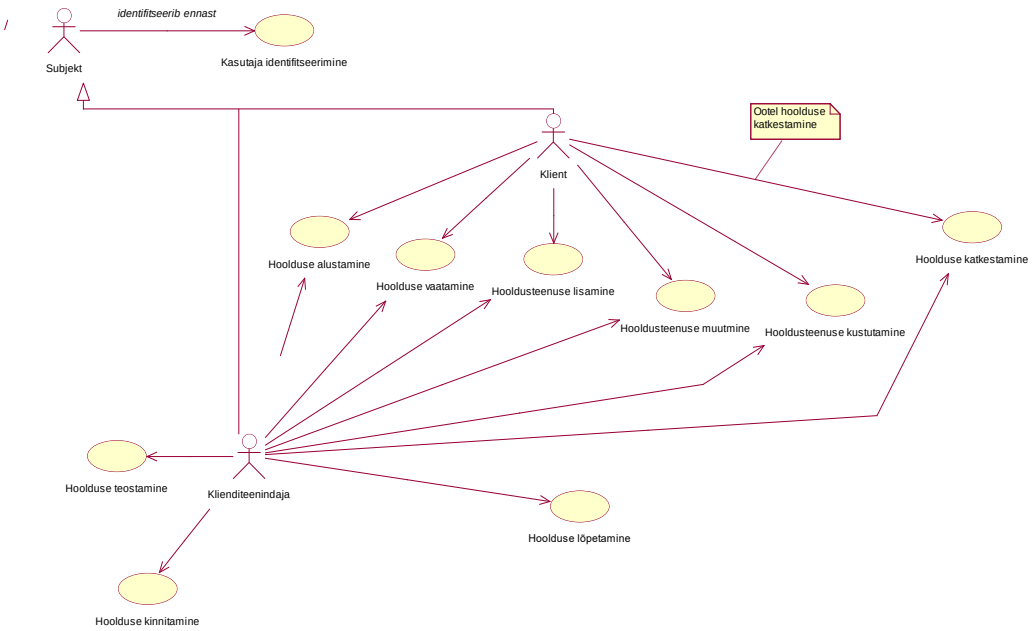
Koodi halb lõhn/heuristika: G10: Vertical Separation

Üldine idee: Halb lahendus oleks hoida kaht teineteisega seotud elementi diagrammi erinevates otstes, mis teeks neid ühendava joone liiga pikaks. Diagramm oleks sel juhul halvasti loetav ja näeks segane välja ning kasvab võimalus ristuvate joonte tekkimiseks.

Refaktoreerimine: Diagrammil asuvad seotud elemendid peaksid asuma üksteisele võimalikult lähedal, et neid ühendavad seoseid tähistavad jooned oleks lühikesed. Sealjuures tuleks vältida ristuvaid jooni. Kui ristuvad jooned on vältimatud, peaks üks joon teisest „üle hüppama“ ehk ristumiskohal üks joontest kõverdub [21]. Samuti tuleks võimalusel vältida diagonaalseid ja ümaraid jooni [21].

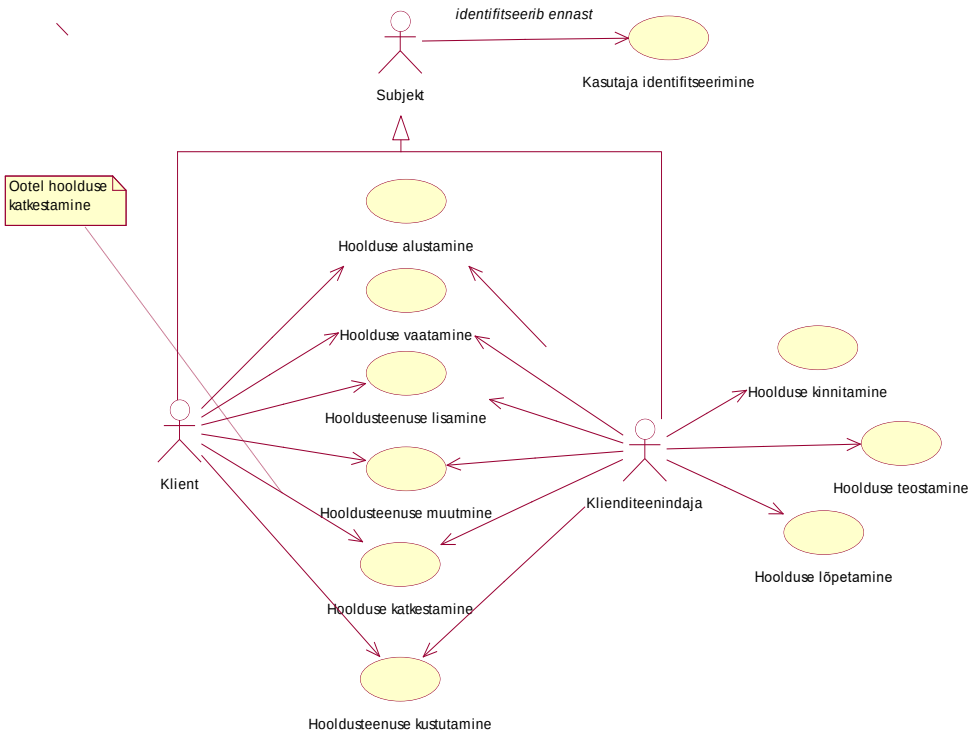
Pildiline näide: esitatud alljärgnevatel joonistel (Joonis 11 ja Joonis 12).

Halvem:



Joonis 11. Liiga pikad jooned [30].

Parem:



Joonis 12. Parandatud joonepikkused.

S3: Levinud modelleerimistavade mittejärgimine

Koodi halb lõhn/heuristika: G24: Follow Standard Conventions

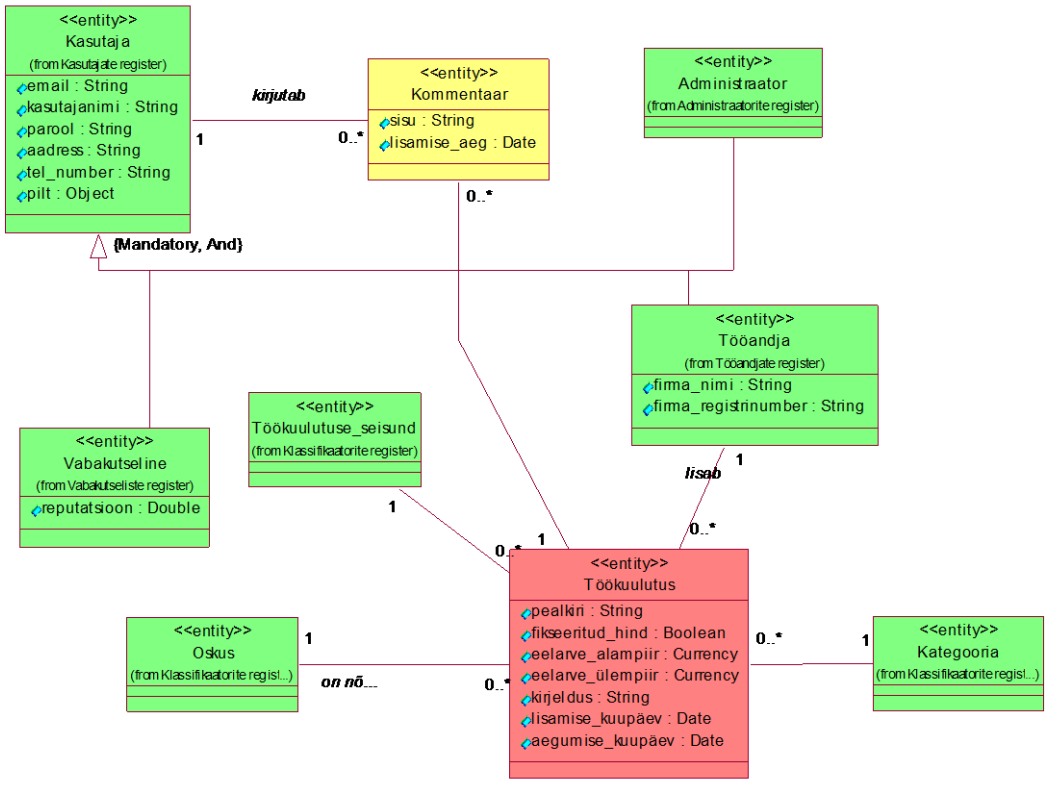
Üldine idee: Iga modelleerija või modelleerijate meeskond peaks järgima levinud modelleerimistavasid ja -harjumusi. Need tavad peaksid näiteks täpsustama, millised peavad välja nägema erinevat tüüpi elemendid, kuidas nimetada mudeleid ja mudelielemente ning kuidas näidata diagrammidel seoseid. Taolisi tavasid poleks vaja tekstilistes mudelielementides lahti selgitada, sest mudelid ise on piisavalt heaks näiteks.

Refaktoreerimine: Kuna levinud modelleerimistavade mittejärgimine võib tähendada seda, et inimesed nii meeskonna sees kui sellest väljas mudeleid ei mõista, tuleks valida mingisugune kindel tavade kogum ning muuta kõiki mudeleid nendele vastavaks.

Näited: Tegevusdiagrammi alguspunkt asub suvaliselt kuskil diagrammi keskel, kuigi üldjuhul peaks see asuma diagrammi üleval vasakus nurgas [47]. Kasutusjuhtude diagrammis asub peamine tegutseja all keskel, kuigi üldjuhul peaks see asuma üleval vasakus nurgas [50]. Olekudiagrammis on lõppseisund üleval vasakus nurgas, kuigi üldjuhul peaks see asuma paremal all nurgas [49]. Üldistusseose korral on üldisem element allpool, kuigi tegelikult peaks see asuma üleval [48].

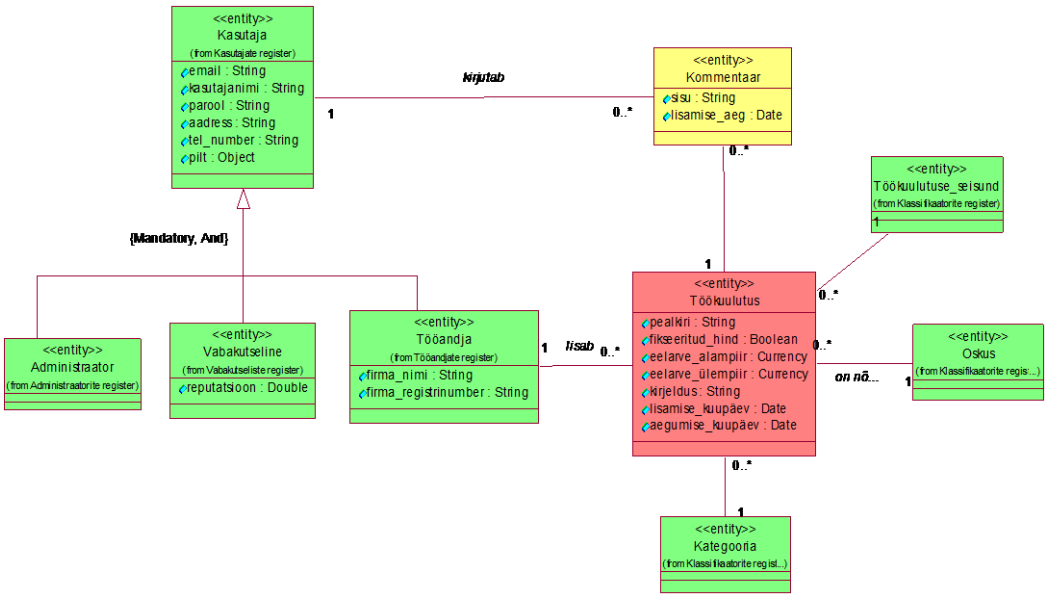
Pildiline näide: esitatud alljärgnevatel joonistel (Joonis 13 ja Joonis 14).

Halvem:



Joonis 13. Üldistusseos ei vasta modelleerimistavadele [10].

Parem:



Joonis 14. Üldistusseos vastab modelleerimistavadele.

S4: Ebakindel struktuur

Koodi halb lõhn/heuristika: G27: Structure over Convention

Üldine idee: Mudeleid luues tuleks kasutada kindlaid ja läbivaid struktuure, mis hoiavad mudeleid laiali valgumast ja teevad need loetavamaks. Näiteks tuleks pakette struktureerida kindlat moodi, kasutusjuhtudel ja andmebaasioperatsioonide lepingutel peaks olema kindel formaat ja kontseptuaalsel andmemudelil (olemi-suhte diagrammid, olemitüüpide ning atribuutide definitsioonid) peaks olema kindel formaat. Mida ebakindlam on mudelite struktuur, seda raskem ja vähem intuitiivsem on ka nende muutmine või neisse uute osade lisamine.

Refaktoreerimine: Mudelitele ja mudelielementidele tuleks valida mingisugune kindel struktuur, kasutada seda läbivalt ning viia sellega vastavusse juba eksisteerivad mudelid ja mudelielemendid.

Näited: Süsteemi mõnede allsüsteemide kohta on eraldi paketid, mis sisaldavad neid allsüsteeme kirjeldavaid mudeleid, kuid ülejäänud allsüsteemide kirjeldused asuvad kõik ühes ebamäärase nimega paketis.

4.3 Nimede halvad lõhnad

N1: Mudelielemendi nimi ei väljenda täpselt selle olemust

Koodi halb lõhn/heuristika: G16: Obscured Intent; G20: Function Names Should Say What They Do; N1: Choose Descriptive Names; N4: Unambiguous Names; N7: Names Should Describe Side-Effects

Üldine idee: Nimed, mis on liiga lühikesed, ebamäärased, lühendatud või üldised ei väljenda täpselt seda, mida mudeli autor silmas pidas. Mudelielementide nimed peaksid võimalikult täpselt väljendama seda, mida need kirjeldavad. On oluline, et nimed teeksid mudeli või mudelielemendi põhiidee arvusaadavaks. Samuti peaksid mudelielementide nimed olema (vähemalt paketi lõikes) unikaalsed.

Refaktoreerimine: Kui erinevaid mudelielemente vaadates on pidevalt vaja täpsustavaid detaile näiteks dokumentatsioonist järgi uurida, tuleks elementidele täpsemad ja selgitavamad nimed välja mõelda.

Näited:

1. Tegevusdiagramm nimega „Loo“, mis kirjeldab uue toote loomist. Parem ja seletavam nimi oleks „Loo uus toode“.

2. Õppejõu rolli kirjeldamiseks nimi „ÕJ“. Selle nime tähendus võib jääda segaseks, parem nimi oleks „Õppejõud“.
3. 50 klassi abil kirjeldatud klassifikaatorite registri esitamiseks kasutatakse mitut diagrammi, millest igaühele on antud nimi stiilis „Klassifikaatorid 1“, „Klassifikaatorid 2“. Need nimed ei ütle diagrammide sisu kohta piisavalt palju. Selle asemel tuleks igale diagrammile anda täpne ja seletav nimi, mille lugemisel saaks kohe aru, millist osa süsteemist selle diagrammi abil kirjeldatakse.

N2: Nimed sobimatul abstraktsioonitasemel

Koodi halb lõhn/heuristika: N2: Choose Names at the Appropriate Level of Abstraction; N5: Use Long Names for Long Scopes

Üldine idee: Mudeleid ja mudelielemente nimetades tuleks silmas pidada nende esitatavat abstraktsioonitaset ja väljendada seda ka nimedes. Halvasti lõhnavad kõrgtaseme mudelielemendid, mille nimi on liialt detailne ja kirjeldav ning detailsed elemendid, mille nimi on liialt üldine.

Refaktoreerimine: Leides mudeli või mudelielemendi, mille nimi tundub olevalt sobimatul abstraktsioonitasemel, tuleks seda muuta abstraktsioonitasemele vastavaks. Sealjuures võivad abiks olla teiste mudelite või mudelielementide nimed, mis asuvad samal abstraktsioonitasemel.

Näited: Kõrgtaseme kasutusjuhu nimetus „Registreeri kaupa“ ning detailsema kasutusjuhu nimetus „Halda kaupa“, kuigi kauba registreerimine on detailsem tegevus kui kauba haldamine.

N3: Levinud nimetamistavade mittekasutamine

Koodi halb lõhn/heuristika: N3: Use Standard Nomenclature Where Possible

Üldine idee: Nimedest on lihtsam aru saada, kui need järgivad mingeid levinud tavasid, sest nendest saavad üldiselt aru ka teised modelleerijad. Seega pole mõistlik ise hakata nimetamistavasid leiutama. Samuti tuleks meeskonnasiseselt kokku leppida teatud sõnavaras, mida mingite kindlate mudelielementide juures kasutada.

Refaktoreerimine: Kui mudelites ei kasutata kindlat nimetamistava või kasutatakse samade mudelitüüpide nimetamisel segamini mitmeid erinevaid nimetamistavasid, tuleks valida üks konkreetne nimetamistava ning muuta sellele vastavalt eksisteerivate mudelielementide nimesid.

Näited: Kasutusjuht nimega „Tahan teenust hallata“. Kasutusjuhte nimetatakse üldiselt vormis tee-midagi (näiteks „Halda teenust“ ja „Kustuta toode“) või millegi-tegemine (nt „Teenuse haldamine“ ja „Toote kustutamine“).

N4: Kodeeritud nimed

Koodi halb lõhn/heuristika: N6: Avoid Encodings

Üldine idee: Mudelielementide nimedes ei tohiks olla kodeeritud info tüübi või skoobi kohta. Samuti ei tohiks mudelielementide nimed koosneda ainult numbritest või lugejaskonnale mõistetamatutest tähekombinatsioonidest. Tuleb teha kõik, et nimi oleks mudeli lugejaskonnale arusaadav.

Refaktoreerimine: Mudelielemendi nimi, milles on kodeeritud info või mis koosneb ainult numbritest/mõistetamatutest tähekombinatsioonidest, tuleks ümber muuta. Sealjuures tuleks lähtuda halvas lõhnas „N1: Mudelielemendi nimi ei väljenda täpselt selle olemust” kirjeldatud juhistest.

Näited: Olemitüübi atribuutide nimedes ei tohiks kodeeritult olla nende tüübi informatsioon, seega nimi „d_synni_aeg“, kus „d“ tähistab andmetüüpi DATE, ei ole sobiv.

4.4 Mudelite valideerimiste halvad lõhnad

V1: Mudelite valideerimine võtab kaua aega

Koodi halb lõhn/heuristika: E2: Tests Require More Than One Step; T9: Tests Should Be Fast

Üldine idee: Mudelite valideerimine peaks olema kiire protsess. Agiilses maailmas pole enamasti kohta aeganõudvatele mudelite inspekteerimistele. Aeglane mudelite valideerimine võib viidata näiteks mingisugusele mudelite struktuuriga seotud probleemile.

Refaktoreerimine: Mudelid tuleks struktureerida nii, et neid oleks lihtne ja kiire üle vaadata, viidates võimalikult vähe ülejäänud meeskonnaliikmete ja kliendi aega. Seega peaksid mudelid olema loogiliselt struktureeritud ja esitatud ülevaatamiseks sobivas järjekorras.

Näited: CASE vahendis asuvad süsteemi kirjeldavad diagrammid suvaliselt arusaamatute nimedega pakettides, mis teeb raskeks mingi konkreetse diagrammi kiiresti üles leidmise.

V2: Puudulikud valideerimised

Koodi halb lõhn/heuristika: T1: Insufficient Tests

Üldine idee: Puudulikud valideerimised tähendavad, et mudeleid valideeritakse kas liiga vähe või ei valideerita neid üldse. Kuigi puudulikud valideerimised ei pruugi otseselt tähendada probleemi, võivad potensiaalsed vead siiski jääda märkamata, põhjustades hiljem palju pahandust ja parandamist.

Refaktoreerimine: Mudelite valideerimisel on oluline keskenduda süsteemi kõige kriitilisematele ja tihedamalt kasutatavatele osadele. Tuleb teha kindlaks, et need toimivad igas (esinemise tõenäosuse huvipakkuvat nivood ületavas) olukorras korrektselt. Mudeleid tuleks valideerida niikaua, kuni ei avastata rohkem vigu. Valideerimised peaksid olema konstruktiivsed, nii et iga korraga vigade arv väheneb. Kuna eraldi kohtumised mudelite inspektsiooniks koos meeskonnaliikmete ja kliendiga on üldiselt liialt aeganõudvad, tuleks valideerimisi üritada teostada jooksvalt, tehes pidevat koostööd [38].

V3: Valideeritud mudeliosade üle ei peeta arvestust

Koodi halb lõhn/heuristika: T2: Use a Coverage Tool!

Üldine idee: Süsteemide kirjeldamisel tekib üldiselt suur hulk mudeleid. Kui juba valideeritud mudeliosade üle ei peeta arvestust, võib juhtuda, et ühtesid ja samu mudeliosi vaadatakse üle liiga tihti, mis raiskab aega, ning mõndasid potensiaalselt vigu peitvaid mudeliosi ei valideeritagi.

Refaktoreerimine: Süsteemi erinevate alamosade üle tuleks pidada arvestust ja hinnata, kui suur osa süsteemi mudelielementidest on valideeritud. Leides suure tähtsusega mudelite osi, mida pole meeskonna või kliendiga üle vaadatud, siis tuleks seda teha.

V4: Triviaalsete valideerimiste ignoreerimine

Koodi halb lõhn/heuristika: T3: Don’t Skip Trivial Tests

Üldine idee: Triviaalne valideerimine on lihtne kontroll mingisuguse mudelielemendi korrektsuses veendumiseks, mida saab läbi viia minimaalse ajaga. Nende ignoreerimisel ei pruugi küll tekkida väga suured vead ja alati pole triviaalne valideerimine piisav, et probleeme avastada, kuid on võimalik, et märgatakse näiteks mingeid halbu lõhnu.

Refaktoreerimine: Kuna triviaalseid valideerimisi on lihtne ja kiire teha, siis ei tohiks neid ignoreerida.

Näited: Jäetakse ülevaatamata näiteks olemitüüpide nimed ja hiljem selgub, et huvitatud osapooled ei saa neist aru.

V5: Valideerimata mudeliosa

Koodi halb lõhn/heuristika: T4: An Ignored Test Is a Question about an Ambiguity

Üldine idee: Nõuete puudulikkus või segasus võib viia raskusteni teatud mudeliosade poolt kirjeldatava käitumise detailide mõistmisel. Seetõttu võib juhtuda, et taolisi mudeliosi valideerimistel pigem ignoreeritakse, lükates nendega tegelemine kunagi tulevikku.

Refaktoreerimine: Igasuguste arusaamatuste puhul oleks parem kohe nõudeid täpsustada, vajadusel mudelit täiendada ning need siis valideerida. Mida kauem arusaamatuste lahendamisega venitatakse, seda rohkem ressursse kulub sellest tekkinud probleemide lahendamiseks.

V6: Valideerimata piirjuhud

Koodi halb lõhn/heuristika: T5: Test Boundary Conditions

Üldine idee: Tegevusdiagrammidel ja kasutusjuhtude mudelis väljendatava käitumise piirjuhtude ignoreerimine valideerimisel võib tulevikus tekitada probleeme. Põhistsenaariumist õigesti aru saamine on kergem, kuid alternatiivsetes stsenaariumites eksitakse tihemini.

Refaktoreerimine: Kuna piirjuhtudes võivad tihemini vead tekkida, tuleks neid valideerimisel mitte ignoreerida. Samuti oleks hea sealjuures kaasata kedagi, kes on piirjuhuga seotud valdkonnaga lähemalt tuttav ning oskab seega paremini vigu märgata.

V7: Leitud vigade ümbruse ignoreerimine

Koodi halb lõhn/heuristika: T6: Exhaustively Test Near Bugs

Üldine idee: Vead mudelites kipuvad levima. Samuti võivad juba parandatud vead muutuda probleemi allikaks, sest alati ei pruugita vigu õigesti parandada.

Refaktoreerimine: Leides kuskilt mudelist mingi vea, tuleks terve mudel kriitilise pilguga üle vaadata, sest on suur tõenäosus, et see viga polnud üksi.

V8: Vigade mustrite ignoreerimine

Koodi halb lõhn/heuristika: T7: Patterns of Failure Are Revealing

Üldine idee: Vead võivad moodustada mustreid. Vahel on mudelites võimalik üles leida mõni üldisem viga, nähes sellest tulenenud väiksemaid ja nähtavamaid vigu. Leides vaid ühe sellise väikese vea, ei pruugi see suuremale veale viidata. Leides mitu sellist väikest viga, võib see juhtida tähelepanu suuremale tehtud veale.

Refaktoreerimine: Leides mudelist mingi vea, võiks vaadata üle ka teisi seotud mudeleid ja mudelielemente ning uurida, kas sarnaseid vigu leidub kuskil veel. Kui sarnaseid vigu leidub läbivalt, võib see viidata laiemale probleemile.
Näited: Modelleerija on valesti mõistnud üht olulist nõuet, mille tulemusena on väikesed vead tekkinud paljudesse mudeli osadesse. Leides selle tagajärjel tekkinud üksiku väikse vea, parandab ta selle ära ja liigub edasi teiste tööülesannete juurde. Kui ta oleks üle vaadanud teisi seotud mudeleid ja mudelielemente, võinuks ta märgata vigade mustrit, mis oleks vihjanud veale nõuetes.

V9: Valideerimata mudeliosade ignoreerimine
Koodi halb lõhn/heuristika: T8: Test Coverage Patterns Can Be Revealing
Üldine idee: Mudelid on üksteisest sõltuvad – erinevad mudelid võivad viidata sama füüsilise või abstraktse olemi erinevatele aspektidele või kirjeldada seda erinevate huvitatud osapoolte jaoks. Valideeritud või valideerimata mudeli või mudelielemendi uurimine võib vihjata teiste vigade tekkepõhjustele.
Refaktoreerimine: Kui järgida soovitust, mis on antud halvas lõhnas „V3: Valideeritud mudeliosade üle ei peeta arvestust“ ehk juba valideeritud mudeliosade üle peetakse arvestust, siis on modelleerijatel teada, milliseid mudeliosi on valideeritud. Kui leidub täiesti valideerimata mudeliosi, tuleks neile tähelepanu pöörata pigem varem kui hiljem.
Näited: Kontseptuaalses andmemudelis on tehtud viga, mis on põhjustanud vea ka andmebaasioperatsioonide eel- ja järeltingimustes. Kui avastatakse viga andmebaasioperatsioonide lepingutes ja kontseptuaalne andmemudel on veel läbi vaatamata, võib see vihjata seal peituvale probleemile.

4.5 Kataloogi loomise protsess

Kokku tekkis raamatus [35] olnud 66st koodi halvast lõhnast 43 mudeli halba lõhna. Igast lähtekoodi halvast lõhnast ei tekkinud mudelite halba lõhna, sest mõned neist olid liiga koodi-spetsiifilised, mida oli raske mudelitele üldistada, näiteks „J1: Avoid Long Import Lists by Using Wildcards“. Mõne mudeli halva lõhna aluseks võeti mitu lähtekoodi halba lõhna, sest nende üldistamisel mudelitele tekkisid väga sarnase põhimõttega halvad lõhnad, mille eraldi esitamine oleks olnud liiga kordav. Näiteks „H2: Aegunud mudelielement“, mille aluseks olid „C2: Obsolete Comment”, „C5: Commented-Out Code” ja „G9: Dead Code“. Kataloog valmis iteratsioonide kaupa:

1. Esimeses iteratsioonis asendati iga halva lõhna juures koodiga seotud sõnavara mudelitega seotud sõnavaraga ning kustutati laused, mida ei olnud vaja erinevatel põhjustel alles jätta – näiteks koodinäited.
2. Teises iteratsioonis tehti parandusi esialgsesse kataloogi ning lisati uusi inglise keelseid lauseid, mis halva lõhna põhiideed mudelitele paremini üldistaks. Lisaks hakati halbade lõhnade juurde tooma mudelipõhiseid näiteid. Näiteid ei toodud mõnede spetsiifilisemate lõhnade juurde, sest nende äratundmiseks piisab üldise idee mõistmisest.
3. Kolmandas iteratsioonis tehti veel mõningaid viimistlevaid parandusi ning kogu kataloog tõlgiti eesti keelde.
4. Neljandas iteratsioonis tehti lisaparandusi ja toodi näiteid teistest puhaste mudelitega seotud allikatest, samuti hakati lõhnu kategoriseerima ning struktureerima – eraldati üksteisest lõhna üldine idee ja näited jne.
5. Viiendas ja viimases iteratsioonis parandati veelkord struktuuri – viidi kokku sarnased lõhnad, lõhnades eraldati teineteisest lõhna üldine idee ja selle refaktoreerimise kirjeldus. Lisaks toodi valikuliselt erinevate lõhnade juures pildilisi näiteid ja muudeti nimede sõnastust.

Valminud kataloogi sai ka erinevates iteratsioonides kasutada kataloogi enese rekursiivseks parandamiseks. Näiteks esines halbade lõhnade hulgas kordusi, kõik halbade lõhnade nimed ei olnud ühtset stiili järgivad, mõned halvad lõhnad olid üleliigsed jne ning neid sai vastavalt kirjeldatud refaktoreerimistele parandada.

4.6 Tulemuste analüüs

Kataloog ei ole ammendav ning tulevikus võiks seda veelgi täiendada. Halbu lõhnu mudelites võib kindlasti esineda lisaks käesolevas töös defineeritutele ning ainult kataloogi juhendite järgimine ei pruugi garanteerida täielikult puhtaid mudeleid. Sellegipoolest annab kataloog ülevaate levinumatest kitsaskohtadest ja probleemidest modelleerimisel ning aitab neid lahendada.

Enamik defineeritud mudelite halbu lõhnu on piisavalt üldised, et neid rakendada mitmetele erinevatele mudelitüüpidele. Spetsiifilisemate mudelite halbade lõhnade jaoks on järgnevalt esitatud tabel (Tabel 1), mis kirjeldab nende esinemise võimalikkust erinevates mudelitüüpides. Tabelisse on märgitud „x“, kui mudeli halb lõhn võib

konkreetses mudelitüübis esineda. Enne tabelit on esitatud tabeli päises olevate numbrite seletused.

1. Allsüsteeme ja nende seoseid kirjeldav mudel
2. Tegevusdiagramm
3. Kasutusjuhtude mudel
4. Kontseptuaalne andmemudel
5. Valdkonnamudel
6. Olekudiagramm
7. Andmebaasioperatsioonide lepingud
8. Jadadiagramm

Tabel 1. Spetsiifilisemate mudelite halbade lõhnade esinemisvõimalikkus.

Mudeli halva lõhna nimi	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
H4: Halvasti kirjutatud mudelielement			x	x			x	
H7: Valel abstraktsioonitasemel olevad mudelielemendid	x	x	x	x	x	x		x
H10: Kõrge sõltuvus	x			x	x			x
H11: Mudelielemendid vales asukohas			x	x	x			x
H12: Ebaselge kirjeldus		x	x	x			x	
H17: Liigne otsustuspunktide arv		x	x				x	
H18: Sobimatud alamkasutusjuhud			x					
H19: Kasutusjuhtude ebaloogiline järjekord			x					
H20: Väär käitumine piirjuhtudel		x	x					
H21: Ilmselgena tunduvat käitumist ei kirjeldata		x	x					
H25: Läbimõtlemata atribuutide tüübid				x				
H26: Tõeväärtusparameetrid							x	x
S1: Liiga mahukad diagrammid	x	x	x	x	x	x		x
S2: Pikad jooned diagrammis	x	x	x	x	x	x		x
N2: Nimed sobimatul abstraktsioonitasemel	x	x	x	x	x		x	x

Saamaks teada, milliseid lõputöö raames valminud mudelite halbu lõhnu võib pidada unikaalseks või vähemalt vähelevinuks, võrreldi valminud kataloogi varasemate uuringutega ning neis kirjeldatud mudelite halbade lõhnadega. Järgnevalt esitatud tabel (Tabel 2) seab vastavusse siin kirjeldatud mudelite halvad lõhnad ja sarnased varem defineeritud mudelite halvad lõhnad.

Tabel 2. Kataloogi tulemite võrdlemine varasemate uuringutega.

Mudeli halva lõhna nimi	**Viited teistele allikatele, kus on selline mudeli halb lõhn juba välja pakutud**
H9: Üleliigne informatsioon mudelielementides	Large Class [6], Two bits of information in one place [36]
N1: Mudelielemendi nimi ei väljenda täpselt selle olemust	Multiple Definitions of Classes with Equal Names [6]
H5: Mudelielemendid, mida ei lähe vaja teistel mudelielementidel	Unused Element [17], No Incoming [6], Unused Use Case [6], No relationship [36], Lava Flow [17]
H18: Sobimatud alamkasutusjuhud	Extends [6], UCs containing common and exceptional functionality [12], Functional decomposition: Using the include relationship [12], Functional decomposition: Using the extend relationship [12]
H15: Ebatäpsused	An item is on the output that is not in the domain model [36], An item is on model, that are not on the output [36]
H10: Kõrge sõltuvus	BLOB [17]
H19: Kasutusjuhtude ebaloogiline järjekord	Functional decomposition: Using pre and postconditions [12]

Unikaalseks või vähelevinuks võib seega lugeda kõiki ülejäänuid ehk 36 kataloogis defineeritud mudelite halba lõhna.

Edasisteks töödeks võib olla loodud halbade lõhnade kataloogi refaktoreerimine, et tuua iga lõhna tuumprobleem veel selgemalt välja. Praegu leidub lõhnade ideede kirjeldustes kohati ka näiteid, mille eesmärk on lõhna olemust rohkem selgitada. Neid võiks liigutada lõhna kirjelduse osaks olevasse näidete alajaotusesse. Samuti on jätkuvalt põhjust panna kirja uusi mudelite halbu lõhnu ning otsida võimalusi halbade lõhnade tuvastamise ja refaktoreerimise automatiseerimiseks.

4.7 Visioon mudelite halbade lõhnade otsimise ja refaktoreerimise automatiseerimisest

Peatükis 5 vaadati üle 30 süsteemianalüüsi dokumenti, mille eesmärk oli mudelite halbade lõhnade otsimine. Keskmiselt kulus ühe dokumendi läbivaatamiseks 20 minutit, seega kestis kogu protsess kokku ligikaudu kümme tundi, mida võib lugeda väga aeganõudvaks. Olukorra lahendamisel oleks abiks mudelite halbade lõhnade otsimise ja refaktoreerimise automatiseerimisest.

Kuna suur osa töös defineeritud mudelite halbadest lõhnadest on üldised, ei ole neid ka võimalik automaatselt tuvastada. Sellegipoolest leiduvad kataloogis ka mõned spetsiifilisemad mudelite halvad lõhnad, mille otsimise ja refaktoreerimise automatiseerimise läbi oleks võimalik aega säästa. Need on esitatud järgnevalt:

- H5: Mudelielemendid, mida ei lähe vaja teistel mudelielementidel
- H6: Kordused
- H10: Kõrge sõltuvus
- H17: Liigne otsustuspunktide arv
- H19: Kasutusjuhtude ebaloogiline järjekord
- H22: Mitu keelt ühes mudelifailis
- H23: Hoiatuste väljalülitamine
- H26: Tõeväärtusparameetrid
- S1: Liiga mahukad diagrammid
- S2: Pikad jooned diagrammis
- S3: Levinud modelleerimistavade mittejärgimine
- N4: Kodeeritud nimed

Rohkemate defineeritud mudelite halbade lõhnade tuvastamise ja refaktoreerimise automatiseerimine eeldab, et lõhna tuumprobleemi õnnestub formaalselt kirjeldada ning see kirjeldus õnnestub üle kanda modelleerimiskeelt kirjeldavale metamudelile.

5 Andmebaasid I aineprojektide analüüs

Analüüsiks valiti 30 juhuslikku TTÜ „Andmebaasid I“ aine raames valminud ning arvestuse saanud projekti, mis kirjeldasid erinevate üliõpilaste valitud valdkondade infosüsteeme ja sisaldasid süsteemianalüüsi mudeleid. Juhuslikkuse saavutamiseks valiti arvestuse saanud üliõpilaste tähestiku järjekorras olevast nimekirjast 30 esimest projekti, milles võis olla igasuguse kvaliteediga mudeleid. Kõik uuritud mudelid pärinevad ühest õppeaastast.

Andmeprojektide analüüsi eesmärgiks oli uurida, millised mudelite halvad lõhnad esinevad kõige sagedamini. Teades kõige tihedamini esinevaid halbu lõhnu, on neid ka kergem märgata ning modelleerimisel vältida. Tuleb märkida, et kõiki eelnevalt kirjeldatud halbu lõhnu ei olnud võimalik aineprojektides tuvastada, sest mõned neist olid seotud modelleerimisprotsessiga üldiselt ja seega ei ole neid juba valminud projektides näha. Projekte vaadati läbi käsitsi ning iga avastatud mudeli halb lõhn märgiti üles. Kuna esinemissagedust määrati 30 projekti suhtes, ei olnud vahet, kas üks mudeli halb lõhn esines ühes projektis ühe korra või mitmekordselt. Järgnevalt on esitatud tabel (Tabel 3) kümne levinuma mudeli halva lõhnaga ning nende esinemissagedused (kõik esinemissagedused on välja toodud Lisas 1).

Tabel 3. Mudeli halbade lõhnade esinemissagedus.

Mudeli halva lõhna nimi	**Esinemissagedus 30 projekti kohta**
H19: Kasutusjuhtude ebaloogiline järjekord	20
H15: Ebatäpsused	18
S3: Levinud modelleerimistavade mittejärgimine	18
S2: Pikad jooned diagrammis	16
H3: Üleliigne mudelielement	14
H8: Ebakõlad	13
N1: Mudelielemendi nimi ei väljenda täpselt selle olemust	12

Mudeli halva lõhna nimi	Esinemissagedus 30 projekti kohta
H6: Kordused	8
H13: Modelleeritava protsessi mittemõistmine	8
S1: Liiga mahukad diagammid	8

Ilmselt ei saa täie veendumusega öelda, et analüüsi jooksul avastati kõik projektides olevad mudelite halvad lõhnad. Kuna projekte vaadati läbi käsitsi, võisid mõned neist jääda märkamata. Samuti on mõned mudeli halvad lõhnad rohkem silmatorkavamad (näiteks „S2: Liiga pikad jooned diagrammis“) kui teised, mis võis tulemusi kallutada.

Et anda rohkem aimu projektides olevatest mudelite halbadest lõhnadest, tuuakse järgnevalt mõned konkreetsemad näited.

- **H15: Ebatäpsused.** Mittefunktsionaalne nõue ei kirjeldanud konkreetseid arvulisi väärtuseid, vaid kasutas selle asemel omadussõnu nagu „suur“ ja „kõrge“, mida on võimalik mitmeti tõlgendada.
- **H8: Ebakõlad.** Mudelielement oli diagrammil ühe nimega, aga dokumentatsioonis teise nimega.
- **N1: Mudelielemendi nimi ei väljenda täpselt selle olemust.** Mudelielemendi nimetamiseks oli kasutatud vähelevinud lühendit, mille selgitus oli alles dokumendi lõpus.
- **H13: Modelleeritava protsessi mittemõistmine.** Kirjeldatud protsess (piletite ostmine) ei kujutanud korrektselt reaalses elus toimuvat protsessi.

Keskmiselt oli igas aineprojektis kuus erinevat tüüpi mudeli halba lõhna (leitud mediaaniga), millest igaüks võinuks tekitada lühemas või pikemas perspektiivis probleeme tarkvaraarenduses ning suurendada tehnilist võlga. Täpsemalt on mudelite erinevat tüüpi halbade lõhnade esinemiste arv välja toodud Lisas 1. Kuna paljud neist olid hästi märgatavad ning samas lihtsasti parandatavad, võib lõputöö raames valminud kataloogi lugeda kasulikuks abivahendiks mudeli halbade lõhnade tuvastamisel ja seega mudelite puhtana hoidmisel.

6 Kokkuvõte

Mudelite kasutamine tarkvaraarenduses võib olla väga viljakas tegevus. Mudelite kasulikkuse tagamiseks peavad need olema võimalikult lihtsalt ja üheselt arusaadavad. Kuna modelleerimine kipub olema intuitiivsem ja raskemini kontrollitav tegevus kui lähtekoodi kirjutamine, suurendab see vigade tekkimise tõenäosust ja muudab olemasolevate vigade avastamise keerukamaks.

Käesoleva lõputöö lahendatavaks probleemiks olid halvasti loetavad mudelid, mis võivad muutuda arendusprotsessis segavaks faktoriks. Loetavuse parandamiseks võeti eeskujuks puhta koodi disainipõhimõtted, mille üheks eesmärgiks on tagada koodi lihtne loetavus. Neid põhimõtteid mudelitele üldistades kirjeldati mõisteid „puhtad mudelid“ ja „mudelite halb lõhn“.

Lõputöö raames valmis raamatu [35] põhjal kataloog 43st mudelite halvast lõhnast, mis võivad esineda mudelites või modelleerimisprotsessi jooksul. Iga halva lõhna juures kirjeldati selle üldist ideed, anti juhiseid refaktoreerimiseks ning toodi tekstilisi või pildilisi näiteid. Valminud kataloog võiks kasuks tulla äri- ja süsteemianalüütikutele, äriinfotehnoloogia tudengitele jt. Samuti võiksid kirjeldatud mudelite halvad lõhnad muutuda sisendiks edasistele sarnastele uuringutele. Kataloogi abil kontrolliti 30 „Andmebaasid I“ õppeaine raames valminud arvestuse saanud aineprojekti, millest selgus, et mudelite erinevat tüüpi halbade lõhnade arvu mediaan oli kuus. Seega võib järeldada, et kataloogi vajalikkus ja kasulikkus on tõestatud.

Kasutatud kirjandus

[1] Aksioom. [WWW] https://et.wikipedia.org/wiki/Aksioom (24.04.2017)

[2] Aljand, R., Kohtla, M., Lutsar, M. Rollimängu infosüsteem : üliõpilastöö. Tallinna Tehnikaülikool, Tallinn, 2012.

[3] Anti-pattern. [WWW] https://en.wikipedia.org/wiki/Anti-pattern (11.05.2017)

[4] Arcelli, D., Cortellessa, V., Trubiani, C. Performance-based software model refactoring in fuzzy contexts. – *International Conference on Fundamental Approaches to Software Engineering, FASE 2015, London, UK, April 11-18*, 149-164. [*Online*] SpringerLink (15.04.2017)

[5] Arendt, T., Burhenne, M., Taentzer, G. Defining and checking model smells: A quality assurance task for models based on the eclipse modeling framework. – *BElgian-Netherlands Evolution Workshop, BENEVOL 2010, Lille, France, December 16*. [*Online*] Eclipse (07.04.2017)

[6] Arendt, T., Taentzer, G. UML Model Smells and Model Refactorings in Early Software Development Phases. – *Universitat Marburg*. 2010. [*Online*] ResearchGate (07.04.2017)

[7] Broken windows theory. [WWW] https://en.wikipedia.org/wiki/Broken_windows_theory (22.04.2017)

[8] Cabot, J., Clarisó, R., Riera, D. On the verification of UML/OCL class diagrams using constraint programming. – *Journal of Systems and Software*. 2014, 93, 1-23. [*Online*] ScienceDirect (15.04.2017)

[9] Cameron, J. Linguistics: A Short Introduction to the Beating Heart of Human Communications. [WWW] https://www.decodedscience.org/linguistics-short-introduction-beating-heart-human-communications/42808 (25.04.2017)

[10] Dalton, O., Kambla, M., Aasver, T. Freelancer.com infosüsteemi töökuulutuste allsüsteem : üliõpilastöö. Tallinna Tehnikaülikool, Tallinn, 2012.

[11] Degtarjova, I., Babuskin, S. Perearstikeskuse Sinu Arst infosüsteemi suunamiste allsüsteem : üliõpilastöö. Tallinna Tehnikaülikool, Tallinn, 2012.

[12] El-Attar, M., Miller, J. Improving the quality of use case models using antipatterns. – *Software & Systems Modeling*. 2009, 9(2), 141-160. [Online] SpringerLink (04.04.2017)

[13] Evitts, P., Hinchcliffe, D. A UML pattern language. 1st ed. Indianapolis : Macmillan Technical, 2000.

[14] False positives and false negatives. [WWW] https://en.wikipedia.org/wiki/False_positives_and_false_negatives (24.04.2017)

[15] Fernández-Sáez, A. M., Genero, M., Chaudron, M. R. Empirical studies concerning the maintenance of UML diagrams and their use in the maintenance of code: A systematic mapping study. – *Information and Software Technology*. 2013, 55(7), 1119-1142. [*Online*] ScienceDirect (09.05.2017)

[16] Refactoring: improving the design of existing code / M. Fowler, K. Beck, J. Brant, W. Opdyke, D. Roberts, E. Gamma. 1st ed. Boston : Addison-Wesley Professional, 1999.

[17] Fourati, R., Bouassida, N., Abdallah, H. B. A metric-based approach for anti-pattern detection in uml designs. – *Computer and Information Science*. 2011, 17-33. [*Online*] SpringerLink (03.05.2017)

[18] Fowler, M. CodeSmell. [WWW] https://martinfowler.com/bliki/CodeSmell.html (03.04.2017)

[19] Fowler, M. IncludeAndExtend. [WWW] https://martinfowler.com/bliki/IncludeAndExtend.html (25.04.2017)

[20] Galorath, D. D. Software total ownership costs: development is only job one. – *Software Tech. News*. 2008, 11(3). [*Online*] Semantic Scholar (09.04.2017)

[21] General Diagramming Guidelines. [WWW] http://agilemodeling.com/style/general.htm (24.04.2017)

[22] Ghannem, A., El Boussaidi, G., Kessentini, M. Model refactoring using examples: a search-based approach. – *Journal of Software: Evolution and Process*. 2014, 26(7), 692-713. [*Online*] ResearchGate (08.04.2017)

[23] Gogolla, M., Hilken, F., Oberweis, A., Reussner, R. Model Validation and Verification Options in a Contemporary UML and OCL Analysis Tool. – *Modellierung*. 2016, 205-220. [*Online*] Semantic Scholar (15.04.2017)

[24] Grossman, L. Nov. 10, 1999: Metric Math Mistake Muffed Mars Meteorology Mission. [WWW] https://www.wired.com/2010/11/1110mars-climate-observer-report/ (21.04.2017)

[25] Heikkilä, M., Bouwman, H., Heikkilä, J., Solaimani, S., Janssen, W. Business model metrics: an open repository. – *Information Systems and e-Business Management*. 2016, 14(2), 337-366. [*Online*] ResearchGate (11.05.2017)

[26] Heuristic (computer science). [WWW] https://en.wikipedia.org/wiki/Heuristic_(computer_science) (18.05.2017)

[27] Himma, T. E-dokumentide vahendamise infosüsteemi tellimuste allsüsteem (AS Telema näitel) : üliõpilastöö. Tallinna Tehnikaülikool, Tallinn, 2012.

[28] Jalakas, O., Reinok, V., Jürgenson, K. Kuulutuste portaal : üliõpilastöö. Tallinna Tehnikaülikool, Tallinn, 2012.

[29] Kruchten, P., Nord, R. L., Ozkaya, I. Technical Debt: From Metaphor to Theory and Practice. – *Ieee software*. 2012, 29(6), 18-21. [*Online*] IEEE Xplore (09.05.2017)

[30] Kõrm, A., Välman, A. Suusa- ja mäevarustusehoolduse IS : üliõpilastöö. Tallinna Tehnikaülikool, Tallinn, 2012.

[31] Lindland, O. I., Sindre, G., Solvberg, A. Understanding Quality in Conceptual Modeling. – *IEEE Software*. 1994, 11(2), 42-49. [*Online*] ResearchGate (03.04.2017)

[32] Liu, Y., Chen, W., Arendt, P., Huang, H. Z. Toward a better understanding of model validation metrics. – *Journal of Mechanical Design*. 2011, 133(7), 071005. [*Online*] Relialab (13.04.2017)

[33] Major, M. L., McGregor, J. D. Using guided inspection to validate UML models. – *25th Annual Software Engineering Workshop*. 1999, 485-507. [*Online*] Semantic Scholar (13.04.2017)

[34] Manifesto for Agile Software Development. [WWW] http://agilemanifesto.org (18.05.2017)

[35] Martin, R. C. Clean Code: A Handbook of Agile Software Craftsmanship. 1st ed. New York – Prentice Hall, 2008.

[36] McDonald, K., Matts, C. Seven modelling smells. [WWW] https://www.infoq.com/articles/seven-modelling-smells (04.05.2017)

[37] Misbhauddin, M., Alshayeb, M. UML model refactoring: a systematic literature review. – *Empirical Software Engineering*. 2015, 20(1), 206-251. [*Online*] SpringerLink (08.04.2017)

[38] Model Reviews: Best Practice or Process Smell? [WWW] http://agilemodeling.com/essays/modelReviews.htm (15.04.2017)

[39] Model Storming: An Agile Best Practice. [WWW] http://agilemodeling.com/essays/modelStorming.htm (15.04.2017)

[40] Mohagheghi, P., Dehlen, V., Neple, T. Definitions and approaches to model quality in model-based software development–A review of literature. – *Information and Software Technology*. 2009, 51(12), 1646-1669. [*Online*] Sintef (02.04.2017)

[41] Mohagheghi, P., Dehlen, V., Neple, T. Towards a Tool-Supported Quality Model for Model-Driven Engineering. – *Quality in Modeling*. 2008, 74. [*Online*] ResearchGate (02.04.2017)

[42] Soeken, M., Wille, R., Drechsler, R. Verifying dynamic aspects of UML models. – *Design, Automation & Test in Europe Conference & Exhibition, DATE 2011, Grenoble, France, March 14-18*, 1-6. [*Online*] Semantic Scholar (13.04.2017)

[43] Soni, M. Defect Prevention: Reducing Costs and Enhancing Quality. [WWW] https://www.isixsigma.com/industries/software-it/defect-prevention-reducing-costs-and-enhancing-quality/ (01.04.2017)

[44] Surprise! Software Rots! [WWW] http://www.agile-process.org/change.html (09.04.2017)

[45] Süld, M., Pukk, K. Taksofirma infosüsteemi tellimuste haldamise allsüsteem : üliõpilastöö. Tallinna Tehnikaülikool, Tallinn, 2012.

[46] Thramboulidis, K. C., Tranoris, C. S. Developing a CASE tool for distributed control applications. – *The International Journal of Advanced Manufacturing Technology*. 2004, 24(1-2), 24-31. [*Online*] SpringerLink (20.05.2017)

[47] UML 2 Activity Diagramming Guidelines. [WWW] http://agilemodeling.com/style/activityDiagram.htm (19.04.2017)

[48] UML 2 Class Diagramming Guidelines. [WWW] http://agilemodeling.com/style/classDiagram.htm (02.05.2017)

[49] UML 2 State Machine Diagramming Guidelines. [WWW] http://agilemodeling.com/style/stateChartDiagram.htm (19.04.2017)

[50] UML 2 Use Case Diagramming Guidelines. [WWW] http://agilemodeling.com/style/useCaseDiagram.htm (19.04.2017)

[51] Van Galen, W. Use Case Preconditions: A Best-Kept Secret? [WWW] https://www.batimes.com/articles/use-case-preconditions-a-best-kept-secret.html (22.04.2017)

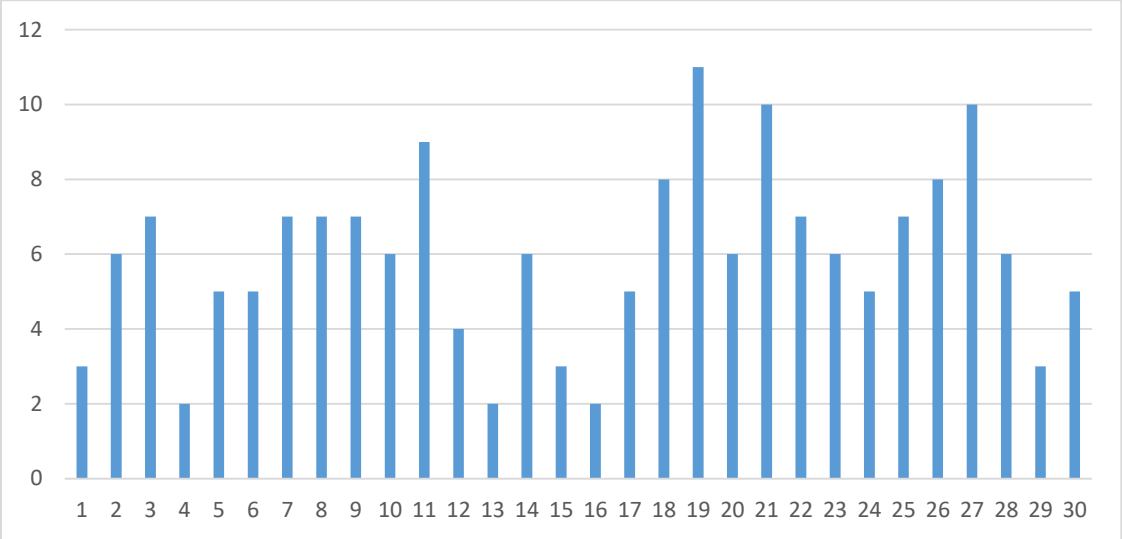
[52] Webley, K. Top 10 NASA Flubs. [WWW] http://content.time.com/time/specials/packages/article/0,28804,1982672_1982673_1982667,00.html (21.04.2017)

Lisa 1 – Mudelite halbade lõhnade esinemine aineprojektides

Tabel 4. Kõikide mudelite halbade lõhnade esinemissagedus.

Mudeli halva lõhna nimi	**Esinemissagedus 30 projekti kohta**
H19: Kasutusjuhtude ebaloogiline järjekord	20
H15: Ebatäpsused	18
S3: Levinud modelleerimistavade mittejärgimine	18
S2: Pikad jooned diagrammis	16
H3: Üleliigne mudelielement	14
H8: Ebakõlad	13
N1: Mudelielemendi nimi ei väljenda täpselt selle olemust	12
H6: Kordused	8
H13: Modelleeritava protsessi mittemõistmine	8
S1: Liiga mahukad diagammid	8
H9: Üleliigne informatsioon mudelielementides	7
H12: Ebaselge kirjeldus	7
H4: Halvasti kirjutatud mudelielement	5
S4: Ebakindel struktuur	5
H17: Liigne otsustuspunktide arv	4
H5: Mudelielemendid, mida ei lähe vaja teistel mudelielementidel	3
H11: Mudelielemendid vales asukohas	2
H20: Väär käitumine piirjuhtudel	2
N3: Levinud nimetamistavade mittekasutamine	2
H1: Sobimatu informatsioon	1
H18: Sobimatud alamkasutusjuhud	1
H21: Ilmselgena tunduvat käitumist ei	1

Mudeli halva lõhna nimi	Esinemissagedus 30 projekti kohta
kirjeldata	
H22: Mitu keelt ühes mudelifailis	1
N2: Nimed sobimatul abstraktsioonitasemel	1



Joonis 15. Erinevat tüüpi mudelite halbade lõhnade arv igas projektis.