TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Liidia Laada 142463IAPB

VIDEOFAILIST SEGMENTIDE VÄLJALÕIKAMISE JA NENDE ÜHENDAMISE FUNKTSIONAALSUSE LISAMINE ORNET MEDITSIINILAHENDUSTE TARKVARASSE

bakalaureusetöö

Juhendaja: Gunnar Piho
PhD

Tallinn 2020

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: Liidia Laada

01.01.2020

Annotatsioon

ORNet meditsiinilahenduste perekonda kuuluvad kolm toodet. ORNet Editor on tarkvarasüsteemiks ilma riistvaralise osata. ORNet Surgery ja ORNet Pathology koosnevad nii tark- kui ka riistvaralistest komponentidest.

Autori ees püstitatud ülesandeks oli arendada uus videofailide taasesitamise ja töötlemise eest vastutav komponent. Seda komponenti hakatakse tulevikus kasutama kõigis ORNet perekonda kuuluvates lahendustes.

Töös on esitatud funktsionaalsed ja mittefunktsionaalsed nõuded uuele komponendile, ORNet tarkvara arhitektuur ja komponendi koodi kirjutamisel kasutatud mustrid. Töö käigus oli loodud 18 uut klassi, 10 uut liidest ja kirjutatud rohkem kui 1550 rida tootmiskoodi. Töös on samuti antud koostööskeemid, mis näitavad, kuidas kood töötab, ning klassidiagrammid, mis näitavad, kuidas klassid ja liidesed suhtlevad omavahel. Lisas on esitatud uute ja täiendatud klasside kirjeldused.

Töö käigus koguti ja analüüsiti järgnevaid meetrikaid: reakate, harukate, jadakate, koodi ridade arv, hooldatavuse indeks, klasside omavaheline sõltuvus, pärimise sügavus, tsüklomaatiline keerukus ja N-Path keerukus. Töö käigus oli kirjutatud 137 ühik- ja integratsioonitesti.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 44 leheküljel, 4 peatükki, 22 joonist, 9 tabelit.

Abstract

Adding functionality of cutting video segments from a video file and merging them to the ORNet medical solutions software

ORNet mediacal solutions group consists of three products. ORNet Editor is a software system without hardware parts. ORNet Surgery and ORNet Pathology have both software and hardware components.

The object of this work is the ORNet medical solutions software. This is a desktop application created using Windows Presentation Foundation technology.

The administration of the company Bait Partner Ltd. has decided to extend the functionality of the software. The author's task was to develop a new component responsible for the playback and processing of video files. This component is going to be used in all the medical solutions belonging to the ORNet family.

Since this work has a limited size, its goal is to add the following functionality to the new component: cutting tagged video segments from a video file and merging them to a new video file.

In the work the test-driven development aproach is partially used. In the work functional and nonfunctional requirements for the new component, the architecture of the ORNet software and design patterns used during writing the code are provided. During the work 18 new classes and 10 new interfaces were created and more than 1550 lines of production code were written. In the work collaboration and class diagrams showing how the classes and interfaces interact and are related to each other are also given. In a supplement the descriptions of the new and complemented classes are provided.

During the work the following metrics were collected and analyzed: line coverage, branch coverage, sequence coverage, source lines of code, maintainability index, class coupling, depth of inheritance, cyclomatic complexity and N-Path complexity. During the work 137

unit and integration tests were written. Despite this the test covereage is insufficient. The written code needs to be refactored. The classes depend too strongly on each other. The methods are too large, take too many arguments and have too high complexity. The problem is also that the code uses poorly object-oriented design and inheritance.

The feedback of the company is given in the supplement 3 Company feedback.

The thesis is in Estonian and contains 44 pages of text, 4 chapters, 22 figures, 9 tables.

Lühendite ja mõistete sõnastik

Aktiivne kirje	*Active record.* Objekt, mis esindab rida andmebaasi tabelis või vaates, kapseldab juurdepääsu andmebaasi ja lisab domeeni loogika andmetele [1].
HI	*Maintainability indeks*. Hooldatavuse indeks.
HK	*Branch coverage*. Harukate.
HM	*Halstead volume*. *Halstead* maht.
JK	*Seguence coverage*. Jadakate.
konkateneerimise „protokoll“	*The concat „protocol“.* Üks programmi ffmpeg videofailide ühendamise võimalustest [2].
KOS	*Class coupling*. Klasside omavaheline sõltuvus.
käsk	*Command*. Sisendi mehhanism Windows Presentation Foundation tehnoloogias.
liides	*Interface*. Obekt-orienteeritud programmeerimiskeele struktuur, mis kirjeldab, mis avalikke meetodeid klass peab omama.
mudel	*Model*. *MVVM* arhetiktuurse disainimustri osa.
MVVM	Model-View-ViewModel. Arhetiktuurne disainimuster.
NPK	*N-Path complexity. N-Path* keerukus.
omadus	*Property*. Klassi liige C# keeles. Omadus on klassi meetod, mis on liigipääsetav samal viisil nagu klassi väli [3]. Omadus võib kaitsta klassi välju muutmisest ilma objekti teadmiseta.
pisipilt	Videofailist kindlas positsioonis võetud pildi madalama kvaliteediga vähendatud koopia.
PS	*Depth of inheritance*. Pärimise sügavus.
SLOC	*Source lines of code*. Koodi ridade arv.
TDD	*Test Driven Development*. Testimisel põhinev arendus.
testitavad read	Kõik programmi read, mida on võimalik katta testidega.
TK	*Cyclomatic complexity.* Tsüklomaatiline keerukus.
vaade	*View*. *MVVM* arhetiktuurse disainimustri osa.
vaate mudel	*Viewmodel*. *MVVM* arhetiktuurse disainimustri osa.
WPF	*Windows Presentation Foundation.*

ümberkodeerimine	Video- ja audiofailide formaadi ja/või kodeerimisviisi muutmine.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Ettevõtte Bait Partner OÜ [4] toodab meditsiinilahenduste rühma, mis kannavad ühist nimetust ORNet [5]. ORNet-i perekonda kuuluvad kolm toodet:

- ORNet Surgery [6, 7];
- ORNet Pathology [8, 9];
- ORNet Capture [10, 11].

ORNet Editor on tarkvarasüsteemiks ilma riistvaralise osata. ORNet Surgery ja ORNet Pathology koosnevad nii tark- kui ka riistvaralistest komponentidest.

Kõigi kolme toote tarkvara põhineb ühisel koodibaasil. Seega, osa funktsionaalsusest on ühine kõigi toodete jaoks. Teine osa funktsionaalsusest on omane mingile konkreetsele tootele.

ORNet Capture on kõige lihtsam kolmest toodetest. Selle toote tarkvara lubab vaadata, salvestada ning töödelda pilte, video- ja audiofaile. ORNet Pathology samuti omab kõiki ülevalpool loetletud funktsioone. Lisaks sellele, ORNet Pathology-l on patoloogia laboris olulised funktsioonid nagu nt:

- Andmete anumatele printimine;
- Patoloogia lampide valguse intensiivsuse reguleerimine;
- Anumate verifitseerimine ruutkoodi abil.

Peamine firma toodang on ORNet Surgery. Peale failide salvestamise ja töötlemisega seotud funktsionide, oskab ORNet Surgery juhtida operatsioonisaalides olevaid seadmeid.

Ettevõte püüab korrapäraselt arendada toodetavate toodete funktsionaalsust.

Antud töös on analüüsitud uue videofailide taasesitamise ja töötlemise eest vastutava komponendi kavandamine ja arendamine.

Töö lõpus autor analüüsib arendatud komponendi kvaliteeti.

1.1 Probleem

Kõik ORNet-i tooted võimaldavad salvestada video- ja audiofaile. Video salvestamise ajal salvestub ka heli. Video salvestamine toimub kaamera abil ning audio salvestamine toimub mikrofoni kaudu. Selle tagajärjel video salvestamisel moodustatakse mitte üks, vaid kaks faili: üks fail sisaldab videoinformatsiooni ning teine – audioinformatsiooni.

Antud töö alustamise hetkel eksisteeris ORNet-i tarkvaras juba komponent, mis võimaldas salvestatud videofaili taasesitada, taasesitamist peatada ja taasalustada, faili ära kustutada või eksportida USB-mäluseadmele, valida failist teatud segmenti ning valitud segmenti eksportida soovitud asukohale arvutis. Antud funktsionaalsus on ühine kõigi kolme ORNet lahenduse tarkvara jaoks.

Firma juhtkond otsustas videofailide taasesitamise ja töötlemisega seotud funktsionaalsust laiendada ning lisada tarkvarasse järgnevaid omadusi.

1. Võimalus märgendada videofaili segmente;
2. Võimalus lõigata videofailist suvaline arv segmente ja liita neid uueks videofailiks;
3. Videofaili eksportimisel ja segmentide väljalõikamisel näidata protsessi progressi;
4. Võimalus lisada videofailile audiokommentaare;
5. Võimalus seadistada heli tugevust;
6. Võimalus seadistada heli sisendeid ja väljundeid;
7. Võimalus liikuda videofaili taasesitamisel tagasi teatud arvu sekundite võrra;
8. Võimalus taasesitada videofaili erineva kiirusega;
9. Võimalus võtta hetketõmmiseid.

Uued funktsionaalsed nõuded tekkisid riigihangetel osalemise tagajärjel ja lähtuvalt olemasolevate klientide soovidest.

Seejärel, otsustati lisada tarkvarale võimet töödelda H.265 standardi [12] järgi kodeeritud videofaile. Enne seda ORNet-i seadmete tarkvara sai töödelda ainult H.264 standardi [13] järgi kodeeritud videofaile. Otsuse põhjuseks oli asjaolu, et H.265 standard on uuem ja kodeerib videofaile efektiivsemalt ning paljud tänapäevased videokaamerad toetavad seda kodeerimisviisi.

Olemasolevas komponendis olid ka muud lahendamist vajavad probleemid. Esiteks, segmendi väljalõikamine videofailist toimus kasutaja arvutis väga aeglaselt. Teine probleem seisnes selles, et komponendi koodi oli äärmiselt raske korras hoida, teostada koodis muudatusi, lisada uusi omadusi. WPF (*Windows Presentation Foundation*) [14] rakenduste kirjutamisel tavaliselt kasutatakse MVVM (*Model-View-ViewModel*) [15] arhetiktuurset disainimustrit. Mainitud komponendi kirjutamisel aga ei võetud seda mustrit arvesse. Kood vajas kindlasti väga palju refaktoreerimist [16].

Pidades silmas mainitud asjaolusid, võeti vastu otsus kirjutada uus video failide taasesitamise ja töötlemise eest vastutav komponent.

1.2 Eesmärk

Töö uue komponendi kallal oli alustatud varasemalt teise töötaja poolt. Oli realiseeritud kasutajaliides ja järgnevad funktsioonid:

1. Videofaili taasesitamine;
2. Videofaili taasesitamise peatamine ja taasalustamine;
3. Videofaili segmentide märgendamine.

Antud töö raames oli autorile püstitatud järgmine eesmärk:

Uus komponent peab võimaldama kasutajal lõigata videofailist segmente määratud kohtades (märgendatud segmente). Videosegmentide väljalõikamine peab olema võimalik nii H.264 kui ka H.265 standardi järgi kodeeritud video puhul. Komponent peab olema võimeline samaaegselt lõikama videofailist mitu segmenti ja ühendama neid üheks uueks videofailiks. Protsess peab toimuma võimalikult kiiresti.

Ülejäänud antud töö probleemist tulenevad eesmärgid jäävad töö skoobist välja töö piiratud mahu tõttu.

2 Metoodika

2.1 ORNet perekonna tooted

Nagu mainitud eespool (vt Sissejuhatus, lk 11), on käeaoleva töö objektiks ORNet meditsiinilahenduste tarkvara. Edasine tekst on kirjutatud tuginedes allikatele [6, 8, 10].

Kolm toodet: ORNet Surgery, ORNet Pathology ja ORNet Capture – omavad ühist aluskoodi. Seega, teatud funtsionaalsus on ühine kõigi kolme toote jaoks. See funktsionaalsus moodustab ORNet Capture-it. ORNet Capture lubab vaadata, salvestada ning töödelda pilte, video- ja audiofaile. ORNet perekonna toodete ühised funktsioonid on esitatud tabelis 1.

Teisest küljest, mõned funktsioonid on spetsifilised konkreetsele tootele.

Lisaks tabelis 1 loetletud funktsioonidele, ORNet Pathology-l on patoloogia laboris olulised funktsioonid nagu nt:

- Andmete anumatele printimine;
- Patoloogia lampide valguse intensiivsuse reguleerimine.
- Anumate verifitseerimine ruutkoodi abil.

ORNet Pathology-le spetsiifilised funktsioonid on toodud tabelis 2.

ORNet Surgery omab, peale failide salvestamise ja töötlemisega seotud funktsionide, operatsioonisaalides olulisi funktsioone, mis seisnevad teiste seadmete juhtimises. Need funktsioonid on toodud tabelis 3.

ORNet lahenduste funktsionaalsus on skemaatiliselt näidatud joonisel 1.

ORNet-i tarkvara on lokaliseeritud viide keelde: eesti, inglise, soome, rootsi ja itaalia. Seadmeid saab juhtida kasutades nii puutetundliku ekraani ja ekraaniklaviatuuri kui ka hiirt ja tavalist klaviatuuri Tarkvara näeb ette tagasisidevormi, mida saab kasutada

probleemide tekkimise korral. Patsientide ja uuringute andmeid on võimalik laadida ORNet-i tarkvarasse haigla infosüsteemist kahel viisil:

- HL7 (Health Level Seven International) [17, 18, 19] sõnumitena;
- DICOM (Digital Imaging and Communications in Medicine) standardile [20, 21] vastavate töönimekirjadena.

Tabel 1. ORNet perekonna toodete põhilised ühised funktsioonid.

Kategooria	**Funktsioonid**
Piltidega seotud funktsioonid	Vaatamine; Salvestamine; Töötlemine; Printimine.
Audiofailidega seotud funktsioonid	Taasesitamine; Salvestamine.
Videofailidega seotud funktsioonid	Taasesitamine; Salvestamine; Töötlemine.
Kaameraga seotud funktsioonid	Videopildi saamine mitmest erinevast kaamerast; Videopildi juhtimine soovitud monitoorile; Videopildi suurendamine/vähendamine; Kaamera positsiooni muutmine; Kaamera fookuse ja kaamerasse sattuva valguse hulga seadistamine; Kaamera signaali ulatumise väljaspool ruumi takistamine;
Mikrofooni/kõlaritega seotud funktsioonid	Heli tugevuse seadistamine; Mikrofooni valimine; Kõlarite valimine.
Teised funktsioonid	Teise ORNet-i tööjaamaga ühenduse loomine ja konsultatsiooni läbiviimine; Raadio kuulamine; Kontroll-loendi kuvamine.

Patsiendid ja uuringud võivad olla lisatud, muudetud ja kustutatud ka käsitsi. Pilte on võimalik eksportida haigla PACS (picture archiving and communication system) arhiivi [21] DICOM salvestamisfunktsiooni abil.

Tabel 2. Põhilised ORNet Pathology tootele spetsiifilised funktsioonid.

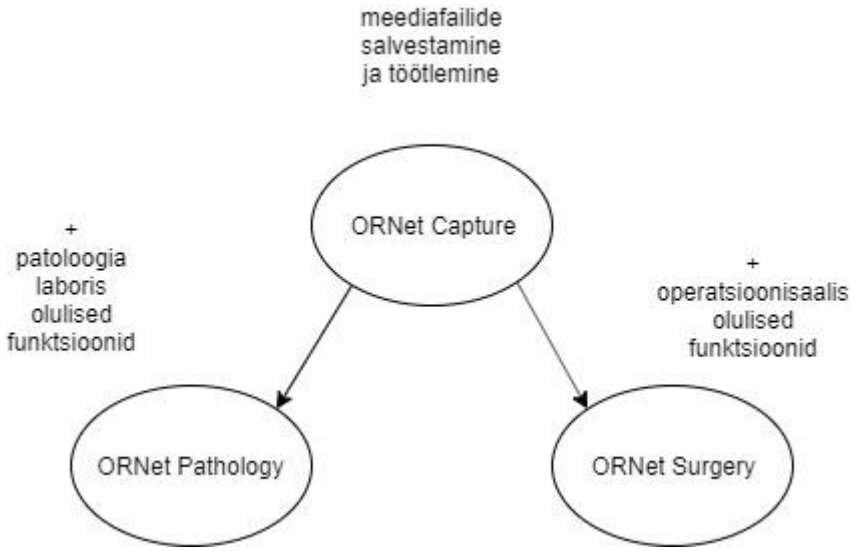
Kategooria	**Funktsioonid**
Anumate nimetamine	Erinevate süsteemide kasutamine anumate nimetamisel.
Teiste seadmete juhtimine	Andmete printimine anumatele kasutades Fa-Tech printerit (Fa-Tech Diagnostics NOVA Laser Cassettes Marker, [22]); Patoloogia lampide sisse-/väljalülitamine; Lampide valguse intensiivsuse reguleerimine.
Piltide töötlemine	Patoloogia piltidele anumate numbrite kandmine.
Teised funktsioonid	Anumate verifitseerimine ruutkoodilugeja abil.

Tabel 3. Põhilised ORNet Surgery tootele spetsiifilised funktsioonid.

Kategooria	**Funktsioonid**
Teiste seadmete juhtimine	Lampide juhtimine ruumis; Operatsioonilaudade positsiooni, kõrguse ja kallaku seadistamine; Temperatuuri, niiskuse ja hapniku taseme reguleerimine; Teiste seadmete häiresituatsioonidest teatamine; Pistikupesade sisse- ja väljalülitamine; Teiste meditsiini- ja muude seadmete kontroll.

Erinevates riikides on paigaldatud umbes 130 ORNet meditsiinilahendust. Tooteid kasutavad mitmed haiglad Soomes [23]. Nende seas on Savonlinna haigla (Itä-Savon Sairaanhoitopiirin Kunatayhtymä), Joensuu haigla (Pohjois-Karjalan keskussairaalaa), Vaasa haigla (Vaasan Keskussairaala), Turku Ülikooli haigla (Turun Yliopistollinen Keskussairaala), Oulu Ülikooli haigla (Oulun Yliopistollinen Sairaala) ja Helsingi Ülikooli haigla laboratoorium (Helsinkin Yliopistollisen Sairaalaan Laboratoriot). Eestis

kasutatakse ORNet lahendusi SA Ida-Viru Keskhaiglas ja SA Tartu Ülikooli Kliinikumis. Tooted on auditeeritud organisatsiooni TÜV Nord poolt.



Joonis 1. ORNet perekonna toodete funktsioonid.

2.2 Tööriistad

Tarkvara kujutab endast töölauarakendust. See on loodud kasutades .NET raamistikus (versioon 4.5) [24] WPF tehnoloogiat ja C# keelt [24].

Käeasoleva töö käigus kasutati arenduskeskkonnana VisualStudio Professional 2017 koos laiendusega ReSharper [25].

Testimise otstarbeks kasutati NUnit raamistikku (versioon 3.0.1) [26]. Libaobjektide (mock objects) loomiseks testimisel kasutati Rhino Mocks raamistikku (versioon 3.6.0.0) [27].

Meediafailide metaandmete (koodeki tüüp ja nimi, resolutsioon, pikkus, bitikiirus, diskreetimissagedus jne) kättesaamiseks kasutati programmi ffprobe.exe [28]. Meediafailide töötlemiseks (lõikamine ja ühendamine) kasutati programmi ffmpeg.exe [29, 30]. Mõlema programmi importimiseks vastavatesse VisualStudio projektidesse kasutati Nuget paketti (FFmpeg.Shared, versioon 4.0.2, [31, 32]). Samuti meediafailidest informatsiooni saamiseks kasutati programmi MediaInfo (versioon 18.12) [33].

Logimisega seotud koodi kirjutamisel kasutati NLog raamistikku (versioon 4.5.11) [34].

Lähtekoodi kompileerimise, testimise ja analüüsimise automatiseerimise eesmärgil kasutati Continua CI rakendust [35].

Versioonikontrolliks kasutati programmi TortoiseSVN [36].

Töö organiseerimiseks kasutati projektihaldustarkvara TargetProcess [37].

Skeemide, koostööskeemide ja klassidiagrammide joonistamiseks kasutati rakendusi draw.io [38] ja yEd [39].

Koodi meetrikate arvutamiseks kasutati tööriista OpenCover [40] (tsüklomaatiline keerukus, N-Path keerukuss, rea-, jada- ja harukate, klassi koodi ridade arv) ja arenduskeskonda VisualStudio Professional 2017 (hooldatavuse indeks, pärimise sügavus ja klasside omavaheline sõltuvus).

Tehnoloogiate valik on antud kliendi poolt ning seetõttu ei saa olla analüüsitud või põhjendatud.

2.3 Tööprotsess

Antud projekti käigus kasutati vähemalt osaliselt arendamisel TDD (*Test Driven Development*, testimisel põhinev arendus) lähenemisviisi [41]. Kõigi klasside jaoks loodi ka liideseid (liidese all peetakse siin silmas obekt-orienteeritud programmeerimiskeele struktuuri, mis kirjeldab, mis avalikke meetodeid klass peab omama). Pärast igat etappi, mis oli seotud programmi koodi kirjutamisega:

1. Kontrolliti, kas kirjutatud kood vastab puhta koodi nõuetele [42]. Vastuolude märkamise korral koodi refaktoreeriti.
2. Kontrolliti, kas olemasolevad ühik- ja integratsioonitestid lähevad endiselt läbi (on rohelised). Juhul, kui testid ei läinud läbi, neid parandati.
3. Kontrolliti, kas kirjutatud kood on kaetud testidega piisaval määral (meetodite jada- ja harukatte väärtused on vähemalt 70% [43]). Ebapiisava koodikatte korral, analüüsiti, kas testimata koodi jaoks on võimalik kirjutada teste. Kui see oli võimalik, lisati uusi ühik- või integratsiooniteste.

Töö koosnes järgmistest etapidest:

1. Loodi klass VideoSegment. See kannab informatsiooni ühest märgendatud videosegmendist.
2. Loodi abstraktne klass VideoCuttingServiceBase ja selle konkreetne klass H265EncodedVideoCuttingService. H265EncodedVideoCuttingService-i funktsioon seisneb segmentide väljalõikamises H.265 standardi järgi kodeeritud videofailidest. H265EncodedVideoCuttingService kasutab oma ülesande täitmiseks programmi ffmpeg.exe.
3. Loodi ka analoogne konkreetne klass H.264 standardi jaoks, mille nimetus on H264EncodedVideoCuttingService.
4. Edasi muudeti videosegmentide väljalõikamisega seotud koodi kontseptsiooni. Oli loodud ühine klass VideoCuttingService, kuna H264EncodedVideoCuttingService-l ja H265EncodedVideoCuttingService-l oli väga palju ühist ning erinevus oli minimaalne (1 koodi rida).
5. Lisati klass VideoMergingService. Selle klassi ülesandeks on videosegmentide ühendamine üheks videofailiks. Klass kasutab oma ülesande täitmiseks programmi ffmpeg.exe.
6. Loodi klassid FfMpegCuttingArgumentsConstructor ja FfMpegMergingArgumentsConstructor. Need klassid konstrueerivad argumente ffmpeg programmi videosegmentide väljalõikamise ja ühendamise käskudele vastavalt.
7. Loodi klass CueTrackConverter. Alguses klass CueTrackConverter omas ainult ühte meetodit Convert(...). Hilem sellele lisandus meetod ConvertBack(...).
8. Kasuatajaliideses (aken, MediaEditorView) loodi nupp „Merge“. Akna vaate mudelis (MediaEditorViewModel, *viewmodel*, *MVVM* arhetiktuurse disainimustri osa) loodi vastav käsk (*command*; sisendi mehhanism *Windows Presentation Foundation* tehnoloogias) MergeVideoSegmentsCommand. Nupu vajutamisel loodud käsu teostamise tagajärjel toimuvad järgmised tegevused:
    1. Märgendatud videosegmente lõigatakse video failist;
    2. Väljalõigatud videosegmendid liidetakse uueks failiks;
    3. Luuakse uus Media tüüpi objekt;
    4. Loodud Media tüüpi objekti salvestatakse andmebaasi;
    5. Uus videofail avatakse kasutajaliideses.
9. Loodi klassi ThumbnailExtractingService. Selle klassi ülesandeks on videofailist pildi võtmine. Klass kasutab oma ülesande täitmiseks programmi ffmpeg.exe.

10. Loodi klassi FfMpegThumbnailExtractingArgumentsConstructor. See klass konstrueerib argumente ffmpeg programmi videofailist pildi võtmise käsu jaoks.
11. Lisati käsule MergeVideoSegmentsCommand videofaili pisipildi (thumbnail, videofailist kindlas positsioonis võetud pildi madalama kvaliteediga vähendatud koopia) loomise funktsionaalsust.
12. Loodi ühine klass FfMpegArgumentsConstructor kolme klassi FFMpegCuttingArgumentsConstructor, FFMpegMergingArgumentsConstructor ja FFMpegThumbnailExtractingArgumentsConstructor asemele.
13. Korraldati kood ümber järgneval viisil:
    1. Klass VideoMergingService saab VideoCuttingService ja ThumbnailExtractingService tüüpi argumente konstruktori kaudu;
    2. VideoMergingService.MergeVideoSegmentsAsync(...) meetodi sees toimub VideoCuttingService.CutVideoSegmentsAsync(...) ja ThumbnailExtractingService.CreateThumbnailAsync(...) meetodite väljakutsumine (varem see toimus MediaEditorViewModel-is käsu MergeVideoSegmentsCommand sees).

    Muudatuse teostamisel lähtuti puhta koodi nõuetest [42].
14. Oli lisatud võimalus töödelda audiofaile, mis on salvestatud paralleelselt videofailidega. Protsess toimub järgnevalt:
    1. Määratud kohtades lõigatakse segmente videofailist välja;
    2. Määratud kohtades lõigatakse segmente audiofailist välja (see toimub eelmise punktiga paralleelselt, kasutades sama ffmpeg.exe programmi käsku);
    3. Väljalõigatud video- ja audiosegmendid ühendatakse üheks videofailiks. Saadud videofail omab audiovoogu.
15. Loodi klass FfMpegFilesCreationProgressCalculator. Selle klassi ülesandeks on programmi ffmpeg poolt failide loomise protsessi kulu arvutamine protsentides.
16. Loodi klass CutMediaSegmentsParamsRecalculator. Selle klassi ülesandeks on märgendatud videosegmentide alguspositsioonide ja kestvuste ümberarvutamine uues ühendamise tagajärjel tekitatud videofailis.
17. Lisati järgmine funktsionaalsus: kui märgendatud videosegmente lõigatakse videofailist välja ja ühendatakse, siis märgendid jäävad alles ka uues tekitatud videofailis.

18. Lisati järgmine funktsionaalsus: sellel ajal, kui märgendatud videosegmente lõigatakse välja ja ühendatakse, näidatakse protsessi kulgu eraldi aknas. Aken võimaldab protsessi katkestada.
19. Lisati järgmine funktsionaalsus: märgendatud videosegmente alati sorteeritakse algupositsiooni järgi enne väljalõikamist ja ühendamist.
20. Teostati järgnev muudatus: videosegmentide väljalõikamise ja ühendamise tagajärjel tekitatud uus video omab sama loomise kuupäeva ja kellaaega nagu esialgne video (andmebaasis).

3 Töö kirjeldus

3.1 Nõuded

3.1.1 Funktsionaalsed nõuded

1. Uue videofailide taasesitamise ja töötlemise eest vastutava komponendi kasutajaliides peab omama nuppu, millele vajutamisel kõiki märgendatud videosegmente lõigatakse välja ning ühendatakse üheks uueks videofailiks.
2. Märgendatud videosegmendid peavad olema ühendatud ajaliselt õiges järjekorras (algusaja järgi kasvavas järjekorras).
3. Pärast ülevalpool mainitud nuppu vajutamist peab uus tekitatud videofail olema avatud uue komponendi kasutajaliideses.
4. Uues tekitatud videofailis videosegmentide märgendid peavad olema säilitatud juhul, kui märgendeid on rohkem kui üks.
5. Videosegmentide väljalõikamise ja ühendamise ajal peab kasutajaliides teavitama kasutajat, et antud protsess on käimas ning näitama protsessi kulgu protsentides.
6. Videosegmentide väljalõikamise ja ühendamise protsessi peab olema võimalik ka katkestada.
7. Pärast ülevalpool mainitud nuppu vajutamist peab uus tekitatud videofail lisanduma käesoleva uuringu meediafailide hulka. Uue videofaili pisipilt peab olema nähtav PathologyMediaDialog aknas (selles aknas on loetletud kõik valitud patsiendile kuuluvad uuringute kaupa grupeeritud meediafailid: pildid, audio- ja videofailid).
8. Uus tekitatud videofail peab omama sama loomise kuupäeva ja kellaaega nagu esialgne videofail.
9. Videosegmentide väljalõikamine ja ühendamine peab olema võimalik nii H.264 standardi järgi kui ka H.265 standardi järgi kodeeritud videofailide puhul.
10. Kui töödeldav videofail omab seotud audiofaili, siis videosegmentide väljalõikamisega paralleelselt peab toimuma segmentide väljalõikamine audiofailist vastavates kohtades. Protsessi lõpus kõik väljalõigatud segmendid peavad olema ühendatud üheks videofailiks, millele on lisatud audiovoog.

3.1.2 Mittefuntsionaalsed nõuded

1. Video- ja audiosegmentide väljalõikamine ja ühendamine peab toimuma võimalikult kiiresti. Tuleb võimaluse korral vältida ümberkodeerimist (video- ja audiofailide formaadi ja/või kodeerimisviisi muutmist).
2. Video ja audio kvaliteet segmentide väljalõikamisel ja ühendamisel ei tohi muutuda halvemaks. Tuleb võimaluse korral vältida ümberkodeerimist.
3. Väljalõigatud segmendi tegelik pikkus võib olla märgendi pikkusest suurem või ebaoluliselt (sekundi murdosad) väiksem.

3.2 Arhitektuur

ORNet tarkvara koosneb 4 kihist. Iga kiht on realiseeritud ühe või mitme Visual Studio projektina. Kihid on järgmised:

1. Esitluskiht. See kiht koosneb kolmest osast. Üks nendest on peamine lõpp-kasutaja rakendus, mis sisaldab kasutajaliidest. Kaks ülejäänud on Windows teenused ilma kasutajaliideseta: DICOM teenus ja HL7 teenus.
2. Loogikakiht. Selles kihis asub rakenduse äriloogika.
3. Andmetele juurdepääsu kiht. Pakub lihtsustatud juurdepääsu andmetele, mida hoitakse andmebaasis.
4. Andmebaasikiht.

ORNet tarkvara arhitektuur on skemaatiliselt näidatud joonisel 2.

Käesoleva töö käigus kirjutatud kood asub suuremal määral loogikakihis ja väiksemal määral esitluskihis. Andmetele juurdepääsu kihti ja andmebaasikihti antud töö käigus ei käsitletud.

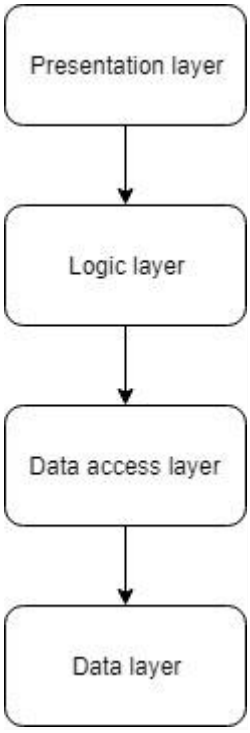
Kasutajaliidese koodi organiseerimiseks kasutatakse ORNet-i tarkvaras MVVM arhitektuurilist disainimustrit.

3.3 Disain

Töös on kasutatud dependency injection muster [44]. See muster seisneb selles, et sõltuvuse korral teatud klass ei loo iseseisvalt teist klassi, millest ta sõltub, vaid saab

liidese realisatsiooni konstruktori kaudu või mingil muul viisil kolmandalt klassilt. See annab järgnevaid eeliseid:

1. Liidestest on alati lihtne luua libaobjekte.
2. Dependency injection kasutamise korral on arusaadav, mis sõltuvusi klass vajab selleks, et toimida korrektselt.
3. Dependency injection aitab vältida mittevajalikke sõltuvusi.
4. Kasutades liideseid on võimalik anda klassile alternatiivseid implementatsioone või täiustama olemasolevaid.



Joonis 2. ORNet tarkvara arhitektuur.

ORNet tarkvaras on osaliseslt rakendatud Stairway mustrer [44]. Lühidalt see seisneb selles, et liideseid ja nende realisatsioone tuleb hoida erinevates pakettides (assemblies). Sellise koodi organisatsiooni puhul on välditud klasside mittevajalikud peidetud sõltuvused. Klassid viitavad ainult nendele pakettidele, mis sisaldavad liideseid, millest nad sõltuvad. See aitab saavutada madalat klasside omavahelist sõltuvust (loose coupling).

Lisaks sellele, antud töö käigus on kasutatud static factory meetod [45, 46]. See on staatiline meetod, mis tagastab klassi eksemplari. Edasi on loetletud selle mustri mõned eelised:

1. Static factory meetoditel on nimed. Nimi seletab, kuidas objekti luuakse ja mis on argumendid.
2. Static factory meetod ei pea alati looma uut objekti.
3. Static factory meetod võib tagastada selle tagastatava klassi igat alamklassi.
4. Static factory meetod võib sisaldada kogu loogikat, mis on vajalik objekti loomiseks.

3.4 Kood

Selle töö käigus autori peamine ülesanne oli kavandada ja kirjutada koodi, mis lisab tarkvarale mainitud funktsionaalsust (vt Eesmärk, lk 13). Töö fookus oli arendusel.

Püstitatid eesmärgi saavutamiseks tarkvarale olid lisatud 18 uut klassi, 10 uut liidest ja kirjutatud rohkem kui 1550 rida koodi (see arv hõlmab ainult tootmiskoodi).

Kood üldjoontes töötab järgmisel viisil:

PathologyMediaDialog aknas videofailile kahekordsel vajutamisel täidetakse meetodit Execute(...) klassis OpenMediaItemCommand. Selle tagajärjel:

1. Luuakse uut MediaEditorView tüüpi objekti.
2. Luuakse uut MediaEditorViewModel tüüpi objekti.
3. MediaEditorView vaate mudeliks (DataContext) määratakse loodud MediaEditorViewModel-it.
4. Kutsutakse MediaEditorViewModel-i meetodit LoadMedia(...).
5. Kutsustakse MediaEditorView-i meetodit ShowDialog().

Ülevalpool kirjeldatud protsessi koostööskeem on esitatud joonisel 3.

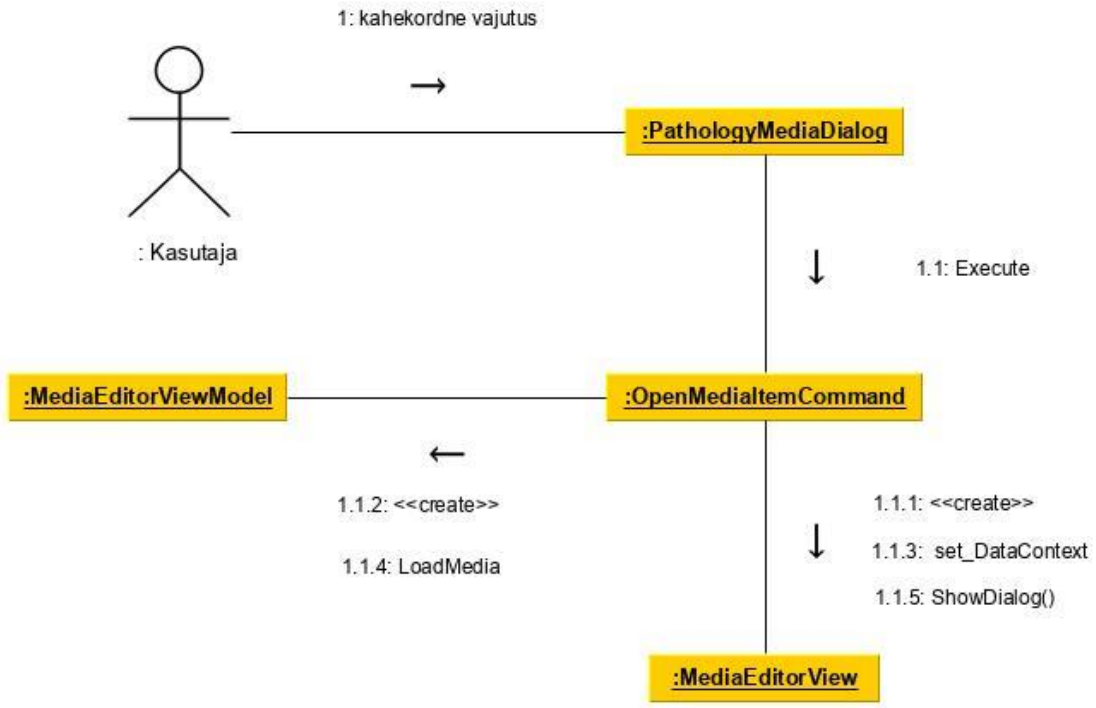
MediaEditorView-s on nupp Merge. Selle nupu vajutamisel täidetakse käsku MergeVideoSegmentsCommand MediaEditorViewModel-is. Selle tagajärjel:

1. CueTrackViewModel-ite kollektsiooni muudetakse IVideoSegments liidese realisatsiooniks.
2. Kutsutakse klassi VideoMergingService meetodit MergeVideoSegmentsAsync(...). Selle tagajärjel:
    1. FfMpegFilesCreationProgressCalculator-ile lisatakse faile, mida hakatakse looma.
    2. Käivitatakse klassi VideoCuttingService meetodit CutVideoSegmentsAsync(...).

3. Väljalõigatud videosegmente ühendatakse üheks uueks videofailiks.
4. Käivitatakse sündmust VideoSegmentsMerged.
5. Luuakse uut Media tüüpi objekti ja salvestatakse seda andmebaasi.
6. Käivitatakse sündmust MediaAdded.
7. Täidetakse klassi CutMediaSegmentsParamsRecalculator meetodit Recalculate(...).
8. Käivitatakse sündmust VideoSegmentsParamsRecalculated.
9. Täidetakse klassi ThumbnailExtractingService meetodit ExtractAsync(...).
10. Täidetakse klassi MediaManager meetodit CreateThumbnailAsync(...).

3. Luuakse MediaItem tüüpi objekti.
4. Käivitatakse sündmust MediaAdded.

Ülevalpool kirjeldatud protsessi koostööskeem on esitatud joonistel 4, 5 ja 6.

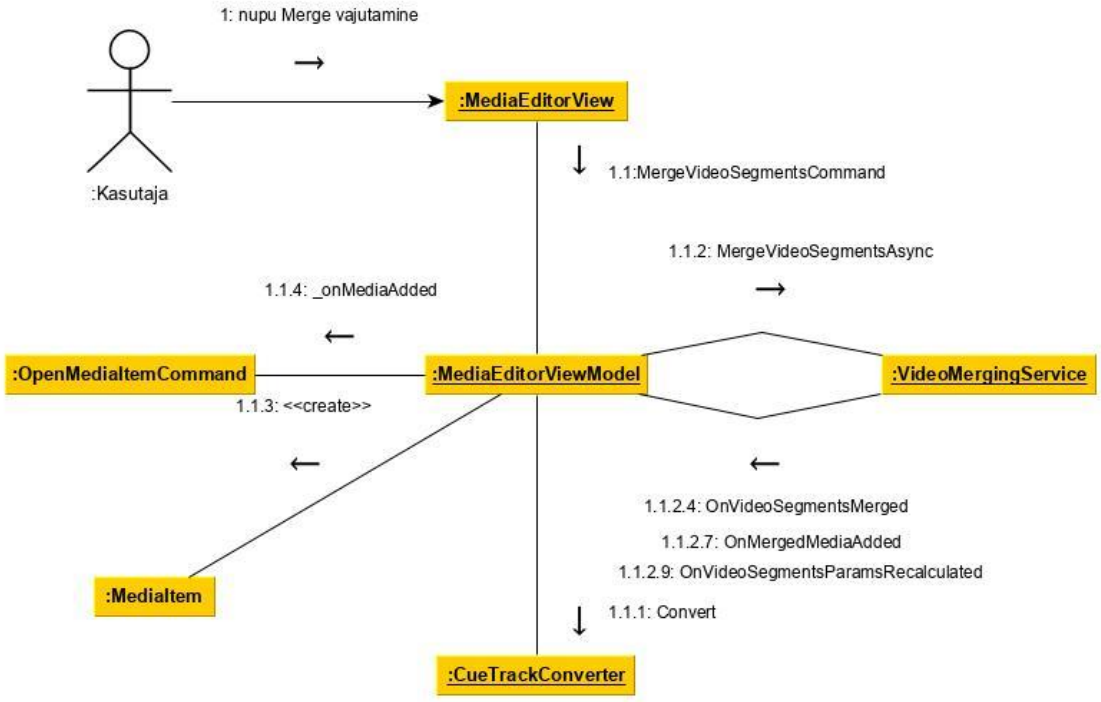


Joonis 3. Akna MediaEditorView avamine.

Selleks, et näidata, kuidas klassid ja liidesed on seotud ja suhtlevad omavahel, olid loodud klassidiagrammid, mis on toodud allpool. Tuleb mainida, et klassidiagramidel on näidatud vaid olulisimad klassid ja nendevahelised interaktsioonid. Kõigi nii uute kui ka juba tarkvaras eksisteerinud klasside kirjeldused on toodud lisas 1 Töö eesmärgi

täitmiseks vajalike klasside kirjeldused. Töö käigus kirjutatud kood on esitud kinnises lisas 4 Programmi kood.

MediaEditorView on uue komponendi kasutajaliides. Tänu sellele kasutaja saab vaadata salvestatud videofaile. MediaEditorView sisaldab ka nuppe, mis on vajalikud videofailiga erinevate toimingute teostamiseks. See klass laiendab klassi Window ning selle visuaalsed elemendid on defineeritud kasutades XAML märgistuskeelt.

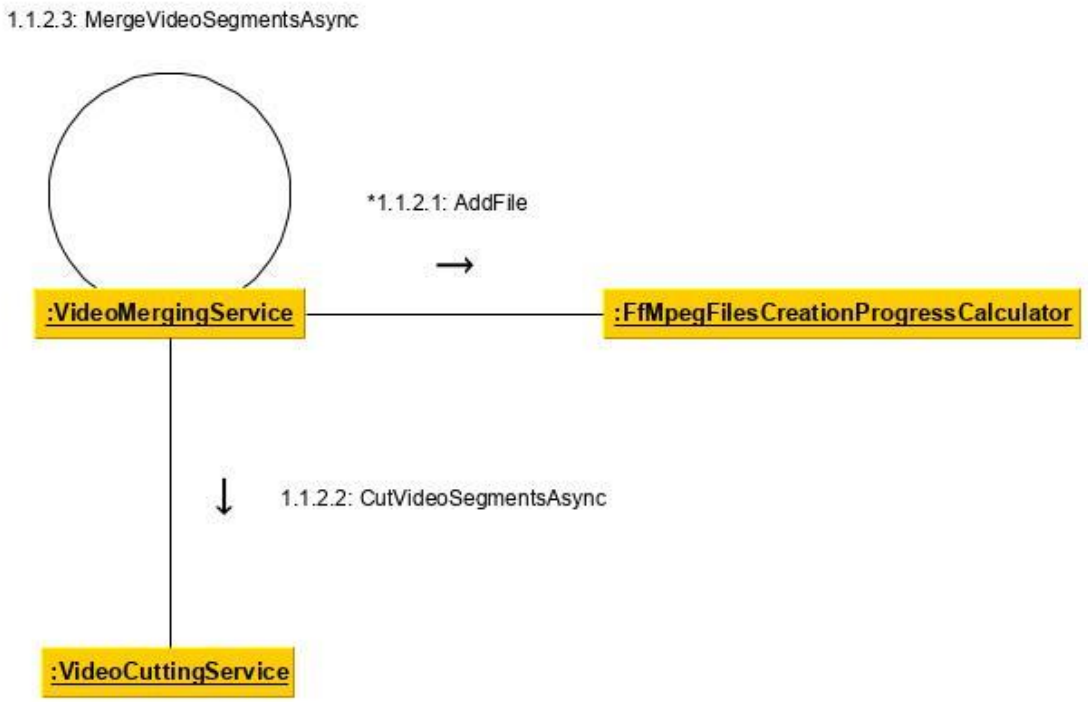


Joonis 4. Videosegmentide väljalõikamine ja ühendamine. Osa 1.

ProgressReportControl on autori poolt loodud kasutajaliidese element ning on MediaEditorView osaks. See klass laiendab klassi UserControl ning nagu ka MediaEditorView koosneb kahest osalisest klassist. Ühes osalises klassis on defineeritud selle visuaalsed elemendid kasutades XAML keelt. Teine osaline klass sisaldab C# keeles kirjutatud koodi ning seal on defineeritud omadused, mis mõjutavad visuaalsete elementide käitumist. Kui toimub videosegmentide väljalõikamine ja ühendamine, ProgressReportControl muutub nähtavaks ning selle edenemisribal kajastatakse protsessi kulgu.

MediaEditorViewModel on MediaEditorView vaate mudeliks. See sisaldab esitlusloogikat. Nupp Merge MediaEditorView-s on seotud käsuga MergeVideoSegmentsCommand. Selle käsu täitmisel kutsutakse välja klassi

VideoMergingService meetodit MergeVideoSegmentsAsync(...). VideoMergingService saadab MediaEditorViewModel-ile sõnumeid (sündmuste kujul) protsessi kulust, MediaEditorViewModel muudab oma omadusi ning selle tulemusel muutub ka kasutajaliides. Käsk CancelMergingCommand on seotud nupuga Cancel ProgressReportControl-is. Selle nupu vajutamisel videosegmentide väljalõikamise ja ühendamise protsessi katkestatakse kasutades CancellationTokenSource mehhanismi.



Joonis 5. Videosegmentide väljalõikamine ja ühendamine. Osa 2.

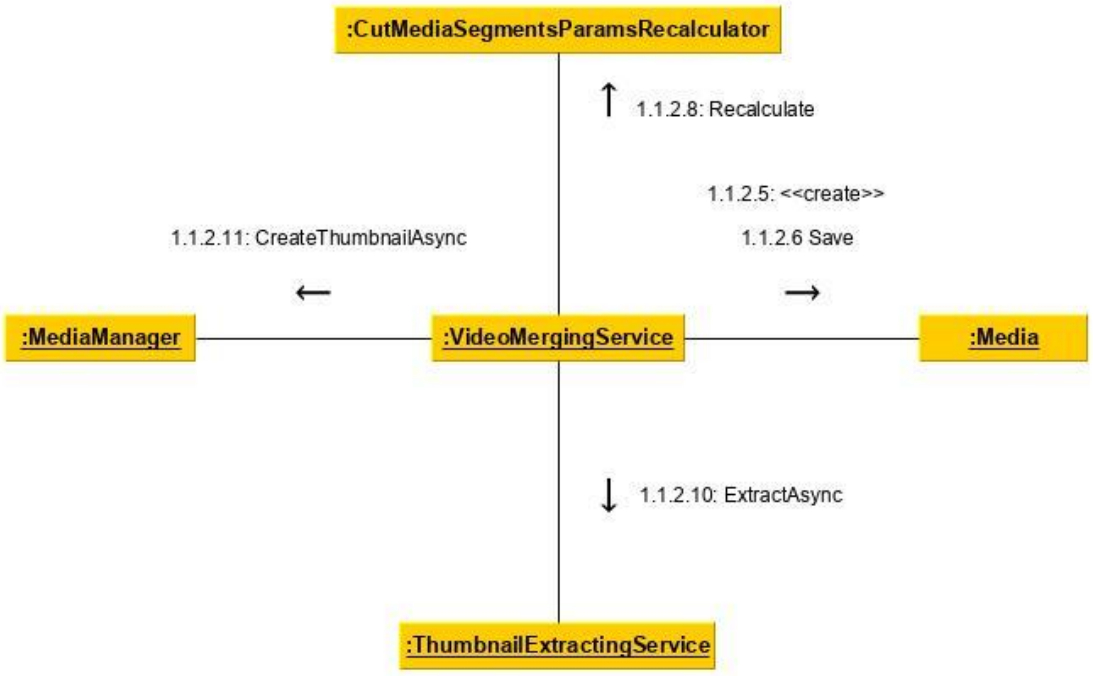
Klass OpenMediaItemCommand kujutab endast käsku, mida täidetakse valitud meediafaili pisipildile (videofailist kindlas positsioonis võetud pildi madalama kvaliteediga vähendatud koopia) hiire kahekordsel vajutamisel. Klassi ülesandeks on avada meediafaili sobivat tüüpi aknas. Klass realiseerib liidest ICommand nimeruumist System.Windows.Input ja seega omab kahte avalikku meetidit: Execute(...) ja CanExecute(...). Kui valitud meediafail on video tüüpi, siis OpenMediaItemCommand loob MediaEditorView-t ja MediaEditorViewModel-t, määrab MediaEditorView vaate mudeliks loodud MediaEditorViewModel-it ning lõpuks avab MediaEditorView-t. See on näidatud klassidiagrammil joonisel 7.

OpenMediaItemCommand loob MediaEditorViewModel tüüpi objekti. See toimub privaatses meetodis GetMediaEditorViewModel(...). MediaEditorViewModel

konstruktor võtab sealhulgas järgmiseid argumente: liideste IMediaFileMetainfoService, IVideoMergingService ja ICueTrackConverter realisatsioone.

Liidese IMediaFileMetainfoService realisatsiooni saab OpenMediaItemCommand konstruktori kaudu.

VideoMergingService tüüpi objekti loob OpenMediaItemCommand privaatses meetodis GetMediaEditorViewModel(...). VideoMergingService konstruktor võtab sealhulgas järgmiseid argumente: liideste IProgramRunner, IFfMpegArgumentsConstructor, IVideoCuttingService, IMediaFileMetainfoService, IThumbnailExtractingService, ICutMediaSegmentsParamsRecalculator ja IFfMpegFilesCreationProgressCalculator realisatsioone.



Joonis 6. Videosegmentide väljalõikamine ja ühendamine. Osa 3.

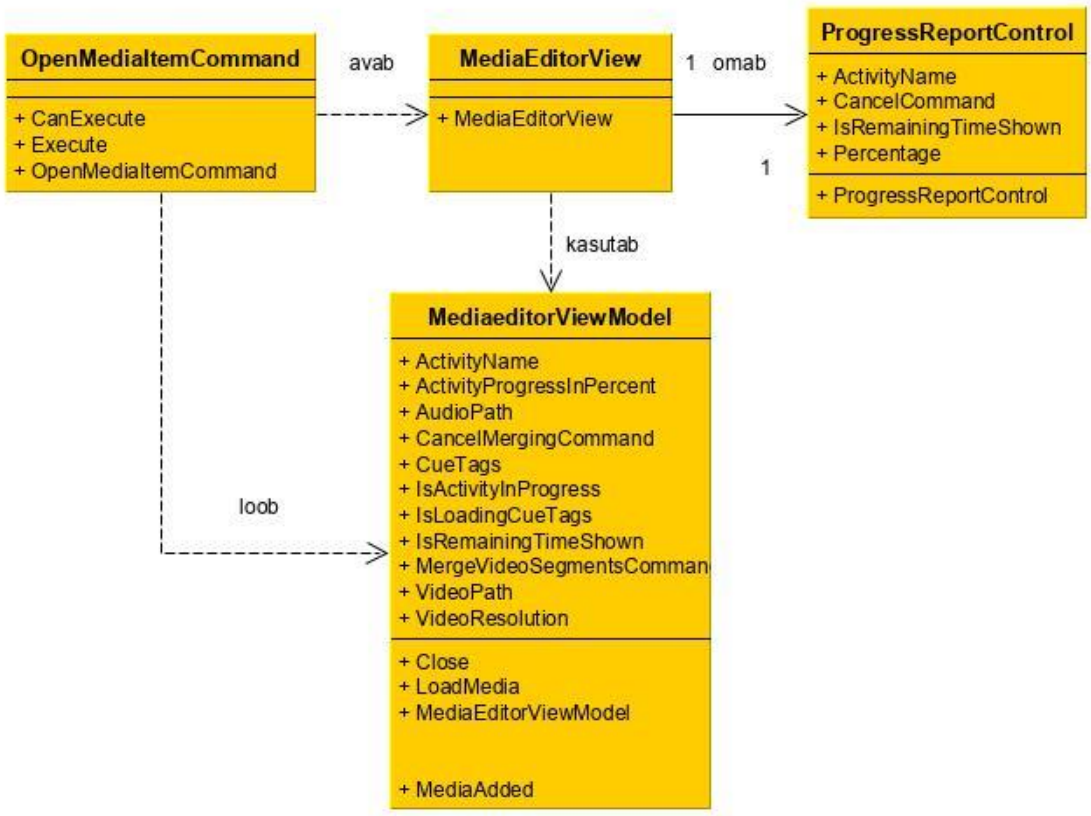
CueTrackConverter tüüpi objekti loob OpenMediaItemCommand privaatses meetodis GetMediaEditorViewModel(...). Liidese IProgramRunner realisatsiooni saab OpenMediaItemCommand konstruktori kaudu. Liidese IFfMpegArgumentsConstructor realisatsiooni saab OpenMediaItemCommand konstruktori kaudu.

VideoCuttingService tüüpi objekti loob OpenMediaItemCommand privaatses meetodis GetMediaEditorViewModel(...). VideoCuttingService konstruktor võtab sealhulgas

järgmiseid argumente: liideste IProgramRunner, IFfMpegArgumentsConstructor ja IFfMpegFilesCreationProgressCalculator realisatsioone.

Liidese IMediaFileMetainfoService realisatsiooni saab OpenMediaItemCommand konstruktori kaudu.

ThumbnailExtractingService tüüpi objekti loob OpenMediaItemCommand privaatses meetodis GetMediaEditorViewModel(...). ThumbnailExtractingService konstruktor võtab sealhulgas järgmiseid argumente: liideste IProgramRunner ja IFfMpegArgumentsConstructor realisatsioone.

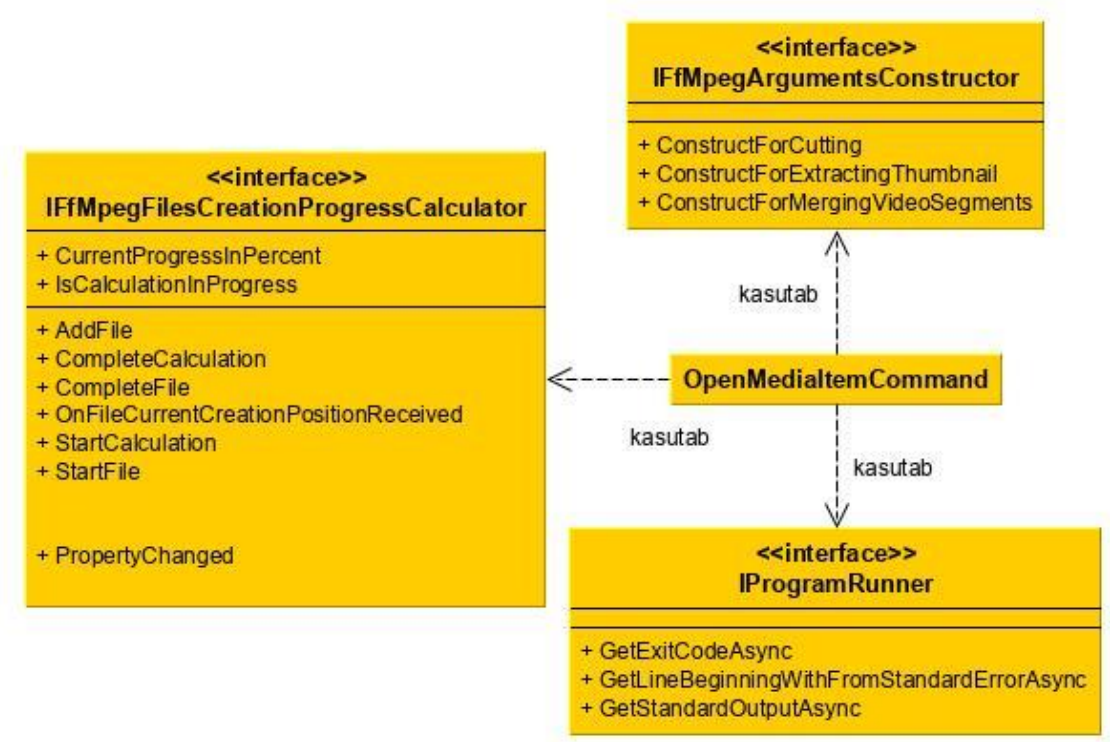


Joonis 7. Klass OpenMediaItemCommand avab MediaEditorView tüüpi akent.

OpenMediaItemCommand saab ICutMediaSegmentsParamsRecalculator liidese realisatsiooni kutsudes klassi CutMediaSegmentsParamsRecalculator static factory meetodit Create(...) välja. Meetodi argumendiks on liidese IMediaFileMetainfoService realisatsioon. See toimub privaatses meetodis GetMediaEditorViewModel(...).

OpenMediaItemCommand saab IFfMpegFilesCreationProgressCalculator liidese realisatsiooni kutsudes klassi FfMpegFilesCreationProgressCalculator static factory meetodit Create() välja. See toimub privaatses meetodis GetMediaEditorViewModel(...).

Klassi OpenMediaItemCommand sõltuvused on toodud klassidiagrammidel joonistel 8, 9, 10 ja 11.

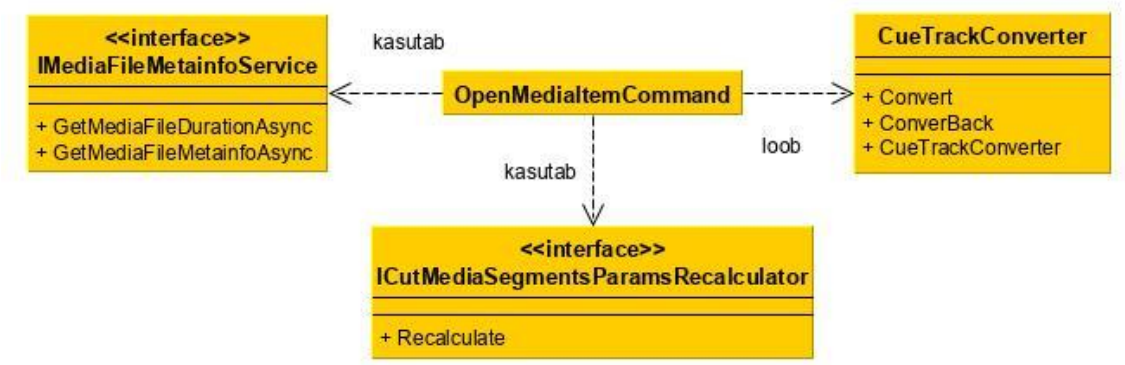


Joonis 8. Klassi OpenMediaItemCommand sõltuvused.

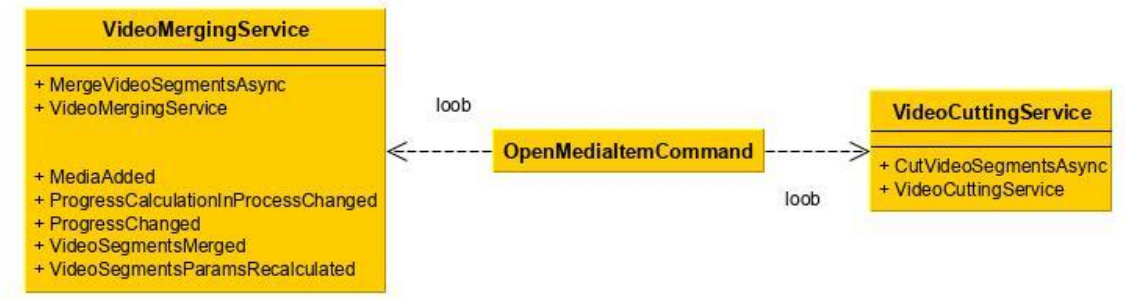
Klass MediaEditorViewModel kasutab märgendatud videosegmentide väljalõikamiseks ja ühendamiseks järgmiste liideste realisatsioonide teenuseid: IMediaFileMetainfoService, IVideoMergingService ja ICueTrackConverter. Klassi MediaEditorViewModel sõltuvused on toodud klassidiagrammil joonisel 12.

Klassi MediaFileMetainfoService ülesandeks on meediafaili metaandmete saamine. Selleks on selles klassis kaks meetodid: GetMediaFileMetainfoAsync(...) ja GetMediaFileDurationAsync(...). Oma eesmärkide täitmiseks kasutab MediaFileMetainfoService programmi ffprobe ning liidese IProgramRunner realisatsiooni ja klassi FFProbeXmlParser teenuseid. See on näidatud klassidiagrammil joonisel 13.

Liidese IMediaFileMetainfoService meetodi GetMediaFileMetainfoAsync(...) väljakutsumine toimub klassis MediaEditorViewModel privaatse meetodi LoadMedia(...) sees. Hiljem informatsiooni videofaili konteineri tüübist kasutatakse uue videosegmentide ühendamise tagajärjel tekitatud videofaili audiovoo kooderimisviisi valimiseks (klass MediaEditorViewModel, meetodid MergeVideoSegments() ja SelectSegmentsAudioEncoding(...)).



Joonis 9. Klassi OpenMediaItemCommand sõltuvused.

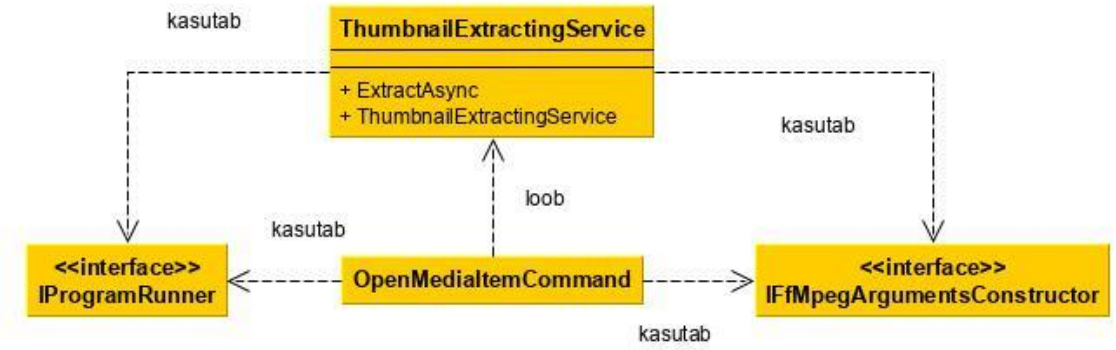


Joonis 10. Klassi OpenMediaItemCommand sõltuvused.

Klassi VideoMergingService ülesandeks on märgendatud videosegmentide ühendamine üheks videofailiks. Oma eesmärkide täitmiseks ta kasutab sealhulgas programmi ffmpeg ning järgmiste liideste realisatsioone: IProgramRunner, IFfMpegArgumentsConstructor, IVideoCuttingService, IMediaFileMetainfoService, IThumbnailExtractingService, ICutMediaSegmentsParamsRecalculator ja IFfMpegFilesCreationProgressCalculator. Klassi VideoMergingService sõltuvusel on toodud klassidiagrammidel joonistel 14 ja 15.

Klassi VideoMergingService meetodi MergeVideoSegmentsAsync väljakutsumisel toimuvad järgmised sündmused:

1. Liidese IFfMpegFilesCreationProgressCalculator realisatsioonile lisatakse faile, mida hakatakse looma (alguses väljalõigatavaid segmente ja seejärel ühendamise käigus tekitatavat videofaili);
2. Algab videosegmentide väljalõikamise ja ühendamise protsessi kulu arvutamine.
3. Kutsutakse välja liidese IVideoCuttingService realisatsiooni meetodit CutVideoSegmentsAsync(...).
4. Väljalõigatud videosegmente ühendatakse uueks videofailiks. Selleks kasutatakse programmi ffmpeg ja IProgramRunner liidese realisatsiooni.
5. Lõpetatakse videosegmentide väljalõikamise ja ühendamise protsessi kulu arvutamist.
6. Uut videofaili salvestatakse andmebaasi.
7. Arvutatakse ümber videosegmentide parameetreid (alguspositsiooni ja kestvust).
8. Kutsutakse välja liidese IThumbnailExtractingService realisatsiooni meetodit ExtractAsync, mille tulemusel uuest videofailist lõigatakse välja pilti.
9. Väljalõigatud pildist luuakse pisipilti. Sellega tegeleb liidese IMediaManager implementatsioon.

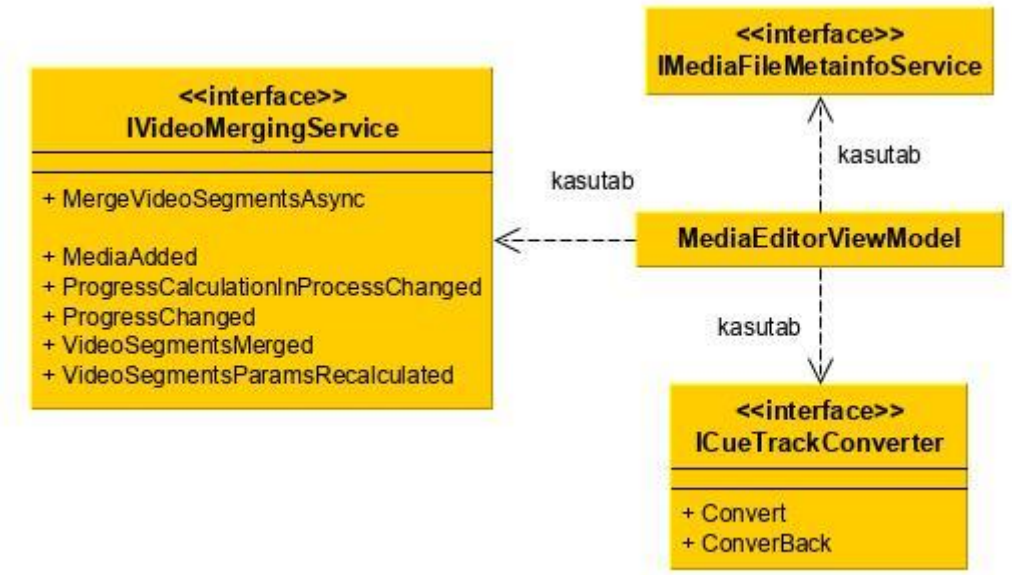


Joonis 11. Klasside OpenMediaItemCommand ja ThumbnailExtractingService sõltuvused.

Klassi CueTrackConverter ülesandeks on liideste IList<CueTrackViewModel> ja IVideoSegments realisatsioonide omavaheline teisendamine mõlemas suunas. Klass CueTrackViewModel on põhimõtteliselt märgendatud videosegmendi vaate mudel. See klass sisaldab esitlusloogikat ja asub estluskihis. Videosegmente töötlevad klassid asuvad aga loogikakihis. Loogikakiht ei tohi esitluskihist sõltuda. Selleks, et anda edasi videosegmentide parameetreid loogikakihti, olid loodud klassid VideoSegment ja VideoSegments. Viimane on põhimõtteliselt liidese IVideoSegment realisatsioonide loend. MediaEditorViewModel-i meetodis MergeVideoSegments() ICueTrackConverter liidese realisatsioon teisendab liidese IList<CueTrackViewModel> realisatsiooni liidese

IVideoSegments realisatsiooniks. Meetodis SaveVideoSegmentsToCueSheet(...) teisendamist teostatakse vastupidises suunas.

Klassi ProgramRunner ülesandeks on käivitada etteantud programmi (antud töö kontekstis ffmpeg või ffprobe) etteantud argumentidega. Oma eesmärkide täitmiseks kasutab ProgramRunner klassi Process nimeruumist System.Diagnostics.



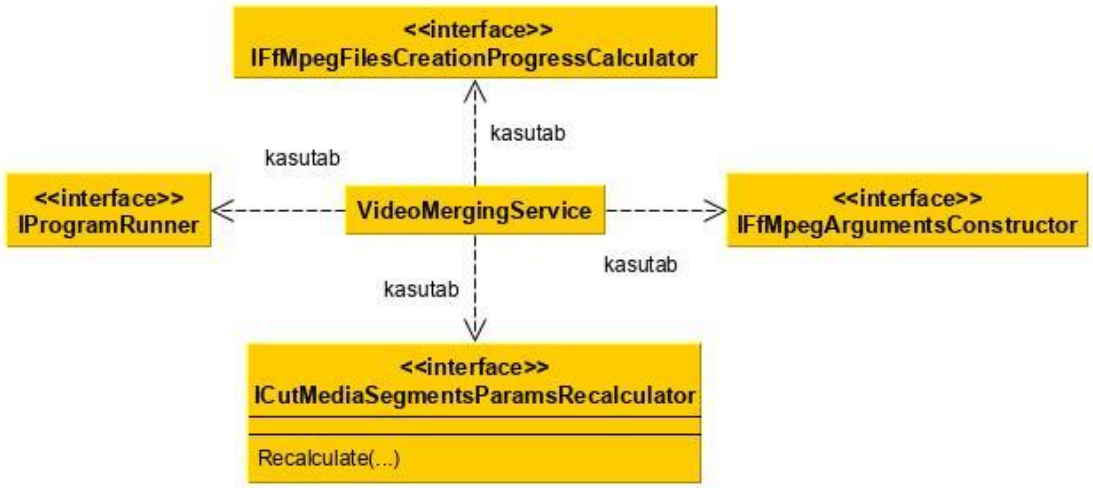
Joonis 12. Klassi MediaEditorViewModel sõltuvused.

Joonis 13. Klassi MediaFileMetainfoService sõltuvused.

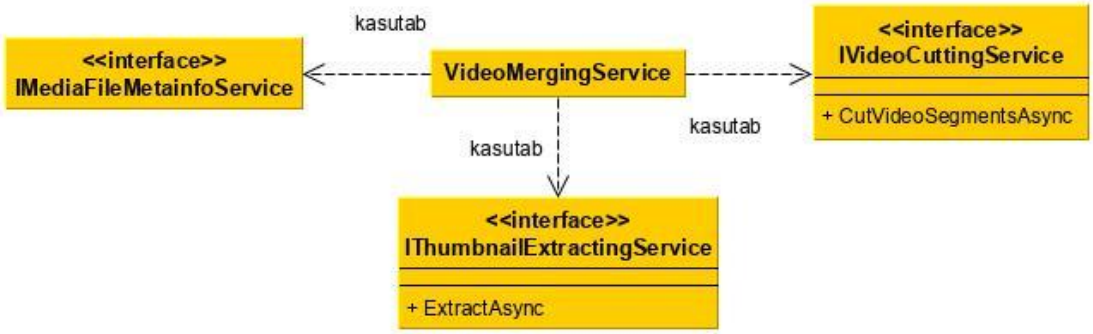
Klassi FfMpegArgumentsConstructor ülesandeks on argumentide konstrueerimine programmiga ffmpeg erinevate operatsioonide teostamiseks.

Klassi VideoCuttingService ülesandeks on videofailist märgendatud videosegmentide väljalõikamine. Oma eesmärkide täitmiseks ta kasutab sealhulgas programmi ffmpeg ning järgmiste liideste realisatsioone: IProgramRunner, IFfMpegArgumentsConstructor ja IFfMpegFilesCreationProgressCalculator. Klassi VideoCuttingService sõltuvused on toodud klassidiagrammil joonisel 16.

Klassi ThumbnailExtractingService ülesandeks on videofailist pildi võtmine. Oma eesmärkide täitmiseks ta kasutab sealhulgas programmi ffmpeg ning järgmiste liideste realisatsioone: IProgramRunner ja IFfMpegArgumentsConstructor. Klassi ThumbnailExtractingService sõltuvused on esitatud klassidiagrammil joonisel 11.



Joonis 14. Klassi VideoMergingService sõltuvused.

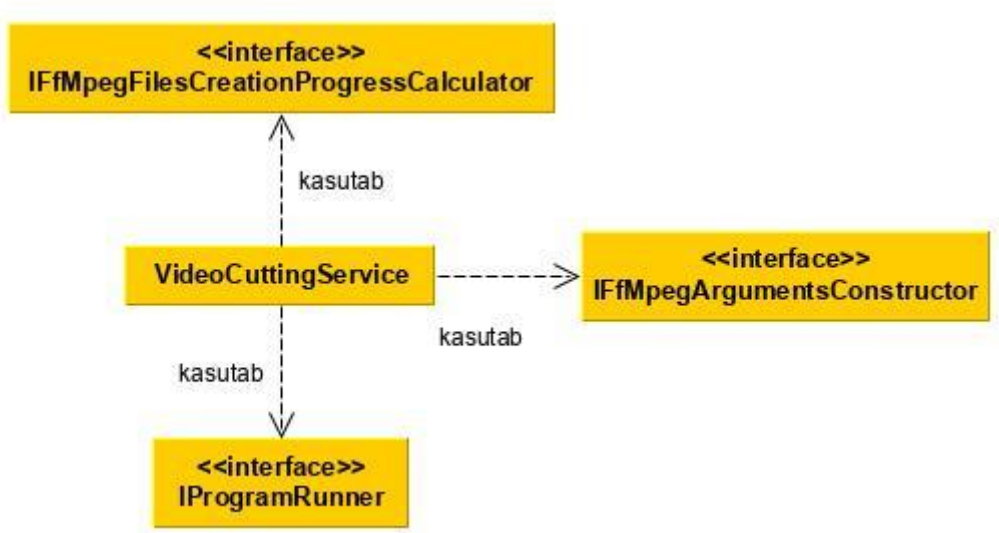


Joonis 15. Klassi VideoMergingService sõltuvused.

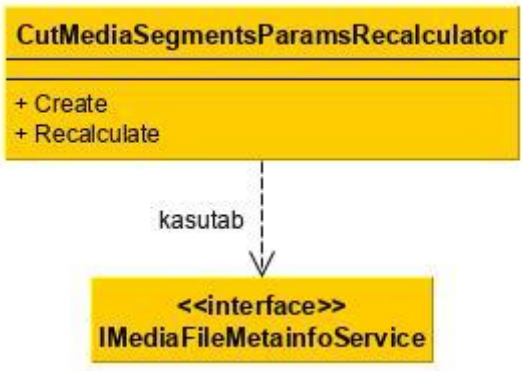
Klassi CutMediaSegmentsParamsRecalculator ülesandeks on väljalõigatud ja ühendatud videosegmentide parameetrite (alguspositsiooni ja kestvuse) ümberarvutamine. Oma eesmärkide täitmiseks ta kasutab liidese IMediaFileMetainfoService realisatsiooni (meetod GetMediaFileDurationAsync(...)). See on näidatud klassidiagrammil joonisel 17.

Selleks, et videofailist segmentide väljalõikamise protsess oleks võimalikult kiire ega mõjutaks video kvaliteeti, selles töös püütakse vältida ümberkodeerimist. Videosegmendi väljalõikamist alustatakse mitte täpsest kasutaja poolt määratud positsioonist, vaid

lähimast eelmisest võtmekaadrist. See tähendab, et väljalõigatud videosegmendi lõplik pikkus on soovitid pikkusest suurem. Teisest küljest, kui videofailis on märgendatud videosegmente rohkem kui üks, märgendid peavad säilima ka uues videosegmentide ühendamise tagajärjel tekitatud videofailis. Selleks, et õigesti asetada märgendeid uude videofaili, nende parameetreid on vaja ümber arvutada. Sellega tegeleb klass CutMediaSegmentsParamsRecalculator.



Joonis 16. Klassi VideoCuttingService sõltuvused.



Joonis 17. Klass CutMediaSegmentsParamsRecalculator kasutab liidese IMediaFileMetainfoService realisatsiooni.

Programm ffmpeg kirjutab videofailide loomisel (antud töö kontekstis videosegmentide väljalõikamisel ja nende ühendamisel) standardvea väljundisse, mitu sekundit videofailist on antud hetkel loodud. See informatsioon on standardvea väljundis regulaaravaldisena järgmisel kujul: time=[0-9]{2}:[0-9]{2}:[0-9]{2}.[0-9]{2}. Klassis ProgramRunner on meetod GetExitCodeAsync(...). See meetod võtab ühe parameetrina ülevalpool mainitud

mustrit. Programmi ffmpeg käivitamisel standardvea väljundist otsistakse vastavusi sellele mustrile ning leidmise korral antakse seda informatsiooni edasi (sündmuse kujul) klassi OpenMediaItemCommand. Seal kutsutakse välja klassi FfMpegFilesCreationProgressCalculator meetodit OnFileCurrentCreationPositionReceived(...). Saadud informatsiooni alusel FfMpegFilesCreationProgressCalculator arvutab videosegmentide väljalõikamise ja ühendamise protsessi kulgu protsentides.

4 Analüüs ja arutelu

4.1 Üldine

Antud töö osas analüüsitakse ainult töö eesmärkide realiseerimiseks vajalikku ja autori poolt kirjutatud koodi. Enumeraatorid, sündmuste argumendid ja klass Constants ei ole töö käigus analüüsitud.

Kvaliteetne kood peab olema kergesti hooldatav, loetav, testitud ja muudatustele kohandatav [44].

Autor otsustas iseseisvalt, mis meetrikaid ta kogus koodi analüüsimiseks.

Kõigepealt, antud töö osas anlüüsiti programmi koodikatet. Mida suurem on koodikate, seda suuremat osa programmi lähtekoodist käivitatakse testimise käigus, see tähendab, et programm sisaldab vähem ennustamatuid defekte. Koodikatte analüüsimise eesmärgil koguti ja analüüsiti järgnevaid meetrikaid:

1. klassi tasemel:
    - reakate;
    - harukate (HK).
2. meetodi tasemel:
    - jadakate (JK);
    - harukate.

Koodi meetrikaid sisaldavates tabelites on samuti toodud konkreetse klassi jaoks kirjutatud testide arv. Testide arvu lugemisel võeti arvesse ainult neid teste, mis testivad meetodeid, mis on vajalikud töö eesmärkide realiseerimiseks ja on autori poolt kirjutatud.

Samuti koguti ja analüüsiti koodi keerukust kajastavaid meetrikaid:

1. klassi tasemel:
    - koodi ridade arv (SLOC);
    - hooldatavuse indeks (HI);

- klasside omavaheline sõltuvus (KOS);
- pärimise sügavus (PS).

2. meetodi tasemel:

   - koodi ridade arv (SLOC);
   - tsüklomaatiline keerukus (TK);
   - N-Path keerukus (NPK);
   - hooldatavuse indeks (HI);
   - klasside omavaheline sõltuvus (KOS).

On oluline, et koodi kerukus oleks võimalikult väiksem. Suur keerukus tähendab, et:

- Koodi on raskem lugeda, sellest aru saada ja hinnata;
- Eksisteerib rohkem potentsiaalseid vigade tekkimise kohti;
- Koodi on raskem muuta, pidada üleval ja taaskasutada;
- Koodi on raskem testida.

Allpool on selgitatud, mida tähendab iga konkreetne meetrika.

Klassi tasemel meetrikad on esitatud ainult siis, kui klass on kirjutatud tervikult autori poolt ning kõik selle meetodid on seotud töö eesmärkide realiseerimisega, vastasel juhul on toodud ainult testide arv ja meetodi tasemel meetrikad.

Töö käigus samuti analüüsiti kirjutatud koodi vastavust puhta koodi põhimõtetele.

4.2 Koodikate

Koodikate kirjeldab, kui suur osa programmi lähtekoodist täidetakse konkreetse testide komplekti käivitamisel [47]. Koodikate näitab programmi piirkondi, mis ei ole kaetud testidega [48, 49, 50]. Koodikate suurendamise eesmärgiga luuakse uusi testjuhtumeid. Koodikatte väärtused on puhtalt kvantitatiivsed ega näita programmikoodi ega testimise kvaliteeti. Mida suurem on koodikate, seda suuremat osa programmi lähtekoodist käivitatakse testimise käigus, see tähendab, et programm sisaldab vähem ennustamatuid defekte.

Eksisteerib mitu erinevat koodikatte kriteeriumit. Antud töös kasutatakse nendest järgmiseid:

1. Reakate. Näitab, kui suur osa lähtekoodi ridadest täidetakse testimisel [47, 51]. OpenCover tööriista puhul kasutatakse reakatte arvutamiseks valemit (1).

$$a = \frac{b}{c} \times 100\%, \text{ kus} \quad (1)$$

   a – reakate protsentides;
   b – täidetud programmi read;
   c – testitavad read (kõik programmi read, mida on võimalik täita testidega).

2. Jadakate (JK). Näitab, kui suur osa programmi järjepunktidest täidetakse testimisel. Järjepunkt programmeerimises on programmi iga punkt, kus tagatakse, et kõik eelmiste tegevuste kõrvalefektid juba ilmnesid, aga järgnevate tegevuste kõrvalefektid veel puuduvad [52]. Arvutamisel kasutatakse valemit (2).

$$a = \frac{b}{c} \times 100\%, \text{kus} \quad (2)$$

   a – jadakate protsentides;
   b – täidetud programmi järjepunktid;
   c – kõik programmi järjepunktid.

3. Harukate (HK). Näitab, kui suur osa juhtimisstruktuuride (näiteks, if- ja case-avaldused) harudest täidetakse testimisel [47, 51]. Arvutamisel kasutatakse valemit (3).

$$a = \frac{b}{c} \times 100\%, \text{kus} \quad (3)$$

   a – harukate protsentides;
   b – täidetud programmi harud;
   c – kõik programmi harud.

Koodikatet mõõdetakse protsentides.

Koodikatte mõõtmise eesmärk on töötada välja range, aga juhitav regressioonitestide komplekt.

Usaldusväärne tarkvara omab koodikatet väärtust 70% ja 90% vahel [49, 43]. 100% koodikate näeb välja kahtlane (sellisel juhul testija eesmärgiks on saavutada kõrgeid

koodikatte väärtusi, aga ta ei mõtle sellest, mida ta teeb). Koodikate alla 50% on kindlasti muret tekitav.

4.3 Koodi ridade arv

SLOC (koodi ridade arvu) kasutatakse programmi suuruse määramiseks lähtekoodi tekstis sisalduvate ridade lugemise teel [53]. SLOC abil ennustatakse aega ja jõudu, mida kulutatakse programmi arendamiseks. Seda samuti kasutatakse programmeerimise produktiivsuse ja tarkvara hooldatavuse hindamiseks.

SLOC jaguneb kaheks tüübiks:

- Füüsiline;
- Loogiline.

Füüsilise SLOC levinuim definitsioon on ridade arv programmi lähtekoodi tekstis väljaarvatud kommentaarid. Füüsiline SLOC on tundlik vormindamise erinavate stiilide suhtes.

Loogiline SLOC mõõdab käivitatavate avalduste arvu programmis. Üks loogilise SLOC-i definitsioonidest C-laadsete keelte jaoks on avaldusi lõpetavate semikoolonite arv [53]. Loogiline SLOC võib märgatavalt erineda füüsilisest SLOC-ist.

Käesoleval ajal puudub standard, mis täpselt defineeriks, mida tähendab koodi rida.

Selles töös klasside ja funktsioonide koodi ridade lugemisel võetakse arvesse kõiki ridu, sealhulgas kommentaare ja tühju ridu. Seega mõõdetakse füüsilist SLOC-i. Samuti peetakse, et funktsioon, mis on rohkem kui 30 rida pikk, on liiga suur ja vajab refaktoreerimist.

4.4 Koodi keerukus

Koodi keerukuse mõõtmiseks kasutatakse antud töös kahte meetrikat:

- Tsüklomaatiline keerukus (TK);
- N-Path keerukus (NPK).

Üldjoontes TK (tsüklomaatiline keerukus) arvutab otsustamise punkte (decision points) funktsioonis ning NPK (N-Path keerukus) arvutab kõiki võimalikke teid [54].

On oluline, et koodi keerukus oleks võimalikult väiksem. Lihtne kood on alati keerulisest koodist parem [55]. Suur keerukus tähendab, et [54]:

- Koodi on raskem lugeda, sellest aru saada ja hinnata;
- Eksisteerib rohkem potentsiaalseid vigade tekkimise kohti;
- Koodi on raskem muuta, pidada üleval ja taaskasutada;
- Koodi on raskem testida.

Selleks, et vähendada funtsiooni keerukust, seda tuleb jagada mitmeks väiksemaks funktsiooniks.

4.4.1 Tsüklomaatiline keerukus

TK kasutatakse arvutiprogrammi keerukuse määramiseks [56]. TK saab samuti määrata iga funktsiooni või klassi jaoks programmis. Projekti, nimeruumi või klassi keerukus võrdub selle konstruktorite ja meetodite keerukuse summaga [57].

TK võrdub lineaarselt sõltumatute teede arvuga läbi programmi lähtekoodi. Lihtsamate sõnadega, funktsiooni TK võrdub otsustamise punktide (decision points) arvuga funktsioonis plus üks meetodisse sisenemise eest [58].

Näiteks, kui lähtekood ei sisalda kontrollvoo avaldusi (tingimusi või otsustamise punkte), keerukus on 1, kuna sellisel juhul on ainult 1 tee läbi koodi. Kui koodis on üks ühe tingimusega if-avaldus, siis on 2 teed läbi koodi: üks, kui if-avaldus on õige, ja üks, kui if-avaldus on väär, seega TK on 2. Siis, kui üks ühe tingimusega if-avaldus on teise ühe tingumisega if-avalduse sees, või siis, kui if-avaldusel on kaks tingimust, TK on 3.

Peale if-avalduste annavad programmile lisakeerukust:

- Tsüklid (while, for, foreach);
- Switch-blokid (case/default);
- Hüpped (continue, goto)
- Erandid (catch)
- Operaatorid, mis võimaldavad liittingimusi (&&, ||, kolmekomponentne operaator).

Iga nendest konstruktsioonidest lisab uut teed läbi koodi.

TK arvutamise algoritm ei ole standartiseeritud.

Üldiselt funktsiooni jaoks on 1-4 madal, 5-7 – mõõdukas, 8-10 – kõrge ja 11+ – väga kõrge TK [55, 58, 54].

4.4.2 N-Path keerukus

Funktsiooni NPK on atsükliliste täitmisteede (acyclic execution path, teiste sõnadega, unikaalsete teede) arv läbi selle funktsiooni [55, 59].

Selle mõiste selgitamiseks kasutatakse siin näiteid. Näidisfunktsioonid on kirjutatud PHP keeles. Esimene näidisfunktsioon on toodud joonisel 18:

```
function insert_default_value($mixed)
{
    if (empty($mixed)) {
        $mixed = 'value';
    }

    return $mixed;
}
```

Joonis 18. Näidisfunktsioon 1.

Eksisteerib kaks unikaalset teed läbi funktsiooni insert_default_value:

1. Kui muutuja mixed on tühi;
2. Kui muutuja mixed ei ole tühi.

Neid teid on võimalik visualiseerida.

Teine näidis on funktsiooni insert_default_value täiendatud versioon ning on toodud joonisel 19. Täiendatud funtsioon sisaldab ühte if-avaldust, mille sees:

1. Kontrollitakse, kas muutuja mixed on sõne;
4. Kontrollitakse, kas muutuja mixed on tühi.

Nüüd on kolm unikaalset teed läbi funktsiooni insert_default_value. Kõiki unikaalseid teid on endiselt võimalik visualiseerida. Need on:

1. Kui muutuja mixed ei ole sõne;
2. Kui muutuja mixed on sõne, aga on tühi.

3. Kui muutuja mixed on sõne ega ole tühi.

```
function insert_default_value($mixed)
{
    if (!is_string($mixed) || empty($mixed)) {
        $mixed = 'value';
    }

    return $mixed;
}
```

Joonis 19. Näidisfuntsioon 2.

Kolmas täiendatud insert_default_value funktsiooni versioon kasutab liidest ToStringInterface, mille ülesandeks on objeti teisendamine sõneks. Funtsioon on toodud joonisel 20.

```
function insert_default_value($mixed)
{
    if ($mixed instanceof ToStringInterface) {
        $mixed = $mixed->to_string();
    }

    if (!is_string($mixed) || empty($mixed)) {
        $mixed = 'value';
    }

    return $mixed;
}
```

Joonis 20. Näidisfunktsioon 3.

Funktsiooni insert_default_value uues versioonis kõigepealt kontrollitakse, kas muutuja mixed realiseerib liidest ToStringInterface. Juhul, kui realiseerib, kutsutakse välja meetodit to_string(). Tagastatavat väartust määratakse muutujale mixed. Ülejäänud funktsiooni insert_default_value kood jääb samaks.

Läbi esimese if-avalduse eksisteerib kaks teed:

1. Muutuja mixed realiseerib ToStringInterface liidest;
5. Muutuja mixed ei realiseeri ToStringInterface liidest.

Unikaalsete teede arv läbi funktsiooni insert_default_value kahekordistub. Nüüd neid on kuus. Nii väikese funktsiooni puhul kõigi võimalike teede visualiseerimine on endiselt võimalik.

NPath keerukus on eksponentsiaalne. Selle puhul iga if-avalsuse lisamine on multiplikatiivne. See tähendab, et selleks, et saada unikaalsete teede koguarvu, me peame korrutama teede arvu läbi iga if-avalduse läbi. Seega, tingimuste lisamine funktsiooni on ohtlik. Kui lisada viimasele näidisfunktsioonile veel ühte if-avaldust, unikaalsete teede arv vähemalt kahekordistub. Ka 12 tee visualiseerimine on võimalik, kuigi võtab rohkem aega. Funktsioon hakkab muutuma keerulisemaks. Funktsioon, mille sees on 12 tingimust, omab vähemalt 4096 (2^{12}) unikaalset teed. 4096 unikaalse tee visualiseerimine ei ole juba võimalik. Kood on liiga keeruline.

Selleks, et efektiivselt testida igat teed funtsioonis, ühiktestide arv peab võrduma NPK väärtusega [54].

Tavaliselt NPK väärtus on kõrgem võrreldes TK väärtusega.

Funktsiooni NPK läve vaikeväärtus on 200 [55]. Teistes allikates on selle väärtuseks 50 ja 140 [54, 59].

4.5 Hooldatavuse indeks

HI (hooldatavuse indeks) näitab, kui kerge on koodi hooldada [60]. HI väärtus võib olla vahemikus 0-100 (46). Mida kõrgem on väärtus, seda parem on hooldatavus. Väärtus 20-100 tähendab head, 10-19 – keskmist ja 0-9 - madalat koodi hooldatavust.

HI arvutatakse, kasutades valemit (4).

$$\mathrm{HI} = \mathrm{MAX}(0, \frac{(171 - 5.2 * \log(\mathrm{HM}) - 0.23 * \mathrm{TK} - 16.2 * \log(\mathrm{SLOC})) * 100}{171}), \text{ kus} \quad (4)$$

HI – hooldatavuse indeks;

HM - Halstead maht;

TK – tsüklomaatiline keerukus;

SLOC – koodi ridade arv.

HM (Halstead maht) kasutatakse tarkvara arendamiseks vajamineva jõu ja aja mõõtmiseks. Selle arvutamine põhineb täidetud operatsioonide ja algoritmis käsitletud operandide arvul. Seega see meetrika on vähemal määral tundlik koodi vormindamise suhtes võrreldes koodi ridade arvuga.

4.6 Klasside omavaheline sõltuvus

KOS (klasside omavaheline sõltuvus, class coupling) näitab, kui palju teisi klasse konkreetne klass või meetod kasutab [61].

Mida kõrgem on selle meetrika väärtus, seda halvem. Uuringud näiavad, et selle meetrika optimaalne lävi on 9 [61].

KOS arvutamisel võtab Visual Studio arvesse meetodite parameetreid, lokaalseid muutujaid, tagastatavaid tüüpe, meetodite väljakutseid, alusklasse, liideste implementatsioone, atribuutide dekoratsioone jms.

Kõrge KOS näitab seda, et klassi on raske taaskasutada ja pidada üleval, kuna see sõltub tugevalt teistest klassidest.

4.7 Pärimise sügavus

PS (pärimise sügavus) samuti nimetatakse pärimispuu sügavuseks [62]. Selle definitsiooniks on maksimaalne pikkus antud sõlmest puu juursõlmeni klasside hierarhias [63].

Mida kõrgem on selle meetrika väärtus, seda kõrgem on koodi keerukus, kuna seda rohkem klasse ja meetodeid on kaasatud [64]. Mida kõrgem on klassi PS, seda suuremat arvu atribuute ja meetodeid antud klass tõenäoliselt pärib. See tähendab, et vigade tõenäosus kasvab ja klassi käitumist on raske ennustada. Teisest küljest, kõrgema PS väärtusega klassid taaskasutavad suurema tõenäosusega meetodeid läbi pärimise.

Pärimise sügavuse ülemiseks läveks peaks olema 5 või 6 [65]. Kui PS väärtused on enamasti alla 2, see võib rääkida sellest, et koodis kasutatakse vähe objekt-orienteeritud disaini ja pärimist.

4.8 Koodi analüüs

Antud töö osas on toodud näiteks kolme klassi analüüs: VideoMergingService, VideoCuttingService ja ThumbnailExtractingService. Kõik kolm klassi on täielikult loodud autori poolt. Klassi VideoMergingService jaoks on esitatud ka refaktoreerimise kava. Ülejäänud koodi analüüs on toodud lisas 2 Loodud ja täiendatud klasside analüüs.

4.8.1 Klass VideoMergingService

Tabel 4. Klassi VideoMergingService koodi meetrikad.

SLOC	347						
Testitavad koodi read	217						
Testimata koodi read	2						
Testide arv	51						
RK	99						
HK	95						
HI	72						
PS	1						
KOS	43						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
CompleteMergingProgressCalculation()	4	100	100	81	2	2	1
CreateThumbnailAsync(...)	11	88,9	66,7	68	6	2	5
Konstruktor	27	100	100	52	14	4	15
MergeVideoSegmentsAsync(...)	106	98,6	96,3	65	25	8192	9
MergeVideoSegmentsAsync(...)	37	100	100	67	7	16	8
NullifyVariables()	2	100	100	86	1	0	0
OnCancellationRequested(...)	10	100	100	67	3	2	4
OnMediaAdded(...)	1	100	100	95	2	2	2
OnProgressCalculationInProcessChanged(...)	1	100	100	95	2	2	2
OnProgressCalculatorPropertyChanged(...)	10	100	80	71	3	4	4
OnProgressChanged(...)	1	100	100	95	2	2	2
OnVideoSegmentsMerged(...)	1	100	100	95	2	2	2
OnVideoSegmentsParamsRecalculated(...)	1	100	100	95	2	2	2
SaveMediaCopyAsync(...)	8	100	100	68	3	0	6
StartMergingProgressCalculation(...)	13	100	100	63	4	2	7
minimaalne	1	88,9	66,7	52	1	0	0
maksimaalne	106	100	100	95	25	8192	15
keskmine	15,5	99,2	96,2	77,5	5,2	548,9	4,6

Tabel 5. Klassi VideoMergingService analüüs.

Kategooria	**Märkused**	**Järeldused**
Testkate	• Meetodi CreateThumbnailAsync(...) HK on liiga madal. • Ülejäänud meetodid on piisavalt testitud.	Meetodit CreateThumbnailAsync(...) peab rohkem testima.
Koodi keerukus	• Klassi KOS väärtus on liiga kõrge. • Konstruktori TK ja KOS väärtused on liiga kõrged. • Avalik meetod MergeVideoSegmentsAsync(...) on liiga suur ning selle TK ja NPK väärtused on liiga kõrged. • Privaatne meetod MergeVideoSegmentsAsync(...) on liiga suur. • Ülejäänud koodi meetrikad on normi piires.	Klass vajab refaktoreerimist.
Puhas kood	• Konstruktor võtab liiga palju argumente (13). • Avalik meetod MergeVideoSegmentsAsync(...) võtab liiga palju argumente (6). • Privaatne meetod MergeVideoSegmentsAsync(...) võtab liiga palju argumente (4). • Meetod CreateThumbnailAsync(...) võtab liiga palju argumente(3). • Meetod SaveMediaCopyAsync(...) võtab liiga palju argumente (3).	Konstruktor ja mainitud meetodid vajavad refaktoreerimist.

Refaktoreerimise kava:

1. Luua klassi SystemWrapper. Selle omadusteks on:
    - Liidese IFile realisatsioon;
    - Liidese IPath realisatsioon.

Selle klassi objekti anda VideoMergingService konstruktorisse.

2. Luua klaasi FfMpegTools. Selle omadusteks on:

   - Liidese IProgramRunner realisatsioon;
   - Sõne tüüpi omadus FfMpegProgramPath;
   - Liidese IFfMpegArgumentsConstructor realisatsioon;
   - Liidese ILogger realisatsioon (NLog raamistikust).

   Selle klassi objekti anda VideoMergingService konstruktorisse.

3. Luua klass VideoMergingSupportingServices. Selle omadusteks on:

   - Liidese IVideoCuttingService realisatsioon;
   - Liidese IMediaFileMetainfoService realisatsioon;
   - Liidese IThumbnailExtractingService realisatsioon;
   - Liidese ICutMediaSegmentsParamsRecalculator realisatsioon;
   - Liidese IFfMpegFilesCreationProgressCalculator realisatsioon;
   - Liidese IMediaManager realisatsioon;
   - SystemManager tüüpi objekt.

   Selle klassi objekti anda VideoMergingService konstruktorisse.

4. Luua klass VideoFileSummary. Selle omadusteks on:

   - Sõne tüüpi omadus VideoPath;
   - Sõne tüüpi omadus AudioPath;
   - Sõne tüüpi omadus VideoDirectoryPath;
   - Liidese IVideoSegments realisatsioon;
   - Sõne tüüpi omadus CopyName;
   - Sõne tüüpi omadus CopyPath;
   - Liidese IMedia realisatsioon OriginalMedia.

   Selle klassi objekti anda parameetrina järgmistesse meetoditesse:

   - Avalik MergeVideoSegmentsAsync(...);
   - Privaatne MergeVideoSegmentsAsync(...);
   - CreateThumbnailAsync(...);
   - SaveMediaCopyAsync(...);

5. Luua meetodit SetCancellationCallback(...). Asetada sellesse meetodisse järgmine kood (vt joonis 21).

```
CancellationTokenRegistration cancellationTokenRegistration =
cancel.Register(() =>
{
      OnCancellationRequested(segments);
});
```

Joonis 21. Kood, mida refaktoreerimise käigus asetatakse meetodisse SetCancellationCallback(...).

6. Luua meetodit TryCutVideoSegmentsAsync(...). Tõsta sellesse meetodisse avalikust meetodist MergeVideoSegmentsAsync(...) kogu koodi, mis on seotud videosegmentide lõikamisega.
7. Luua meetodit CompleteVideoMerging(). Asetada sellesse meetodisse järgmist koodi (vt joonis 22).

```
_systemManager.DeleteFolder(segments.SegmentsDirectoryPath, true);
NullifyVariables();
```

Joonis 22. Kood, mida refaktoreerimise käigus asetatakse meetodisse CompleteVideoMerging().

Klassis VideoMergingService meetodit CompleteVideoMerging() saab kasutada mitmes kohas.

8. Luua meetodit TryMergeVideoSegmentsAsync(...). Tõsta sellesse meetodisse avalikust meetodist MergeVideoSegmentsAsync(...) kogu koodi, mis on seotud videosegmentide ühendamisega.
9. Luua meetodit TrySaveMediaCopyAsync(...). Tõsta sellesse meetodisse avalikust meetodist MergeVideoSegmentsAsync(...) kogu koodi, mis on seotud uue meedia andmebaasi salvestamisega.
10. Luua meetodit RecalculateVideoSegmentsParams(...). Tõsta sellesse meetodisse avalikust meetodist MergeVideoSegmentsAsync(...) kogu koodi, mis on seotud videosegmentide parameetrite ümberarvutamisega.
11. Luua meetodit TryCreateThumbnailAsync(...). Tõsta sellesse meetodisse avalikust meetodist MergeVideoSegmentsAsync(...) kogu koodi, mis on seotud uuest videofailist pisipildi saamisega.
12. Privaatses meetodis MergeVideoSegmentsAsync(...) kasutada meetodi GetExitCodeAsync(...) asemel meetodit ReturnIfProcessSucceededAsync(...). Nii GetExitCodeAsync(...) kui ka ReturnIfProcessSucceededAsync(...) asuvad klassis ProgramRunner. Käesoleval hetkel meetod ReturnIfProcessSucceededAsync(...) nõuab parendamist.

See on esialgne refaktoreerimise kava ega lahenda kõiki probleeme.

4.8.2 Klass VideoCuttingService

Tabel 6. Klassi VideoCuttingService koodi meetrikad.

SLOC	102						
Testitavad koodi read	60						
Testimata koodi read	4						
Testide arv	17						
RK	93,3						
HK	90,9						
HI	62						
PS	1						
KOS	16						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
CutVideoSegmentsAsync(...)	52	90,2	89,5	67	14	512	7
Konstruktor	16	100	100	60	7	4	8
keskmine	34	95,1	94,7	63,5	10,5	258	7,5

Tabel 7. Klassi VideoCuttingService analüüs.

Kategooria	**Märkused**	**Järeldused**
Testkate	Klass on piisavalt testitud.	Uusi teste ei ole vaja lisada.
Koodi keerukus	• Klassi KOS väärtus on liiga kõrge. • Meetod CutVideoSegmentsAsync(...) on liiga suur ning selle TK ja NPK väärtused on liiga kõrged. • Ülejäänud koodi meetrikad on normi piires.	Klass vajab refaktoreerimist.
Puhas kood	• Konstruktor võtab liiga palju argumente (6). • Meetod CutVideoSegmentsAsync(...) võtab liiga palju argumente (4).	Konstruktor ja meetod CutVideoSegmentsAsync(...) vajavad refaktoreerimist.

4.8.3 Klass ThumbnailExtractingService

Tabel 8. Klassi ThumbnailExtractingService koodi meetrikad.

SLOC	95						
Testitavad koodi read	56						
Testimata koodi read	0						
Testide arv	19						
RK	100						
HK	100						
HI	63						
PS	1						
KOS	16						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
ExtractAsync(...)	45	100	100	68	10	128	6
Konstruktor	18	100	100	59	8	4	8
keskmine	31,5	100	100	63,5	9	66	7

Tabel 9. Klassi ThumbnailExtractingService analüüs.

Kategooria	**Märkused**	**Järeldused**
Testkate	Klass on piisavalt testitud.	Teste ei ole vaja lisada.
Koodi keerukus	• Klassi KOS väärtus on liiga kõrge. • Konstruktori TK väärtus on liiga kõrge. • Meetod ExtractAsync(...) on liiga suur ning selle TK väärtus on liiga kõrge. • Ülejäänud koodi meetrikad on normi piires.	Klass vajab refaktoreerimist.
Puhas kood	• Konstruktor võtab liiga palju argumente (6). • Meetod ExtractAsync(...) võtab liiga palju argumente (3).	Konstruktor ja meetod ExtractAsync(...) vajavad refaktoreerimist.

4.8.4 Üldised järeldused.

Töö käigus oli kirjutatud 129 ühik- ja 8 integratsioonitesti.

Kõige levinumad probleemid koodis on järgmised:

1. Koodi ebapiisav testkate;
2. Omadused ei ole testitud. See on tingitud sellest, et tööriist OpenCover ei näita koodikatet omaduste jaoks ega võta omadusi arvesse klassi koodikatte arvutamisel.
3. Liiga suured meetodid;
4. Liiga kõrged TK väärtused;
5. Liiga kõrged NPK väärtused;
6. Liiga kõrged KOS väärtused;
7. Meetodid võtavad liiga palju argumente.

Lisaks sellele, koodi probleemiks on see, et ühe ja sama asja nimetamisel ei kasutata samu termineid. Klassis MediaEditorViewModel on omadus CueTags, mis on märgendatud videosegmentide kollektsiooniks. See on SortedObservableCollection tüüpi objekt, mis sisaldab CueTrackViewModel tüüpi objekte. Samas CueTrackViewModel objektide loendit teisendab VideoSegments tüüpi objektiks ja vastupidi CueTrackConverter. Õigemini oleks kasutada kõikjal ühte ja sama terminit, kas CueTag või CueTrack.

Samuti pärimise sügavus on loodud klassides enamasti 1, mis räägib sellest, et autor kasutab vähe pärimist ja objekt-orienteeritud disaini.

Vaatamata sellele, et vajalik funktsionaalsus on realiseeritud, kood ei ole ideaalne ning vajab refaktoreerimist.

4.9 Tulevikuplaanid

Käesoleval hetkel videosegmentide väljalõikamine ja ühendamine toimub järgnevalt:

1. Kasutaja loob märgendatud segmente;
2. Kasutaja vajutab nuppu Merge;
3. Videosegmentide väljalõikamise ja ühendamise protsess võtab aega, seetõttu kasutaja peab ootama kuni protsessi lõpuni.

4. Pärast seda, kui videosegmendid on ühendatud, võib kasutaja MediaEditorView-t kinni panna. Vastasel juhul videosegmentide väljalõikamise ja ühendamise protsessi katkestatakse.

Sellisel kujul lahendus ei sobi kasutamiseks. Tulevikus on vaja viia videosegmentide väljalõikamise ja ühendamise protsessi tausta. Sellisel juhul saavad samaaegselt toimuda mitu videosegmentide väljalõikamise ja ühendamise protsessi. Lisaks sellele, on kasutajal võimalus sulgeda MediaEditorView-t enne seda, kui väljalõikamise ja ühendamise protsess on lõppenud.

Kokkuvõte

ORNet meditsiinilahenduste perekonda kuuluvad kolm toodet. ORNet Editor on tarkvarasüsteemiks ilma riistvaralise osata. ORNet Surgery ja ORNet Pathology koosnevad nii tark- kui ka riistvaralistest komponentidest.

Käesoleva töö objektiks on mainitud toodete tarkvara, mis on Windows Presentation Foundation tehnoloogia abil loodud töölauarakendus.

Ettevõtte BaitPartner OÜ juhtkond otsustas laiendada antud tarkvara funktsionaalsust. Autori ees püstitatud ülesandeks oli arendada uus video failide taasesitamise ja töötlemise eest vastutav komponent. Seda komponenti hakatakse tulevikus kasutama kõigis ORNet perekonda kuuluvates lahendustes.

Kuna antud töö omab piiratud mahtu, on selle eesmärgiks ainult uuele komponendile järgmise funktsionaalsuse lisamine: märgendatud segmentide väljalõikamine videofailist ja nende liitmine uueks failiks.

Töös on osaliselt kasutatud testimisel põhineva arenduse lähenemisviisi. Töös on esitatud funktsionaalsed ja mittefunktsionaalsed nõuded uuele komponendile, ORNet tarkvara arhitektuur ja komponendi koodi kirjutamisel kasutatud mustrid. Töö käigus oli loodud 18 uut klassi, 10 uut liidest ja kirjutatud rohkem kui 1550 rida tootmiskoodi. Töös on samuti antud koostööskeemid, mis näitavad, kuidas kood töötab, ning klassidiagrammid, mis näitavad, kuidas klassid ja liidesed suhtlevad omavahel. Lisas on esitatud uute ja täiendatud klasside kirjeldused.

Töö käigus koguti ja analüüsiti järgnevaid meetrikaid: reakate, harukate, jadakate, koodi ridade arv, hooldatavuse indeks, klasside omavaheline sõltuvus, pärimise sügavus, tsüklomaatiline keerukus ja N-Path keerukus. Töö käigus oli kirjutatud 137 ühik- ja integratsioonitesti. Vaatamata sellele, koodi testkate on ebapiisav. Kirjutatud kood vajab refaktoreerimist. Klassid on liiga tugevalt seotud üksteisega. Koodis leiduvad meetodid, mis on liiga suured, võtavad liiga palju argumente ja omavad liiga kõrget keerukust. Probleemiks on ka see, et koodi kirjutamisel kasutati vähe objekt-orienteeritud disaini.

Ettevõttepoolne tagasiside on antud lisas 3 Ettevõttepoolne tagasiside.

Kasutatud kirjandus

[1] M. Fowler, "Active Record," [Online]. Available: https://www.martinfowler.com/eaaCatalog/activeRecord.html. [Accessed 09 Dec. 2019].

[2] The FFmpeg project, "Wiki: concatenate," [Online]. Available: https://trac.ffmpeg.org/wiki/Concatenate. [Accessed 10 Oct. 2019].

[3] Microsoft, "Members (C# Programming Guide)," 20 Jul. 2015. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/members. [Accessed 27 Sept. 2019].

[4] Bait Partner Ltd., "Bait Partner," Bait Partner Ltd., [Online]. Available: https://baitpartner.eu/avaleht/. [Accessed 01 Sept. 2019].

[5] Bait Partner Ltd., "ORNet," Bait Partner Ltd., [Online]. Available: https://ornet.eu/. [Accessed 01 Sept. 2019].

[6] Bait Partner Ltd., "ORNet Surgery user manual," Bait Partner Ltd., Tallinn, 2017.

[7] Bait Partner Ltd., "ORNet Surgery," Bait Partner Ltd., [Online]. Available: https://ornet.eu/ornet-surgery/. [Accessed 01 Sept. 2019].

[8] Bait Partner Ltd., "ORNet Pathology User Manual," Bait Partner Ltd., Tallinn, 2018.

[9] Bait Partner Ltd., "ORNet Pathology," Bait Partner Ltd., [Online]. Available: https://ornet.eu/pathology/. [Accessed 01 Sept. 2019].

[10] Bait Partner Ltd., "ORNet Capture User Manual," Bait Partner Ltd., Tallinn, 2015.

[11] Bait Partner Ltd., "ORNet Capture," Bait Partner Ltd., [Online]. Available: https://ornet.eu/ornet-capture/. [Accessed 01 Sept. 2019].

[12] ITU-T, "Recommendation H.265 (06/19)," ITU-T, 2019.

[13] ITU-T, "Recommendation H.264 (06/19)," ITU-T, 2019.

[14] A. Nathan, WPF 4.5 Unleashed, Indianapolis, Indiana: Sams Publishing, 2014.

[15] J. Likness, "Model-View-ViewModel (MVVM) Explained," 24 Apr. 2014. [Online]. Available: https://www.wintellect.com/model-view-viewmodel-mvvm-explained/. [Accessed 31 Aug. 2019].

[16] M. Fowler, Refactoring: improving the design of existing code, Boston: Addison-Wesley, 2000.

[17] Health Level Seven International, "HL7 Messaging Standard Version 2.3," Health Level Seven International, Ann Arbor, 1997.

[18] Health Level Seven International, "HL7 Messaging Standard Version 2.3.1," Health Level Seven International, Ann Arbor, 1999.

[19] Health Level Seven International, "HL7 Messaging Standard Version 2.4," Health Level Seven International, Ann Arbor, 2000.

[20] National Electrical Manufacturers Association, "Digital Imaging and Communications in Medicine (DICOM)," National Electrical Manufacturers Association, 2019.

[21] O. S. Pianykh, Digital imaging and communication in medicine (DICOM): a practical introduction and survival guide, Berlin: Springer, 2012.

[22] Fa-Tech Diagnostics Europe BV, "Fa-Tech NOVA Laser Cassettes Marker," [Online]. Available: https://www.fatechdiagnostics.com/wp-content/uploads/2019/10/Nova_ENG.pdf. [Accessed 26 Nov. 2019].

[23] Bait Partner Ltd., "ORNet Story," [Online]. Available: https://ornet.eu/ornet-story/. [Accessed 30 Oct. 2019].

[24] A. Troelsen, PRO C# 5.0 and the .NET 4.5 framework., New York: Apress, 2012.

[25] JetBrains, "ReSharper," JetBrains, [Online]. Available: https://www.jetbrains.com/resharper/. [Accessed 02 Sept. 2019].

[26] C. Poole, R. Prouse, S. Busoli and N. Colvin, "NUnit," [Online]. Available: https://nunit.org/. [Accessed 02 Sept. 2019].

[27] Hibernating Rhinos, "Rhino Mocks," Hibernating Rhinos, [Online]. Available: https://hibernatingrhinos.com/oss/rhino-mocks. [Accessed 03 Sept. 2019].

[28] The FFmpeg project, "FFprobe documentation," The FFmpeg project, [Online]. Available: https://ffmpeg.org/ffprobe.html. [Accessed 03 Sept. 2019].

[29] The FFmpeg project, "FFmpeg documentation," The FFmpeg project, [Online]. Available: https://ffmpeg.org/ffmpeg.html. [Accessed 03 Sept. 2019].

[30] The FFmpeg project, "FFmpeg documentation," The FFmpeg project, [Online]. Available: https://ffmpeg.org/ffmpeg-all.html. [Accessed 03 Sept 2019].

[31] "FFmpeg.Shared 4.0.2," [Online]. Available: https://www.nuget.org/packages/FFmpeg.Shared/. [Accessed 04 Sept. 2019].

[32] K. Schwarz, "FFmpeg builds," [Online]. Available: https://ffmpeg.zeranoe.com/builds/. [Accessed 03 Sept. 2019].

[33] MediaArea, "MediaInfo," MediaArea, [Online]. Available: https://mediaarea.net/en/MediaInfo. [Accessed 03 Sept. 2019].

[34] J. Kowalski, K. Christensen and J. Verdurmen, "NLog," [Online]. Available: https://nlog-project.org/. [Accessed 04 Sept. 2019].

[35] VSoft Technologies Pty Ltd., "Continua CI," VSoft Technologies Pty Ltd., [Online]. Available: https://wiki.finalbuilder.com/. [Accessed 04 Sept. 2019].

[36] TortoiseSVN, "TortoiseSVN," TortoiseSVN, [Online]. Available: https://tortoisesvn.net/. [Accessed 04 Sept. 2019].

[37] Targetprocess, "Targetprocess," Targetprocess, [Online]. Available: https://www.targetprocess.com/. [Accessed 04 Sept. 2019].

[38] JGraph Ltd., "draw.io," JGraph Ltd., [Online]. Available: https://about.draw.io/. [Accessed 04 Sept. 2019].

[39] yWorks, "yEd graph editor: high quality diagrams made easy," yWorks, [Online]. Available: https://www.yworks.com/products/yed. [Accessed 04 Sept. 2019].

[40] "OpenCover," [Online]. Available: https://github.com/OpenCover/opencover. [Accessed 04 Sept. 2019].

[41] K. Beck, Test-driven development: by example, Boston: Addison-Wesley, 2003.

[42] R. C. Martin, Clean Code: a handbook of agile software craftsmanship, Upper Saddle River: Prentice Hall, 2009.

[43] SeaLights, "Code Coverage Metrics," [Online]. Available: https://www.sealights.io/test-metrics/code-coverage-metrics/. [Accessed 05 Sept. 2019].

[44] G. M. Hall, Adaptive code via C#: Agile coding with design patterns and SOLID principles, Redmond: Microsoft Press, 2014.

[45] M. Moskala, "Effective Java in Kotlin, item 1: Consider static factory methods instead of constructors," 19 Mar. 2018. [Online]. Available: https://blog.kotlin-academy.com/effective-java-in-kotlin-item-1-consider-static-factory-methods-instead-of-constructors-8d0d7b5814b2. [Accessed 31 Oct. 2019].

[46] E., "Java Constructors vs Static Factory Methods," 11 Sept. 2019. [Online]. Available: https://www.baeldung.com/java-constructors-vs-static-factory-methods. [Accessed 31 Oct. 2019].

[47] "Code coverage," [Online]. Available: https://en.wikipedia.org/wiki/Code_coverage. [Accessed 06 Oct. 2019].

[48] "Code Coverage Tutorial: Branch, Statement, Decision, FSM," [Online]. Available: https://www.guru99.com/code-coverage.html. [Accessed 06 Oct. 2019].

[49] M. Fowler, "TestCoverage," 17 Apr. 2012. [Online]. Available: https://www.martinfowler.com/bliki/TestCoverage.html. [Accessed 06 Oct. 2019].

[50] JMockit , "Measuring code coverage," [Online]. Available: https://jmockit.github.io/tutorial/CodeCoverage.html. [Accessed 06 Oct. 2019].

[51] S. Pittet, "An introduction to code coverage," [Online]. Available: https://www.atlassian.com/continuous-delivery/software-testing/code-coverage. [Accessed 06 Oct. 2019].

[52] "Точка следования," [Online]. Available: https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D1%87%D0%BA%D0%B0_%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F. [Accessed 06 Oct. 2019].

[53] "Source lines of code," [Online]. Available: https://en.wikipedia.org/wiki/Source_lines_of_code. [Accessed 03 Oct. 2019].

[54] B. Savage, "Code complexity and clean code," 22 May 2013. [Online]. Available: https://www.brandonsavage.net/code-complexity-and-clean-code/. [Accessed 01 Oct. 2019].

[55] N. Modess, "NPath complexity and cyclomatic complexity explained," 19 May 2013. [Online]. Available: https://modess.io/npath-complexity-cyclomatic-complexity-explained/. [Accessed 01 Oct. 2019].

[56] "Cyclomatic complexity," [Online]. Available: https://en.wikipedia.org/wiki/Cyclomatic_complexity. [Accessed 01 Oct. 2019].

[57] "Understanding cyclomatic complexity," [Online]. Available: https://blog.ndepend.com/understanding-cyclomatic-complexity/. [Accessed 01 June 2019].

[58] J. Lian, "NPath complexity and cyclomatic complexity explained," 19 Feb. 2017. [Online]. Available: https://www.w3c-lab.com/npath-complexity-cyclomatic-complexity-explained/. [Accessed 01 Oct. 2019].

[59] C. A., "What is software complexity and how can you manage it?," [Online]. Available: https://carlalexander.ca/what-is-software-complexity/. [Accessed 09 Nov. 2019].

[60] Z. Naboulsi, "Code Metrics – Maintainability Index," 26 May 2011. [Online]. Available: https://blogs.msdn.microsoft.com/zainnab/2011/05/26/code-metrics-maintainability-index/. [Accessed 07 Oct. 2019].

[61] Z. Naboulsi, "Code Metrics – Class Coupling," 25 May 2011. [Online]. Available: https://blogs.msdn.microsoft.com/zainnab/2011/05/25/code-metrics-class-coupling/. [Accessed 07 Oct. 2019].

[62] Z. Naboulsi, "Code Metrics – Depth of Inheritance (DIT)," 19 May 2011. [Online]. Available: https://blogs.msdn.microsoft.com/zainnab/2011/05/19/code-metrics-depth-of-inheritance-dit/. [Accessed 07 Oct. 2019].

[63] J. Ingeno, Software architect's handbook: become a successful software architect bi implementing effective architecture concepts, Birmingham: Packt Publishing Ltd., 2018.

[64] K. Nayyeri, "Depth of Inheritance for WPF and Windows Forms Applications," 28 Dec. 2007. [Online]. Available: http://web.archive.org/web/20101216141549/http://nayyeri.net/depth-of-inheritance-for-wpf-and-windows-forms-applications. [Accessed 07 Oct 2019].

[65] Lite Solutions, "Depth of Inheritance Tree," [Online]. Available: https://www.cachequality.com/docs/metrics/depth-inheritance-tree. [Accessed 07 Oct. 2019].

[66] S. V. Deursen and M. Seeman, Dependency injection: principles, practices, and patterns, Shelter Island: Manning Publications Co., 2019.

[67] H. Roberts, "Cyclomatic Complexity: Logic in CSS," 26 Apr. 2015. [Online]. Available: https://csswizardry.com/2015/04/cyclomatic-complexity-logic-in-css/?utm_source=html5weekly&utm_medium=email. [Accessed 01 Oct. 2019].

Lisa 1. Töö eesmärgi täitmiseks vajalike klasside kirjeldused.

Enumeraatorid, sündmuste argumendid ja klass Constants ei ole koodi kirjelduses mainitud.

Klasside kirjeldamisel kasutatakse järgmiseid märkuseid:

- [1] – v.a konstruktor;
- [2] - klass ei ole täielikult autori poolt kirjutatud;
- [3] - kuigi antud klassis on rohkem avalikke liikmeid, selles töös on oluline mainida ainult mõned;
- [4] – klass ei ole autori poolt kirjutatud;
- [5] – klass kasutab oma tööks programmi ffmpeg.

Klasside kirjeldused on esitatud klasside tähestikulises järjekorras.

Tabel 10. Klass CueTrackConverter.

Ülesanne	Teisendab liidese IList<CueTrackViewModel> realisatsiooni liidese IVideoSegments realisatsiooniks ning vastupidi.	
Realiseerib liidest	ICueTrackConverter	
Mustrid	Stairway	
Avalikud liikmed[1]		
Nimi	**Tüüp**	**Selgitus**
Convert(...)	meetod	Teisendab liidese IList<CueTrackViewModel> realisatsiooni liidese IVideoSegments realisatsiooniks.
ConverBack(...)	meetod	Teisendab liidese IVideoSegments realisatsiooni liidese IList<CueTrackViewModel> realisatsiooniks.

Tabel 11. Klass CueTrackViewModel[2].

Ülesanne	Märgendatud videosegmendi vaate mudel.	
Laiendab klassi	Abstraktne BaseViewModel	
Avalikud liikmed[1,3]		
Nimi	**Tüüp**	**Selgitus**
FromCueFile(...)	Meetod	Loeb märgendatud videosegmente .cue laiendiga failist ja tagastab CueTrackViewModel tüüpi objektide massiivi.
ToCueFile(...)	Meetod	Salvestab märgendatud videosegmente .cue laiendiga faili.
EndPosition	Omadus	Märgendatud videosegmendi lõpp-positsioon videofailis, s
StartPosition	Omadus	Märgendatud videosegmendi alguspositsioon videofailis, s
Title	Omadus	Märgendatud videosegmendi nimetus

Tabel 12. Klass CutMediaSegmentsParamsRecalculator.

Ülesanne	Arvutab videosegmentide alguspositsioone ja kestvusi uues videosegmentide ühendamise tagajärjel tekitatud videofailis.	
Realiseerib liidest	ICutMediaSegmentsParamsRecalculator	
Mustrid	Stairway, dependency injection, static factory method	
Avalikud liikmed		
Nimi	**Tüüp**	**Selgitus**
Create(...)	Meetod	Tagastab uut ICutMediaSegmentsParamsRecalculator liidese realisatsiooni.
Recalculate(...)	Meetod	Arvutab videosegmentide alguspositsioonide ja kestvuste uuei väärtusi.

Tabel 13. Klass EnumHelper[2].

Ülesanne	Võimaldab teostada erinevaid operatsioone enumeraatoritega.	
Märkused	Staatiline klass	
Avalikud liikmed[3]		
Nimi	**Tüüp**	**Selgitus**
GetDescription(...)	Meetod	Võimaldab saada enumeraatori liikme Description atribuudi värtust sõnena.

Tabel 14. Klass FfMpegArgumentsConstructor.

Ülesanne	Konstrueerib argumente programmi ffmpeg jaoks erinevate operatsioonide teostamise eesmärgil	
Realiseerib liidest	IFfMpegArgumentsConstructor	
Mustrid	Stairway, dependency injection	
Avalikud liikmed[1,3]		
Nimi	**Tüüp**	**Selgitus**
ConstructForCutting(...)	meetod	Konstrueerib argumente videofailist segmendi väljalõikamiseks. Kui on antud tee audiofailini, segmenti lõigatakse välja ka audiofailist. Saadud segmente ühendatakse üheks videofailiks, mis omab nii video- kui ka audiovoogu.
ConstructForExtractingThumbnail(...)	meetod	Konstrueerib argumente videofailist pildi võtmiseks.
ConstructForMergingVideoSegments(...)	meetod	Konstrueerib argumente videosegmentide ühendamiseks.

Tabel 15. Klass FfMpegFilesCreationProgressCalculator.

Ülesanne	Arvutab programmi ffmpeg poolt failide loomise protsessi kulgu protsentides. Arvutamisel kasutatakse valemit (5): $a = \frac{b}{c} \times 100\%$, kus (5) a – protsessi kulg protsentides; b – loodud sekundid arv; c – protsessi käigus loodavate failide kogupikkus sekundites.	
Realiseerib liidest	IFfMpegFilesCreationProgressCalculator, INotifyPropertyChanged	
Mustrid	Stairway, static factory method	
Avalikud liikmed[1]		
Nimi	**Tüüp**	**Selgitus**
AddFile(...)	meetod	Liidab faili pikkust loodavate failide kogupikkusele
CompleteCalculation()	meetod	Lõpetab falide loomise protsessi kulu arvutamist.

Nimi	Tüüp	Selgitus
CompleteFile()	meetod	Arvutab failide loomise protsessi kulgu lähtuvalt sellest, et eelmise faili loomine on lõpetatud.
Create()	meetod	Tagastab uut IFfMpegFilesCreationProgressCalculator liidese realisatsiooni.
OnFileCurrentCreationPositionReceived	meetod	Arvutab failide loomise protsessi kulgu arvestades saadud jooksva faili loodud sekundite arvu.
StartCalculation()	meetod	Alustab uue failide loomise protsessi kulu arvutamist.
StartFile()	meetod	Jätab meelde, et edaspidi saadud loodud sekundite arv kehtib järgmise faili kohta.
CurrentProgressInPercent	omadus	Jooksev failide loomise protsessi kulg protsentides.
IsCalculationInProgress	omadus	On tõene, kui failide loomise protsessi kulu arvutamine on käimas, ning väär vastasel juhul.
PropertyChanged	Sündmus	Käivitatakse siis, kui omadused muutuvad.

Tabel 16. Klass FFMpegMergingInputConstructor.

Ülesanne	Konstrueerib sisendit konkateneerimise "protokolli" (the concat "protocol", üks programmi ffmpeg videofailide ühendamise võimalustest) jaoks. Antud sisend on osa programmile ffmpeg antavatest argumentidest.	
Realiseerib liidest	IFFMpegMergingInputConstructor	
Mustrid	Stairway	
Märkused	Kasutades programmi ffmpeg on võimalik ühendada videosegmente, kasutades 3 viisi: Konkateneerimise "demultipleksor" (the concat "demuxer"). Konkateneerimise "protokoll" (the concat "protocol"). Konkateneerimise "filter" (the concat "filter"). Antud töös kasutatakse konkateneerimise "protokolli".	
Avalikud liikmed[1]		
Nimi	**Tüüp**	**Selgitus**
Construct(...)	meetod	Täidab klassi ülesannet.

Tabel 17. Klass FFProbeXmlParser[2].

Ülesanne	Töötleb programmi ffprobe standardväljundit, mis on XML kujul.	
Avalikud liikmed[1]		
Nimi	**Tüüp**	**Selgitus**
ParseMetainfo(...)	Meetod	Täidab klassi ülesannet.

Tabel 18. Klass FileNameGenerator[4].

Ülesanne	Genereerib patsientide kataloogides asuvate failide nimesid.	
Realiseerib liidest	IFileNameGenerator	
Mustrid	Stairway, dependency injection	
Avalikud liikmed[1,3]		
Nimi	**Tüüp**	**Selgitus**
Folder	omadus	Absoluutne tee kataloogini.
GenerateNextFileName()	meetod	Genereerib järgmist faili nime.
GetName()	meetod	Tagastab jooksvat faili nime.

Tabel 19. Klass Media2.

Ülesanne	Hoiab informatsiooni meedia kohta, olemuselt on Active record (Objekt, mis esindab rida andmebaasi tabelis või vaates, kapseldab juurdepääsu andmebaasi ja lisab domeeni loogika andmetele).	
Laiendab klassi	EntityBase	
Realiseerib liidest	IMedia	
Avalikud liikmed[1,3]		
Nimi	**Tüüp**	**Selgitus**
Save()	meetod	Salvestab meediat andmebaasi.

Tabel 20. Klass MediaEditorView[2].

Ülesanne	Uue videofailide taasesitamise ja töötlemise eest vastutava komponendi kasutajaliides.
Realiseerib liidest	INotifyPropertyChanged
Laiendab klassi	Window
Avalikud liikmed[1,3]	

Tabel 21. Klass MediaEditorViewModel[2].

Ülesanne	Uue videofailide taasesitamise ja töötlemise eest vastutava komponendi vaate mudel.	
Laiendab klassi	Abstraktne BaseViewModel	
Mustrid	Dependency injection	
Avalikud liikmed[1,3]		
Nimi	**Tüüp**	**Selgitus**
CancelMergingCommand	Käsk	ProgressReportControl-is on seotud nupuga „Cancel“. MediaEditorViewModel-is käivitab privaatset meetodit CancelMerging, kus katkestatakse alustatud videosegmentide ühendamise protsessi.
MergeVideoSegmentsCommand	Käsk	MediaEditorView aknas on seotud nupuga „Merge“. MediaEditorViewModel-is käivitab privaatset meetodit MergeVideoSegments, kus ühendatakse märgendatud videosegmente.
Close()	Meetod	Sündmuste küljest võetakse ära sündmusetöötlejaid.
LoadMedia(...)	Meetod	Hangitakse informatsiooni meediafaili/-failide (juhul kui videofailiga paralleelselt salvestati ka audiofaili) kohta ja laetakse märgendeid.
ActivityName	Omadus	Tegevuse nimetus (nt videofaili töötlemine)
ActivityProgressInPercent	Omadus	Tegevuse (nt videosegmentide ühendamine) edasiminek protsentides.
AudioPath	Omadus	Absoluutne tee audiofailini
CueTags	Omadus	Märgendatud videosegmentide (CueTrackViewModel objektide) kollektsioon (ObservableCollection).
IsActivityInProgress	Omadus	On tõene siis, kui mingi tegevus on käimas, ja väär vastasel juhul
IsLoadingCueTags	Omadus	On tõene siis, kui käimas on väljalõigatud ja ühendatud videosegmentide parameetrite ümberarvutamise protsess, ja väär vastasel juhul.
IsRemainingTimeShown	Omadus	On tõene siis, kui käimasoleva protsessi lõpuni jäänud aega on vaja näidata, ja väär vastasel juhul.
VideoPath	Omadus	Absoluutne tee videofailini
VideoResolution	Omadus	Videovoo resolutsioon
MediaAdded	Sündmus	Käivitatakse siis, kui videosegmentide väljalõikamine ja ühendamine on lõpetatud.

Tabel 22. Klass MediaFileMetainfoService[2].

Ülesanne	Võtab meediafailist metaandmeid.	
Realiseerib liidest	IMediaFileMetainfoService	
Mustrid	Stairway, Dependency injection	
Märkused	Kasutab oma tööks programmi ffprobe	
Avalikud liikmed[1]		
Nimi	**Tüüp**	**Selgitus**
GetMediaFileDurationAsync(...)	Meetod	Tagastab meediafaili pikkust sekundites.
GetMediaFileMetainfoAsync(...)	Meetod	Tagastab MediaMetainfo tüüpi objekti, mis kannab informatsiooni antud meediafaili kohta.

Tabel 23. Klass MediaItem[2].

Ülesanne	Kasutajaliidese elemendi MeediaItemControl vaate mudel.
Realiseerib liidest	IMediaItem, INotifyPropertyChanged
Mustrid	Dependency injection
Avalikud liikmed[1,3]	

Tabel 24. Klass MediaManager[2].

Ülesanne	Teostab erinevaid meediafailidega seotud tegevusi.	
Realiseerib liidest	IMediaManager	
Mustrid	Dependency injection	
Avalikud liikmed[1,3]		
Nimi	**Tüüp**	**Selgitus**
CreateThumbnailAsync(...)	meetod	Loob antud pildist õigete mõõtmetega pisipilti.
GetMediaCopy(...)	meetod	Loob meedia objektist koopiat, kus on muudetud meedia nimi, pikkus ja DICOM staatus.

Tabel 25. Klass MediaMetainfo[2].

Ülesanne	Hoiab meediafaili metaandmeid.	
Realiseerib liidest	IMediainfo	
Mustrid	Stairway	
Avalikud liikmed1		
Nimetus	**Tüüp**	**Tähendus**
AudioBitRate	Omadus	Audiovoo bitikiirus, kb/s
AudioSampleRate	Omadus	Audiovoo diskreetimissagedus, Hz
ContainerFormat	Omadus	Videofaili puhul konteineriformaat
Resolution	Omadus	Videovoo resolutsioon
VideoBitRate	Omadus	Videovoo bitikiirus, kb/s
VideoEncoding	Omadus	Videovoo kodeerimisviis

Tabel 26. Klass OpenMediaItemCommand[2].

Ülesanne	Avab valitud meediafaili sobivat tüüpi kasutajaliideses.	
Realiseerib liidest	ICommand	
Mustrid	Dependency injection	
Avalikud liikmed[1]		
Nimi	**Tüüp**	**Selgitus**
CanExecute(...)	meetod	Tagastab, kas meediafaili avamine on võimalik.
Execute(...)	meetod	Juhul kui valitud meedia faili tüüp on video, avab seda MediaEditorView tüüpi aknas. Teiste meediafailide tüüpide korral kasutatakse teist tüüpi aknaid.

Tabel 27. Klass ProgramRunner[2].

Ülesanne	Käivitab etteantud programmi	
Realiseerib liidest	IProgramRunner	
Avalikud liikmed[1,3]		
Nimi	**Tüüp**	**Selgitus**
GetExitCodeAsync(...)	meetod	Käivitab programmi ja tagastab väljumiskoodi.
GetLineBeginningWithFromStandardErrorAsync(...)	meetod	Käivitab programmi ja tagastab standardvea voo rida, mis algab etteantud sümbolite kombinatsiooniga.
GetStandardOutputAsync(...)	meetod	Käivitab programmi ja tagastab standardväljundit.

Tabel 28. Klass ProgressReportControl.

Ülesanne	Kasutajaliidese element. Näitab tegevuse nimetust ja kulgu protsentides. Omab edenemisriba. Omab nuppu, mida on võimalik kasutada tegevuse katkestamiseks. Soovi korral on võimalik näidata aega, mis on jäänud tegevuse lõpuni, kuid see ei puutu antud töösse.	
Laiendab klassi	UserControl	
Avalikud liimed[1,3]		
Nimi	**Tüüp**	**Selgitus**
ActivityName	omadus	Tegevuse nimetus
CancelCommand	omadus	Käsk, mida käivitatakse nupu „Cancel“ vajutamisel.
IsRemainingTimeShown	omadus	On tõene, kui tegevuse lõpuni jäänud aega on vaja näidata, ning on väär vastasel juhul.
Percentage	omadus	Tegevuse jooksev kulg protsentides.

Tabel 29. Klass SortedObservableCollection<T>.

Ülesanne	Kasutatakse, kui on vaja näidata kasutajaliideses sorteeritud kollektsiooni.	
Laiendab klassi	ObservableCollection<T>	
Avalikud liikmed[1]		
Nimi	**Tüüp**	**Selgitus**
Sort(...)	meetod	Lisab kollektsioonile sorteerimise viisi.
UpdateOrder()	meetod	Uuendab kollektsiooni esitlust kasutajaliideses.

Tabel 30. Klass SystemManager[2].

Ülesanne	Võimaldab saata päringuid ja käske operatsioonisüsteemile.	
Mustrid	Dependency injection	
Avalikud liikmed[1,3]		
Nimi	**Tüüp**	**Selgitus**
CreateNewFolder(...)	meetod	Loob kausta
DeleteFolder(...)	meetod	Kustutab kausta

Tabel 31. Klass SystemSetting[4].

Ülesanne	ORNet programmi seadistused, mida hoitakse andmebaasis.
Laiendab klassi	SystemSettingBase
Avalikud liikmed[1,3]	

Tabel 32. Klass ThumbnailExtractingService[5].

Ülesanne	Võtab videofailist pilti.	
Realiseerib liidest	IThumbnailExtractingService	
Mustrid	Stairway, dependency injection	
Avalikud liikmed1		
Nimi	**Tüüp**	**Selgitus**
ExtractAsync(...)	Meetod	Täidab klassi ülesannet.

Tabel 33. Klass VideoCuttingService[5].

Ülesanne	Lõikab videofailist segmente välja.	
Realiseerib liidest	IVideoCuttingService	
Mustrid	Stairway, Dependency injection	
Avalikud liikmed[1]		
Nimi	**Tüüp**	**Selgitus**
CutVideoSegmentsAsync(...)	meetod	Teostab klassi ülesannet. Kui on antud tee audiofailini, segmenti lõigatakse välja ka audiofailist. Saadud segmente ühendatakse üheks videofailiks, mis omab nii video- kui ka audiovoogu.

Tabel 34. Klass VideoMergingService[5].

Ülesanne	Ühendab videosegmente üheks videofailiks.	
Realiseerib liidest	IVideoMergingService	
Mustrid	Stairway, dependency injection	
Avalikud liikmed[1]		
Nimi	**Tüüp**	**Selgitus**
MergeVideoSegmentsAsync(...)	meetod	Täidab klassi ülesannet.
MediaAdded	sündmus	Käivitatakse siis, kui uus meedia on lisatud andmebaasi.
ProgressCalculationInProcessChanged	sündmus	Käivitatakse, kui videosegmentide ühendamise protsess alustatakse või lõpetatakse.
ProgressChanged	sündmus	Käivitatakse siis, kui videosegmentide ühendamise protsessi kulule antakse uut väärtust (protsentides).
VideoSegmentsMerged	sündmus	Käivitatakse siis, kui videosegmendid on ühendatud uueks videofailiks.
VideoSegmentsParamsRecalculated	sündmus	Märgendatud videosegmendid jäävad ka uues loodud videofailis, kuid on vaja arvutada nende parameetreid (alguspositsioon ja pikkus) ümber. Seda sündmust käivitatakse siis, kui need parameetrid on ümberarvutatud.

Tabel 35. Klass VideoSegment.

Ülesanne	Hoiab informatsiooni videofaili märgendatud segmendi kohta.	
Realiseerib liidest	IVideoSegment	
Mustrid	Stairway	
Avalikud liikmed[1]		
Nimi	**Tüüp**	**Tähendus**
AudioEncoding	Omadus	Videosegmendi audiovoo kodeerimisviis
Duration	Omadus	Videosegmendi pikkus, s
FileName	Omadus	Programmi poolt videosegmendi failile antud nimi
StartPosition	Omadus	Videosegmendi alguspositsioon, s
TagTitle	Omadus	Kasutaja poolt märgendatud segmendile antud nimi
VideoEncoding	Omadus	Videosegmendi videovoo kodeerimisviis

Tabel 36. Klass VideoSegments.

Ülesanne	On videosegmentide loendiks.	
Laiendab klassi	List<IVideoSegment>	
Realiseerib liidest	IVideoSegments	
Mustrid	Stairway	
Avalikud liikmed[1]		
Nimi	**Tüüp**	**Selgitus**
Add(...)	meetod	Lisab videosegmenti loendisse.
GetEnumerator()	meetod	Tagastab enumeraatorit, mis võimaldab itereerida üle videosegmentide loendi.
AudioEncoding	omadus	Videosegmentidele lisatava/lisatud audiovoo kodeerimisviis
Count	omadus	Videosegmentide arv loendis.
OriginalMedia	omadus	Meedia, millest antud videosegmente hakatakse lõikama/lõigati.
SegmentsDirectoryPath	omadus	Absoluutne tee kaustani, kuhu salvestatkse videosegmentide failisid.
this[int index]	omadus	Tagastab videosegmenti määratud positsioonis loendis (määratud indeksiga).
VideoEncoding	omadus	Videosegmentides sisalduva videovoo kodeerimisviis.

Lisa 2. Uute ja täiendatud klasside analüüs.

Antud töö osas analüüsitakse ainult töö eesmärkide realiseerimiseks vajalikku ja autori poolt kirjutatud koodi. Enumeraatorid, sündmuste argumendid ja klass Constants ei ole töö käigus analüüsitud.

Klasside analüüsil kasutatakse järgnevaid märkuseid:

- [1] – klass on autori poolt loodud;
- [2] – klass on autori poolt muudetud/täiendatud;
- [3] – meetod/omadus on autori poolt loodud;
- [4] – meetod/omadus on autori poolt muudetud/täiendatud;

Klasside analüüs on esitatud klasside tähestikulises järjekorras.

Klass CueTrackConverter[1]

Tabel 37. Klassi CueTrackConverter koodi meetrikad.

SLOC	72						
Testitavad koodi read	45						
Testimata koodi read	1						
Testide arv	11						
RK	97,7						
HK	87,5						
HI	64						
PS	1						
KOS	15						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
Convert(..)[3]	32	100	100	57	4	8	11
ConvertBack(...)[3]	17	92,9	66,7	63	3	2	9
keskmine	24,5	96,4	83,3	60	3,5	5	10

Tabel 38. Klassi CueTrackConverter analüüs.

Kategooria	Märkused	Järeldused
Testkate	• Meetodi ConvertBack(...) HK väärtus on liiga madal. • Meetod Convert(..) on piisavalt testitud.	Meetodi ConvertBack(...) jaoks on vaja lisada ühikteste.
Koodi keerukus	• Klassi KOS väärtus on liiga kõrge. • Meetod Convert(..) on liiga suur ning selle KOS väärtus on liiga kõrge. • Ülejääud koodi meetrikad on normi piires.	Klass vajab refaktoreerimist.
Puhas kood	• Mõne muutuja nimi ei ole täpne (nt cueTracksCollection on tegelikult IList<CueTrackViewModel> liides). • Meetod Convert(..) võtab liiga palju argumente (5). • Meetodite nimed ei ole täpsed ja arusaadavad, ei ole selge, mida ja milleks need teisendavad.	Klass vajab refaktoreerimist.

Klass CutMediaSegmentsParamsRecalculator[1]

Tabel 39. Klassi CutMediaSegmentsParamsRecalculator koodi meetrikad.

SLOC	52						
Testitavad koodi read	26						
Testimata koodi read	0						
Testide arv	5						
RK	100						
HK	100						
HI	77						
PS	1						
KOS	10						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
Create(..)[3]	4	100	100	77	2	2	4
Konstruktor[3]	1	100	100	87	1	0	2
Recalculate(...)[3]	20	100	100	71	7	4	7
minimaalne	1	100	100	71	1	0	2
maksimaalne	20	100	100	87	7	4	7
keskmine	8,3	100	100	78,3	3,3	2	4,3

Tabel 40. Klassi CutMediaSegmentsParamsRecalculator analüüs.

Kategooria	**Märkused**	**Järeldused**
Testkate	Klass on piisavalt testitud.	Ühikteste ei ole vaja lisada.
Koodi keerukus	• Klassi KOS väärtus on liiga kõrge. • Ülejäänud koodi meetrikad on normi piires.	Klass vajab refaktoreerimist.
Puhas kood	-	Klass vajab refaktoreerimist.

Klass EnumHelper[2]

Tabel 41. Klassi EnumHelper koodi meetrikad.

Testide arv	0						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
GetDescription(...)[3]	11			65	5		5

Tabel 42. Klassi EnumHelper analüüs.

Kategooria	**Märkused**	**Järeldused**
Testkate	Meetodi jaoks ei ole kirjutatud ühtegi testi. Ei õnnestunud saada informatsiooni JK ja HK kohta.	On vaja kirjutada ühikteste meetodi jaoks
Koodi keerukus	• Meetodi SLOC, HI, TK ja KOS on normi piires. • Ei õnnestunud saada informatsiooni NPK kohta.	Meetod ei vaja refaktoreerimist.
Puhas kood	-	Meetod ei vaja refaktoreerimist.

Klass FfMpegArgumentsConstructor[1]

Tabel 43. Klassi FfMpegArgumentsConstructor koodi meetrikad.

Testide arv	24						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
ConstructForCutting(...)[3]	32	100	100	51	5	16	9
ConstructForExtractingThumbnail(...)[3]	18	100	100	61	4	8	6
ConstructForMergingVideoSegments(...)[3]	41	100	100	57	5	8	12
GetAudioEncoderInCaseOfCuttingVideo-Segments(...)[3]	17	100	100	67	5	8	2
GetVideoBitStreamFilter(...)[3]	17	100	100	66	3	4	2
Konstruktor3	2	100	100	80	1	0	3
minimaalne	2	100	100	51	1	0	2
maksimaalne	41	100	100	80	5	16	12
keskmine	21,2	100	100	63,7	3,8	7,3	5,7

Tabel 44. Klassi FfMpegArgumentsConstructor analüüs.

Kategooria	Märkused	Järeldused
Testkate	Meetodid on piisavalt testitud.	Uusi ühikteste ei ole vaja lisada.
Koodi keerukus	• Meetod ConstructForCutting(...) on liiga suur. • Meetod ConstructForMergingVideoSegments(...) on liiga suur ning selle KOS väärtus on liiga kõrge. • Ülejäänud koodi meetrikad on normi piires.	Mainitud meetodid vajavad refaktoreeri-mist.
Puhas kood	• Meetod ConstructForCutting(...) võtab liiga palju argumente (4). • Meetod ConstructForExtractingThumbnail(...) võtab liiga palju argumente (3). • Meetodi GetAudioEncoderInCaseOfCuttingVideoSegments(...) nimi on liiga pikk ja keeruline. • Meetod GetAudioEncoderInCaseOfCuttingVideoSegments(...) sisaldab switch-konstruktsiooni. • Meetod GetVideoBitStreamFilter(...) sisaldab switch-konstruktsiooni. • Meetod ConstructForMergingVideoSegments(...) sisaldab väljakommenteeritud vana koodi.	Mainitud meetodid vajavad refaktoreeri-mist.

Klass FfMpegFilesCreationProgressCalculator[1]

Tabel 45. Klassi FfMpegFilesCreationProgressCalculator koodi meetrikad.

SLOC		132						
Testitavad koodi read		64						
Testimata koodi read		4						
Testide arv		7						
RK		93,7						
HK		100						
HI		76						
PS		1						
KOS		13						
Meetod/omadus		**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
AddFile(...)[3]		10	100	100	69	2	2	2
CompleteCalculation()[3]		7	100	100	71	1	0	2
CompleteFile()[3]		7	100	100	72	1	0	2
Create()[3]		1	0	0	89	1	0	1
CurrentProgressInPercent[3]	**get()**	1			98	1		0
	set(...)	3			79	2		1
IsCalculationInProgress[3]	**get()**	1			98	1		0
	set(...)	3			81	2		0
OnFileCurrentCreationPositionReceived(...)[3]		17	100	100	60	3	4	3
OnPropertyChanged(...)[3]		1	100	100	92	2	2	4
StartCalculation()[3]		4	100	100	81	1	0	1
StartFile()[3]		3	100	100	82	1	0	1
	minimaalne	1	0	0	60	1	0	0
	maksimaalne	17	100	100	98	3	4	4
	keskmine	4,8	91,7	91,7	81	1,5	1	1,4

Tabel 46. Klassi FfMpegFilesCreationProgressCalculator analüüs.

Kategooria	Märkused	Järeldused
Testkate	• Meetodi Create() nii JK kui HK väärtus on 0. • Omadused ei ole testitud. • Ülejäänud meetodid on piisavalt testitud.	On vaja kirjutada lisaühikteste meetodi Create() ja omaduste jaoks.
Koodi keerukus	• Klassi KOS väärtus on liiga kõrge. • Ülejäänud koodi meetrikad on normi piires.	Klass vajab refaktoreerimist.
Puhas kood	• Mõne lokaalse muutuja nimi on liiga pikk (_allFilesTotalDurationInSeconds). • Kuna klass omab static factory meetodit klassi eksemplaari loomiseks, konstruktorit peab muutma privaatseks.	Klass vajab refaktoreerimist.

Klass FFMpegMergingInputConstructor[1].

Tabel 47. Klassi FFMpegMergingInputConstructor koodi meetrikad.

SLOC	29						
Testitavad koodi read	17						
Testimata koodi read	0						
Testide arv	3						
RK	100						
HK	100						
HI	68						
PS	1						
KOS	6						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
Construct(...)[3]	15	100	100	60	6	16	5

Tabel 48. Klassi FFMpegMergingInputConstructor analüüs.

Kategooria	**Märkused**	**Järeldused**
Testkate	Klass on piisavalt testitud.	Ühkteste ei ole vaja lisada.
Koodi keerukus	Kõik koodi meetrikad on normi piires.	Klass ei vaja refaktoreerimist.
Puhas kood	-	Klass ei vaja refaktoreerimist.

Klass FFProbeXmlParser2

Tabel 49. Klassi FFProbeXmlParser koodi meetrikad.

Testide arv	2						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
ParseMetainfo(...)[4]	72	95,4	76	39	19	4096	20

Tabel 50. Klassi FFProbeXmlParser analüüs.

Kategooria	**Märkused**	**Järeldused**
Testkate	Klass on piisavalt testitud.	Ühkteste ei ole vaja lisada.
Koodi keerukus	Meetod ParseMetainfo(...) on liiga suur ning selle TK, NPK ja KOS väärtused on liiga kõrged.	Meetod vajab refaktoreerimist.
Puhas kood	Meetodis ParseMetainfo(...) on kaks lokaalset muutujat: format ja containerFormat. Ei ole arusaadav, mille poolest nad erinevad.	Meetod vajab refaktoreerimist.

Klass Media[2]

Tabel 51. Klassi Media koodi meetrikad.

Testide arv	0						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
Create()[1]	8	100	100	71	2	2	3
Modify()[1]	2	100	100	98	1	0	1
Save()[1]	8	100	100	79	2	2	1
minimaalne	2	100	100	71	1	0	1
maksimaalne	8	100	100	98	2	2	3
keskmine	6	100	100	82,7	1,7	1,3	1,7

Tabel 52. Klassi Media analüüs.

Kategooria	**Märkused**	**Järeldused**
Testkate	Meetodite Create(), Modify() ja Save() jaoks ei ole kirjutatud ühtegi ühiktesti, vaatamata sellele nende JK ja HK väärtused on normi piires.	Meetodite jaoks on vaja kirjutada ühikteste.
Koodi keerukus	Kõik klassi koodi meetrikad on normi piires.	Meetod ei vaja refaktoreerimist.
Puhas kood	-	Meetod ei vaja refaktoreerimist.

Klass MediaEditorViewModel[2]

Töö käigus autor jagas avalikku meetodi LoadMedia(...) kaheks: mõlema uue meetodi nimeks on LoadMedia(...), üks nendest on privaatne.

Tabel 53. Klassi MediaEditorViewModel koodi meetrikad.

Testide arv	**0**							
Meetod/omadus		**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
ActivityName[3]	**get()**	1			98	1		0
	set(...)	3			80	2		1
ActivityProgressInPercent[3]	**get()**	1			98	1		0

Meetod/omadus		SLOC	JK	HK	HI	TK	NPK	KOS
ActivityProgressInPercent[3]	set(...)	3			79	2		2
CancelMerging()[3]		1	0	0	98	1	0	1
CancelMergingCommand[3]	get()	1			98	1		1
CanMergeVideoSegments()[3]		1	0	0	83	2	0	2
Close()[3]		7	0	0	67	1	0	10
CueTags[4]	get()	1			98	1		2
IsActivityInProgress[3]	get()	1			98	1		0
	set(...)	3			81	2		1
IsLoadingCueTags[3]	get()	1			98	1		0
	set(...)	3			81	2		1
IsRemainingTimeShown[3]	get()	1			98	1		0
	set(...)	3			81	2		1
Konstruktor[4]		62	0	0	32	12	0	32
LoadMedia(...)[3]		8	0	0	71	4	0	6
LoadMedia(...)[3]		44	0	0	71	26	256	5
LoadTags(...)[3]		7	0	0	71	4	2	3
MergeVideoSegments()[3]		39	0	0	75	5	4	4
MergeVideoSegmentsCommand[3]	get()	1			98	1		1
OnIsMergingProgressCalculation-InProcessChanged(...)[3]		2	0	0	84	2	2	1
OnMediaAdded(...)[3]		1	0	0	95	2	2	2
OnMergedMediaAdded(...)[3]		5	0	0	75	2	2	2
OnMergingProgressChanged(...)[3]		1	0	0	94	1	0	1
OnVideoSegmentsParamsRecalculated(...)[3]		6	0	0	71	2	2	3
SaveCueTags(...)[3]		1	0	0	95	1	0	2
SaveVideoSegmentsToCueSheet(...)[3]		6	0	0	79	2	2	4
SelectSegmentsAudioEncoding(...)[3]		12	0	0	70	4	8	4
SortCueTagsByStartPosition(...)[3]		1	0	0	76	2	0	5
	minimaalne	1	0	0	32	1	0	0
	maksimaalne	62	0	0	98	26	256	32
	keskmine	8,2	0	0	82,7	3,2	19,7	3,3

Tabel 54. Klassi MediaEditorViewModel analüüs.

Kategooria	Märkused	Järeldused
Testkate	Klassi jaoks ei ole kirjutatud teste, seega meetodite nii JK kui ka HK on 0.	Klassi on vaja testida.
Koodi keerukus	• Konstruktor on liiga suur, selle TK ja KOS väärtused on liiga kõrged. • Meetod MergeVideoSegments() on liiga suur. • Privaatne meetod LoadMedia(...) on liiga suur, selle TK ja NPK väärtused on liiga kõrged. • Ülejäänud koodi meetrikad on normi piires.	Klassi meetodid ja konstruktor vajavad refaktoreerimist.
Puhas kood	• Konstruktor võtab liiga palju argumente (11). • Avalik meetod LoadMedia(...) võtab liiga palju argumente (3). • Meetodis SelectSegmentsAudioEncoding(...) on kasutatud switch-konstruktsioon.	Klassi meetodid ja konstruktor vajavad refaktoreerimist.

Klass MediaFileMetainfoService[2]

Tabel 55. Klassi MediaFileMetainfoService koodi meetrikad.

Testide arv	1						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
GetMediaFileDurationAsync(...)[3]	28	66,7	60	71	5	4	6

Tabel 56. Klassi MediaFileMetainfoService analüüs.

Kategooria	Märkused	Järeldused
Testkate	Meetodi GetMediaFileDurationAsync(...) JK ja HK väärtused on liiga madalad.	Mainitud meetodi jaoks on vaja lisada ühikteste.
Koodi keerukus	Kõik meetodi koodi meetrikad on normi piires.	Meetodit ei ole vaja refaktoreerida.
Puhas kood	-	Meetod ei vaja refaktoreerimist.

Klass MediaManager[2]

Tabel 57. Klassi MediaManager koodi meetrikad.

Testide arv	1						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
GetMediaCopy(...)[3]	12	100	100	78	1	0	2

Tabel 58. Klassi MediaManager analüüs.

Kategooria	**Märkused**	**Järeldused**
Testkate	Meetod on piisavalt testitud.	Ühikteste ei ole vaja lisada.
Koodi keerukus	Meetodi kõik koodi meetrikad on normi piires.	Meetod ei vaja refaktoreerimist.
Puhas kood	Meetod võtab liiga palju argumente (3).	Meetod vajab refaktoreerimist.

Klass MediaMetainfo[2]

Tabel 59. Klassi MediaMetainfo koodi meetrikad.

Testide arv	0							
Meetod/omadus		**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
ContainerFormat[3]	**get()**	1			98	1		1
Konstruktor[4]		6	100	100	68	1	0	2
VideoEncoding[3]	**get()**	1			98	1		1
	minimaalne	1	100	100	68	1	0	1
	maksimaalne	6	100	100	98	1	0	2
	keskmine	2,7	100	100	88	1	0	1,3

Tabel 60. Klassi MediaMetainfo analüüs.

Kategooria	Märkused	Järeldused
Testkate	Klass ei ole testitud.	On vaja kirjutada ühikteste konstruktori ja omaduste jaoks.
Koodi keerukus	Kõik klassi koodi meetrikad on normi piires.	Konstruktor ja omadused ei vaja refaktoreerimist.
Puhas kood	Konstruktor võtab liiga palju argumente (6).	Konstruktor vajab refaktoreerimist.

Klass OpenMediaItemCommand[2]

Tabel 61. Klassi OpenMediaItemCommand koodi meetrikad.

Testide arv	0						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
Execute(...)[4]	155	0	0	31	27	2097152	54
GetMediaEditorViewModel(...)[3]	32	0	0	53	2	2	38
Konstruktor[4]	51	0	0	41	23	16	31
OnProgramRunnerMatchGot(...)[3]	1	0	0	92	1	0	2
minimaalne	1	0	0	31	1	0	2
maksimaalne	155	0	0	92	27	2097152	54
keskmine	59,8	0	0	54,3	13,3	524292,5	31,3

Tabel 62. Klassi OpenMediaItemCommand analüüs.

Kategooria	Märkused	Järeldused
Testkate	Klassi jaoks ei ole kirjutatud ühik- ega muid tüüpi teste, seega meetodite nii JK kui ka HK on 0.	Meetodeid on vaja testida.
Koodi keerukus	• Konstruktor on liiga suur, selle TK ja KOS väärtused on liiga kõrged. • Meetod Execute(...) on liiga suur, selle TK, NPK ja KOS väärtused on liiga kõrged. • Meetod GetMediaEditorViewModel(...) on liiga suur ning selle KOS väärtus on liiga kõrge. • Ülejäänud koodi meetrikad on normi piires.	Klassi meetodid ja konstruktor vajavad refaktoreerimist.
Puhas kood	Konstruktor võtab liiga palju argumente (31).	Konstruktor vajab refaktoreerimist.

Klass ProgramRunner[2]

Tabel 63. Klassi ProgrammRunner koodi meetrikad.

Testide arv	0						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
GetExitCodeAsync(...)[3]	86	87,2	68,4	65	19	512	7
GetLineBeginningWithFrom-StandardErrorAsync(...)[3]	35	100	100	67	6	2	6
GetStandardOutputAsync(...)[4]	56	81,8	64,3	67	18	80	6
OnMatchGot(...)[3]	1	100	66,7	95	2	2	2
minimaalne	1	81,8	64,3	65	2	2	2
maksimaalne	86	100	68,4	95	19	512	7
keskmine	47,7	89,7	66,5	75,7	13	198	5

Tabel 64. Klassi ProgrammRunner analüüs.

Kategooria	**Märkused**	**Järeldused**
Testkate	Klassi jaoks ei ole kirjutatud ühik- ega muud tüüpi teste. Vaatamata sellele, meetodite JK väärtused on piisavalt kõrged. Kolme meetodi HK on alla 70%.	Kõigi meetodite jaoks on vaja kirjutada ühikteste.
Koodi keerukus	• Meetod GetStandardOutputAsync(...) on liiga suur ning selle TK väärtus on liiga kõrge. • Meetod GetExitCodeAsync(...) on liiga suur ning selle TK ja NPK väärtused on liiga kõrged. • Meetod GetLineBeginningWithFromStandardErrorAsync(...) on liiga suur. • Ülejäänud koodi meetrikad on normi piires.	Mainitud meetodid vajavad refaktoree-rimist.
Puhas kood	• Meetod GetStandardOutputAsync(...) võtab liiga palju argumente (4). • Meetod GetExitCodeAsync(...) võtab liiga palju argumente (6). • Meetod GetLineBeginningWithFromStandardErrorAsync(...) võtab liiga palju argumente (4). • Suur osa koodist meetodites GetExitCodeAsync(...), GetStandardOutputAsync(...) ja GetLineBeginningWithFromStandardErrorAsync(...) kordub. • Meetodi GetLineBeginningWithFromStandardErrorAsync(...) nimi on liiga keeruline.	Mainitud meetodid vajavad refaktoree-rimist.

Klass ProgressReportControl[1]

Tabel 65. Klassi ProgressReportControl koodi meetrikad.

Testide arv	**0**							
Meetod/omadus		**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
ActivityName[3]	**get()**	1			94	1		2
	set(...)	1			95	1		2
CancelCommand[3]	**get()**	1			94	1		2
	set(...)	1			95	1		3
IsRemainingTimeShown[3]	**get()**	1			94	1		2
	set(...)	1			95	1		2
konstruktor[3]		8	0	0	64	1	0	7
Percentage[3]	**get()**	1			94	1		2
	set(...)	1			95	1		2
	minimaalne	1	0	0	64	1	0	2
	maksimaalne	8	0	0	95	1	0	7
	keskmine	1,8	0	0	91,1	1	0	2,7

Tabel 66. Klassi ProgressReportControl analüüs.

Kategooria	**Märkused**	**Järeldused**
Testkate	Klass kujutab endast kasutajaliidese elementi, seetõttu selle jaoks ei ole kirjutatud ühtegi testi ning konstruktori ja omaduste HK ja JK on 0.	Lisateste ei ole vaja kirjutada.
Koodi keerukus	Kõik koodi meetrikad on normi piires.	Konstruktor ja omadused ei vaja refaktoreerimist.
Puhas kood	-	Konstruktor ja omadused ei vaja refaktoreerimist.

Klass SortedObservableCollection<T>[1]

Tabel 67. Klassi SortedObservableCollection<T> koodi meetrikad.

SLOC	31						
Testitavad koodi read	14						
Testimata koodi read	14						
Testide arv	0						
RK	0						
HK	0						
HI	82						
PS	3						
KOS	10						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
konstruktor[3]	1	0	0	86	1	0	9
Sort(...)[3]	5	0	0	73	3	0	8
UpdateOrder()[3]	1	0	0	98	1	0	1
minimaalne	1	0	0	73	1	0	1
maksimaalne	5	0	0	98	3	0	9
keskmine	2,3	0	0	85,7	1,7	0	6

Tabel 68. Klassi SortedObservableCollection<T> analüüs.

Kategooria	**Märkused**	**Järeldused**
Testkate	Klass kuulub kasutajaliidese koodi, seetõttu seda ei ole testitud. Siit tulenevalt on RK, HK ja JK väärtused 0.	Ühkteste lisada ei ole vaja.
Koodi keerukus	Kõik koodi meetrikad on normi piires.	Klass ei vaja refaktoreerimist.
Puhas kood	-	Klass ei vaja refaktoreerimist.

Klass SystemManager[2]

Tabel 69. Klassi SystemManager koodi meetrikad.

Testid	2						
Meetod/omadus	**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
CreateNewFolder(...)[3]	3	100	100	83	1	0	2
DeleteFolder(...)[3]	4	100	100	77	2	2	2
keskmine	3,5	100	100	80	1,5	1	2

Tabel 70. Klassi SystemManager analüüs.

Kategooria	**Märkused**	**Järeldused**
Testkate	Nende meetodite jaoks on kirjutatud kaks testi, kusjuures mõlemad testivad meetodit DeleteFolder(...).	On vaja kirjutada ühikteste, mis testiks meetodit CreateNewFolder(...).
Koodi keerukus	Meetodite kõik koodi meetrikad on normi piires.	Meetodid ei vaja refaktoreerimist.
Puhas kood	Üks DeleteFolder(...) argumentidest on boolean tüüpi.	Mainitud meetod vajab refaktoreerimist.

Klass VideoSegment[1]

Tabel 71. Klassi VideoSegment koodi meetrikad.

SLOC	15							
Testitavad koodi read	0							
Testimata koodi read	0							
Testide arv	0							
RK								
HK								
HI	93							
PS	1							
KOS	3							
Meetod/omadus		**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
AudioEncoding[3]	**get()**	1			98	1		1
	set(...)	1			95	1		1
Duration[3]	**get()**	1			98	1		0
	set(...)	1			95	1		0
FileName[3]	**get()**	1			98	1		0
	set(...)	1			95	1		0
StartPosition[3]	**get()**	1			98	1		0
	set(...)	1			95	1		0
TagTitle[3]	**get()**	1			98	1		0
	set(...)	1			95	1		0
VideoEncoding[3]	**get()**	1			98	1		1
	set(...)	1			95	1		1
	minimaalne	1	0	0	95	1	0	0
	maksimaalne	1	0	0	98	1	0	1
	keskmine	1			96,5	1		0,3

Tabel 72. Klassi VideoSegment meetrikad.

Kategooria	Märkused	Järeldused
Testkate	Klass ei ole testitud.	Klassi on vaja testida.
Koodi keerukus	Kõik klassi koodi meetrikad on normi piires	Klassi ei ole vaja refaktoreerida.
Puhas kood	-	Klassi ei ole vaja refaktoreerida.

Klass VideoSegments[1]

Tabel 73. Klassi VideoSegments koodi meetrikad.

SLOC	15							
Testitavad koodi read	0							
Testimata koodi read	0							
Testide arv	0							
RK								
HK								
HI	93							
PS	2							
KOS	7							
Meetod/omadus		**SLOC**	**JK**	**HK**	**HI**	**TK**	**NPK**	**KOS**
AudioEncoding[3]	**get()**	1			98	1		1
	set(...)	1			95	1		1
OriginalMedia[3]	**get()**	1			98	1		1
	set(...)	1			95	1		1
SegmentsDirectoryPath[3]	**get()**	1			98	1		0
	set(...)	1			95	1		0
VideoEncoding[3]	**get()**	1			98	1		1
	set(...)	1			95	1		1
	minimaalne	1	0	0	95	1	0	0
	maksimaalne	1	0	0	98	1	0	1
	keskmine	1			96,5	1		0,8

Tabel 74. Klassi VideoSegments analüüs.

Kategooria	**Märkused**	**Järeldused**
Testkate	Klass ei ole testitud.	Klassi on vaja testida.
Koodi keerukus	Kõik klassi koodi meetrikad on normi piires	Klassi ei ole vaja refaktoreerida.
Puhas kood	-	Klassi ei ole vaja refaktoreerida.

Lisa 3. Ettevõttepoolne tagasiside.