Projektdokumentation

Mission-Uncuttable

Sascha Rittig (Product-Owner) – sascha.rittig@stud.htwk-leipzig.de
Tim Jeske (Software-Architekt) – tim.jeske@stud.htwk-leipzig.de
Alexander Bonin – alexander.bonin@stud.htwk-leipzig.de
Clemens Zwinzscher – clemens.zwinzscher@stud.htwk-leipzig.de
Felix Dittrich – felix.dittrich@stud.htwk-leipzig.de
Markus Leupold – markus.leupold@stud.htwk-leipzig.de
Maximilian Fornaçon – maximilian.fornacon@stud.htwk-leipzig.de
Jeremy Risy – jeremy.risy@stud.htwk-leipzig.de
Johannes Müller – johannes.mueller@stud.htwk-leipzig.de
Julian Oliver Böttcher – julian_oliver.boettcher@stud.htwk-leipzig.de
Valentin Ackva – valentin.ackva@stud.htwk-leipzig.de
HTWK Leipzig

WS18/19 - SS19

Inhaltsverzeichnis

I. Anforderungsspezifikation

Autoren: Sascha Rittig

In Lehrveranstaltungen wie "Algorithmen und Datenstrukturen" werden per UbiCast Vorlesungsmitschnitte angefertigt. Diese sind nur bedingt für einen attraktiven Einsatz als Lehrvideo geeignet. Daher wird eine Software benötigt, die den einfachen und gut unterstützten Schnitt von ganzen Vorlesungen und kleineren thematischen Sequenzen realisiert.

I.1 Produkteinsatz

Autoren: Sascha Rittig

Videoschnitt für die Aufzeichnungen der Lehrveranstaltungen "Algorithmen und Datenstrukturen" und "Softwaretechnik". Im Zielformat 4:3.

I.2 Initiale Kundenvorgaben

Autoren: Sascha Rittig

- ein odere mehrere Videomitschnitte müssen eingelesen werden
- Modus Tafel, Folie, Visualizer, Demo am Rechner sowie ein Folienwechsel sind automatisch zu identifizieren
- Wechsel im Schnitt zwischen Tafel, Sprecher, Folie/Visualizer
- Zoom auf den Sprecher mit Tracking
- Einblendung des Sprechers, wenn weiße Fläche zur Verfügung steht
- automatisierter Vorschlag der Software zum Schnitt
- der Schnitt muss vollständig bearbeitbar sein
- Audio-Korrekturen müssen nachträglich einsprechbar sein
- Teile müssen entfernbar sein
- zusätzliche Sequenzen können eingefügt werden
- das Bild unter der Audio-Aufzeichnung muss durch ein späteres Standbild oder ein externes Bild ersetzbar sein
- Gliederung und aktueller Gliederungspunkt am unteren Rand transparent einblendbar
- Auch Zwischentafeln sollten einblendbar sein.

I.3 Produktvision

Autor: Sascha Rittig

Im Rahmen des Projekts sollte auch eine Produktvision erstellt werden. Eine Produktvision dient insbesondere der Orientierung des Teams und der gemeinsamen Ausrichtung auf ein Ziel. Die Vision soll beschreiben warum das Projekt überhaupt umgesetzt wird und was der gewünschte Zustand ist, der erreicht werden soll. Hierzu orientiert man sich an den folgenden Fragen: Wer ist unsere Zielgruppe? Welche Kundenbedürfnisse sollen mit dem Produkt befriedigt werden? Was unterscheidet unser Produkt von den Produkten der Konkurrenz? Was sind wesentliche Eigenschaften, die sich unterscheiden und die Kunden vom Kauf überzeugen können? Was ist es für ein Produkt und wie heißt das Produkt?

Um diese und weitere Fragen zu klären, hat das Team recherchiert und am 8.11.2018 in Form einer Folien-Präsentation die Ergebnisse präsentiert. (Die Folien sind im Git Doku Ordner zu finden!) Im folgenden sind die Ergebnisse unserer Produktvision zusammengefasst.

I.4 Liste der funktionalen Anforderungen

Autor: Valentin Ackva

Basierend auf der Vorgabe des Kunden wurde folgende Anforderungen spezifiziert. Diese sind unterteilt in funktionale und nicht funktionale Anforderungen.

- Programm soll schnell und zuverlässig laufen
- Bearbeitung und Zusammenstellung eines Vorlesungsvideos soll nicht zu lange dauern
- Programm soll so einfach wie möglich zu bedienen sein

I.5 Liste der nicht-funktionalen Anforderungen

Autor: Valentin Ackva

- Schlichtes Videobearbeitungsprogramm, dass mehrere Video-, Bild- und Audiodateien schnell und einfach mit möglichst wenig Aufwand in ein fertiges und qualitativ hochwertiges Vorlesungsvideo bearbeitet werden kann
- Erweiterbarkeit des Vorlesungsvideos mit Bildern, Videos und Audiokommentaren
- Automatisierung einiger nötiger Bearbeitungen um den Benutzer der Software zu entlasten

I.6 Weitere Zuarbeiten zum Produktvisions-Workshop

I.6.1 Graphical user interface

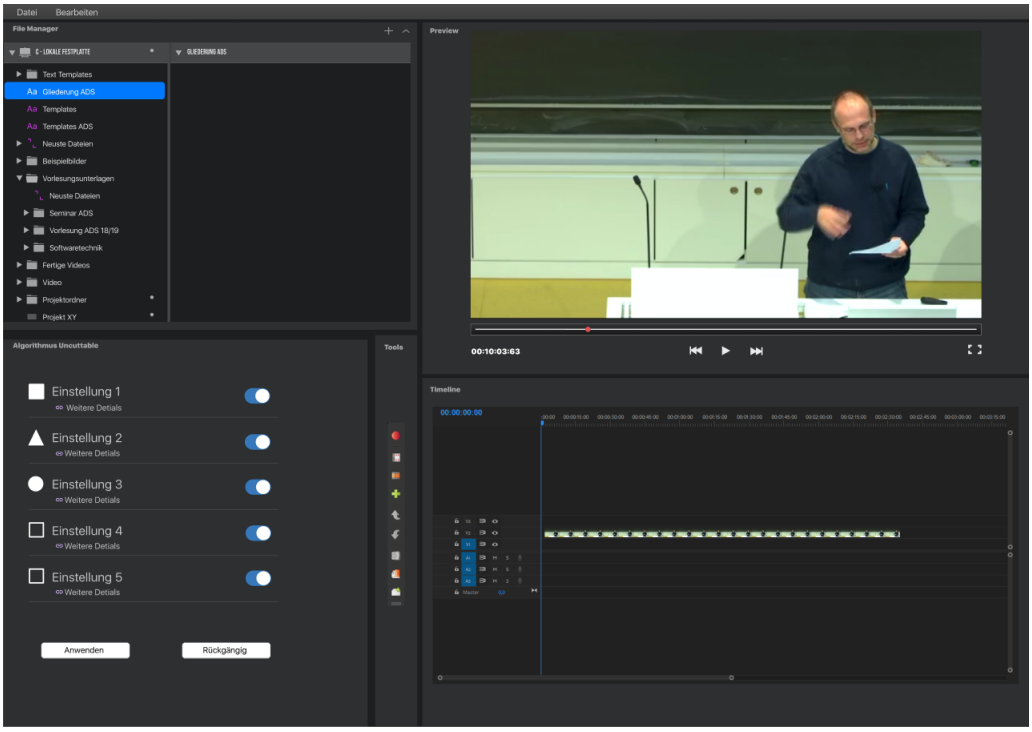
I.6.1.1 GUI-Varianten

Auf der Grundlage dieser Anforderungen wurden folgende drei GUI-Varianten entworfen:
Variante 1

Autor: Julian Oliver Böttcher

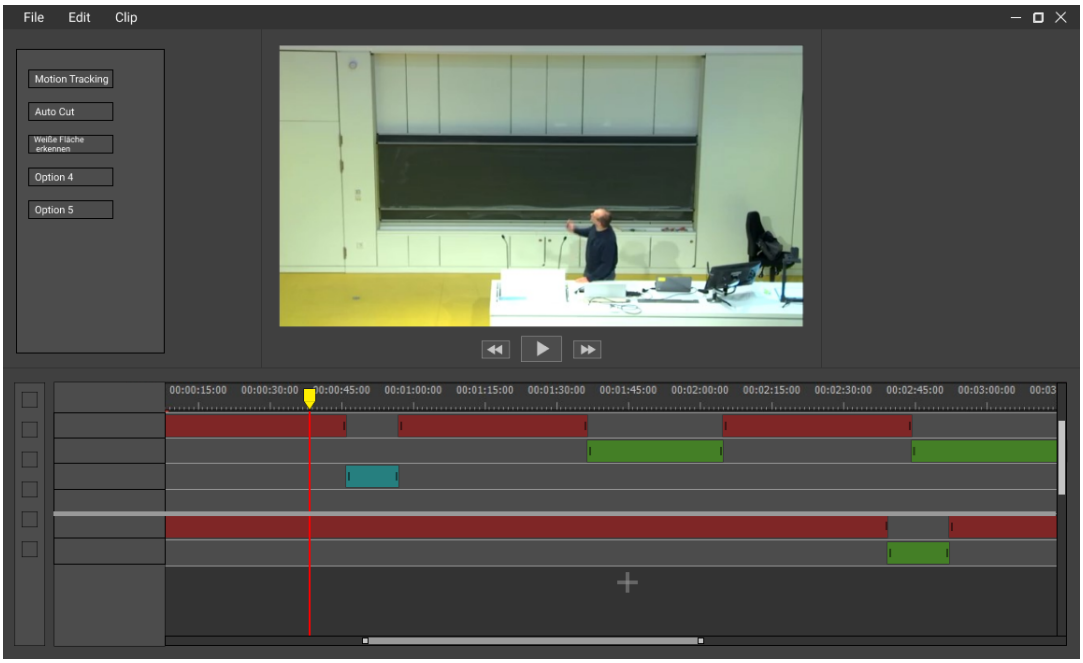
Variante 2

Autor: Valentin Ackva



Variante 3

Autor: Alexander Bonin



I.6.2 Zielgruppe

Autor: Felix Dittrich, Maximilian Fornaçon

I.6.2.1 Übersicht

Als Zielgruppe für unser Projekt wurden Dozenten aus den Bereichen:

- Universitäten, Hochschulen
- e-Learning

festgelegt.

Poweruser Allgemein
Der Poweruser wird so definiert, dass bei ihm die Funktonalität des Programmes bzw. der Anwendung im Vordergrund liegt.
Er möchte ein umfangreiches Programm wobei alles abgedeckt sein soll was er damit umsetzen möchte.
Dafür ist er auch bereit sich in ein Programm langfristig einzuarbeiten und damit auseinanderzusetzen um das gewünschte Ziel zu erreichen.

DAU - Dümmster Anzunehmender User Allgemein
Im Gegensatz zu dem Poweruser steht bei dem DAU die einfache Bedienung und schnelle Erlernbarkeit der Funktionalitäten des Programms bzw. der Anwendung im Vordergrund.
Der DAU möchte ein fehlertolerantes Programm welches nicht nach jedem Fehlschritt gleich bestraft dafür gibt er sich auch mit den Grundfunktionalitäten zufrieden.

I.6.2.2 Poweruser

spezielle Eigenschaften für unser Projekt:

- Dozent mit Erfahrung im Umgang mit Videoschnittsoftware
- technikaffin
- hat den Willen und die Zeit sich mit der Funktionalität der Software auseinanderzusetzen
- ist bereit mehr Zeit für die Bearbeitung aufzubringen

Anforderung an die Software Der Poweruser möchte die Anwendung bestmöglich an seine eigene Arbeitsweise anpassen um so schneller und effizienter ans Ziel zu kommen.
Dabei legt er allerdings genauso viel Wert auf die Übersichtlichkeit des Programmes, da diese einen Großteil zum schnellen und effizienten Arbeiten beiträgt.
Auf eine Backupfunktion möchte er nicht verzichten, da Fehler auch einem Profi passieren können und es unschön und ärgerlich für ihn wäre wenn er dann von vorne beginnen müsste.

I.6.2.3 DAU

spezielle Eigenschaften für unser Projekt:

- Dozent hat wenig Erfahrung mit Videoschnittsoftware
- Dozent hat wenig Zeit für Einarbeitung und den Schnitt des Videos
- wenig technisches Verständnis

Anforderung an die Software Der DAU möchte im Gegenteil zum Poweruser so wenig wie möglich selbst machen bzw. vorab einstellen müssen um zu seinem Ziel zu kommen.
Er legt viel Wert auf Hilfestellungen wie Tooltips oder zu Beginn ein Einführungsvideo in die Software oder ähnliches.

Genau deshalb ist es ihm auch wichtig eine saubere, moderne und einfach zu bedienende GUI vorzufinden.
Eine Backupfunktion ist für ihn unerlässlich, da gerade Einsteigern doch öfters Fehler passieren und falls diese dann nicht rückgänig zu machen sind würde er sich höchstwahrscheinlich eine andere Software suchen.

I.6.3 Erweiterungspotenzial

Autoren: Clemens Zwinzscher, Markus Leupold

Erkennung

- automatische Erkennung der Gliederung aus Beamer-Präsentation
- Handschrifterkennung (Tafel und Visualizer)
- Spracherkennung

Suchfunktion

- einzelne Videos nach Begriffen durchsuchen (Voraussetzung Sprach- und Texterkennung)

Zensierung

- Unkenntlichmachung von Objekten im Video
- Verpixelung oder Unschärfe
- automatisches Tracking des Objekts

anpassbare GUI

- das Farbdesign sollte anpassbar sein
- die Anordnung der Teilfenster sollte geändert werden können
- Schriftart und Schriftgröße auch änderbar

Speicherung

- Speicherung des aktuellen Bearbeitungsstandes
- Pausierung der Bearbeitung

I.7 Liste der Kundengespräche mit Ergebnissen

Autor: Sascha Rittig

1.11.18 Der Kunde möchte ein möglichst einfaches Schnitt Programm, mit dem er schnell kleine Themen Videos zusammen schneiden kann. Es soll bequem zu bedienen sein. Man muss es schnell benutzen können, da er ansonsten auch andere Schnittprogramm nutzen könnte.
Der Begriff Zwischentafeln meint einfach externe Bilder die man im Videoschnitt einfügen können sollte. Bildererkennung soll das letzte sein mit dem sich im Projekt beschäftigt werden sollte. Als letztes wurde unser GUI-Entwurf vom Kunden als Gut befunden.

27.11.18 Email Austausch über ein Zweites Kundengespräch und klären der Frage ob das Buildsystem FBS in unserem Projekt genutzt werden kann. Ergebniss war ein Termin für das Kundegespräch und die Erlaubnis für die Nutzung von FBS.

Im Januar bis Juni gab es immer wieder kurze Treffen nach Lehrveranstaltungen mit dem entsprechenden Kunden, um ihn immer auf den neuesten Stand des Projekts zu bringen und abzugleichen ob sich die Software in die richtige Richtung entwickelt.

17.5.19 Grund des Treffens war es zusätzliche Lehr-Videos zu erhalten. Dies war wegen technischen Schwierigkeiten nicht möglich. Dafür haben wir Zugang zur ADS-Gruppe im Opal erhalten, welcher uns zwar den Stream der Videos ermöglicht aber nicht den Download. Der Kunde hat außerdem gesagt, dass es ok ist wenn die Folien im Video nicht mit den PDF-Folien verglichen und hochauflösenderen Bildern ersetzt werden.

27.5.19 Es wurden Ideen zum Automatischen-Schneiden abgeglichen. Ergebnis des Gesprächs war außerdem, dass es reicht wenn wir die Software für Linux entwickeln. Wir haben den Kunden um zusätzliche Lehr-Videos gebeten und wir sind uns einig gewesen, dass eine zusätzliche Zoomfunktion für die Timeline sehr wichtig ist.

24.6.19 Wir haben dem Kunden grundlegende Funktionen vorgeführt. Diese wurden vom Kunden positiv wahrgenommen und das Overlay Fenster wurde abgenickt. Der Kunde hat um die Möglichkeit gebeten mehrere Vorlesungen im Programm gleichzeitig/nacheinander in einem Projekt einfügen zu können. Außerdem hat der Kunde Verständnis für die langen Rechenzeiten im Autocut gezeigt und diese ebenfalls abgenickt.

II. Architektur und Entwurf

II.1 Zuarbeiten der Teammitglieder

II.1.1 GUI Frameworks

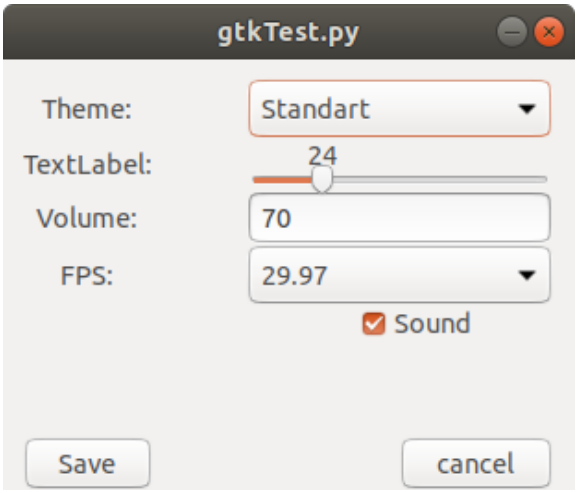
Im folgenden werden verschiedene GUI-Frameworks vorgestellt, die in der Technologierecherche behandelt wurden.

II.1.1.1 pyGTK mit Glade

Autor: Alexander Bonin

- Plattformunabhängigkeit
  - Installation unter Windows gestaltet sich extrem kompliziert. Es ist jedoch unter Windows, MacOs und Linux lauffähig.
- Videodarstellung
  - Nicht ohne weiteres möglich.
- Qualität der Dokumentation
  - Es gibt leider keine besonders ausfürhliche Dokumentation.
- Schwierigkeitsgrad der Benutzung
  - Benutzung ist theoretisch einfach da der WYSIWYG Editor "Glade"keine Programmierkenntnisse erfordert. Jedoch ist selbst die bedienung dieses Editors kompliziert und begrenzt die möglichkeiten stark.

- Lernbarkeit
  - Die Lernbarkeit ist ziemlich einfach, da es sich um einen WYSIWYG Editor handelt.
- Maintenance
  - PyGObject wird aktiv weiterentwickelt. Das neueste Release ist vom 17.11.2018
- Lizenz (academic/commercial use)
  - GNU Lesser General Public License
- Anpassbarkeit (eigene Widgets)
  - Die Anpassbarkeit ist hoch, denn die Objekte lassen sich z.B mit CSS in ihrem Desin anpassen.
- Python3 Unterstuetzung
  - Der Nachfolger von pyGTK namens PyGObject unterstützt auch Python 3.
- Stärken
  - theoretisch schnelle Bearbeitung durch WYSIWYG Editor
- Schwächen
  - Glade ist extrem umständlich zu bedienen und bietet keinen wirklichen Vorteil im gegensatzt zu manuellen Methoden.
  - Glade schränkt die Gestaltungsmöglichkeiten stark ein



II.1.1.2 Tkinter

Autor: Maximilian Fornaçon

Überblick: Tkinter ist seit vielen Jahren das standard GUI-Toolkit für Python. Tkinter steht für Tk-Interface, denn es baut auf Tcl/Tk auf.

Plattformunabhängigkeit: Tkinter ist vollständig Plattformunabhängig. ES wird bei der Installation von Python unter Windows und MacOS standardmäßig mit installiert. Unter Linux erforderd es eine separate Installation.

Videodarstellung: Tkinter bringt keine Möglichkeit mit Videos abzuspielen. Hier müsste man auf externe Bibliotheken zurückgreifen.

Qualität der Dokumentation: Es gibt mehrere Anlaufstellen für Dokumentation, jedoch sind einige recht unvollständig. Aufgrund der langjährigen Existenz und Benutzung von Tkinter gibt es viel Material in Foren und Videoplattformen.

Lernbarkeit und Schwierigkeitsgrad der Benutzung: Durch besagte Foren und Videotutorials fällt es leicht sich in die Benutzung von Tkinter einzuarbeiten. Layoutertellung und und das Platzieren der Widgets ist sehr intuitiv. Wenn es jedoch um Styling geht, hat Tkinter große schwächen.

Lizenz: Tkinter ist unter der BSD-Lizenz veröffentlicht, d.h. es darf frei verwendet, kopiert, verändert oder verbreitet werden. Lediglich bestehende Copyright-Vermerke dürfen nicht entfernt werden.

GUI-Beispiel: Grundsätzlich war es möglich das vorgegebene Layout und Design in beiden Farbvariationen mit Tkinter nachzubauen. Die Platzierung der Widgets war relativ einfach und die Farben zu verändern war auch möglich, jedoch erwies sich das als nicht sehr elegant.

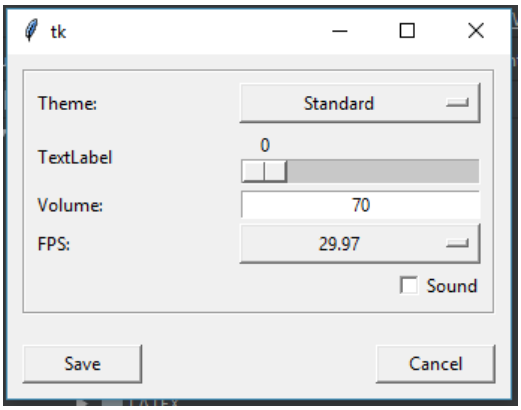


Abbildung 1: Tkinter hell

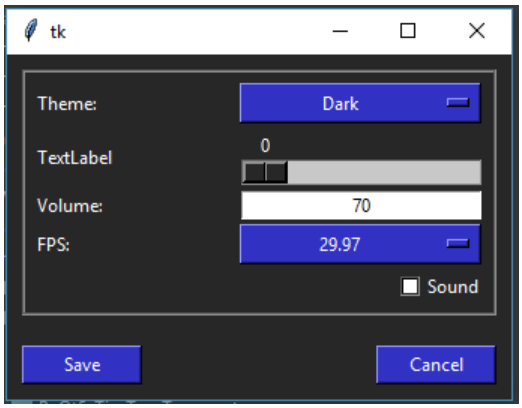


Abbildung 2: Tkinter dunkel

Stärken:

- Ist bei der Pythoninstallation unter Windows und MacOS standardmäßig dabei
- Plattformunabhängig
- schnell
- viele Onlineressourcen
- Layout- und Widgetplatzierung ist simpel

Schwächen:

- optisch sehr altbacken
- schlecht customizable
- kein Videoplayback

Fazit: Aufgrund von Lizenz, vielen Onlineressourcen und der Tatsache, dass Tkinter Plattformunabhängig läuft ist es ein Kandidat für unser Projekt, jedoch ist die veraltete Optik und das komplizierte Handling von Styles eher negativ zu bewerten.

II.1.1.3 PyGUI

Autor: Markus Leupold

Überblick: PyGUI ist ein Framework für grafische Nutzeroberflächen, das Plattformübergreifend funktionieren soll (Mac, Linux und Windows). Der Entwickler dieses Frameworks bezeichnet es selbst als experimentell. Er nennt auf der Projekt-Website unter anderem den Anspruch einer sehr leichtgewichtigen Implementierung – sowohl für die darunterliegende API, als auch für die Einbindung in Anwendungen: Programme sollen nicht unnötig aufgebläht werden. Dafür nutzt PyGUI Frameworks, die Zugriff auf System-APIs bieten, um die systemeigenen GUI-Funktionen auszunutzen.
Eine zentrale Eigenschaft von PyGUI ist die feste Integration des Document-View-Patterns, wodurch sich andere Design-Patterns nicht auf sinnvolle Weise implementieren lassen.

Website: `http://www.cosc.canterbury.ac.nz/greg.ewing/python_gui/`
Lizenz: unbekannt

Bewertung der Schlüsseleigenschaften:

Eigenschaft	Note	Anmerkung
Plattformunabhängigkeit	2	plattformabhängige Zusatzmodule benötigt
Videodarstellung	4	höchstens über OpenGL-Integration
Dokumentationsqualität	4	gesamte Dokumentation nur im Code lesbar, einige Dinge nur über ein Beispielprogramm ersichtlich
Schwierigkeitsgrad	3	für die Verwendung zuerst eine umfassende Vorstellung vom Zusammenspiel und der Funktionsweise der Komponenten benötigt
Lernbarkeit	3	anfangs sehr unübersichtlich aufgrund der mangelhaften Dokumentation
Maintenance	3	neueste Version vom 25. März 2017[1], jedoch auch kein hoher Anspruch durch Nutzung von Standard-APIs
Anpassbarkeit	4	stark gebunden an die Funktionalität der System-APIs, jedoch erweiterbare Grundklassen und Canvas-Klasse zum Zeichnen und Anzeigen von Bildern

Stärken:

- Vorbereitete (und erweiterbare) Möglichkeit zur Datenspeicherung (Teil des Design-Patterns)
- Eventuelle Möglichkeit der Nutzung von GTK auf beliebigen Systemen (Die normale Installation nutzt nur Standard-APIs des jeweiligen Systems, ließe sich aber vielleicht anpassen)

Schwächen:

- Eingeschränkte Anpassbarkeit durch Festlegung einer festen System-API bei der Installation des Frameworks:
  - Windows: Win32
  - Linux: GTK
  - MacOSX: Cocoa
- Einschränkung auf das Document-View-Pattern

- Für den Anfang leicht undurchsichtige Implementierung, da zentrale Dokumentation nicht den essentiellen Überblick über die generischen Klassen schafft
- Fehler in der Windows-Implementierung von `GScrollableViews.py` macht Nutzung des Moduls unmöglich

GUI-Beispiel: Es hat sich als sehr schwierig und mühsam erwiesen, die Beispiel-GUI zu erstellen. Das Hauptproblem war das korrekte Layout, da das Grid-Layout von PyGUI zu unflexibel ist. Die Lösung war hier, eine eigene Klasse zu schreiben, über die das Layout wesentlich anpassbarer ist. Für so einen simplen Dialog ist das unerwartet viel Programmieraufwand.

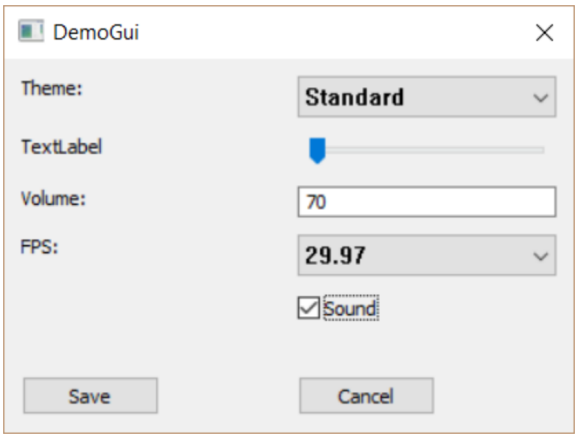


Abbildung 3: Mit PyGUI unter Windows erstellte Beispiel-GUI.

Eignungseinschätzung für das Projekt: Wie bereits oben erwähnt, bezeichnet der Entwickler von PyGUI sein Werk als experimentell. Diese Eigenschaft kann man tatsächlich erkennen, während man eine Grafische Oberfläche mit dem Framework erstellt: Zum Beispiel sind die Module durchzogen von auskommentierten Zeilen und in mindestens einem Modul befinden sich aktive Debug-Ausgaben. Durch den Fehler im Modul `GScrollableViews.py` der Windows-Installation ist es außerdem unmöglich, scrollbare Objekte zu erzeugen.
Neben diesen Punkten spricht wohl auch bereits die nicht sehr zufriedenstellende Bewertung der Schlüsseleigenschaften für sich. PyGUI ist deshalb gänzlich ungeeignet für das Projekt.

II.1.1.4 libavg

Autor: Alexander Bonin

- Plattformunabhängigkeit
  - Läuft auf Linux, Mac OS X und Windows
- Videodarstellung
  - Sehr einfach und gut optimiert möglich
- Qualität der Dokumentation
  - Auf der offiziellen Website ist eine sehr gute Dokumentation vorhanden.
- Schwierigkeitsgrad der Benutzung
  - Wahrscheinlich nicht besonders hoch, da viele Objekte und Funktionen vorgeben sind.
- Lernbarkeit

  - Verhältnismäßig schwierig zu lernen, da es komplett eigene Strukturen und Objekte benutzt. Python ist nur die Skripting-Language um die in C++ geschriebenen Plug-Ins zu steuern.
- Maintenance
  -
- Lizenz (academic/commercial use)
  - GNU Lesser General Public License
  - auch für proprietäre Programme nutzbar
- Anpassbarkeit (eigene Widgets)
  - Nicht wirklich gegeben, da mit vorgegebenen Objekten, die in einer anderen Sprache geschrieben sind, gearbeitet wird.
- Python3 Unterstuetzung
  - Leider nicht vorhanden, nur Unterstützung für Python 2.7.

Stärken

Die größte Stärke dieses Frameworks ist die gut Performance, weswegen libavg auch für live-Animationen und Videowiedergabe verwendet wird.

Schwächen

Leider unterstützt dieses Framework nicht die aktuelle Python-Version und wird auch nicht aktiv weiterentwickelt. Dies verrät zumindest ein Blick in die GitHub Commit-History, die in den letzten Jahren nur aus wenigen Bugfixes besteht. Der neueste Stabile Release ist von Mitte 2014. Außerdem eignet sich libavg, aufgrund der in C++ geschrieben Plug-Ins, eher für aufwendige 3d-Animationen und Spiele als für eine komplexe Videoschnitt GUI.

II.1.1.5 PyGObject

Autor: Valentin Ackva

Überblick
PyGObject ist der Nachfolger von PyGTK. Man kann sagen, dass PyGObject = Python + GTK3 ist. Das Framework nutzt dabei GObject, um C Bibliotheken mit einzubinden.

Stärken

- Liefert viele Bibliotheken mit: u.a. GTK+, GStreamer, WebKitGTK+, GLib, GIO.
- Pitivi - ein Open Source Video Bearbeitungsprogramm nutzt PyGObject.

Schwächen

- Aufgrund der vielen Bibliotheken aufwändig zu installieren (Windows).
- Nutzt keine nativen Widgets.

Plattformunabhängigkeit
Wie schon unter pyGTK: Das Framework läuft auf allen Plattformen (Win, Linux, Mac), jedoch kann es unter Mac zu einzelnen Anzeigefehlern kommen.

Videodarstellung

Die Videodarstellung ist mit der mitgelieferten Bibliothek GTK+ leicht umzusetzen. Zusätzlich gibt es viele Hilfestellungen dazu online.

Qualität der Dokumentation
Dokumentationen und Hilfestellungen zu PyGObject sind eher rar. Zusätzlich haben sich bei der neuen Version von GTK+ viele grundlegenden Strukturen geändert.

Schwierigkeitsgrad der Benutzung und Lernbarkeit
Mit einem GUI Builder (z.B. Glade) recht gut umsetzbar, jedoch erfordert die Nutzung des Programmes eine intensivere Einarbeitung.

Maintenance
PyGObject wird aktiv weiterentwickelt. Last released: Nov 30, 2018

Lizenz (academic/commercial use): GNU Lesser General Public License

Anpassbarkeit (eigene Widgets)
Mit GTK+ viele Möglichkeiten eigene Widgets zu erstellen, welche sich mit CSS anpassen lassen.

Python3 Unterstützung: Vorhanden

II.1.1.6 Pyforms

Autor: Maximilian Fornaçon

Überblick: Pyforms v4 ist ein junges GUI-Framwork für Python3, welches dem Entwickler die möglichkeit geben soll schnell Anwendungen, welche unter drei verschiedenen Umgebungen laufen, zu erstellen: Web, Terminal und Desktop GUI. Ich habe mich auf das erstellen von Desktop-Applikationen konzentriert.
Dass komplexe Projekte, mit Pyforms möglich sind, zeigen die Beispielprojekte, welche in der Dokumentation verlinkt sind:

- `https://github.com/UmSenhorQualquer/pythonVideoAnnotator`
- `https://github.com/UmSenhorQualquer/3D-tracking-analyser`

Plattformunabhängigkeit: Pyforms läuft unter Windows, MacOS und Linux.
Videodarstellung: Mit dem ControlPlayer-Widget ist es möglich Videos abzuspielen. Das Widget bringt Play-/Pause-Buttons, einen Framescounter und eine Progressbar mit Zeitanzeige mit. Mir ist es jedoch nicht gelungen die Elemente des Widgets anzupassen. Ton abzuspielen ist mir ebenfalls nicht gelungen.
Qualität der Dokumentation: Die offizielle Dokumentation ist noch in der Erstellung, ist deshalb sehr lückenhaft und enthält defekte Links. Sie gibt zwar einen guten Einstieg in das Framwork, erklärt z.B. das MVC-Pattern, jedoch vermisst man tieferführende Informationen.
Lernbarkeit und Schwierigkeitsgrad der Benutzung: Pyforms verwendet eine eigne Art und Weise, wie Widgets angeordnet werden können. Damit ist es sehr einfach grobe Layouts zu erstellen. Die Feinjustierung erwies sich jedoch als schwierig. Pyforms erlaubt es mittels CSS-Stylesheets die Widgets anzupassen. Aufgrund der unvollständigen Dokumentation und fast gar keinen weiteren Materialien außer dieser Dokumentation ist es sehr schwierig komplexere Projekte umzusetzten.
Lizenz: Das Projekt steht unter der MIT-Lizenz, und ist deshalb geeignet für unser Projekt.
GUI-Beispiel: Es ist mir, ohne einen extremen Zeitaufwand betreiben zu wollen, nicht möglich gewesen die GUI exakt nachzubauen. Auch die Anpassung der Farben stellte sich, aufgrund der fehlenden Dokumentation, als sehr schwierig heraus.
Stärken:

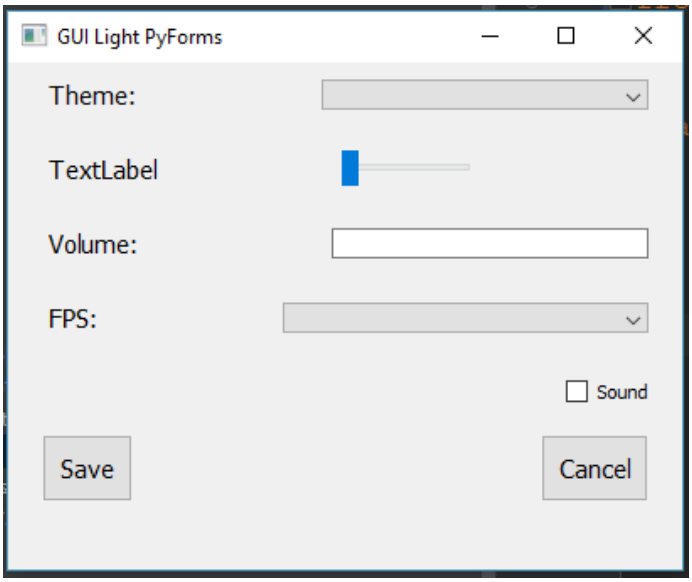


Abbildung 4: PyForms hell

- 3 Modi
  PyForms-GUI
  PyForms-Terminal
  PyForms-Web für Django
- Gut für Prototyping
- Style und Layout mit CSS
  Für verschiedene Betriebssysteme auch verschiedene Styles möglich

Schwächen:

- Dokumentation sehr unvollständig
- wenig Onlineressourcen

Fazit: Pyforms hat hohes Potential. Aufgrund der Neuheit des Projektes und der schwachen Dokumentation ist es aber zu diesem Zeitpunkt nicht geeignet für unser Projekt.

II.1.1.7 PyQt5 mit QtCreator

Autor: Markus Leupold

Überblick: PyQt5 ist eine Python-Schnittstelle für das Anwendungs-Toolkit „Qt", ein äußerst umfangreiches und praktisch grenzenlos einsetzbares Framework für C++ und JavaScript. Es existieren zum Beispiel Qt-Bibliotheken für Netzwerkkommunikation, Multimedia und Verarbeitung geographischer Daten. Für das Projekt viel interessanter ist jedoch, dass sich mit Qt sehr leistungsstarke grafische Nutzeroberflächen plattformübergreifend entwickeln lassen. Der Qt Creator – ein WYSIWYG-Editor – vereinfacht dabei den Entwicklungsprozess stark.
Der Qt Creator ist auch zusammen mit PyQt5 verwendbar. Dazu erstellt man zuerst mithilfe des Creators ein .ui-File. Diese Datei würde normalerweise mit dem Qt-Framework geladen, das den Inhalt interpretieren und die entsprechende Oberfläche dazu aufbauen würde. Das .ui-File lässt sich jedoch auch mit PyQt5 laden. Dieses übernimmt dann die Übermittlung des Inhalts an das Qt-Framework und sorgt dafür, dass das Backend nun nicht mehr ein C++-,

sondern ein Python-Programm ist. Auf diese Art und Weise lassen sich die Einfachheit von Python und die Leistung von Qt kombinieren.

Websites: PyQt5: `https://www.riverbankcomputing.com/software/pyqt/intro`
Qt: `https://www.qt.io`
Lizenz: GNU GPL v3; Riverbank Commercial License

Bewertung der Schlüsseleigenschaften:

Eigenschaft	Note	Anmerkung
Plattformunabhängigkeit	1	
Videodarstellung	1	
Dokumentationsqualität	3	Qt bietet umfangreiche Dokumentationen, jedoch ist PyQt5 noch fast komplett dokumentationslos
Schwierigkeitsgrad	3	Qt Creator ist einfach zu nutzen, allerdings ist die Python-Integation etwas mühsam zu verstehen aufgrund der fehlenden Dokumentation von PyQt5
Lernbarkeit	1	Anfangs zwar schwierig, aber Durchbruch nach kurzer Zeit möglich
Maintenance	1	Sowohl Qt mit Qt Creator, als auch PyQt5 werden durchgehend verbessert
Anpassbarkeit	1	Erstellung eigener Widgets/Komponenten und Nutzung von CSS möglich

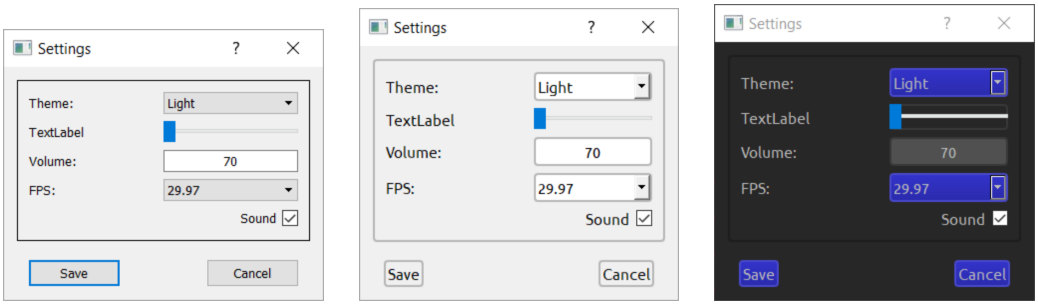
Stärken:

- Qt Creator erleichtert Entwicklung durch visuelle Komponente
- Volle Anpassbarkeit und Möglichkeiten zur eigenen Widget-Erstellung
- großer Komponenten-Pool

Schwächen:

- vermutlich lange Einarbeitungszeit nötig, um zufriedenstellende Ergebnisse zu erhalten
- mangelhafte Dokumentation für PyQt5

GUI-Beispiel: Die Gestaltung der Beispiel-GUI erweist sich mit Qt anfangs als etwas mühsam, da zuerst eine gewisse Einarbeitungszeit vonnöten ist, um das Zusammenspiel aller beteiligten Systeme zu verstehen. Selbstverständlich geht dem auch noch ein wenig Zeit zur Installation der benötigten Systeme voraus. Hier kann es tatsächlich etwas unfreundlich werden, denn Qt präsentiert die zwei Themen der Nutzungslizenz und des Installers etwas unübersichtlich. Sind diese Hürden dann schließlich überwunden, lässt sich die Oberfläche geradezu spielerisch leicht aufbauen:
Das grundlegende Layout der Komponenten wird mit dem Qt Creator „zusammengeklickt". Jederzeit ist es möglich, über wenige Befehle in einem Python-Skript das Ergebnis anzuzeigen. Das Design der Oberfläche definiert man – unabhängig vom Layout – über Stylesheets, deren Syntax und Elemente stark an CSS angelehnt sind. Damit lässt sich sogar das Ubuntu-Design der Vorlage auf einem Windows-System relativ gut imitieren.
Abb. 5 zeigt eine unstilisierte und die zwei stilisierten Varianten der Beispiel-GUI. Ein paar Dinge wurden nicht neu designt, da der Design-Prozess insgesamt sehr aufwendig ist. Bei der Checkbox wäre es beispielsweise wohl nötig gewesen, das Aussehen für alle Zustände komplett neu zu definieren, da bei der Anwendung auch nur eines beliebigen Stils die Checkbox ein einfaches Quadrat wird. Das Häkchen geht dabei gänzlich verloren.



(a) Ohne Stylesheets. *(b) Helles Theme.* *(c) Dunkles Theme.*

Abbildung 5: Beispiel-GUI mit PyQt5 und Qt Creator unter Windows.

Eignungseinschätzung für das Projekt: Es steht wohl außer Frage, dass ein so großes und kommerziell genutztes Anwendungs-Framework wie Qt überwiegend Vorteile bietet. Damit erstellte grafische Nutzeroberflächen sind leistungsstark und vielfältig anpassbar. Qt selbst ist also bestens geeignet. Interessant ist vielmehr die Integration der Qt-Oberflächen in ein Python-Programm: Auch hier ist alles sehr einfach gehalten, wobei es aber an der PyQt5-Dokumentation mangelt. Fast vollständig ausgeglichen wird das dadurch, dass die Python-Methoden in ihrer Funktion nach außen 1:1-Kopien der C++-Funktionen von Qt sind, weshalb die Qt-Dokumentation oft ausreicht.
Im Abschnitt der Beispiel-GUI ist ja bereits deutlich geworden, dass zwar der Anfang mühsam ist. Der Lern-Aufwand rentiert sich jedoch, da PyQt5 mit dem Qt Creator vor allem drei Dinge bringt: Einfachheit, Ordnung und Übersichtlichkeit. Deshalb wird für diese Framework-Kombination eine klare Empfehlung ausgesprochen.

II.1.1.8 WxPython

Autor: Valentin Ackva

Überblick
WxPython ist ein Open Source GUI-Toolkit, das auf der in C++ implementierten Bibliothek wxWidgets aufbaut. WxPython ist plattformübergreifend und bietet auf Windows, Linux und Mac nativ aussehende GUIs. Es enthält alle modernen GUI-Elemente und bietet deshalb eine gute Basis für professionelle GUIs.

Stärken

- Große Anzahl an Widget-Bibliotheken
- Nativer Look
- Relativ Flexibel (Modifikationen)

Schwächen

- Nicht vorinstalliert (wie Tkinter)
- UI-Layout mit vielen Hierarchien und Fenstern (schnell unübersichtlich)

Plattformunabhängigkeit
WxPython ist ein Cross-Plattform Toolkit. Somit läuft es auf allen gängigen Plattformen ohne vorherige Modifikationen.

Videodarstellung
Die Bibilothek "wx media.MediaCtrl"kann Videos + Audiodateien (MP3, ...) anzeigen und rendern (gängige Formate). Dabei nutzt "wx media"native backends um Mediendatei zu renden (Mac - QuickTime, Win - ActiveMovie/DirectShow).

Qualität der Dokumentation
Dokumentation im Internet ist ausreichend aber nicht überragend. Dennoch existiert eine große Community mit vielen Foren.

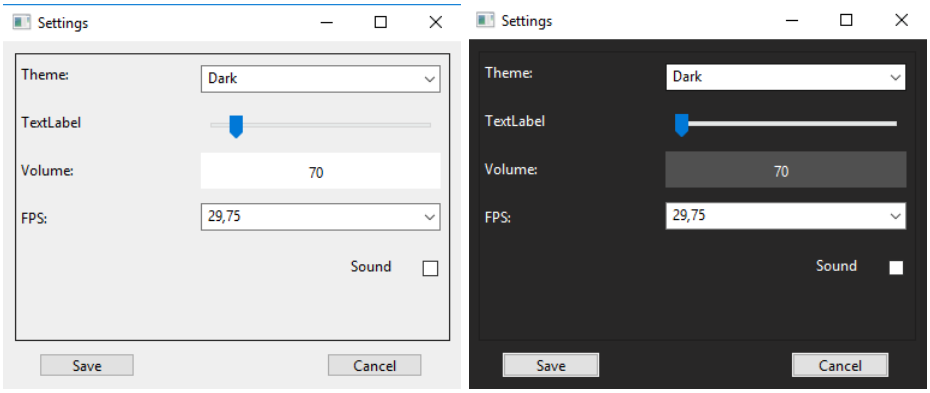
Schwierigkeitsgrad der Benutzung und Lernbarkeit
WxPython erweist sich als relativ simple, einfach zu schreiben und einfach zu verstehen. Jedoch könnte es bei vielen GUI-Ebenen unübersichtlich werden. **Maintenance**
Wird häufig aktualisiert, letztes Update am 25.06.18.

Lizenz (academic/commercial use): GNU Lesser General Public License

Anpassbarkeit (eigene Widgets)
Vorhandene Widgets können modifiziert werden. z.B. mit wxWidgets (C++) können 'generic' Widgets erstellt werden.

Python3 Unterstützung: Vorhanden

Beispiel GUI mit WxPython



II.1.2 OpenCV

Autor: Clemens Zwinzscher

website: `https://opencv.org/`
github: `https://github.com/opencv/opencv`
opencv-python documentation: `https://opencv-python-tutroals.readthedocs.io/en/latest/`

OpenCV (Open Computer Vision) ist eine quelloffene Bibliothek, die Algorithmen zur Bildverarbeitung und zum maschinellen Sehen (d.h. computergestützte Lösungen von Aufgabenstellungen, die sich an den Fähigkeiten des menschlichen visuellen Systems orientieren) bereitstellt. Die Bibliothek wurde hauptsächlich in C++ geschrieben, aber wird auch in anderen Sprachen angeboten, darunter Python. OpenCV läuft auf vielen verschiedenen Betriebssystemen (z.B. Windows, Linux, Mac) und ist somit plattformunabhängig nutzbar. Da es unter der BSD-Lizenz lizensiert ist, kann es frei benutzt und auch verändert werden. Des weiteren wird OpenCV auch immer noch gut unterstützt und weiterentwickelt (die letzte Version 4.0 erschien im November 2018). Die Installation von OpenCV gestaltet sich nicht sehr schwierig, da es sehr einfach über pip installiert werden kann.

Nutzer von Ubuntu können es auch aus dem Ubuntu-Repository installieren, jedoch ist diese Version sehr veraltet. OpenCV ist außerdem sehr gut dokumentiert und bietet viele Beispiele an, sodass es relativ leicht zu erlernen und zu verstehen ist und auch außerhalb der offiziellen Dokumentation findet man online viele Ressourcen zu OpenCV.

Einige Anwendungsbereiche von OpenCV wären:

- Bearbeitung einzelner Bilder (z.B. Größe ändern, einzelne Regionen ausschneiden, Farbraum ändern, Filter anwenden)
- Video einlesen und frame für frame bearbeiten
- Video aus mehreren Bildern erstellen und exportieren
- Objekterkennung in Bildern und Videos → haar cascade Daten in OpenCV enthalten
- Erkennung von Gesichtern, Augen, Körpern ...
- Text detection (Regionen mit Text erkennen → hat jedoch nicht so gut funktioniert)
- beinhaltet auch eine machine learning Bibliothek

Eine Gesichtserkennung in OpenCV zu schreiben ist nicht wirklich schwer:

```
import cv2

faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
image = cv2.imread('bild.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30)
)
```

Eine Voraussetzung für OpenCV ist Numpy. Numpy ist eine Bibliothek zur Handhabung von Vektoren, Matrizen oder generell mehrdimensionalen Arrays. OpenCV hängt von Numpy ab, da es alle Bilder als Numpy array speichert, weshalb es auch bei der Installation von OpenCV mitinstalliert wird. Numpy ermöglicht einen einfachen Zugriff auf einzelne Pixel eines Bildes oder auf bestimmte Regionen, woraufhin diese bearbeitet und verändert werden können.
Beispiel:

```
import cv2
import numpy as np
img = cv2.imread('bild.jpg')
px = img[100,100]  # auf einen Pixel zugreifen
region = img[100:150, 120:180]  # auf eine Region zugreifen
```

II.1.3 Face-Tracking-Bibliotheken

Autor: Felix Dittrich

II.1.3.1 Face Recognition

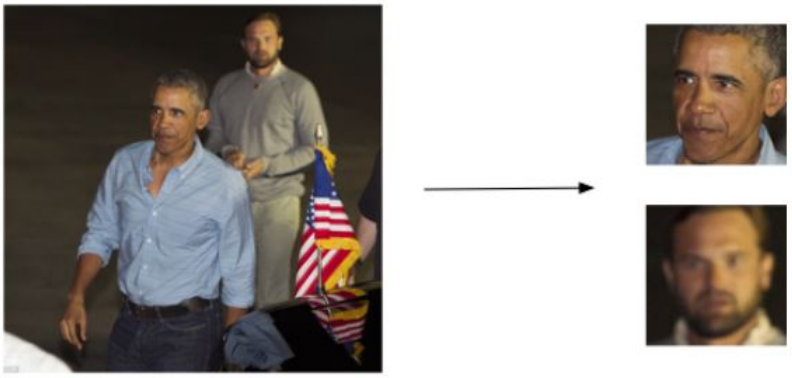
Link: `https://github.com/ageitgey/face_recognition`

Die Bibliothek Face Recognition konzentriert sich hauptsächlich auf eine saubere und schnelle Erkennung von Gesichtern wobei der Schwerpunkt wirklich nur auf Gesichtern liegt.

Die Bibliothek umfasst nur wenige Methoden was es dem Benutzer einfach machen soll seine Zielfunktion umzusetzen, Manko dabei ist das Objekte nicht erkannt werden und die weitere Verwendung ausserhalb dieses Themengebietes nicht gegeben ist.
Voraussetzung für diese Bibliothek wären:

- Mac, Linux (Windows möglich allerdings nur über Umwege)
- Python 3.3+ oder 2.7
- cmake, dlib
- OpenCV

Die Gesichtserkennung hat sich als sehr gut herausgestellt.



Input Output

Die Qualität der Dokumentation war ebenfalls gut.

face_recognition.api.compare_faces(*known_face_encodings, face_encoding_to_check, tolerance=0.6*)
[source]

Compare a list of face encodings against a candidate encoding to see if they match.

Parameters:
- **known_face_encodings** – A list of known face encodings
- **face_encoding_to_check** – A single face encoding to compare against the list
- **tolerance** – How much distance between faces to consider it a match. Lower is more strict. 0.6 is typical best performance.

Returns: A list of True/False values indicating which known_face_encodings match the face encoding to check

Der Schwierigkeitsgrad der Benutzung sah aufgrund der guten Dokumentation sowie der Größe und Übersichtlichkeit einfach aus.

Automatisch alle Gesichter in einem Bild erkennen

```
import face_recognition

image = face_recognition.load_image_file("my_picture.jpg")
face_locations = face_recognition.face_locations(image)

# face_locations is now an array listing the coordinates of each face!
```

Als Ausgabedaten wird ein Array mit den Koordinaten der Fläche/n in denen sich Gesichter befinden zurück gegeben.
Die Bibliotheksgröße ist im Vergleich zu OpenFace und TensorFlow Object Detection eher klein.
Die Bibliothek steht unter der MIT Lizens und ist somit frei verfügbar.
Zum Support kann man sagen das Sie dauerhaft und regelmäßig erweitert und aktualisiert wird.
Es gibt eine Übersicht mit 41 Projekten in denen Face Recognition verwendet wurde:
Link: `https://www.hackster.io/projects/tags/face+recognition`
Dort könnte man sich Anregungen / Ideen für das eigene Projekt suchen oder auch nach Lösungen suchen falls ein Problem auftritt.
Methode der Benutzung:

- BGR color aus OpenCV wird in RGB konvertiert
- dieser Wert wird als Parameter an die jeweilige vorprogrammierte Methode übergeben und anhand dieser wird das/die Gesicht/er im Bild oder Video erkannt

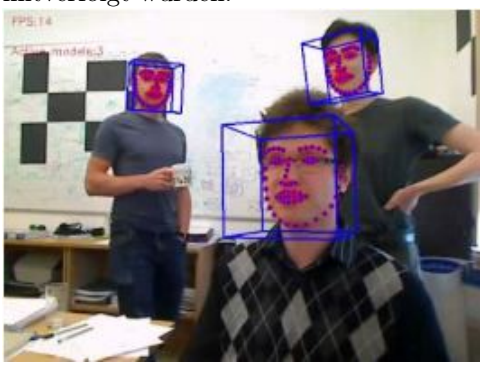
II.1.3.2 OpenFace

Link: `https://cmusatyalab.github.io/openface/`

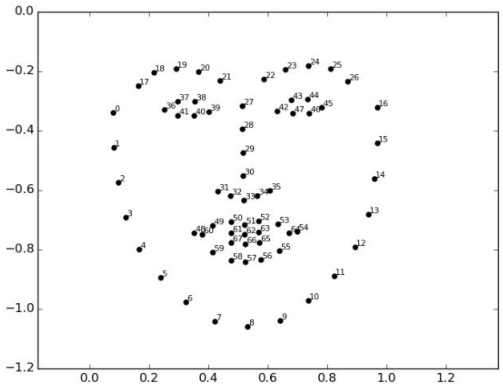
Die Bibliothek OpenFace richtet sich hauptsächlich auf die Mimikerkennung verfügt allerdings auch über reine Methoden zur Gesichtserkennung.
Voraussetzung für diese Bibliothek wären:

- Linux (bestimmte Distribution), Mac, Windows (allerdings nur über Docker)
- Python x.x - unbekannt
- OpenCV
- dlib

Die Gesichtserkennung war eher durchschnittlich, da schnelle Bewegungen nicht zeitgleich mitverfolgt wurden.



Zur Qualität der Bibliotheksdokumentation kann man nur sagen das sie sehr unübersichtlich und definitiv nicht ausreichend ist.
Die Bibliotheksgröße liegt im Vergleich zu OpenFace und TensorFlow Object Detection im Mittelfeld.
Die Bibliothek steht unter der Apache Lizens und wäre somit frei verfügbar.
Zur Verwendung in anderen Projekten wurde leider nichts gefunden.
Zum Support kann man sagen, dass der letzte commit am 18.06.2018 war und somit eher nicht zu empfehlen ist.
Als Ausgabedaten erhält man bestimmte Punkte aus einem Koordinatensystem, was aus der Dokumentation allerdings sehr schlecht ersichtlich war.



```
OUTER_EYES_AND_NOSE= [36, 45, 33]
```

Methode der Benutzung: (nicht richtig ersichtlich aus der Dokumentation)

- unterteilt in Mimikerkennung(Landmarks) und Gesichtserkennung(Block)

II.1.3.3 TensorFlow Object Detection

Link: `https://github.com/tensorflow/models/tree/master/research/object_detection`

Die Bibliothek TensorFlow Object Detection bietet uns ein breites Feld an Möglichkeiten da sie soviel im Video als auch im Bild Gesichter, Personen und Objekte erkennen und verfolgen kann, sie verfügt über vorgerfertigte Module die das tracken für uns wesentlich vereinfachen würden. Darunter auch das Erkennen mehrerer Objekte und Personen in einem Video.
Voraussetzungen für diese Bibliothek wären:

- Windows, Linux, Mac
- TensorFlow
- Python 3.6 oder darunter (Update fehlt noch)
- OpenCV

Die Gesichtserkennung bzw. allgemein die Erkennung von Personen, Gesichtern und Objekten war im Vergleich zu den anderen beiden Bibliotheken die beste, schnelle Bewegungen und auch ein Perspektivenwechsel machten keine Probleme.

Die Qualität der Dokumentation war ebenfalls sehr gut und ausführlich mit dem extra Pluspunkt das es zu TensorFlow und TensorFlow Object Detection eine riesige Community gibt sowie Tutorials, etc.

Die Bibliotheksgröße ist im Vergleich zu OpenFace und Face Recognition sehr groß.
Der Support der Bibliothek ist regelmäßig und sie wird stetig erweitert.
TensorFlow Object Detection findet dementsprechend in sehr vielen Projekten Anwendung und man findet viele Beispiele im Internet.
Der Schwierigkeitsgrad der Benutzung sah aufgrund der guten Dokumentation einfach aus benötigt allerdings ein wenig Einarbeitungszeit.
Methode der Benutzung:

- viele Möglichkeiten !
- z.B.:
  - Video einlesen
  - Model verwenden
  - für Erkennung in Bilder teilen
  - bearbeitetes Video ausgeben
- oder:
  - komplettes Video übergeben
  - Gesichter erkennen
  - bearbeitetes Video ausgeben
- etc.

II.1.4 PDF-Konverter

Autor: Julian Oliver Böttcher

II.1.4.1 Ghostscript

Ghostscript ist ein Interpreter der Seitenbeschreibungssprache PDF, welches mehrere Einsatzmöglichekeiten bietet. Zum einen kann man es für die Rasterung von PDF-Dateien nutzen, also um aus PDF-Code eine Datei zu erstellen. Außerdem kann man es benutzen um PDF-Dateien in fast alle möglichen Bildformate umzuwandeln, um sie so besser benutzen zu können, zum Beispiel zum einblenden in Videos.
Ghostscript ist auf allen gängigen Platformen verfügbar, darunter Windows, Linux und Mac OS. Die Installation ist auch sehr einfach und kann über pip mit "pip(3) install ghostscriptërledigt werden.
Oft aktualisiert wird Ghostscript allerdings nicht. Das letzte Update gab es Anfang 2018.

II.1.4.2 ImageMagick (Wand)

Wand ist eine auf Bibliothek programmiert in C für Python und basiert auf ImageMagick. Wand kann zusätzlich zum Konvertieren von PDF-Dateien zu Bildern, diese Bilder sogar noch zusammenfügen oder anders bearbeiten, z.B. mit Effekten.
Die Installation ist etwas komplizierter als bei Ghostscript, da man erst ImageMagick herunterladen muss und erst danach Wand per pip mit "pip(3) install Wandïnstallieren kann. Python 3 wird unterstützt.

II.1.5 UserSettings

Autor: Julian Oliver Böttcher

II.1.5.1 ConfigParser

ConfigParser benutzt .ini Dateien um Einstellungen abzuspeichern.
Die Installation ist sehr einfach und kann über pip mit "pip(3) install configparserërledigt werden.
Es scheint nicht so als würde ConfigParser noch regelmäßig geupdatet werden, da das letzte Update vor über 2 Jahren war.
Python 3 wird unterstützt.
Die Benutzung ist sehr einfach (siehe Bilder).

Standard .ini Datei:

```
[SectionOne]
Status: Single
Name: Derek
Value: Yes
Age: 30
Single: True

[SectionTwo]
FavoriteColor = Green
[SectionThree]
FamilyName: Johnson

[Others]
Route: 66
```

Erstellen einer .ini Datei:

```
# lets create that config file for next time...
cfgfile = open("c:\\next.ini",'w')

# add the settings to the structure of the file, and lets write it out...
Config.add_section('Person')
Config.set('Person','HasEyes',True)
Config.set('Person','Age', 50)
Config.write(cfgfile)
cfgfile.close()
```

Einlesen einer .ini Datei:

```
>>> Name = ConfigSectionMap("SectionOne")['name']
>>> Age = ConfigSectionMap("SectionOne")['age']
>>> print "Hello %s. You are %s years old." % (Name, Age)
Hello Derek. You are 30 years old.
```

II.1.5.2 JSON

JSON benutzt .json Dateien. Diese haben keine eine einheitliche Formatierung, weswegen man sie flexibler einsetzen kann.

Ein weiterer Vorteil ist, dass JSOn bereits in Python enthalten ist und somit nicht extra installiert werden muss.

Hier ein Beispiel:

```
data = {
    "president": {
        "name": "Zaphod Beeblebrox",
        "species": "Betelgeusian"
    }
}

with open("data_file.json", "w") as write_file:
    json.dump(data, write_file)


with open("data_file.json", "r") as read_file:
    data = json.load(read_file)


>>> blackjack_hand = (8, "Q")
>>> encoded_hand = json.dumps(blackjack_hand)
>>> decoded_hand = json.loads(encoded_hand)

>>> blackjack_hand == decoded_hand
False
>>> type(blackjack_hand)
<class 'tuple'>
>>> type(decoded_hand)
<class 'list'>
>>> blackjack_hand == tuple(decoded_hand)
True
```

II.1.6 Crossplatforming

Autor: Jeremy Risy

II.1.6.1 pyInstaller

- Wird immer mal wieder Aktualisiert
- Ist kostenlos
- einfach über pip oder pip3 Installierbar
- unter jedem Betriebssystem Nutzbaraber nur auf dem System auf dem das Skript gefreezt wurde läuft es auch also unter Windows10 gefrezzt dann ist die .exe Datei auch nur unter Windows10 Ausführbar

II.1.6.2 cx Freeze

- Es ist keine aktuelle Version vorhanden letztes Update 2017

- Es läuft auf jedem Betriebssystem aber nur auf dem System auf dem das Skript gefreezt wurde läuft es auch also unter Windows10 gefrezzt dann ist die .exe Datei auch nur unter Windows10 Ausführbar
- Probleme beim Installieren da keine Updates auf Neuere Pythonversionen

II.1.6.3 FBS

- Läuft unter jedem Betriebssystem
- Ist Aktuell und wird Ständig mit Updates versorgt
- Zur Zeit nur bedingt mit Python 3.7 kompatibel Updates sollen Folgen
- Ermöglicht eine Direkte Nutzung von PyQT
- ist auch kostenlos nutzbar
- basiert auf pyInstaller
- ermöglicht durch integrierten Installer die .exe ohne FBS zu Installieren und zu Nutzen

II.2 Entscheidungen des Technologieworkshops

Im Technologieworkshop haben wir uns entschieden, die Grafische Oberfläche mit `PyQt5` umzusetzen, da es mit dem QtCreator einfach zu benutzen ist, eine große Community und Unterstützung hat, häufig geupdatet wird, auf allen benötigten Plattformen funktioniert und durch seine umfangreichen Möglichkeiten und große Anpassbarkeit, die Umsetzung komplexer Widgets ermöglicht, die unser Projekt benötigt (z.B. Timeline). Außerdem hat `PyQt5` eine gute Performance, da es nativ ist und in C/C++ geschrieben wurde. Für die Videomanipulationen wird die `Openshot`-Bibliothek verwendet, die speziell dafür entwickelt wurde. Sie hat, wie oben beschrieben, viele Vorteile gegenüber anderen Bibliotheken und eine gute Performance, da sie in C++ geschrieben wurde. Für die Analyse der Videos, werden die Bibliotheken `OpenCV` und `Tensorflow` verwendet. Welche Bibliothek für das Rendern von PDFs als einzelne Bilder verwendet wird, wurde im Workshop noch nicht festgelegt. Sollten Bildbearbeitungsmechanismen benötigt werden, wird die Bibliothek `ImageMagick` bzw. PythonMagick verwendet und auch für das Rendern der PDFs benutzt. Sollten keine weiteren Bildbearbeitungsfunktionen benötigt werden, kann auch eine kleinere Bibliothek, wie `pdf2image` verwendet werden. Für das Freezen und das Erstellen von Installern für verschiedene Plattformen, wird die Bibliothek `fbs` verwendet werden. Diese vereinfacht das Freezen und gibt dem Team dadurch mehr Zeit für das eigentliche Programmieren. Für Unittests werden die Module `pytest` und `pytest-cov` verwendet.

II.3 Überblick über Architektur

Das Ziel ist es eine einfach wartbare und testbare Software zu entwerfen, die keine zu komplizierten Architekturvorgaben hat, damit die Bachelorstudenten diese auch mit relativ wenig Vorwissen umsetzen können. Da die grafische Oberfläche einen Hauptbestandteil eines Videobearbeitungsprogramms darstellt, ist es schwierig Tests zu ermöglichen, wenn keine gute Trennung von Darstellung und Logik vorhanden ist. Daher haben wir uns für den Model-View-Presenter-Ansatz entschieden der auf dem Entwurfsmuster Model-View-Controller basiert. Bei diesem Ansatz werden Modell und View komplett voneinander getrennt und nur durch einen Controller (auch Presenter genannt) verbunden. Für die Umsetzung der Timeline war dieser Ansatz jedoch nicht so gut geeignet, da hier Modell und View sehr voneinander abhängig sind. Daher haben wir uns dort für eine Art des Model-View-Viewmodel-Entwurfsmuster entschieden. Da wir die Bibliothek Openshot verwenden, welche ein Modell für Timeline und Clips vorgibt, wirkt sich das auch mit auf die Architektur aus. Das durch die Bibliothek vorgegebene Modell musste allerdings um einige Funktionen erweitert werden, da nicht alle Anforderungen unserer Software mit diesem Modell umsetzbar waren. Für Daten, die in vielen Klassen zur Verfügung stehen müssen, wie zum Beispiel Einstellungen oder Ressourcen

haben wir uns für Singletons oder statische Utility-Klassen entschieden, da diese sehr einfach zu implementieren sind und auch mit geringem Vorwissen gut umsetzbar sind. Dadurch enstehen zwar Nachteile durch zusätzliche Abhängigkeiten und für das Testen, allerdings stellen diese in Python kein großes Problem dar. Die Einstellungen so umzusetzen, hat den großen Vorteil, dass man die Einstellungen an einer Stelle ändern kann und alle anderen Klassen, die auf die Einstellungen zugreifen, greifen auf dieselbe Instanz zu und erhalten somit, sofort die geänderten Einstellungen. Bei der grafischen Oberfläche viel die Entscheidung das Component-Pattern zu verwenden und Views in einer Hierachie zu ordnen. Dabei wird das Observer-Pattern angewendet, damit Kinderelemente geupdated werden, wenn ihr Elternelement geupdated wird. Views, die Kinder einer anderen View sind, sind also Observer und jede View selbst ist ein Observable. Geplant war auch, dass Daten und Controller bzw. Presenter über das Observer-Pattern verbunden sind. Dies wurde jedoch nicht umgesetzt, was auch damit zusammen hängt, dass dies mit Verwendung der vorgegebenen Klassen der Bibliothek Openshot nicht ganz so einfach umsetzbar war. Die Projektordnerstruktur und der Zugriff auf Ressourcen wurden durch die Bibliothek fbs vorgegeben. Um den Zugriff auf Ressourcen zu vereinfachen wird ein eigenes Ressourcensystem entwickelt.

II.4 Definierte Schnittstellen

Schnittstelle zwischen Controller und View wird von Presenter festgelegt. Die View hat keine Logik und zeigt nur an, was der Presenter ihr sagt. Der Presenter setzt die Callback-Funktionen der View und aktualisiert so, was die View anzeigt. Die View selbst weiß nichts über die Daten. Außnahme bildet die Timeline, die ein eigenes Viewmodel hat. Das Viewmodel soll sich über das Observer-Pattern mit den Daten aktualisieren. Eine weitere Schnittstelle befindet sich zwischen Timeline und dem Autocut und zwischen Filemanager und Autocut. Der muss neue Dateien zum Filemanager hinzufügen, Spuren in der Timeline erstellen und Clips zur Timeline hinzufügen können. Diese Schnittstelle soll über einen Autocut-Controller umgesetzt werden, der Zugriff auf dem Timeline-Controller und den Filemanager-Controller hat.

II.5 Liste der Architekturentscheidungen

II.5.1 Architekturentscheidungen vor Programmierbeginn

- Model-View-Presenter
- Model-View-Viewmodel für Timeline
- Views nach Component-Pattern
- Projektstruktur wie von fbs vorgegeben
- Verwendung der Bibliotheken, wie in Abschnitt II.2 beschrieben

II.5.2 Architekturentscheidungen nach Sprint

II.5.2.1 Sprint 4

- Vermeidung von Magic Numbers durch Speichern aller Einstellungen in einer JSON Datei
- Settings als Singleton

II.5.2.2 Sprint 6

- Resources als Utility Klasse

II.5.2.3 Sprint 7

- Aktualisierung der Views mit Parent-View über Observer Pattern

II.6 Entwürfe

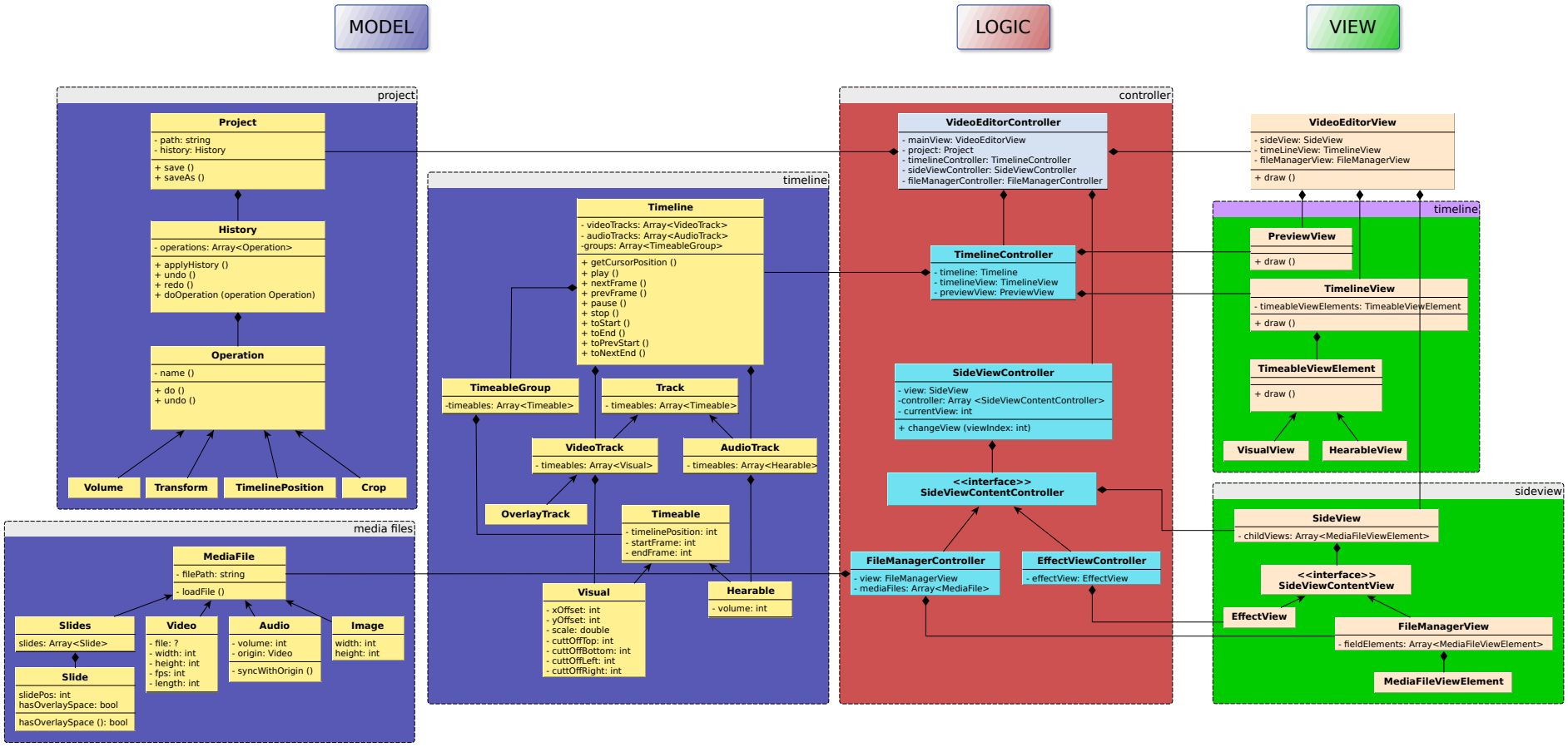
II.6.1 Architekur

Autor: Tim Jeske

II.6.1.1 Programmstruktur

Das folgende UML Diagram zeigt den ursprünglichen Entwurf der Software.

Undo-Konzept
Das Undo-Konzept ist erforderlich, damit man im Schnittprogram Benutzerinteraktionen wieder rückgängig machen kann. Dafür gibt es eine Klasse History, die alle Interaktionen speichert. Die Interaktionen selbst müssen als Kind der Klasse Operation implementiert werden. Diese Klasse hat ein Methode `do()`, welche die Interaktion ausführt und ein Methode `undo()`, die eine Interaktion ausführt, die die `do()`-Aktion rückgängig macht. Zum Beispiel kann es eine Aktion geben, „Verschiebe ein Timelineelement in der Timeline um x Frames nach links“. Wird diese Aktion benutzt, zum Beispiel mit 20 als Wert für x, wird eine solche Operation erstellt, deren `do()`-Methode, das Objekt um 20 Frames nach links verschiebt und deren `undo`-Methode, das Objekt um 20 Frames nach rechts verschiebt und damit wieder zu ursprünglichen Position.


MODEL
LOGIC
VIEW
project
Project
- path: string
- history: History
+ save ()
+ saveAs ()
History
- operations: Array<Operation>
+ applyHistory ()
+ undo ()
+ redo ()
+ doOperation (operation Operation)
Operation
- name ()
+ do ()
+ undo ()
Volume
Transform
TimelinePosition
Crop
media files
MediaFile
- filePath: string
- loadFile ()
Slides
slides: Array<Slide>
Slide
slidePos: int
hasOverlaySpace: bool
hasOverlaySpace (): bool
Video
- file: ?
- width: int
- height: int
- fps: int
- length: int
Audio
- volume: int
- origin: Video
- syncWithOrigin ()
Image
width: int
height: int
timeline
Timeline
- videoTracks: Array<VideoTrack>
- audioTracks: Array<AudioTrack>
-groups: Array<TimeableGroup>
+ getCursorPosition ()
+ play ()
+ nextFrame ()
+ prevFrame ()
+ pause ()
+ stop ()
+ toStart ()
+ toEnd ()
+ toPrevStart ()
+ toNextEnd ()
TimeableGroup
-timeables: Array<Timeable>
Track
- timeables: Array<Timeable>
VideoTrack
- timeables: Array<Visual>
AudioTrack
- timeables: Array<Hearable>
OverlayTrack
Timeable
- timelinePosition: int
- startFrame: int
- endFrame: int
Visual
- xOffset: int
- yOffset: int
- scale: double
- cuttOffTop: int
- cuttOffBottom: int
- cuttOffLeft: int
- cuttOffRight: int
Hearable
- volume: int
controller
VideoEditorController
- mainView: VideoEditorView
- project: Project
- timelineController: TimelineController
- sideViewController: SideViewController
- fileManagerController: FileManagerController
TimelineController
- timeline: Timeline
- timelineView: TimelineView
- previewView: PreviewView
SideViewController
- view: SideView
-controller: Array <SideViewContentController>
- currentView: int
+ changeView (viewIndex: int)
<<interface>>
SideViewContentController
FileManagerController
- view: FileManagerView
- mediaFiles: Array<MediaFile>
EffectViewController
- effectView: EffectView
VideoEditorView
- sideView: SideView
- timeLineView: TimelineView
- fileManagerView: FileManagerView
+ draw ()
timeline
PreviewView
+ draw ()
TimelineView
- timeableViewElements: TimeableViewElement
+ draw ()
TimeableViewElement
+ draw ()
VisualView
HearableView
sideview
SideView
- childViews: Array<MediaFileViewElement>
<<interface>>
SideViewContentView
EffectView
FileManagerView
- fieldElements: Array<MediaFileViewElement>
MediaFileViewElement

II.6.1.2 Userflow

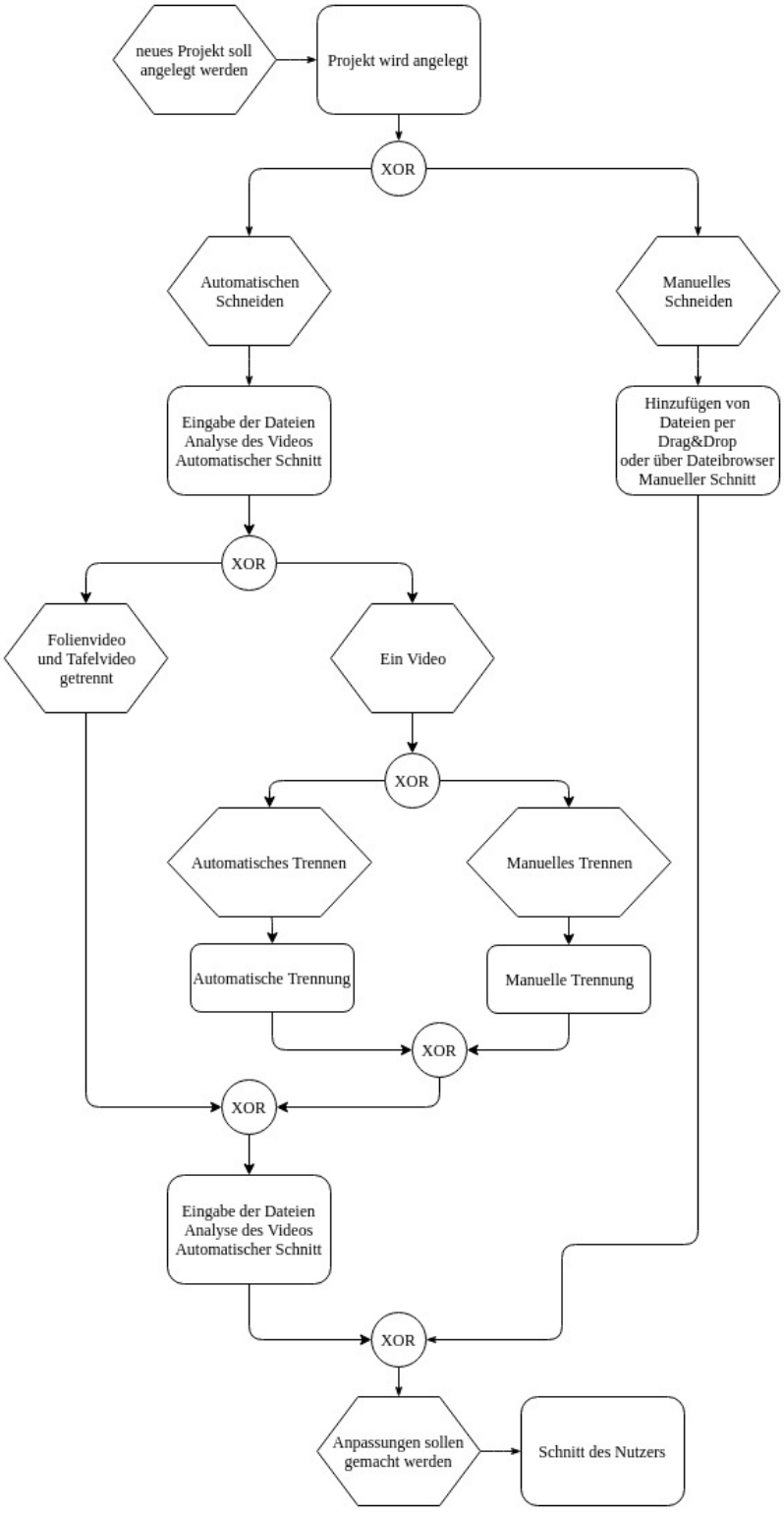


Abbildung 6: Visualisierung des Userflows

II.6.1.3 Datenverarbeitung

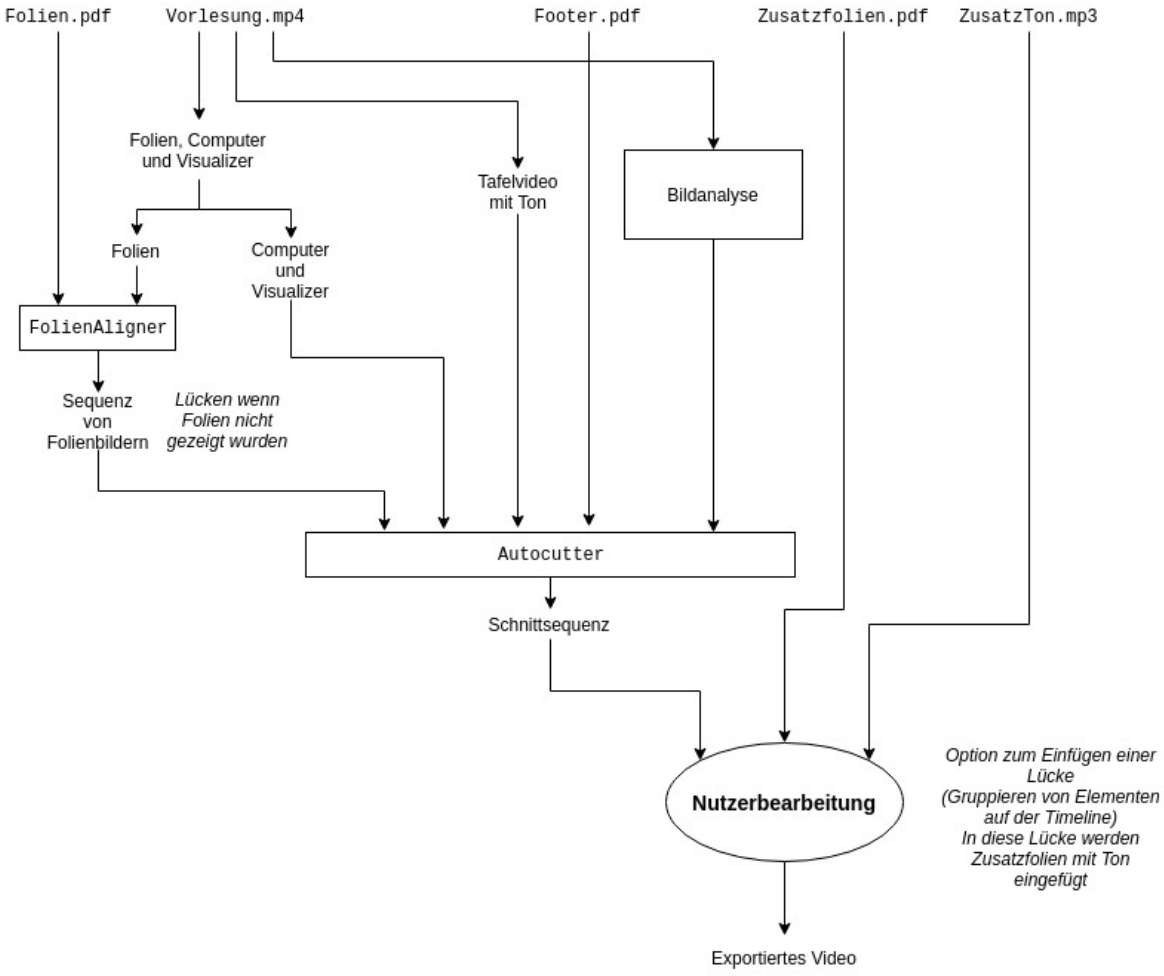


Abbildung 7: Schritte der Datenverarbeitung

II.6.2 Designentwurf

Autor: Maximilian Fornaçon

II.6.2.1 Gesamtlayout

Aufgabenstellung
Entwicklung eines vorerst endgültigen und einheitlichen gesamt Layouts für das Startfenster und das Editorfenster. Hierfür sollen auch die Ergebnisse der Entwurfsaufgaben für die Timeline, den Filemanager und die Videopreview berücksichtigt werden. Das Ergebnis soll ein Bild sein, dass für das Team als Vorlage für die Programmierung der GUI dienen soll. Weitere Detailbilder sind möglich. Maßangaben sind wünschenswert, aber optional!

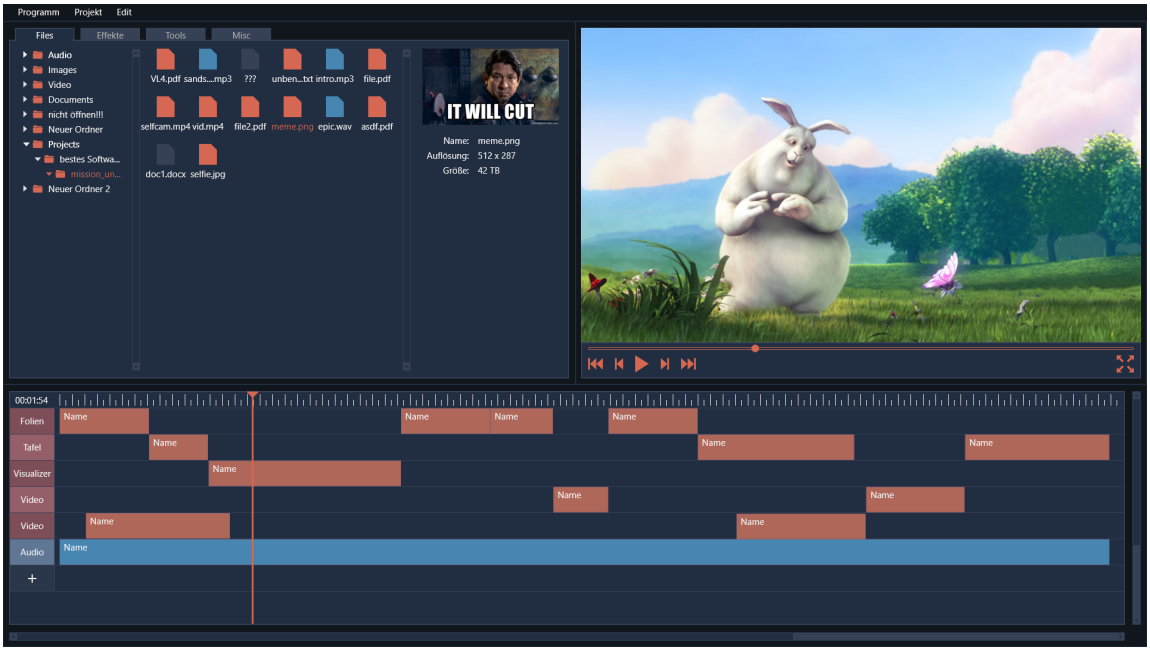


Abbildung 8: Hauptfenster im dunklen Design

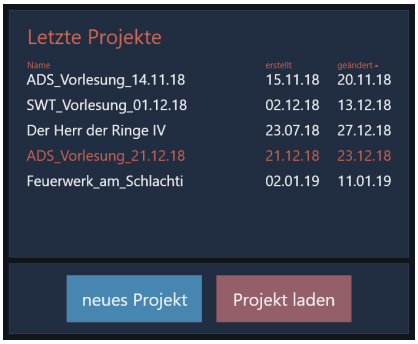


Abbildung 9: Startfenster 1

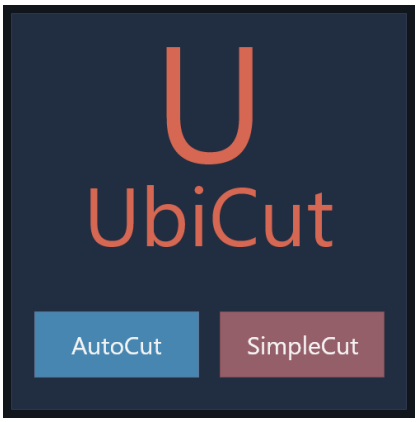


Abbildung 10: Startfenster 2

#11151C	Hintergrund
#212D40	Boxen
#2A3649	Timeline-plus-button
#364156	Rahmen, inaktive Tabs, Scollbars, nicht unterstützte Datein
#5F7695	Timeline-audio-feld
#4686B1	Timeline Audiospur, Audiofiles
#7D4E57	Timeline-optisches-Feld 1
#955F69	Timeline-optisches-Feld 2
#AE6759	Timeline Bildspur
#D66853	Playerbuttons, Fileicons, Foldericon, Timeneedle, Highlights
#B1B1B1	inactive Text
#F5F5F5	Text, Zeitleiste

Abbildung 11: Farbpalette

II.6.2.2 Buttons

Aufgabenstellung Es soll ein Konzept für das Design von Buttons und Symbolen erstellt werden. z.B. Play/Pause-Button usw.

II.6.2.3 Konzept

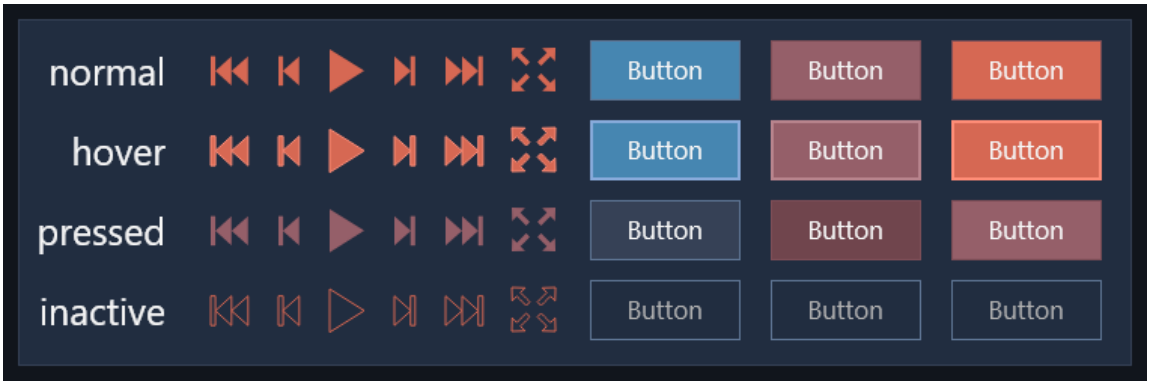


Abbildung 12: Buttons in verschiedenen Zuständen

II.6.3 Schnittstelle zwischen TimelineController und TimelineView

Autor: Markus Leupold

Für die Kommunikation sind Mime-Types sehr gut geeignet! Zu bedenkende Funktionen:

- Verschieben von Objekten
- Vorschau des eines einzelnen Clips
- Bearbeiten der gesamten Spuren (z.B. Mute der Audio-Spur oder Sichtbarkeit der Video-Spur)
- Bearbeiten einzelner Clips (Effekte, Schnitt)
- Überlappen von Videos/Übergänge
- Drag and Drop
- Markierungen, um mehrere Clips auszuwählen welche nicht unbedingt in einer gemeinsamen Spur liegen

II.6.3.1 Eigenschaften eines Timeables in der View

- ID
- Name
- Startzeitpunkt
- Länge
- Vorschaubild
- Ausgewählt (Boolean)
- (Typ)

II.6.3.2 Was die TimelineView dem Controller sagt

- Timeable hinzufügen
- Timeable löschen
- Timeable zerteilen
- Timeable umbenennen
- Timeable verschieben (lediglich neue Startzeit)
- Teil eines Timeables entfernen (Start- und Endzeit des entfernten Stücks sind relevant)
- Timeable auswählen
- Timeable abwählen

II.6.3.3

Von Controller zu View:

- alle Infos, die benötigt werden, damit ein Widget erstellt werden kann. Folgende Angaben sind Möglichkeiten:
  - Objekt-ID
  - Vorschaubild
  - Start und Ende (Zeit)

II.6.4 AutoCut - Konzept

Autor: Felix Dittrich

II.6.4.1 Aufgabenstellung

Es soll ein Konzept ausgearbeitet werden, welchen Nutzen, welche Funktionen und welche Auswirkungen die Modi haben. Dazu sollen Entwuerfe sowie Ideen festgehalten werden, wie das Ganze in der aktuellen GUI integriert werden kann bzw. wie es allgemein umsetzbar wäre. Im Unterpunkt Konzepte werden die einzelnen Entwuerfe grafisch dargestellt sowie die Idee dahinter kurz erklärt.

II.6.4.2 Konzepte

Konzept 1
Nach dem auswählen des AutoCut Buttons soll sich ein weiteres Fenster öffnen in welchem eine VideoDatei sowie eine Pdf hinzugefügt werden muss über die entsprechenden Buttons. Nach dem erfolgreichen hinzufügen soll sich das x in einen grünen Haken verändern und der ok Button soll erscheinen bzw. benutzbar sein.

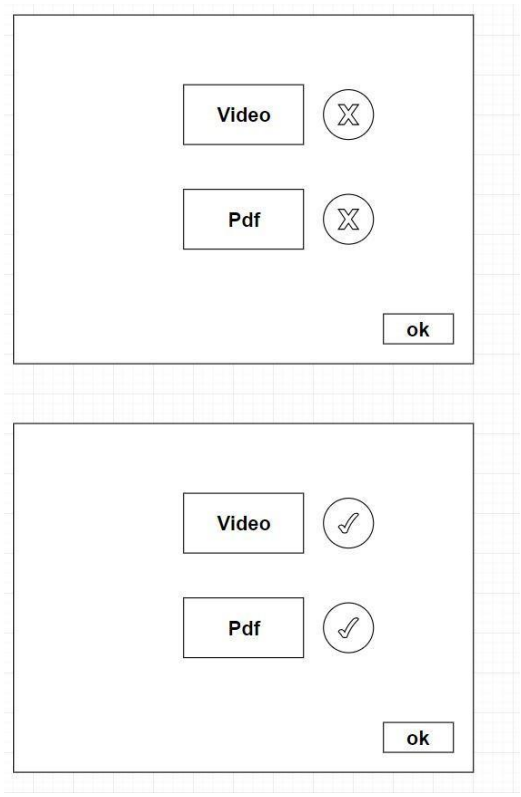


Abbildung 13: GUI nach AutoCut-Button

Nun muss das Tafelvideo einmal analysiert werden und setzt zwei Timestamps einmal für den Visualisermodus und einmal für den Tafelmodus. Für den Visualisermodus soll der Zeitraum (von bis) aus dem Video festgehalten werden, wenn sich die Person im rechten Rechteck befindet bzw. wenn dort Bewegungen erkannt werden.

Abbildung 14: Visualiser-Rect.

Für den Tafelmodus soll der Zeitraum aus dem Video festgehalten werden, wenn sich der Kopf der Person im Rechteck an der Tafel befindet.

Abbildung 15: Board-Rect.

Die PDF soll in JPG Bilder gesplittet auf einer eigenen Spur liegen und anhand der Länge des Videos gleichmäßig auf der Spur angezeigt werden. (Die einzelnen Bilder sollen wenn unten rechts ausreichend Platz vorhanden ist das aktuelle Tafelvideo als Overlay anzeigen ansonsten soll nur das „Bild“ angezeigt werden) Die gesplittete AudioFile soll auf einer eigenen Spur liegen (gleiche Länge wie Video). Anhand der Timestamps soll in der Videospur nun entweder das Tafelvideo oder das Visualiservideo angezeigt werden als Start und Endpunkt gelten die entsprechenden Timestamps. Fazit: das Tafelvideo muss einmal analysiert werden und Zeiten setzen sowie ob es sich um Tafel oder Visualiser handelt...anhand dieser Daten wird dann das Tafelvideo und Visualiservideo geschnitten und nacheinander in der Timeline eingefügt. Ziel: Es müssen anhand der Videoanalyse Teilvideos erstellt werden, welche dann nacheinander in der Timeline automatisch eingefügt werden. Idee: Klasse AutoCut bekommt das Tafelvideo und analysiert es (2 Methoden einmal Visualiser Persondetection und einmal Tafeldetection Kopfdetection, benötigt Liste). Die Unterklasse Tafel erbt die Timestamps der Tafeldetection und schneidet das Video (Tafel) entsprechend -> einfügen in Liste. Die Unterklasse Visualiser erbt die Timestamps der Visualiserdetection und schneidet das Video (Visualiser) entsprechend -> einfügen in Liste. Am Ende wird aus der Klasse AutoCut die Liste mit den Videos nacheinander in der Timeline eingefügt.

Vorteile

- schnell und effizient
- echter AutoCut

Nachteile

- Implementierung benötigt etwas Zeit
- wenn die Videoqualität zu gering ist kann es fehlerhafte Cuts geben oder allgemein zu Fehlern führen

Konzept 2
In Videopreview wird das hinzugefügte Video angezeigt so das man ein Vergleichsobjekt hat. Hinter

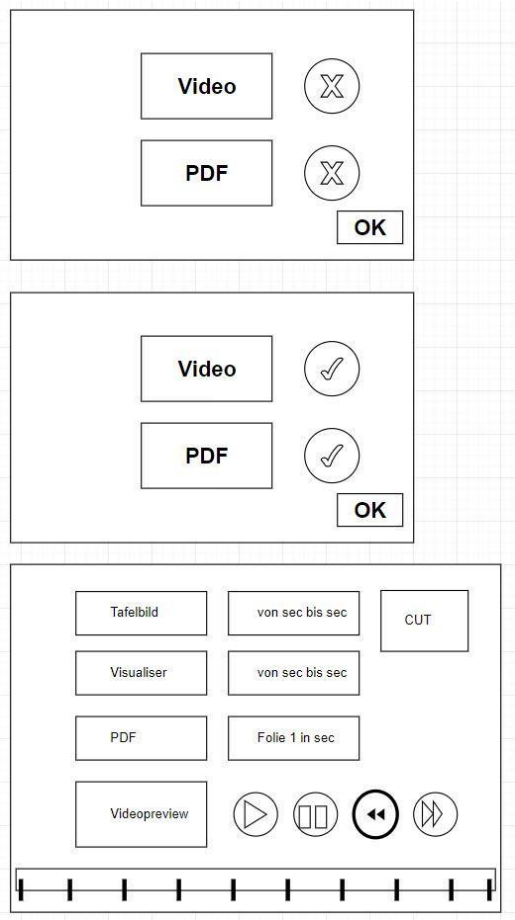


Abbildung 16: GUI-Entwurf

den einzelnen Elementen fügt man ein, von wann bis wann welcher Modi angezeigt werden soll dementsprechend muss es ein extra Fenster mit einer Liste geben (ein add Button fehlt hier noch), entsprechend dieser Liste wird das Video nach dem Cut button dann erstellt und in den Filemanager geladen. Bei bis sollte der maximale Wert die „End"-Länge des Videos sein dementsprechend bei von: 0.

Vorteile

- ist ohne Objectdetection umsetzbar/ schnellere Umsetzung als Konzept 1

Nachteile

- kann ungenau werden durch "menschliches Versagen"

Konzept 3
Im Modus AutoCut bekommt der Filemanager zusätzliche Buttons, welche auf Klick die Audioda-

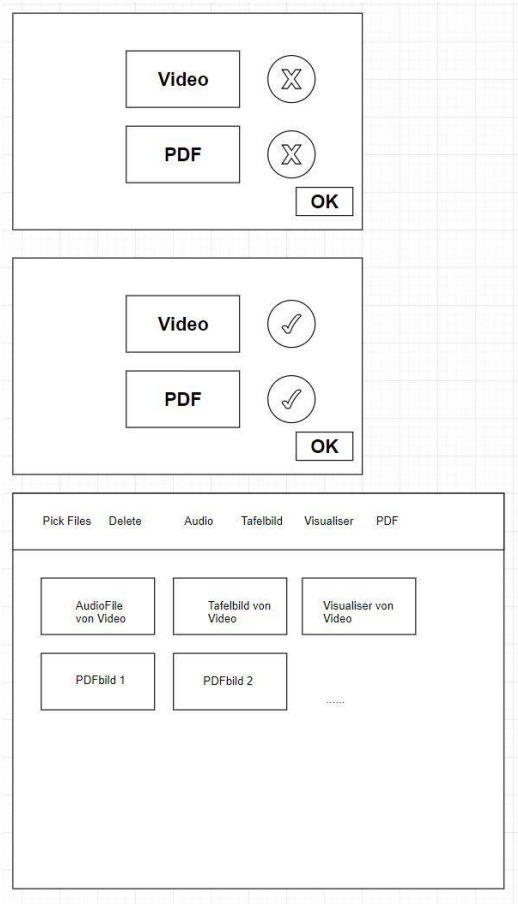


Abbildung 17: veränderte Filemanager nach AutoCut-Modus

tei, Tafelbild -, Visualiser -videos, gesplittete PDF als Bilder/Dateien in den Filemanager lädt.

Vorteile

- einfach implementiert

II.6.5 Nachteile

- hat nicht viel mit AutoCut zu tun

Konzept 4
Beschreibung: Das eingefügte Video wird im „Hintergrund“ berechnet sowie die PDF gesplittet und direkt in der Timeline eingefügt heißt: Danach ist eine Spur belegt mit dem Tafelbildvideo, eine mit

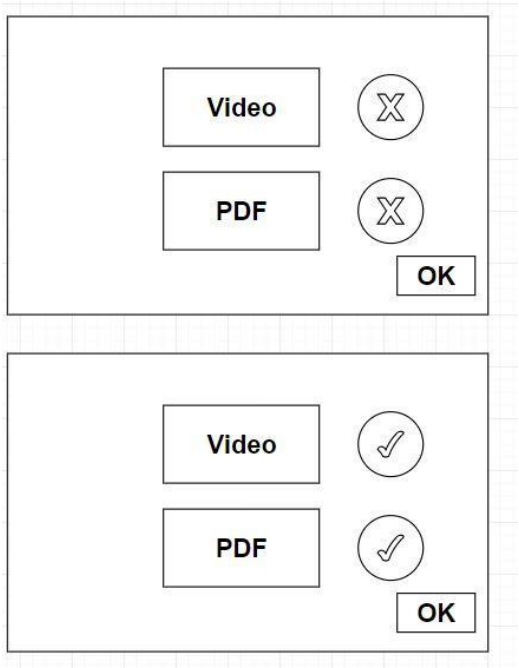


Abbildung 18: GUI-Entwurf nach AutoCut-Button

dem Visualiservideo, eine mit der extrahierten Audiodatei und eine mit den Bildern der PDF. Das Programm kann danach wie im ManCut benutzt werden um das Video zusammenzuschneiden.

Vorteile

- kann schnell implementiert werden

II.6.6 Nachteile

- benötigt viel Nachbearbeitungszeit für den Benutzer

III. Prozess- und Implementationsvorgaben

III.1 Definition of Done

III.1.1 Konzept- und Entwurfsaufgaben

1. Beschreibung und Abbildungen im Produktdokumentations-LaTeX-Dokument
2. im Team diskutiert
3. akzeptiert vom Product-Owner

III.1.2 Programmieraufgaben

1. Funktionalität nach Anforderungen (z.B. Codestyle) implementiert
2. Methoden sind im Code dokumentiert
3. komplizierte Codestellen sind kommentiert
4. Code in Development gepushed

5. Erfolgreich getestet nach Anforderungen (in der Regel Unittests)
6. Code kompiliert und Funktionalität funktioniert auch nach dem Freezen
7. Codereview vom Software-Architekten akzeptiert
8. akzeptiert vom Product-Owner

III.2 Coding Style

Autor: Tim Jeske

Für den Programmierstil gelten die Vorschriften von PEP-8. Im Folgenden werden einige wichtige Regeln beschrieben und erweitert.

III.2.1 Benennung

Alle Namen müssen aus Zeichen aus dem ASCII-Code bestehen. Umlaute dürfen nicht verwendet werden und es sind englische Bezeichnungen zu verwenden. Abkürzungen sollten vermieden werden. Der Name einer Variable sollte Informationen über den Inhalt und weniger über die Art des Objektes enthalten.

Richtig

```
window_width = 600
class VideoEditorView: (...)
```

Falsch

```
überschuss = 5 # Umlaut
win_with = 600 # Abkürzung
var1 = 15 # Variablenname sagt nichts über Bedeutung der Variable aus
bool1 = true # Variable nach Typ benannt, sollte nach Zweck benannt werden
verylongvariable = "something" # Keine Trennung
```

III.2.1.1 Ordner und Dateinamen

Ordner- und Dateinamen sind klein zu schreiben. Wenn es der Lesbarkeit hilft, können Wörter durch Unterstriche getrennt werden. Sie sollten jedoch sparsam eingesetzt werden. Ordner sollten kurze Namen haben.

Richtig

```
videoeditor_controller.py
startview.py
```

Falsch

```
Classes.py # großer Anfangsbuchstabe
start_view.py # Trennung nicht nötig zur Lesbarkeit
```

III.2.1.2 Funktionen- und Variablennamen

Funktionen- und Variablennamen sind klein geschrieben und einzelne Wörter werden mit einem Unterstrich von einander abgetrennt, um die Lesbarkeit zu verbessern. Wörter ohne Bedeutung, wie *my* oder *var* sollen in Bezeichnungen nicht verwendet werden.

Richtig

```
def get_height(): (...)
def draw_elements(): (...)
__private_variable = "Ihr seht mich nicht"
CONSTANT_VALUE = 10
to_long_name = "ok"
```

Falsch

```
# Variblenname sagt nichts über Funktion/Inhalt der Variablen aus
s = 1
str = "Hello World"
# langer Name hat keine Unterstriche
tolongname = "to long"
# CamelCase bei Variablen in der Regel nicht erwünscht
# Name hat keine Aussagekraft
myBox = Box(...)
```

III.2.1.3 Klassennamen

Klassennamen sollen im CamelCase geschrieben werden.

Richtig

```
class FileManager: (...)
class JsonImporter: (...)
```

Falsch

```
class File_Manager: (...)
class filemanager: (...)
class fileManager: (...)
class file_manager: (...)
```

III.2.2 Layout

Für die Einrückung einen Blockes sind 4 Leerzeichen zu benutzen. Tabs sind nicht erlaubt. Eine Methode soll weniger als 15 Zeilen lang sein. Eine Zeile soll maximal 79 Zeichen enthalten. Bei langen Variablennamen sind Ausnahmen von 90 Zeichen möglich. Dies ist zu vermeiden. Die Beschränkung der Zeilenlänge ermöglicht es zwei Codefenster nebeneinander anzuzeigen.

Hinter Operanden sollten keine Zeilenumbrüche gesetzt werden.

Richtig

```
# einfach zu lesen
income = (gross_wages
          + taxable_interest
          + (dividends - qualified_dividends)
          - ira_deduction
          - student_loan_interest)
```

Falsch

```
# Operatoren sind weit entfernt von Operanden
income = (gross_wages +
          taxable_interest +
          (dividends - qualified_dividends) -
          ira_deduction -
          student_loan_interest)
```

III.2.3 Kommentare

Block-Kommentare sollten auf Englisch in ganzen, kurzen Sätzen geschrieben sein. Kommentare werden nur verwendet, wenn die Funktionsweise des Programmcodes ohne Erklärung schwer verständlich ist. Es sind Inline- und Blockkommentare möglich.

Richtig

```
Erklären warum x um 1 erhöht wird.
x = x + 1  # Compensate for border
```

Falsch

```
Offensichtliche Funktionsweise kommentieren.
x = x + 1  # Increment x
```

III.2.4 Dokumentation

Die Dokumentation wird durch drei Anführungszeichen begonnen und beendet. Die Dokumentation ist auf Englisch zu schreiben. Sie besteht aus einer kurzem Beschreibung in wenigen Zeichen in der ersten Zeile, einer längeren Beschreibung in der zweiten und gegeben falls weiteren Zeilen. Für die Dokumentation wird die Epytext-Markup-Language[2] verwendet. Diese verwendet Tags, wie *@type, @param* oder *@return.*

[2] siehe `http://epydoc.sourceforge.net/manual-epytext.html`

Richtig

```
def x_intercept(m, b):
    """
    Return the x intercept of the line M{y=m*x+b}.
    The X{x intercept} of a line is the point at which it
    crosses the x axis (M{y=0}).

    This function can be used in conjuction with L{z_transform} to
    find an arbitrary function's zeros.

    @type  m: number
    @param m: The slope of the line.
    @type  b: number
    @param b: The y intercept of the line.  The X{y intercept} of a line is
              the point at which it crosses the y axis (M{x=0}).
    @rtype:   number
    @return:  the x intercept of the line M{y=m*x+b}.
    """
    return -b/m
```

III.2.5 Commits

Die Nachricht eines Commits sollte als Erstes ein # und die Ticketnummer (ohne MU) stehen und danach ein kurzer Titel/Beschreibung (< 50 Zeichen), welches Feature implementiert wurde. Dabei soll es sich nicht um eine Beschreibung handeln, was ihr gemacht habt, sondern die Funktionalität beschreiben, die ihr implementiert habt. Die Commit-Nachricht soll in Englisch geschrieben werden. In den meisten Fällen sollte der Commit noch eine ausführlichere Beschreibung haben. Bei einem Bugfix kann es sich um eine um eine Beschreibung handeln, was den Bug verursacht hat oder bei einem Feature um eine genauere Beschreibung. Auch diese sollte kurz sein (< 100 Zeichen). Der erste Buchstabe des Titels und der Beschreibung sind groß und beide sollten immer in der Gegenwartsform geschrieben sein. Am Ende befindet sich kein Punkt.

Richtig

```
git commit -m "#142 Refactor GUI-State-System" -m "Remove unnecessary code f.."
git commit -m "#[bugnr] Fix [bugname]" -m "..."
```

Falsch

```
# erster Buchstabe klein und nicht Gegenwartsform
git commit -m "#[bugnr] fixed [bugname]" -m "..."
```

III.2.6 Magische Werte

Magische Werte oder auf Englisch *Magic Numbers* sind Werte, die irgendwo im Quelltext stehen, ohne dass sie einer Variablen zugeordnet werden. Sie heißen Magische Werte, weil man ihre Bedeutung nicht immer nachvollziehen kann. Magische Werte dürfen im Quellcode nicht vorkommen.

Richtig

```
WINDOW_HEIGHT = 800
WINDOW_WIDTH = 600
window = Window(WINDOW_HEIGHT, WINDOW_WIDTH)
```

Falsch

```
window = Window(800, 600)
```

III.3 Zu nutzende Werkzeuge

Für das Erstellen von PyQt5 ui-Dateien ist der QtCreator zu benutzen. Für das gemeinsame Arbeiten an LaTeX-Dokumenten ist Overleaf zu benutzen. Für das Erstellen von UML-Diagrammen ist das kostenlose Programm yEd zu benutzen.

IV. Sprint 1 (26.11. – 13.12.2018)

Autor: Julian Oliver Böttcher

IV.1 Ziel des Sprints

Erstellung einer grundlegenden GUI

IV.2 User-Stories des Sprint-Backlogs

- Preview Videoplayer - Alexander Bonin und Valentin Ackva
- Programmlayout - Felix Dittrich, Maximilian Fornacon, Julian Oliver Böttcher
- Timeline Darstellung - Markus Leupold, Clemens Zwitscher
- FileManager - Jeremy Risy, Johannes Müller

IV.3 Liste der durchgeführten Meetings

- 26.11.2018 - Sprintplanning
- 29.11.2018 - reguläres Treffen
- 3.12.2019 - reguläres Treffen
- 6.12.2019 - reguläres Treffen
- 13.12.2019 - Review & Retrospektive

IV.4 Ergebnisse des Planning-Meetings

Absprache wie Jira benutzt werden soll, Festlegen des Sprint-Ziel, Fokus liegt erst auf Konzepten, erst danach soll der Code geschrieben werden, Festlegen welche technologien hauptsächlich benutzt werden sollen, Aufteilung der Aufgaben unter allen Mitgliedern

IV.5 Aufgewendete Arbeitszeit pro Person und Arbeitspaket

Tabelle 1 enthält eine Übersicht über die aufgewendeten Arbeitszeiten pro Person und Arbeitspaket.

Arbeitspaket	Person	Start	Ende	h	Artefakt
Menüleiste	Julian Böttcher	30.11.18	30.11.18	2	Klasse.java
Filemanager	Jeremy Risy	30.11.18	2.12.18	3,5	–
Video Preview	Valentin Ackva	30.11.18	2.12.18	3,5	preview.py
Video Preview	Alexander Bonin	30.11.18	2.12.18	3,5	preview.py
Startview	Maximilian Fornaçon	30.11.18	2.12.18	4h 5m	start_view.py
Fenster erstellen	Felix Dittrich	30.11.18	2.12.18	4	videoeditor_view.py
Layout erstellen	Maximilian Fornaçon	30.11.18	2.12.18	1	main_window.ui

Tabelle 1: Aufgewendete Arbeitszeit pro Person und Arbeitspaket in Sprint 1

IV.6 Konkrete Code-Qualität im Sprint

Es wurde wenig Code geschrieben. Der vorhandene Code hat gute Qualität und bedarf kein Refactoring.

IV.7 Konkrete Test-Überdeckung im Sprint

Es wurden noch keine Tests geschrieben.

IV.8 Ergebnisse des Reviews

- Endergebnis Erster Sprint: nicht erfüllt
- Vorstellen der Einzelteile nach dem Sprint
- Akzeptieren/Ablehnen dieser mit Verbesserungsvorschlägen

IV.9 Ergebnisse der Retrospektive

- Erstellen einer Pro- und Contra Liste zum Sprint
- Stellungnahme der einzelnen Parteien zum Erfolg/Misserfolg
- PyQt4 Doku lesen könnte helfen

IV.10 Abschließende Einschätzung des Product-Owners

Autor: Sascha Rittig

Auch wenn noch nicht viel Code geschrieben wurde, kann man sich über eine erstaunlich hohe Anwesenheit in allen(!) Meetings erfreuen und einer hohen Aktivität in der Git-Doku. Unsere erste Retrospektive war sehr aufschlussreich und ein paar der dort angesprochene Punkte werden uns hoffentlich in kommenden Sprints helfen effektiver zu arbeiten.

IV.11 Abschließende Einschätzung des Software-Architekten

Es wurde nur wenig Programmcode geschrieben. Die Programmcode-Dokumentation des geschriebenen Codes ist gut. Die Codestyle-Vorgaben wurden beachtet. Der geschriebene Code hat eine gute Qualität. Entwurfsaufgaben waren unfertig und unzureichend. In diesem Sprint hat sich jeder Bachlor mit einem Softwareentwurfsmuster beschäftigt und die Ergebnisse in einem Vortrag mit dem Team geteilt. Es wurde sich außerdem vorbereitend auf das Projekt mit dem Model-View-Presenter-Entwurfsmuster beschäftigt. Es waren bei dem Vortrag leider nicht alle Teammitglieder aufmerksam. Statt einem Vortrag wäre vllt etwas praktisches oder eine Diskussion besser gewesen. Ein Beispiel für das Model-View-Presenter-Entwurfsmuster befindet sich im Git, damit die Bachelor sich dort das Entwurfsmuster nocheinmal ansehen können.

V. Sprint 2 (17.12.2018 – 14.01.2019)

Autor: Maximilian Fornaçon

V.1 Ziel des Sprints

Ziel des Sprints war die Implementierung der grundsätzlichen GUI-Funktionen.

V.2 User-Stories des Sprint-Backlogs

ID	Bezeichnung	Schätzung
MU-1	Produktvision	13
MU-2	Technologierecherche	13
MU-27	Preview Videoplayer	5
MU-29	Programmlayout	8
MU-36	Verschiedene Modi für Video	3
MU-37	Automatische Erkennung der Modi	13
MU-41	Undo/Redo	5
MU-59	Timeline Darstellung	13
MU-61	ShortCuts	5
MU-62	FileManager	13
MU-67	Farbdesign	8
MU-99	Programmeinstellungen	3

V.3 Liste der durchgeführten Meetings

- 17.12.2018 - Planning-Meeting
- 20.12.2018 - normales Meeting
- 10.01.2019 - Review
- 10.01.2019 - Retrospektive

V.4 Ergebnisse des Planning-Meetings

Sprint 2 muss im Vergleich zum ersten Sprint wesentlich produktiver werden. Es soll sich weiterhin mit der GUI befasst werden, darunter auch Aufgaben, die im ersten Sprint nicht fertiggestellt werden konnten. Hier soll ein Schwerpunkt auf der Timeline und dem allgemeinen Layout liegen. Hinzu kommen neue Aufgaben, welche sich mit erster Funktionalität befassen. Aufwandsschätzung soll ab Sprint 3 über eine App ablaufen.

V.5 Aufgewendete Arbeitszeit pro Person und Arbeitspaket

Tabelle 2 enthält eine Übersicht über die aufgewendeten Arbeitszeiten pro Person und Arbeitspaket.

V.6 Konkrete Code-Qualität im Sprint

Die Programmcode-Dokumentation des geschriebenen Codes ist gut. Die Codestyle-Vorgaben wurden beachtet. Der geschriebene Code hat eine gute Qualität.

Arbeitspaket	Person	Start	Ende	h	Artefakt
Filemanager	Jeremy Risy	6.1.19	9.1.19	12	Klasse.java
Video Preview	Valentin Ackva	30.11.18	2.12.18	10	videoeditor-view.py
Video Preview	Alexander Bonin	30.11.18	2.12.18	10	videoeditor-view.py
Entwurf Gesamtlayout	Maximilian Fornaçon	04.01.19	08.01.19	8h 15m	entwürfe.tex
Erkennung Tastendruck	Maximilian Fornaçon	04.01.19	06.01.19	6	shortcut.py, shortcut_loader.py
Timeline	Clemens Zwinzscher	10.01.19	11.01.19	7	timeable_view.py
Timeline	Markus Leupold	11.12.18	06.01.19	30h	view.timeline.timelineview
Implementiere Grundklassen	Felix Dittrich	11.12.18	06.01.19	1	history.py, operation.py, project.py
Settings	Julian Oliver Böttcher	9.1.19	10.1.19	4	settingsview.py

Tabelle 2: Aufgewendete Arbeitszeit pro Person und Arbeitspaket in Sprint 2

V.7 Konkrete Test-Überdeckung im Sprint

Es wurden in diesem Sprint keine Test geschrieben.

V.8 Ergebnisse des Reviews

Es gab Fortschritte der GUI (Videoplayer und Timeline). Desweiteren wurden Grundlagen für die Shortcuts und Undo/Redo geschaffen. Es gibt nun ein Konzept für das Programmdesign erstellt.

V.9 Ergebnisse der Retrospektive

PRO

- Layouttask abgeschlossen
- mehr Erfahrung mit Werkzeugen
- Development funktioniert
- dringende Sachen am Anfang erledigt
- erster sichtbarer Erfolg
- Hilfe bei Problemen
- Anwesenheit

CON

- unabgemeldete Abwesenheit
- viel andere Arbeit
- Feiertage (keine Zeit)
- verlorene Daten durch Computerdefekt
- Probleme mit Schnittstelle QtCreator - Code
- Projektdoku hängt hinterher
- Tasks zu groß (Timeline)
- zu spät im Sprint mit Programmieren angefangen
- viel Einarbeitung (Qt)
- zu wenig kommuniziert
- Sprintziel nicht erreicht
- zu wenige Tasks abgeschlossen
- Konzeptordner im Git fehlt (Aufgabe der Master)

ZIELE

- Abhängigkeit der Tasks beachten

V.10 Abschließende Einschätzung des Product-Owners

Da wir noch am Anfang des Projektes sind, ist es nicht weiter verwunderlich, dass viele noch nicht mit Python vertraut sind. Dies ist momentan für die meisten eine Hürde, die erst überwunden werden muss bevor man mit dem Programmieren anfängt. Trotzdem gab es schon einige Teammitglieder die sich umfassend mit einigen programmiertechnischen Tickets beschäftigt haben. Was sich im nach hinein auch als sehr positiv und hilfreich herausgestellt hat, war der in diesem Sprint angefertigte Entwurf des Gesamtlayouts von Maximilian Fornaço. Er hat neben einem Dummybild ein Farbdesign mit Farbcodes erstellt und alle Button als SVG zur Verfügung gestellt.

V.11 Abschließende Einschätzung des Software-Architekten

Weniger Fortschritt als erwartet. Der geschriebene Programmcode hat eine gute Qualität. Einige Teammitglieder haben gute Arbeit geleistet, aber die Leistung des gesamten Teams war weniger als erwartet. Effektivität beim Programmieren und Aufgaben lösen muss verbessert werden. Mehr Motivation für Bachelor wird gebraucht. Aufgaben scheinen zu komplex zu sein. Kommunikation zwischen Software-Architekt und Team ist wenig. Wenige Fragen der Bachelor zu ihren Aufgaben. Teilweise keine Antworten auf Fragen des Software-Architekten an Bachelor über Telegram.

VI. Sprint 3 (14.01.2019 – 31.01.2019)

Autor: Clemens Zwinzscher

VI.1 Ziel des Sprints

Das Ziel des dritten Sprints war vor allem die Fertigstellung der Hauptkomponenten der GUI. Dazu zählen der Filemanager, die Timeline und die Videopreview. Auch mit dem Speichern von Nutzer-Einstellungen soll begonnen werden (Shortcuts, Farbesign). Außerdem sollte es mit der Doku vorangehen (vor allem Produktvision und Technologierecherche).

VI.2 User-Stories des Sprint-Backlogs

ID	Bezeichnung	Schätzung
MU-1	Produktvision	13
MU-2	Technologierecherche	13
MU-29	Programmlayout	5
MU-36	Verschiedene Modi für Video	3
MU-37	Automatische Erkennung der Modi	3
MU-43	Undo/Redo	2
MU-59	Timeline Darstellung	20
MU-61	Shortcuts	5
MU-62	FileManager	5
MU-67	Farbdesign	8
MU-99	Programmeinstellungen	3
MU-115	Sprintdokumentation	5

VI.3 Liste der durchgeführten Meetings

- 14.01.2019 - Planning-Meeting
- 21.01.2019 - Retrospektive
- 24.01.2019 - normales Meeting
- 28.01.2019 - erste Zwischenstandspräsentation
- 31.01.2019 - Sprint-Review

VI.4 Ergebnisse des Planning-Meetings

Timeline und Filemanager wurden aus dem letzten Sprint übernommen. Vor allem beim Filemanager war es das Ziel, dass dieser in diesem Sprint fertig wird. Die Timeline wird am höchsten geschätzt und es sollen in Zukunft mehr Leute daran arbeiten. Der Dokumentation wird eine erhöhte Priorität gegeben (Sprintdoku, Produktvision, Technologierecherche). Bei der Sprintdokumenation wurde sich darauf geeinigt, dass die ersten beiden Sprints von allen zusammen gemacht werden. Außerdem gibt es einige Designtasks (Farbdesign anpassen, Gesamtlayout).

VI.5 Aufgewendete Arbeitszeit pro Person und Arbeitspaket

Tabelle 3 enthält eine Übersicht über die aufgewendeten Arbeitszeiten pro Person und Arbeitspaket.

Arbeitspaket	Person	Start	Ende	h	Artefakt
Timeline	Clemens Zwinzscher	14.1.19	30.1.19	4h 30m	timeable_view.py
Timeline	Clemens Zwinzscher	14.1.19	30.1.19	7h	track_view.py
Timeline	Markus Leupold	19.1.19	27.1.19	12 h	view.timeline.timelineview
Filemanager	Johannes Müller	14.1.19	30.1.19	–	filemanager.py
Filemanager	Jeremy Risy	14.1.19	30.1.19	4h	filemanager.py
Settings	Julian Oliver Böttcher	12.1.19	22.1.19	3h	settings_view.py
Startview	Maximilian Fornaçon	28.1.19	28.1.19	30m	start_view.py
Entwurf Buttons	Maximilian Fornaçon	14.1.19	14.1.19	45m	entwürfe.tex
Settings	Maximilian Fornaçon	17.1.19	28.1.19	3h 30m	settings.py
Settings	Maximilian Fornaçon	17.1.19	22.1.19	2h 20m	config.py
Folienverarbeitung	Felix Dittrich	12.1.19	30.1.19	5	presentation.py

Tabelle 3: Aufgewendete Arbeitszeit pro Person und Arbeitspaket in Sprint 3

VI.6 Konkrete Code-Qualität im Sprint

VI.7 Konkrete Test-Überdeckung im Sprint

VI.8 Ergebnisse des Reviews

Bis auf eine Person haben alle die Produktvision und Technologierecherche in das Latex-Dokument geschrieben. Der Filemanager funktioniert erstmal, aber es werden noch keine Vorschaubilder angezeigt, sondern nur der Dateiname. Bei der Timeline wurden keine Tasks beendet, aber es ging gut voran.

VI.9 Ergebnisse der Retrospektive

pro:

- Fortschritt Timeline
- Programmierworkshop
- es geht mit Doku voran

contra:

- Pünktlichkeit bei Meetings
- fehlende Motivation
- fehlende Kommunikation
- Doku nicht vollständig
- wenig Zeit (wegen Prüfungen)
- Doku aufwändig
- nicht so viel geschafft wie geplant war
- fehlende Anwesenheit beim Team Building

Maßnahmen:

- sagen wenn man nicht kommen kann
- zu Programmierworkshops kommen
- Doku schreiben
- fester Zeitplan (Zeit zum Programmieren einplanen)
- Tasks kleiner machen, damit Fortschritt spürbar wird
- Teambuilding früher planen

Vergleich mit letzter Retrospektive:

- jetzt mehr Untertasks
- langsam ist man in Qt eingearbeitet
- es hat mehr Kommunikation stattgefunden

VI.10 Abschließende Einschätzung des Product-Owners

Autor: Sascha Rittig

Der gebrachte Arbeitsaufwand ist weiterhin noch ausbaufähig! Außerdem mangelt es mittlerweile auch an Motivation um an der Doku weiter zu arbeiten. Wir haben diesen Sprint einen Programmier-Workshop durchgeführt, der mit viel positiven Feedback angenommen wurde.

VI.11 Abschließende Einschätzung des Softwarearchitekten

Es gabe einige Fortschritte an der GUI. Es gab einigen Fortschritt beim Layout der Timeline. Es wurde ein Programmierworkshop durchgeführt um besseren Codestyle zu fördern und die Arbeit im Team zu verbessern. Der Gesamtfortschritt war nicht so gut. Es fehlte etwas an Motivation, Zeit und die Kommunikation war nicht ausreichend.

VII. Sprint 4 (31.01. – 26.04.2019)

Autor: Felix Dittrich

VII.1 Ziel des Sprints

Das Ziel des vierten Sprints ist die Fertigstellung der Timeline sowie des Filemanagers. Desweiteren sollen erste Funktionen der Videoverarbeitung implementiert werden. Dieser Sprint beinhaltet ausserdem alles, was über die Semesterferien erledigt wurde.

VII.2 User-Stories des Sprint-Backlogs

ID	Bezeichnung	Schätzung
MU-2	Technologierecherche	3
MU-27	Preview Videoplayer	8
MU-29	Programmlayout	5
MU-30	Importieren verschiedener Quellen	13
MU-36	verschiedene Modi für Video	3
MU-37	automatische Erkennung der Modi	8
MU-43	Undo/Redo	2
MU-59	Timeline Darstellun	20
MU-61	ShortCuts	3
MU-62	FileManager	5
MU-67	Farbdesign	8
MU-99	Programmeinstellungen	5
MU-107	Hauptfenster: Maximize Button fehlt	–
MU-112	Projekteinstellungen	5
MU-115	Sprintdokumentation	5
MU-131	Werte im Programm sollen einstellbar sein	–
MU-134	Timeline Logik	2
MU-137	Resource System	3
MU-147	Timeable löschen	5
MU-148	Timeable verschieben	5
MU-149	Timeable zerschneiden	5
MU-151	Timeable Anfang und Ende entfernen	5
MU-153	Timeable hinzufügen	13
MU-154	Preview Schnittstelle	5
MU-155	Preview "Nadel"	3
MU-157	Programm Design	13

VII.3 Liste der durchgeführten Meetings

- 15.04.2019 - Planning-Meeting
- 26.04.2019 - Retrospektive/Sprint-Review

VII.4 Ergebnisse des Planning-Meetings

Die User Stories wurden überarbeitet und übersichtlicher angeordnet. Es wurde vorgestellt, was über die Semesterferien erledigt wurde. Desweiteren wurde festegelegt, was im neuen Sprint erledigt werden muss und wer welche Aufgabe übernimmt. Festgehalten wurde, dass im Sommersemester

mehr Zeit für das Projekt vorhanden ist und diese auch effektiv dafür genutzt werden sollte. Es wurde festgelegt, dass 2 von 4 Terminen als Programmierworkshops anstelle von Meetings genutzt werden.

VII.5 Aufgewendete Arbeitszeit pro Person und Arbeitspaket

Tabelle 4 enthält eine Übersicht über die aufgewendeten Arbeitszeiten pro Person und Arbeitspaket.

VII.6 Konkrete Code-Qualität im Sprint

Die Programmcode-Dokumentation des geschriebenen Codes ist gut. Die Codestyle-Vorgaben wurden beachtet. Der geschriebene Code hat eine gute Qualität.

VII.7 Konkrete Test-Überdeckung im Sprint

Es wurden in diesem Sprint keine Tests durchgeführt.

VII.8 Ergebnisse des Reviews

Der Fortschritt der Timeline und des Filemanagers war sichtbar, zudem wurde einiges an der Logik und Verarbeitung von Daten geändert und hinzugefügt.

VII.9 Ergebnisse der Retrospektive

PRO

- Programmierworkshops
- zusätzliche Treffen ausserhalb der regulären Zeit
- Fortschritt an der Timeline
- Fortschritt allgemein am Projekt

CON

- keine Videopreview
- Tasks die fertig sind können nicht abgeschlossen werden, da Sie niemand testet
- Beschreibung von Tasks ist nicht aussreichend
- Tasks die eine hohe Priorität besitzen wurden zu spät bearbeitet
- die Umsetzung der Meetingprotokolle ist nicht ausreichend

ZIELE

- Die Zusammenarbeit untereinander soll weiter gefördert und erhöht werden
- Die einzelnen Teammitglieder sollen in etwa den selben Arbeitsaufwand leisten

VII.10 Abschließende Einschätzung des Product-Owners

Autor: Sascha Rittig

Man sieht nun, dass es im Projekt Fortschritte gibt, was auch an der deutlich angestiegen Kommunikation innerhalb des Teams liegen dürfte. Die nicht vorhandene Videopreview macht das Testen der Timeline momentan noch sehr mühsam.

Arbeitspaket	Person	Start	Ende	h	Artefakt
Folienverarbeitung	Felix Dittrich	18.2.19	05.4.19	8.5	presentation.py
Platz für Sprecher Erkennung/Einarbeitung OpenCV und Tensorflow, Keras	Felix Dittrich	18.2.19	05.4.19	23	presentation.py
Projekteinstellungen in JSON	Felix Dittrich	18.2.19	05.4.19	1	project settings.py
Video in Tafelvideo und Folien/Visualizervideo teilen	Felix Dittrich	08.4.19	12.4.19	5.5	presentation.py
Anzeige in Filemanager	Felix Dittrich	08.4.19	12.4.19	7.5	filemanager controller.py
Projekteinstellungen	Julian Oliver Böttcher	17.4.19	25.4.19	2.5	projectsettings.py
Schnittstelle Timeline	Julian Oliver Böttcher	25.4.19	26.4.19	2.5	timeline controller.py
Drag und Drop	Felix Dittrich	08.4.19	12.4.19	2.5	filemanager controller.py

Tabelle 4: Aufgewendete Arbeitszeit pro Person und Arbeitspaket in Sprint 4

VII.11 Abschließende Einschätzung des Software-Architekten

Der Codestyle-Richtlinien wurden nicht immer eingehalten. Einige Dateien müssen dahin gehend noch einmal überarbeitet werden. Die Bibliothek Openshot wurde erstmals verwendet und muss in den nächsten Sprints der richtigen Architektur entsprechend eingebaut werden. Der Filemanager wurde nicht nach dem Model-View-Presenter-Entwurfsmuster umgesetzt.

VIII. Sprint 5 (29.04. – 10.05.2019)

Autor: Johannes Müller

VIII.1 Ziel des Sprints

Ziel des Sprints sind die einerseits die Implementierung des automatischen Schnitts und andererseits das Vornehmen diverser Verbesserungen.

VIII.2 User-Stories des Sprint-Backlogs

ID	Bezeichnung	Schätzung
MU-360	Design-Pattern-Korrektur Timeline	1

VIII.3 Liste der durchgeführten Meetings

- 29.04.2019 Sprintstart
- 10.05.2019 Retrospektive/Sprint-Review

VIII.4 Ergebnisse des Planning-Meetings

In Sprint 5 wurde festgehalten, dass wir deutlich produktiver Arbeiten sollen als zuvor. Desweiteren wurden mehrere User-Stories erstellt und aufgeteilt. Dies ist der erste Sprint im Sommersemester.

VIII.5 Aufgewendete Arbeitszeit pro Person und Arbeitspaket

Tabelle 5 enthält eine Übersicht über die aufgewendeten Arbeitszeiten pro Person und Arbeitspaket.

VIII.6 Konkrete Code-Qualität im Sprint

Die Programmcode-Dokumentation des geschriebenen Codes ist gut. Die Codestyle-Vorgaben wurden beachtet. Der geschriebene Code hat eine gute Qualität.

VIII.7 Konkrete Test-Überdeckung im Sprint

Es wurden in diesem Sprint keine Tests durchgeführt.

VIII.8 Ergebnisse des Reviews

Es fand eine 33 Minütige Zwischenstandspresentation statt. Es konnte ein Großteil der Aufgaben bearbeitet werden.

VIII.9 Ergebnisse der Retrospektive

PRO

- Fortschritt am Projekt
- Programmierworkshops
- Zusammenarbeit im Team

CON

Arbeitspaket	Person	Start	Ende	h	Artefakt
Anzeige in Filemanager	Felix Dittrich	29.4.19	10.5.19	18.5	filemanager controller.py
Drag und Drop	Felix Dittrich	29.4.19	10.5.19	2.5	filemanager controller.py
Filepicker nur unterstützte Dateiformate anzeigen	Felix Dittrich	29.4.19	10.5.19	1	filemanager controller.py
mehrere Dateien auf einmal importieren	Felix Dittrich	29.4.19	10.5.19	1	filemanager controller.py
Entfernen von MediaFiles	Felix Dittrich	29.4.19	10.5.19	0.2	filemanager controller.py
Folienverarbeitung	Felix Dittrich	29.4.19	10.5.19	1.5	presentation.py, audio.py, media file.py, slide.py, video.py
Folienverarbeitung	Felix Dittrich	29.4.19	10.5.19	2.5	video splitter.py, removedcode/other.txt
Versuchte Openshot-Installation	Markus Leupold	–	–	$\geq$ 12h	–

Tabelle 5: Aufgewendete Arbeitszeit pro Person und Arbeitspaket in Sprint 5

- Arbeitszeiten von einzelnen Studenten
- Problem mit Openshot auf Windows
- Alte Tasks die nicht bearbeitet wurden

ZIELE

- mehr Zusammenarbeit unter den Studenten

VIII.10 Abschließende Einschätzung des Product-Owners

Autor: Sascha Rittig

In diesem Sprint hat sich hauptsächlich etwas am Filemanager getan. Dieser wurde komplett Refactort. Der Unterschied zwischen den erbrachten Arbeitszeiten der einzelnen Bachelor war in diesem Sprint etwas beunruhigend.

VIII.11 Abschließende Einschätzung des Software-Architekten

Der Codestyle und die Codedokumentation sind gut. Der Filemanager wurde refactored und hat jetzt einen Controller. Es sind in den meisten anderen Bereichen keine großen Veränderungen geschehen.

IX. Sprint 6 (13.05. – 27.05.2019)

Autor: Markus Leupold

IX.1 Ziel des Sprints

Ziel des Sprints sind die einerseits die Implementierung des automatischen Schnitts und andererseits das Vornehmen diverser Verbesserungen.

IX.2 User-Stories des Sprint-Backlogs

ID	Bezeichnung	Schätzung
MU-36	Verschiedene Modi für Video	3
MU-199	Overlay Spur	
MU-27	Preview Videoplayer	8
MU-197	Timeline: Timeable lässt sich scrollen	–
MU-198	Timeline: Bilder und Audio keine Länge	–
MU-192	Filemanager: Fehlerhafte Darstellung bei Selektion	–
MU-145	Hauptfenster: Menüleisten Hover funktioniert nicht	–
MU-193	Filemanager: Videoimport Absturz	–
MU-195	SettingsView: Mehrere Fenster	–
MU-194	SettingsView: Maximize Button	–
MU-196	Timeline: Lücke Dragging von Timeables	–
MU-2	Technologierecherche	3
MU-155	Preview „Nadel”	2
MU-157	Programm Design	13
MU-112	Projekteinstellungen	5
MU-153	Timeable hinzufügen	5
MU-149	Timeable zerschneiden	3
MU-132	Dokumentation Grundlayout und Mechanik der TimelineScrollArea	–
MU-32	Audiospur	8
MU-61	ShortCuts	3
MU-115	Sprintdokumentation	5

IX.3 Liste der durchgeführten Meetings

- 13.05.2019 (Planning)
- 20.05.2019 (Reguläres Treffen)
- 24.05.2019 (Reguläres Treffen)
- 27.05.2019 (Retrospektive)

IX.4 Ergebnisse des Planning-Meetings

Arbeitsaufteilung: Es wurde festgelegt, dass sich das Team in zwei Gruppen zu je vier Personen aufteilt, die sich mit folgenden zwei Hauptaufgaben beschäftigen:

- Liegen gebliebene Dinge und Bugfixes
  - Speicherung des Projekts
  - Bugfixes

- Automatischer Schnitt (Konzept)
  - Welche Modi es gibt/geben sollte
  - Welchen Nutzen der Nutzer aus den Modi zieht
  - Wie der Nutzer in Kontakt mit den Modi kommt

Problemfaktor Openshot: Die Videoschnittbibliothek Openshot bereitet zwei Mitgliedern des Entwicklerteams[3] Probleme, da sie sich nur schwer auf Windows-Systemen installieren lässt. Diese Einschränkung verlangsamt das Projekt insofern, dass die betroffenen Personen seit der Integrierung Openshots ins Projekt nicht mehr daran arbeiten können. Die anderen Entwickler sind nicht betroffen, da sie Linux-Systeme verwenden, auf denen die Installation der Bibliothek sehr einfach ist.
Es wurde deshalb beschlossen, UbiCut vorerst nur für Linux-Systeme zu entwickeln. Die Entwickler, die bisher Windows verwendet haben, steigen für das Projekt auf Linux um. Dies wird zuerst durch Virtualisierung mit Oracle VirtualBox probiert. Sollte es hierbei zu Problemen kommen, kann auch eine Linux-Distribution direkt (z.B. als Dual-Boot) installiert werden.

IX.5 Aufgewendete Arbeitszeit pro Person und Arbeitspaket

Tabelle 6 enthält eine Übersicht über die aufgewendeten Arbeitszeiten pro Person und Arbeitspaket.

IX.6 Konkrete Code-Qualität im Sprint

Es wurden in diesem Sprint keine Code-Reviews durchgeführt. Deshalb gibt es keine offizielle Bewertung der konkreten Codequalität.

Subjektiv bewertet aber der Protokollant diese als mangelhaft – vor allem aufgrund nicht eingehaltener Architekturvorgaben und Designpatterns. Nach der Auffassung des Protokollanten entspricht diese Bewertung auch der Meinung des Softwarearchitekten und während der Retrospektive kam dieses Problem dann zur Sprache.

IX.7 Konkrete Test-Überdeckung im Sprint

Es wurden in diesem Sprint keine Tests durchgeführt.

IX.8 Ergebnisse des Reviews

Es sind viele Fortschritte zu sehen. Die Implementierung der AutoCut-Funktion hat begonnen, nachdem ein Konzept dazu entwickelt war. Die Timeline unterstützt nun die Manipulation der Timeables durch den Nutzer und im Zuge dessen wurde auch die History implementiert. Dadurch können Operationen auch wieder rückgängig gemacht werden. Die TimelineView wurde erweitert durch die Timeneedle und viele Design-Aufgaben wurden abgeschlossen.

IX.9 Ergebnisse der Retrospektive

PRO

- Sichtbare Fortschritte (viel Zustimmung aus dem Team zu diesem Punkt)
- Gute Planungsgespräche für Features

CON

- Persönliche Arbeitsmotivation bzw. wenige Ergebnisse

[3] Die betroffenen Mitglieder sind Markus Leupold und Johannes Müller

Arbeitspaket	Person	Start	Ende	h	Artefakt
Konzept	Felix Dittrich	13.05.19	27.05.19	3.5	–
Bugfixes	Felix Dittrich	13.05.19	27.05.19	0.5	–
Autocut	Felix Dittrich	13.05.19	27.05.19	12	autocut controller.py, videosplitter.py, ...
Versuchte Openshot-Installation	Markus Leupold	–	–	$\geq$ 12h	–

Tabelle 6: Aufgewendete Arbeitszeit pro Person und Arbeitspaket in Sprint 6

- Ungleichverteilung der Arbeitsmenge (in Bezug auf Erledigtes)
- Openshot macht auf den Windows-Systemen Probleme (wichtig)
- Mangelhafte Umsetzung des Codestyles
  - Kommentare (Doc-Kommentare als normale Kommentare)
  - Ungenutzte imports
- Fehlende Codereviews
- Resultat in Bezug auf fehlende Codereviews: Mangel bei
  - Architektur
  - Codestyle
- Viele alte Tasks, die nicht abgeschlossen werden

ZIELE

- Mehr Code-Reviews

IX.10 Abschließende Einschätzung des Product-Owners

Autor: Sascha Rittig

In diesem Sprint wurden im Projekt großer Fortschritt erzielt, der diesmal auch für alle Sichtbar war. Tim und ich haben aus den letzten Retrospektiven gelernt und das Backlog fast komplett verworfen. Wir haben um künftige Sprints effektiver zu gestalten, die meisten Tickets total überarbeitet und ganz neu strukturiert.

IX.11 Abschließende Einschätzung des Software-Architekten

Der Codestyle-Richtlinien wurden nicht immer eingehalten. Einige Dateien müssen dahin gehend noch einmal überarbeitet werden. Manche Kommentare wurden falsch formatiert. Es sind viele ungenutzte Imports vorhanden. Um die Projektsettings umzusetzen wurde der Code der Programmsettings kopiert. Das muss dringend refactored werden, sodass beide Klassen den selben Code mit kleinen Anpassungen verwenden. Dafür empfehle ich eine Parentklasse zu erstellen, die den gemeinsamen Code beinhaltet.

X. Sprint 7 (27.05. – 07.06.2019)

Autor: Alexander Bonin

X.1 Ziel des Sprints

Autocut abschließend implementieren

X.2 User-Stories des Sprint-Backlogs

ID	Bezeichnung	Schätzung
MU-44	Speichern und Laden	13
MU-36	Verschiedene Modi für Video	3
MU-39	Einblendung Sprecher	8
MU-61	Shortcuts	2
MU-224	Spur deaktivieren	8
MU-201	Audiopegel	8
MU-38	Zoom auf Sprecher	8
MU-60	Automatischer Schnitt	8
MU-221	Projektordner	3
MU-242	Spur entfernen/hinzufüchen	5

X.3 Liste der durchgeführten Meetings

- 27.05 Planning
- 03.06 Reguläre Treffen
- 07.06 Review

X.4 Ergebnisse des Planning-Meetings

Valentin will Audiopegel und Maximize-Button machen Clemens macht Speichern/Laden und Automatisch Spuren füllen Felix macht Einblendung Sprecher Max macht Preview Nadel Markus macht Spur entfernen/hinzufügen

X.5 Aufgewendete Arbeitszeit pro Person und Arbeitspaket

Tabelle 7 enthält eine Übersicht über die aufgewendeten Arbeitszeiten pro Person und Arbeitspaket.

X.6 Konkrete Code-Qualität im Sprint

Die Konkrete Code Qualität wurde nicht besprochen oder bewertet, da dies bei diesem Projekt aufgrund des hohen UI Anteils nicht so einfach ist.

X.7 Konkrete Test-Überdeckung im Sprint

Die Konkrete Test-Überdeckung beträgt exakt 0/

Arbeitspaket	Person	Start	Ende	h	Artefakt
AutoCut	Felix Dittrich	27.05.19	07.06.10	5	autocut controller.py
AutoCut-GUI	Felix Dittrich	27.05.19	07.06.10	3.5	autocut view.py, autocut controller.py
Audiospur	Alexander Bonin	27.05.19	07.06.10	6	audiospur.py
	Clemens Zwinzscher	27.05.19	07.06.10	9.5	
	Julian Bötticher	27.05.19	07.06.10	0.75	
	Johannes Müller	27.05.19	07.06.10	0	
	Maximilian Fornacon	27.05.19	07.06.10	9	
	Markus Leupold	27.05.19	07.06.10	0	
	Valentin Ackva	27.05.19	07.06.10	10	

Tabelle 7: Aufgewendete Arbeitszeit pro Person und Arbeitspaket in Sprint 7

X.8 Ergebnisse des Reviews

Welche Task abgeschlossen wurde?

- MU-225 Speichern der Timeline
- MU-226 Speichern der Filemanager MediaFiles in uc.
- MU-229 Laden des Projektes
- MU-251 Spuren Füllen
- MU-121 Fullscreen Button
- MU-209 Slider
- MU-247
- MU-262
- MU-76
- MU-227
- MU-136
- MU-216
- MU-211

X.9 Ergebnisse der Retrospektive

aufgrund von Feiertagen ausgefallen

X.10 Abschließende Einschätzung des Product-Owners

Autor: Sascha Rittig

In diesem Sprint wurden mehrere kritische Funktionen die für die für den sinnvollen Einsatz des Programms unabdinglich sind erfolgreich implementiert. Dazu gehören das Speichern und Laden eines Projektes, sowie das Füllen und Erstellen der Videospuren, mit den aus dem Autocut extrahierten Videoclips.

X.11 Abschließende Einschätzung des Software-Architekten

Der Codestyle und die Dokumentation sind gut. Die Architektur wurde an verschiedenen Stellen nicht eingehalten. Anstatt das Model-View-Presenter-Entwurfsmuster zu verwenden, wurde die Logik direkt in die Views geschrieben. Das muss dringend refactored werden. Die Timeline wurde als Singleton implementiert. Das sollte auch geändert werden. Hier hätte aus Sicht des Softwarearchitekten eine stärkere Kontrolle der Arbeit der Bachelor erfolgen sollen. Die Fehler sollten durch Refactorings in den folgenden Sprinten behoben werden.

XI. Sprint 8 (07.06. – 21.06.2019)

Autor: Valentin Ackva

XI.1 Ziel des Sprints

Ziel des Sprints ist einerseits die ordentlich Implementierung der Modi in der Timeline. Andererseits das Vornehmen diverser Verbesserungen hinsichtlich: Clean up, Refactoring und kleine Tickets abschließen.

XI.2 User-Stories des Sprint-Backlogs

ID	Bezeichnung	Schätzung
MU-44	Speichern und Laden	13
MU-36	Verschiedene Modi für Video	3
MU-39	Einblendung Sprecher	8
MU-38	Zoom auf Sprecher	8
MU-67	Farbdesign	8
MU-256	MVC Refactoring	8
MU-201	Audiopegel	8
MU-238	Logo	2
MU-221	Projektordner	3
MU-156	Gruppierung von Elementen	13
MU-242	Spur entfernen/hinzufüchen	5
MU-42	Gliederungspunkte anzeigen	13
MU-32	Audiospur	8
MU-157	Programm Design	13

XI.3 Liste der durchgeführten Meetings

- 07.06.2019 - Review / Sprintplanung
- 14.06.2019 - Reguläres Treffen
- 17.06.2019 - Reguläres Treffen
- 21.06.2019 - Review

XI.4 Ergebnisse des Planning-Meetings

Personelle Zuständigkeiten:

- Markus:
  Timeline soll mit Frames arbeiten anstatt mit Pixel(im Verhältnis zur Zeitlänge)
- Clemens:
  Gruppieren von Elementen in der Timeline
- Felix:
  Automatischer Zoom auf Sprecher
  Text einfügen
  Filemanger überarbeiten
- Alex:
  Audiospur

- Valentin
  Refactoring
  Timeline mit Player connecten
  Audiopegel anzeigen und einstellen
  Logo

XI.5 Aufgewendete Arbeitszeit pro Person und Arbeitspaket

Tabelle 8 enthält eine Übersicht über die aufgewendeten Arbeitszeiten pro Person und Arbeitspaket.

XI.6 Konkrete Code-Qualität im Sprint

Es wurden in diesem Sprint keine Code-Reviews durchgeführt. Grund hierfür ist die Fokussierung auf die Abschließung der Tickets. Deshalb gibt es keine offizielle Bewertung der konkreten Codequalität.

XI.7 Konkrete Test-Überdeckung im Sprint

Es wurden in diesem Sprint keine Test-Überdeckung durchgeführt.

XI.8 Ergebnisse des Reviews

Folgende Task wurden fertiggestellt:

- MU-222 Projektordner erstellen
- MU-254 Eingabe Feld Projektnam
- MU-39 Einblendung Sprecher
- MU-248 Overlayspur
- MU-239 Logo als Vektorgrafik
- MU-240 Logo an Farbdesign anpassen
- MU-241 Logo in Projekt einbinden
- MU-222 Projektordner erstellen
- MU-112 Projekteinstellungen
- MU-142 Vorhandenes Projekt bei Start auswählen
- MU-252 Audiopegel einstellen

XI.9 Ergebnisse der Retrospektive

Es gab lediglich eine kurze Retrospektive:
PRO

- Jeder ist ein Experte in seinem Gebiet und daher kommen gute Fortschritte

CON

- Zu wenig Kommunikation

ZIELE

- Mehr Programmiertreffen

Arbeitspaket	Person	Start	Ende	h	Artefakt
Automatischer Schnitt	Felix Dittrich	07.06.19	21.06.19	15	board video.py, autocut controller.py, speaker video.py
Audio Timeline	Alexander Bonin	07.06.19	21.06.19	12	operations.py
Gruppieren von Elementen	Clemens Zwinzscher	07.06.19	21.06.19	22	timeable-view.py, timeable-settings
Refactoring	Julian Oliver Böttcher	07.06.19	21.06.19	1	
Shortcut und Autocut	Maximilian Fornaçon	07.06.19	21.06.19	25	config.py, settings-controller.py, autocut controller.py
Scroll Bar	Markus Leupold	07.06.19	21.06.19	7	timeline scroll area.py
Logo, Audiopegel	Valentin Ackva	07.06.19	21.06.19	7	timeable-settings-view.py
	Johannes Müller	07.06.19	21.06.19	0	

Tabelle 8: Aufgewendete Arbeitszeit pro Person und Arbeitspaket in Sprint 8

XI.10 Abschließende Einschätzung des Product-Owners

Autor: Sascha Rittig

In diesem Sprint wurden weitere wichtige zeitkritische Tasks beendet. Nun ist es möglich eine Videos in eine Overlayspur zu ziehen, die das entsprechende Video dann in einem kleinen Kasten unten rechts im gerenderten Video anzeigt. Diese Spur soll im Autocut den Sprecher anzeigen. Weiterhin wurde die Videoanalyse fertig bzw. weiter ausgebaut, so dass es uns nun möglich ist den Sprecher und die Tafel mit einer zufriedenstellenden Präzision zu erkennen/identifizieren.

XI.11 Abschließende Einschätzung des Software-Architekten

Die Architekturfehler aus den vorherigen Sprints wurden nicht refactored und bestehen immer noch. Der Codestyle hat sich sehr verbessert. Die Codedokumentation ist immer noch gut. Außerdem wurde auch viel Dokumentation außerhalb des Codes geschrieben.

XII. Sprint 9 (24.06. – 05.07.2019)

Autor:

XII.1 Ziel des Sprints

XXX

XII.2 User-Stories des Sprint-Backlogs

- x

XII.3 Liste der durchgeführten Meetings

- x

XII.4 Ergebnisse des Planning-Meetings

XII.5 Aufgewendete Arbeitszeit pro Person und Arbeitspaket

Tabelle 9 enthält eine Übersicht über die aufgewendeten Arbeitszeiten pro Person und Arbeitspaket.

XII.6 Konkrete Code-Qualität im Sprint

XII.7 Konkrete Test-Überdeckung im Sprint

XII.8 Ergebnisse des Reviews

XII.9 Ergebnisse der Retrospektive

PRO

- x

CON

- x

ZIELE

- x

XII.10 Abschließende Einschätzung des Product-Owners

Autor: Sascha Rittig

Im letzten Sprint wurde hauptsächlich Clean-up und Bug-fixing betrieben. Außerdem haben wir die Produktmesse vorbereitet und konnten hierfür ein sehr gelungenes Poster entwerfen. Zusätzlich wurde noch das Benutzer- und Anwendungshandbuch fertig.

Arbeitspaket	Person	Start	Ende	h	Artefakt
Bugfixes, etc.	Felix Dittrich	24.06.19	05.07.19	–	–
Doku	Felix Dittrich	24.06.19	05.07.19	3	Latex

Tabelle 9: Aufgewendete Arbeitszeit pro Person und Arbeitspaket in Sprint 9

XII.11 Abschließende Einschätzung des Software-Architekten

Der Codestyle und die Dokumentation sind sehr gut. Die Verständlichkeit des Codes ist an einigen Stellen noch nicht gut genug. Die SettingsView und der SettingsController wurden refactored. Der größte Teil der Logik befindet sich jetzt im Controller, allerdings auch einige Elemente der View, die der Controller benutzt und deren Zugriff eigentlich stattdessen durch z.B. getter und setter der View erfolgen sollte. Refactoring der Projektsettings wurde nicht gemacht. Views können jetzt einfach aktualisiert werden bei Änderungen der Einstellungen.

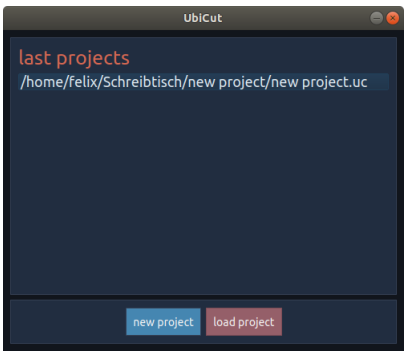
XIII. Dokumentation

XIII.1 Benutzerhandbuch

Autor: Felix Dittrich

XIII.1.0.1 Starten des Programms

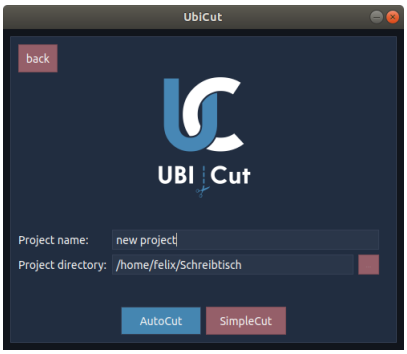
Das Programm „UbiCut“ kann unter Linux über das Terminal oder per Doppelklick auf das Icon geöffnet werden.



Um ein neues Projekt anzulegen betätigt man den Button „new project“.
Um ein bereits vorhandenes Projekt zu Laden betätigt man nach der Auswahl des Projektes den Button „load project“.

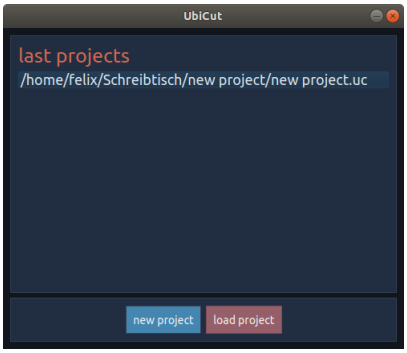
XIII.1.0.2 Neues Projekt

Im project name muss ein Name für das Projekt bzw. den Projektordner angegeben werden.
Unter project directory kann man mithilfe des rechter Hand liegenden Buttons einen Pfad im System auswählen in welchem das neue Projekt/Projektordner erstellt wird.



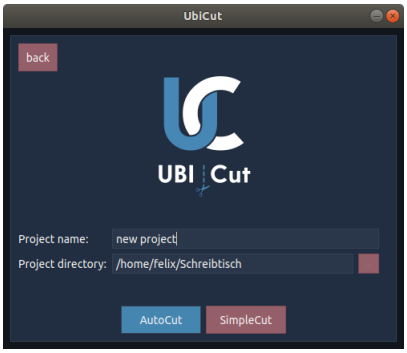
XIII.1.0.3 Projekte laden

Um ein Projekt zu laden muss dieses aus der Liste last projects ausgewählt werden und dann der Button load project betätigt werden.
Eine Alternative innerhalb des Programms bietet die Möglichkeit direkt im Programm zu laden. Unter project -> open kann man eine .uc-Datei auswählen und diese laden.

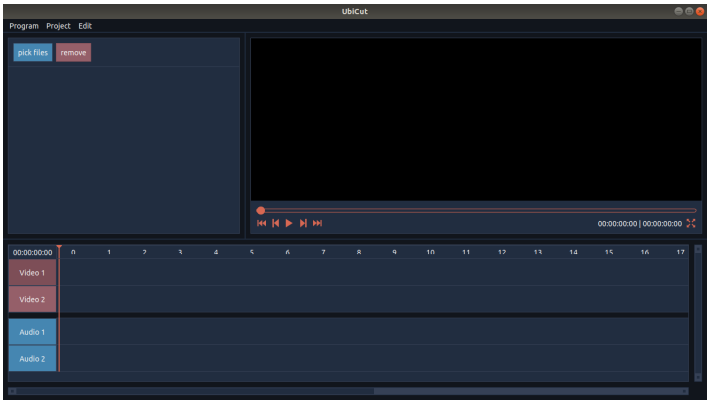


XIII.1.0.4 Einfacher Schnitt

Über den Button SimpleCut kommt man in den normalen Schnittmodus unter welchem es keine vorgeschnittenen Video-/Audio-Dateien gibt.

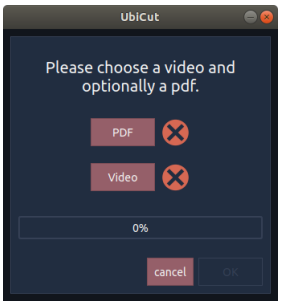


Im Filemanager können nun Dateien des Formates :
.mp4
.jpg
.png
.mp3
.wav
.pdf
über den Plus Button hinzugefügt werden.
Die Auswahl mehrerer Dateien ist möglich.
Über den Ordner Button lassen sich Ordner innerhalb des Filemanagers erstellen und Dateien per Drag and Drop dort hinein ziehen.
Über den Mülleimer Button lassen sich einzelne Elemente aus dem Filemanager entfernen.

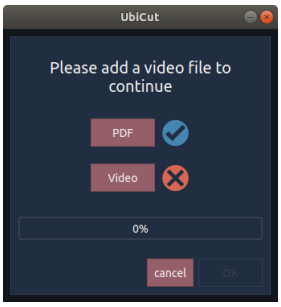


XIII.1.0.5 Automatischer Schnitt

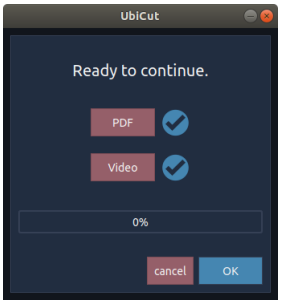
Über den Button AutoCut gelangt man in den automatischen Schnitt Modus welcher Videos zu folgenden Modis erzeugt und automatisch in der Timeline einfügt:
– Visualizermodus(Sprecher arbeitet am Visualizer oder Monitor)
– Tafelmodus (Sprecher zoomt auf die Tafel)
– Sprechermodus(Sprecher wird verfolgt um Ihn als Overlay in den Videos einzufügen)
– Folienmodus(Sprecher zeigt etwas auf den Folien)
Achtung: Es werden ausschließlich die aufgezählten Modis unterstützt ein Zoom auf den Sprecher am Pult wird nicht unterstützt und führt zu einem fehlerhaften Ergebnis des AutoCuts.



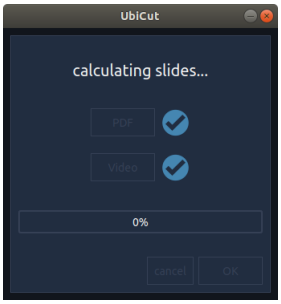
Auswahl eines Videos und optional einer PDF-Datei.



Der Autocut benötigt ein Video um fortzufahren nur eine PDF zu splitten ist nicht möglich.



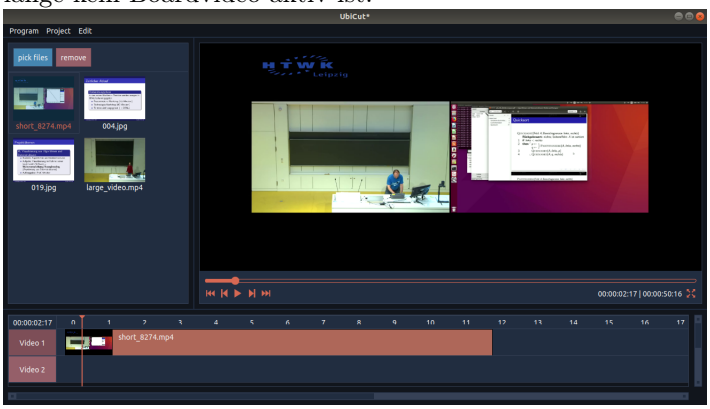
Wenn mindestens ein Video eingefügt wurde kann der Autocut gestartet werden.



Es wird in Form von text angezeigt was gerade abgearbeitet wird und unterhalb wieweit der Fortschritt ist.
Während der Bearbeitung ist kein gewollter Abbruch möglich.

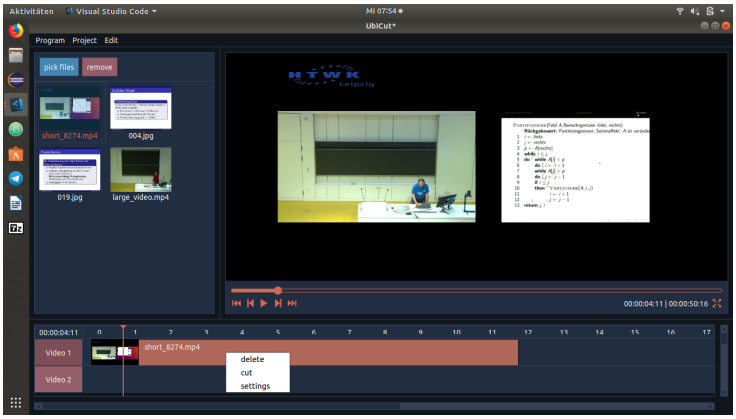
XIII.1.0.6 Allgemein

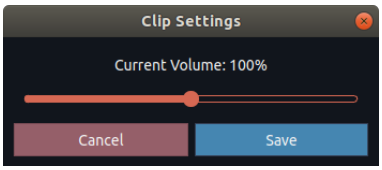
Die Größe der einzelnen Fenster (Previewplayer,Timeline,Filemanager) kann mithilfe der Trennlinien zwischen den Fenstern beliebig angepasst werden.
Die Größe kann auf den Linien per Drag and Drop angepasst werden.
Dateien die im Filemanager liegen können einfach per Drag and Drop in die Timeline gezogen werden.
Die Elemente innerhalb der Timeline lassen sich per Drag and Drop bequem verschieben.
Autocut: Alle Elemente in der Timeline Overlayspur werden im Video unten rechts angezeigt solange kein Boardvideo aktiv ist.



XIII.1.0.7 Schneiden, Löschen, Lautstärke anpassen

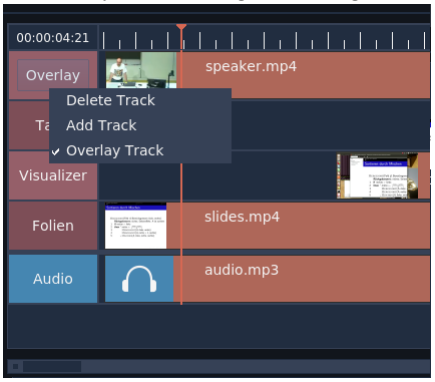
Per rehtsklick auf eine Datei innerhalb der Timeline öffnet sich ein Kontextmenü.
– Löschen: löscht das ausgewählte Element aus der Timeline.
– Cut: schneidet das ausgewählte Element an der Stelle wo das Cut angewendet wird in 2 Teile.
– Cut Timeneedle: schneidet das ausgewählte Element an der Position der Zeitnadel.
– settings: öffnet ein weiteres Fenster in welchem sich die Lautstärke für Audiotracks anpassen lässt.





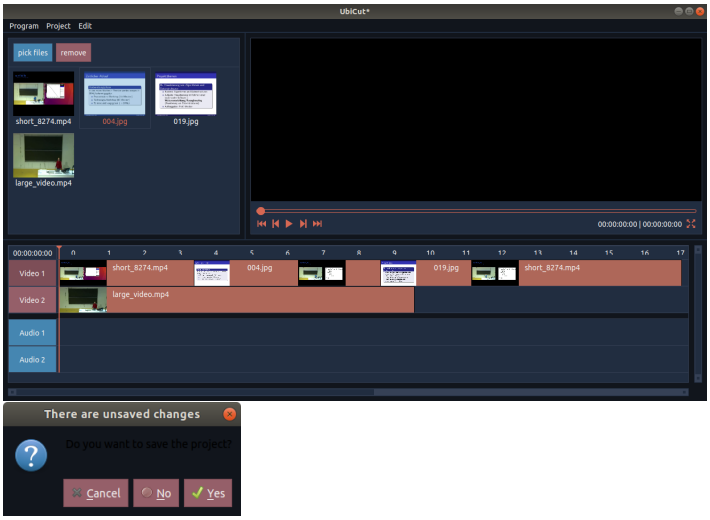
XIII.1.0.8 Spuren hinzufügen, löschen, als Overlayspur festlegen

Durch einen Rechtklick auf einen der Namen der Spuren öffnet sich ein Kontextmenü.
– Delete Track: löscht die ausgewählte Spur.
– Add Track: Öffnet einen Dialog, in dem man auswählen kann, ob eine Video oder Audiospur hinzugefügt werden soll und welchen Namen die Spur haben soll.
– Overlay Track: Legt die ausgewählte Spur als Overlay Spur fest / entfernt sie als Overlay Spur.



XIII.1.0.9 Speichern

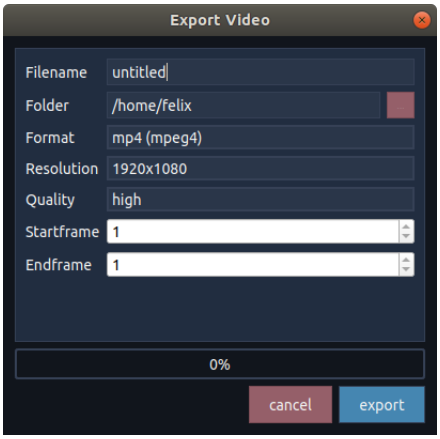
Das aktuelle Projekt kann über project -> save im Projektordner abgespeichert werden oder über project -> save as als Projektordner unabhängige .uc-Datei.
Über dem Programmnamen befindet sich ein kleiner Stern wenn es ungespeicherte Änderungen gibt.



Falls es ungespeicherte Änderungen gibt und man das Programm beendet gibt es eine Abfrage ob man das Programm ohne speichern beenden möchte.

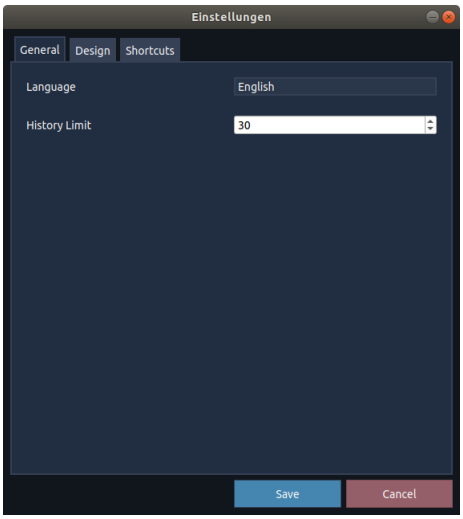
XIII.1.0.10 Exportieren

Nach dem Bearbeiten kann das Video über project -> export exportiert werden.
Darüber hinaus kann in diesem Fenster eingestellt werden:
– der Name des Videos
– der Pfad wo es gespeichert werden soll
– das Foramt in welchem es gespeichert werden soll
– die Auflösung
– die Qualität
– Start sowie Endframe
Ein Fortschrittsbalken unterhalb der Einstellungen zeigt den Fortschritt des exportierens an.



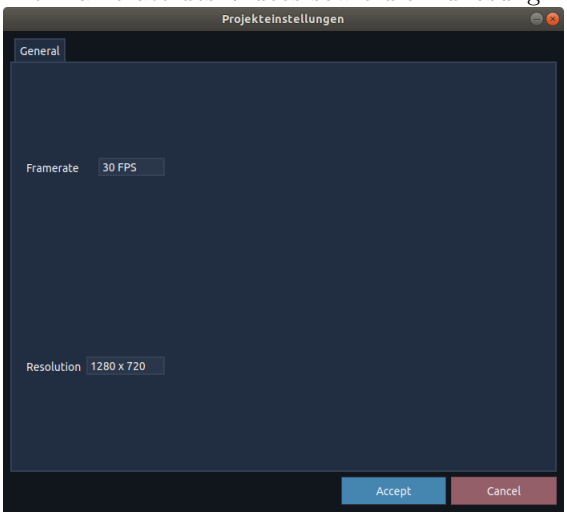
XIII.1.0.11 Programmeinstellungen

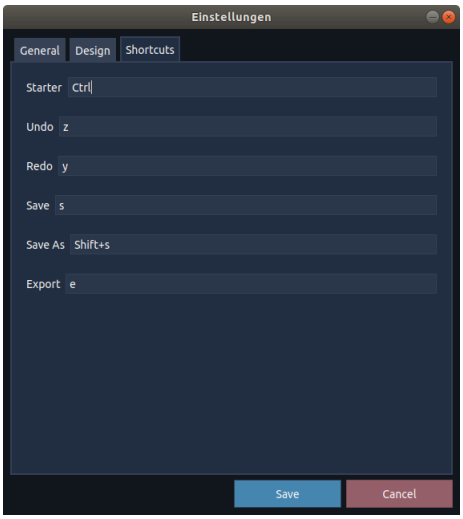
Über program -> settings lassen sich die Programmeinstellungen öffnen.
Unter general lässt sich die Sprache des Programms einstellen (erfordert Programmneustart) sowie die Größe der möglichen Undo/Redo-Operationen (abhängig vom eigenen Arbeitsspeicher).
Unter Design lässt sich das Design des Programms anpassen (in progress!! diese Funktion gibt es leider noch nicht).
Unter Shortcuts lassen sich die einzelnen Shortcuts anpassen.



XIII.1.0.12 Projekteinstellungen

Unter project -> Projectsettings findet man die Projekteinstellungen.
Diese bieten für das aktuell geöffnete Projekt zwei Einstellmöglichkeiten.
Die Framerate des Videos sowie die Auflösung.





XIII.1.0.13 Redo/Undo

Über edit -> Undo lassen sich Schritte rückgängig machen.
Über edit -> redo lassen sich Schritte wiederholen.
Die maximalen Schritte welche man rückgängig machen kann hängt von der Einstellung unter program -> general -> history limit ab.

XIII.1.0.14 Shortcuts

Liste aller Shortcuts nach Standardeinstellung:
Strg + z: Undo
Strg + y: Redo
Strg + s: save
Strg + Shift + s: save as
Strg + e: export
Shortcuts können unter program -> settings -> Shortcuts individuell angepasst werden danach muss das Programm neu gestartet werden um diese zu übernehmen.

XIII.2 Installationsanleitung

Autor: Tim Jeske

XIII.2.1 Linux

XIII.2.1.1 Kompilieren auf einem Linuxsystem

Dieser Abschnitt beschreibt die Einrichtung des Projektes auf einem Linuxsystem und die Erstellung einer unter Linux ausführbaren Datei, sowie das Erstellen einer installierbaren DEB-Datei. Zum Kompilieren wird ein Docker mit Ubuntu 16 verwendet. In einem Dockerfile stehen die Anweisungen zur Installation der Bibliotheken. Will man das Projekt ohne Docker kompilieren, kann man sich an diesen Anweisungen orientieren. Die Installation von Docker wird auf dieser Webseite[4] beschrieben. Zum Einrichten des Dockers geht man in den Ordner `mission-uncuttable/code` und führt das den Befehl `sudo fbs buildvm` aus. Dann wird der Docker mit dem Dockerfile, dass unter `mission-uncuttable/code/src/build/docker/ubuntu` liegt eingerichtet. Mit dem Befehl `sudo fbs runvm` wird der Docker gestartet. Nun kann man das Projekt dort mit `fbs freeze` freezen und anschließend mit `fbs installer` ein `.deb`-File daraus erstellen. Bei der Verwendung des Dockers kann es passieren, dass die Bibliothek Openshot nicht im Virtual-Enviroment verfügbar ist. Normalerweise sollte das behoben werden können, indem man das Virtual-Enviroment neu erstellt mit `python3.6 -m venv venv -system-site-packages`. Weitere Informationen zum Shippen von Projekten mit fbs findet man auf der Manualseite von fbs[5].

XIII.2.1.2 Bekannte Probleme beim Installieren auf Ubuntu

Openshot Fehler beim Importieren
Wenn Openshot beim Importieren den Fehler `Segmentation Fault Error` wirft, können folgende Schritte helfen.

- liegt in `/usr/lib/python3/dist-packages` eine Datei namens `_openshot.so`?
- Falls ja: Lösche aus allen Ordnern `/usr/local/lib/python3.x/dist-packages` die `openshot.py` und die `_openshot.so` Datei
- Falls nein:
  - `sudo nano ~/.bashrc`
  - dort `export LD_LIBRARY_PATH=/usr/local/lib` reinschreiben
  - Terminals neustarten oder jeweils `source ~/.bashrc` ausführen

XIII.2.1.3 Installieren auf einem Linuxsystem

Dieser Abschnitt beschreibt das Installieren der Anwendung mit einer `.deb`-Datei für Linuxsysteme. Die `.deb`-Datei kann entweder mit dem Ubuntu-Softwaremanager installiert werden oder über die Konsole. Für die Installation über die Konsole startet man die Konsole und navigiert zu dem Ordner in dem die `.deb`-Datei liegt. Dann installiert man die `.deb`-Datei mit dem Kommand `sudo dpkg -i UbiCut.deb`. Das Programm wird jetzt installiert und sollte danach ausführbar sein. Ist es nicht ausführbar kann man das Programm unter Ubuntu über die Konsole starten und sehen, warum das Programm nicht startet. Dazu gibt man `cd /opt/UbiCut` ein und startet dann das Programm mit `./UbiCut`. Nun wird der Fehler, der beim Start des Programmes auftritt, angezeigt werden. Handelt es sich bei dem Fehler um einen Fehler mit FFMPEG, dann sollte diese behoben werden in dem man FFMPEG installiert (auf Ubuntu `sudo apt-get install ffmpeg`).

[4] `https://docs.docker.com/install/linux/docker-ce/ubuntu/`
[5] https://build-system.fman.io/manual/

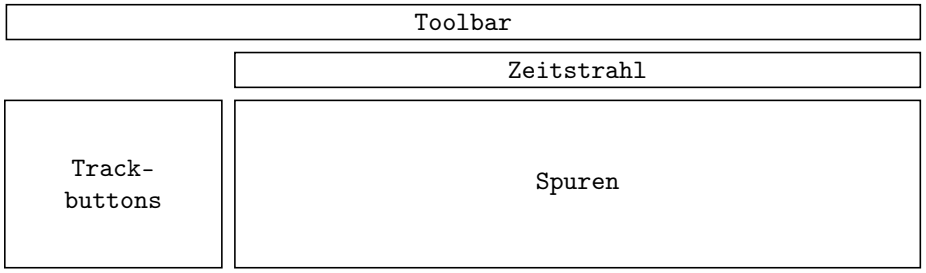


Abbildung 19: Layoutschema der Timelineview

XIII.2.2 Windows

Aus Zeitgründen können wir bisher keine Unterstützung für Windows anbieten, da das Installieren von Openshot auf Windows sehr aufwendig ist.

XIII.3 Entwicklerdokumentation

XIII.3.1 Timelineview

Autor: Markus Leupold

XIII.3.1.1 Aufbau der Timelineview

Anforderungen an den Aufbau und deren Problemfaktor
Hauptsächlich stellt die TimelineView die eigentlichen Spuren dar. Dazu gehören auch die sogenannten Trackbuttons und der Zeitstrahl, die in Abhängigkeit zu den Spuren positioniert werden müssen (siehe Abbildung 19). Dabei gibt es die folgenden Anforderungen:

1. Die Spuren müssen horizontal und vertikal scrollbar sein, da sie im Regelfall zu umfangreich sind, um vollständig gezeigt zu werden.
2. Zu jeder Spur gehört genau ein Trackbutton, der die gleiche vertikale Position besitzt, wie die Spur.
3. Die Ausrichtung der Trackbuttons an den Spuren und die Scrollbarkeit der letzteren machen eine Synchronisierung der vertikalen Position der Spuren und der Trackbuttons notwendig.
4. Der Zeitstrahl muss mit seinen angezeigten Zeitinformationen horizontal an den Spuren ausgerichtet sein.
5. Die Ausrichtung des Zeitstrahls an den Spuren und die Scrollbarkeit der letzteren macht eine Synchronisation der horizontalen Position der Spuren und des Zeitstrahls notwendig.
6. Sowohl die gerade relevanten Trackbuttons als auch der gerade relevante Bereich des Zeitstrahls müssen immer sichtbar sein.

Die Anforderungen 1-5 legen es nahe, die Spuren, die Trackbuttons und den Zeitrahl in einer gemeinsamen Scrollarea unterzubringen, um die Bewegungen auf einfache Art zu synchronisieren. Zwar werden so manche Anforderungen erfüllt, aber auch eine verletzt: Scrollt man die Spuren horizontal, bewegen sich die Trackbuttons aus dem Bild und scrollt man die Spuren vertikal, bewegt sich der Zeitstrahl aus dem Bild.

Es gibt verschiedene Ansätze, dieses Problem zu lösen. Der wahrscheinlich eleganteste aber ist es, drei separate Scrollareas für die drei Bereiche zu verwenden und über einige Observer oder Qt-Signale die Scroll-Positionen der Scroll-Areas zu synchronisieren. Für die Trackbuttons synchronisiert man dann nur die vertikale Bewegung und entsprechend für den Zeitstrahl nur die horizontale

Bewegung. Dazu gesellen sich noch ein paar weitere Abhängigkeiten, auf die später bei den betroffenen Komponenten eingegangen wird.

XIII.3.1.2 Basisklassen der TimelineView

ConnectableScrollArea

Kurzbeschreibung: Erweiterung der QScrollArea zu einer speziellen Scrollarea, die über externe Scrollbars gesteuert werden kann.

Zweck: Steuerung aller drei Scrollareas der TimelineView mit einer gemeinsamen horizontalen bzw. vertikalen Scrollbar.

Methodenübersicht:

`connect_scrollbar()`	Verbinde die Scrollbewegung mit einer Scrollbar.
`disconnect_scrollbar()`	Löse die Verbindung zwischen der Scrollbewegung und einer Scrollbar.

ContentAdjustableConnectableScrollArea

Kurzbeschreibung: Erweiterung der ConnectableScrollArea zu einer ConnectableScrollArea, die ihre eigene Größe an ihren Inhalt anpassen kann.

Zweck: Verhinderung des Scrollens in bestimmten Richtungen. Der Hintergrund ist, dass der Zeitstrahl in der Vertikalen und die Trackbuttons in der Horizontalen immer vollständig zu sehen sein sollen. Mit anderen Worten dürfen sie in diesen Richtungen nicht auf Scrollen angewiesen sein. Dies erreicht man durch eine Anpassung der Größe der Scrollareas an ihren Inhalt.

Methodenübersicht:

`set_adjusting_to_width()`	Aktiviere oder deaktiviere die Breitenanpassung.
`set_adjusting_to_height()`	Aktiviere oder deaktiviere die Höhenanpassung.

SizeLinkableFrame

Kurzbeschreibung: Erweiterung des QFrame zu einem Frame, der seine eigenen Maße mit den Maßen eines anderen SizeLinkableFrame synchronisieren kann. Beide haben dann mit automatischer wechselseitiger Aktualisierung die gleiche Breite und/oder Höhe.

Zweck: Der Frame, der die Trackbuttons enthält, und der Frame, der die Tracks enthält, müssen gleich hoch sein. Das gleiche gilt auch für den Zeitstrahl und die Tracks in der Breite. Nur so können die Scroll-Abhängigkeiten problemlos implementiert werden.

Methodenübersicht:

`link_to_width()`	Synchronisiere die Breite des Frames mit der eines anderen.
`link_to_height()`	Synchronisiere die Höhe des Frames mit der eines anderen.

XIII.3.2 Logik: Modivideos/Autocut

Autor: Felix Dittrich

In diesem Abschnitt wird kurz erklärt, wie das schneiden der einzelnen Modivideos sowie das teilen der PDF funktioniert in Verbindung mit dem Autocut und was es zu beachten gibt.

XIII.3.2.1 Folien

autocut/presentation.py

Kurzbeschreibung: Die Klasse Presentation beeinhaltet eine Methode, die eine PDF-Datei in einzelne .jpg-Bilder teilt und im Projektordner abspeichert.

Methodenübersicht:

`convert_pdf(folder_path,..)`	Teilt die PDF in Ihre einzelnen Bilder und speichert diese im Projektordner ab.

model/data/slide.py

Kurzbeschreibung: Die Klasse Slide enthält eine Liste mit den Pfaden zu den einzelnen erzeugten Bildern der Klasse Presentation.

Methodenübersicht:

`check_color()`	Überprüft ein einzelnes Bild ob in der unteren rechten Ecke genügend Platz vorhanden ist um ein Bild als Overlay einzufügen.

XIII.3.2.2 Videosplitter

autocut/video splitter.py

Kurzbeschreibung: Die Klasse VideoSplitter enthält sämtliche Methoden um das Hauptvideo in die einzelnen Modi wie:

1. Tafelvideo
2. Visualizervideo
3. Folienvideo
4. Audio
5. Sprechervideo

zu zerschneiden und im Projektordner abzuspeichern. Bei Veränderungen ist zu beachten, dass die Werte der einzelnen "Region of Interest"kurz RoI angepasst werden müssen. Diese setzen sich wie folgt zusammen: frame[Anfangswert y-Achse:Endwert y-Achse, Anfangswert x-Achse:Endwert x-Achse] dabei besitzt die linke obere Ecke die Koordinaten (y=0, x=0).
Achtung: Falls diese Werte an ein neues Layout angepasst werden sollte dies für die Auflösung 1280p sowie 1920p vorgenommen werden sowie im Vorraus auf Korrektheit getestet werden, da fehlerhafte Werte das Ergebnis des Autocuts maßgeblich beeinflussen können. Im Projekt liegt unter unittest/python/autocut roi ein Skript zum testen der Bereiche.

Methodenübersicht:

`cut_video(update_progress)`	Erzeugen und speichern der Modivideos (Visualizer,Tafel,Folie).
`cut_zoom_video(update_progress)`	Erzeugen und speichern des Sprechervideos.
`cut_audio_from_video(update_progress)`	Erzeugen und speichern des Audiotracks.

XIII.3.2.3 Tafelvideo

model/data/board video.py

Kurzbeschreibung: Die Klasse BoardVideo enthält das erzeugte Tafelvideo und bietet eine Methode zur Analyse.

Methodenübersicht:

`check_board_area(update_progress)`	Überprüft anhand des schwarz-weiß Frames ob sich das Bild an der Tafel befindet und schreibt bzw. erzeugt entsprechend der Ergebnisse (Sekundenzeitstempel im Video) Teilvideos (Tafel, Sprecher) für die Timeline.

XIII.3.2.4 Visualizervideo

model/data/visualizer video.py

Kurzbeschreibung: Die Klasse VisualizerVideo enthält das erzeugte Visualizervideo und bietet eine Methode zur Analyse.

Methodenübersicht:

`check_visualiser_area(update_progress)`	Überprüft anhand eines kleinen Rechtecks in der oberen linken Ecke(y=0:y1=40, x=0:x1=40) ob der Visualizermodus aktiviert ist oder ob der Folienmodus aktiv ist entsprechend dieser gesetzten Zeitstempel werden Teilvideos für die Timeline erzeugt bzw. gespeichert.

XIII.4 Software-Lizenz

Autor: Sascha Rittig

GNU GPL(General Public License)

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

For precise terms and conditions for copying, distribution and modification
see <https://www.gnu.org/licenses/>.

XIV. Projektabschluss

Autor: Sascha Rittig

XIV.1 Protokoll der Abnahme und Inbetriebnahme beim Kunden

Noch ausstehend. Für Mitte August geplant.

Autor: Sascha Rittig

XIV.2 Präsentation auf der Messe

Unser Stand war erfreulicherweise sehr beliebt und gut besucht. Wir haben auf einem Beamer unser fertiges Autocut-Video präsentiert und konnten auch ein gut entworfenes Poster vorweisen. (Das Poster ist im Git, im Doku Ordner zu finden.) Die Software konnte über drei von uns zur Verfügung gestellten Laptops ausgiebig getestet werden. Wir haben viel positives Feedback erhalten und durften viele interessante Gespräche führen, unter anderen mit den Verantwortlichen für E-Leraning an der HTWK, die das für unser Projekt so wichtige UBIcast System betreuen.

Autor: Sascha Rittig

XIV.3 Abschließende Einschätzung durch Product-Owner

Aus Sicht des Product-Owners war das Projekt einer großer Erfolg. Wir konnten am Ende viel mehr Funktionen umsetzen, als wir am Anfang in unserer schon recht optimistischen Einschätzung vorstellen konnten. Das heißt dass wir in unserer Software eine richtige Videoanalyse durchführen die Gegenstände, Folien und Bewegung im Bild erkennen kann. Außerdem konnte das Team eine praktikable Timeline für den Videoschnitt erstellen, die mit einigen gängigen Videoschnittprogrammen mithalten kann. Zusätzlich war es uns auch möglich einen eigenen Filemanager mit Unterordnern, Vorschaubildern und Drag & Drop zu implementieren, der vollkommen unabhängig von Betriebssystem funktioniert.

Bemerkenswert ist, dass wir einen recht professionellen Workflow zustande bekommen haben, was

das Thema Design angeht. Es wurden zu beginn mehrere Bild-Dummys mit verschiedenen Vorschlägen für das Design angefertigt. Diese wurden dann in der Produktvision im Team besprochen. Aus dieser Diskussion ist dann einer neuer Design-Vorschlag entstanden, dem wir dem Kunden vorgestellt haben. Nach dem der Kunde dem Design zugestimmt hatte, wurde das Farbdesign mit Farbcodes in der Doku festgemacht. Außerdem wurden von Maximilian direkt Entwürfe für alle weiteren Fenster erstellt. Die nötigen Buttons wurden zur direkten Verwendung bereit als SVG, gut geordnet im Git Design Ordner hochgeladen. Das Konzept findet man in groben Zügen beschrieben unter Kapitel II.6.2. Außerdem ist das Design so flexibel angelegt, dass man es im Programm jederzeit wechseln und über das Design-Sheet auch neue Designs hinzufügen kann.

Eine weiterer wichtiger Punkt ist die generelle Motivation und das Interesse der Bachelor am Projekt. Diese waren meiner Meinung nach in unserem Projekt überdurchschnittlich hoch, auch wenn sich das nicht bei jedem in der Programmierzeit widerspiegelt, einige hatten in der Hinsicht Probleme privater Natur. Wir haben alle Treffzeiten im Kalender und auch noch einige darüber hinaus wahrgenommen und es war immer eine hohe Anwesenheit und Gesprächsbereitschaft zu verzeichnen. Ich denke wir konnten in einer stetig steigenden Kurve eine gutes Teamfeeling aufbauen, sei es durch gemeinsame/sichtbare Erfolge im Projekt oder durch ein paar schöne Teambuilding Treffen. Die Retrospektiven in denen sich jeder Aussprechen konnte haben sicher auch ihren Teil dazu beigetragen. Ich bin froh der Project-Owner von diesem Team geworden zu sein.

Unser Kunde hat das Programm bisher auch positiv wahrgenommen und zieht vielleicht in Erwägung es auch zu benutzen, was ich auch schon als Erfolg sehe. Das Programm erfüllt die meisten anfänglich gestellten Anforderungen. Die wichtigen Anforderungen, also die mit denen das Programm für den Verwendungszweck überhaupt genutzt werden kann sind alle erfüllt. Funktionen an die wir gedacht und schon gearbeitet haben, aber aus Zeitlichen gründen nicht mehr umsetzen konnten waren ein Dateifilter für Filemanager, Text Einblendungen wie zum Beispiel in Form von Gliederungspunkten, eine Zoomfunktion für Timeline und Gruppierungsmöglichkeiten für Video- und Audio-Clips. Diese Funktionen waren am Ende des Projektes noch in Arbeit haben es aber nicht ins Endprodukt geschafft. Ich denke nicht, dass es sich lohnt das Projekt noch ein weiteres Jahr im Softwareprojekt laufen zu lassen, da die zusätzlichen Funktionen die man noch hinzufügen könnte schon angefangen und zu dem nicht mehr so aufwendig sind. Tatsächlich könnte das Projekt ein Refactoring vertragen, andererseits ist es jedoch in Python geschrieben was ein Refactoring für *projektfremde* Programmierer nicht wirklich einfach macht.

Ich denke das Softwareprojekt war eine sehr sinnvolle Veranstaltung für mich, da ich wirklich viel gelernt habe was Projektmanagement angeht, besonders in Hinsicht auf Organisation und Kommunikation im Team. Wir mussten zum Beispiel einmal das komplette Backlog verwerfen, neue Task schreiben und neu Strukturieren, da es einfach zu unübersichtlich wurde und nur noch zu große nichtssagende Task überall rumlagen. Solche Dinge würde ich in meinem nächsten Projekt von Anfang an anders angehen.

XIV.4 Abschließende Einschätzung durch Software-Architekt

Die Codestyle-Vorgaben wurden eingehalten. Der Code ist gut leserlich und strukturiert geschrieben. Einige Dateien enthalten etwas viele Codezeilen. Die Projektordnerstruktur ist übersichtlich. Die Architektur In den letzten Wochen des Softwareprojektes wurde noch sehr viel Programmcode geschrieben, was dazu geführt hat, dass einige Architekturverletzungen entstanden sind. Wir haben uns dann entschieden diese durch Refactorings in den folgenden Sprints wieder zu beheben. Durch Zeitmangel, war es dann allerdings nur möglich wenige der Refactorings durchzuführen. Das hat zur Folge, dass das Model-View-Presenter Pattern an vielen Stellen verletzt wurde und große Teile der Logik in der View implementiert wurden, wo sie nicht hingehören. Auch an Stellen, wo schon

refactored wurde, gibt es noch nicht genügend Trennung der Bereiche, um eine gute Testbarkeit zu ermöglichen. Außerdem wurden Views der Timeline als Singleton implementiert, um Zugriffe zu vereinfachen und jetzt ist es schwierig, diesen Schritt rückgängig zu machen, da viele Klassen auf diese View zugreifen. Sollte das Projekt weitergeführt werden, empfehle ich zuerst ein Refactoring der Architektur im Bereich Model-View-Presenter zumachen. Der Programmcode ist in Ordnung und steht häufig nur an den falschen Stellen. Außerdem sollte die Timeline refactored werden, da diese teilweise sehr unübersichtlich wird und sehr viel Code umfasst.

In dem Projekt wurden außer einem Beispieltest keine Tests geschrieben. Das Schreiben von Tests war Anfangs schwierig, wegen dem großem Anteil der GUI und wurde auch durch einige Architekurverletzungen erschwert. Daher wurde das Schreiben der Tests in die letzten Sprints verschoben. Das Team hat sich in den letzten Sprints dann entschieden andere wichtige Aufgaben zu priorisieren und die Tests wegzulassen. Das Fehlen von Tests führt zu einen hohen Fehleranfälligkeit und sollte bei Weiterführung des Projektes nachgeholt werden. Durch die Bibliothek lässt sich das Programm sehr einfach freezen und an den Nutzer bringen. Es gibt allerdings Probleme, wenn der Benutzer ein anderes Betriebssystem als die Entwickler benutzen. Dann muss das Projekt auf dem Zielbetriebssystem eingerichtet und dort gefreezed werden. Das erscheint erstmal einfach. Über eine Virtuelle Maschine kann das erforderliche Betriebssystem installiert werden. Allerdings gab es viele Probleme beim Einrichten unserer Software, was an verwendeten Bibliotheken lag und an inkompatiblen Pythonversionen. Die Bibliothek fbs hat die Funktion das Programm auch über Docker zu freezen, was den ganzen Prozess sehr vereinfacht. Das Problem hierbei ist, dass die Bibliothek Openshot dafür im Docker über ein Dockerscript installiert werden muss, was nicht vollständig geklappt hat.