# IMPLEMENTIERUNGSHEFT

## Multimediatool zum Testen von Videoencodern

Johannes Werner, Noel Schuhmacher, Sascha Rapp,
Simon Grafenhorst, Carina Weber, Jan Benedikt Schwarz

27. Februar 2016

# Inhaltsverzeichnis

# 1 Einleitung

In der Implementierungsphase ist es das Ziel, die Planung aus den zwei vorhergehenden Phasen in Code umzusetzen. Dieses Dokument zeigt dabei den Verlauf der Phase und entstandene Änderungen sowohl an der geplanten Funkionalität als auch am Entwurf auf.

**2 Entwurfsänderungen** In diesem Kapitel werden alle Änderungen, die am Entwurf gemacht wurden, nacheinander mit einer kurzen Begründung aufgelistet. Die Klassen wurden nach Vererbunghierachien und nach Paketen in Unterkapitel sortiert, um das Auffinden einzelner Klassen zu erleichtern.

**3 Ergebnis** Im ersten Teil dieses Kapitels wird der zeitliche Verlauf der Phase kurz dargestellt. Dannach werden alle Änderungen, die bereits im vorigen Kapitel aufgelistet wurden, auf einer gröberen Ebene dargestellt und die Gründe für die größten Änderungen erläutert. Dannach wird auf die Unterschiede zwischen den im Pflichtenheft gemachten Angaben und dem fertigen Programm eingangen. Angefangen wird mit der GUI, über weggelassene Funktionalität hin zu zusätzlich implementierter Funktionalität.

**4 Testüberdeckung** Dieses Kapitel gibt einen kurzen Einblick in die Testüberdeckung des Codes.

# 2 Entwurfsänderungen

## 2.1 Legende

**Aufbau eines Eintrags:**

`{+,-,#} (Sichtbarkeitsbereich) ...`

`(optional) alter Ausdruck => neuer Ausdruck`

Grund der Änderung.

**Änderungszeichen:**

+ –> Hinzugefügt

\- –> Entfernt

\# –> Geändert

**Sichtbarkeitsbereiche:**

- public
- public slots
- signals
- protected
- private slots
- private

## 2.2 Allgemeines

- Korrigierte Rechtschreibfehler in den Klassen-/Methodennamen werden nicht dokumentiert.
- Allen Attributen wird ein Unterstrich zur bessern Lesbarkeit des Codes angehängt.
- Gui-Klassen werden folgende Methoden bei Bedarf hinzugefügt:

  ```
  + (private) void createUi()
  + (private) void connectActions()
  ```

  Diese Methoden dienen allein der Lesbarkeit des Codes.
- Änderungen an den folgenden Methodenspezifizierern werden nicht dokumentiert:

  ```
  const
  noexcept
  ```

- Destruktoren werden in allen Klassen als implzit gegeben betrachtet und bei Bedarf implementiert.

## 2.3 Entfernte Klassen

### 2.3.1 ApplyFilter

Wurde entfernt, da das Anwenden der Filter nicht Rückgängig gemacht werden kann.

### 2.3.2 BlendingFilter

Wurde entfernt, da der Filter im Konflikt mit der Filtervorschau Funktionalität stand.

### 2.3.3 ZoomFilter

Wurde entfernt, da ein Bug in libavfilter auftrat, aufgrunddessen das Bild falsch skaliet wurde.

## 2.4 Umbenannte Klassen

### 2.4.1 BlackWhiteFilter

Wurde umbenannt in GrayscaleFilter.
Korrekte Übersetzung ins Englische.

## 2.5 Hinzugefügte Klassen

### 2.5.1 ChangeFilter

War implementierungstechnisch kein Mehraufwand.

### 2.5.2 GraphVideo

Um Rgb Histogramme für Videos anzeigen zu können.

### 2.5.3 ResultSaver

Trennen vom Speichern des Projektes und der Ergebnisse.

### 2.5.4 VideoSaver

Um encodierte Videos speichern zu können.

### 2.5.5 AnalysisGraph

Enum für bessere Schnittstelle.

### 2.5.6 AnalysisVideo

Enum für bessere Schnittstelle.

### 2.5.7 GraphCalculator

Um das Berrechnen vom Anzeigen zu trennen.

### 2.5.8 GraphPlayer

Um Rgb Histogramme abspielen zu können.

## 2.6 Filterkofigurationsboxen

### 2.6.1 Basisklasse: FilterConfigurationBox

```
+ (public) static std::unique_ptr<FilterConfigurationBox>
CreateConfigurationBox(FilterTab& filterTab, Model::Filter& filter)
```

Fabrikmethode zur Erstellung einer FilterConfigurationBox für einen bestimmten Filter.

```
+ (public) void setFilterTab(FilterTab& filterTab)
+ (public) void setFilterIndex(std::size_t index)
+ (private) std::size_t index
+ (private) FilterTab* filterTab
```

Dass der UndoCommand zum Rückgaengig machen der Änderung am Filter korrekt erstellt werden kann.

```
+ (public) virtaul void updateUi()
```

Um der FilterConfigurationBox bescheid sagen zu können, dass der Filter sich ausserhalb der Box geändert hat.

```
+ (protected) void updatePreview()
```

Wird von der erbenden Klasse aufgerufen, um die Vorschau entsprechend den Änderungen des Benutzers anzupassen.

```
+ (protected) void updateTempFilter();
```

Wird von der erbenden Klasse aufgerufen, um den temporären Filter auf den Stand des originalen Filters zu bringen.

```
+ (private slots) applyFilter()
+ (private slots) resetFilter()
```

Slots für die entsprechenden Buttons.

```
+ (private) static QImage& getDefaultImage()
+ (private) static std::unique_ptr<QImage> defaultImage
```

Stellt das Standardbild für die Vorschau zur Verfuegung.

```
+ (protected) std::unique_ptr<Model::Filter> tempFilter
```

Temporäres Filterobjekt, auf welches die erbende Klasse die Änderungen des Benutzers anwendet.

```
+ (protected) QScrollArea* filterOptionsArea
+ (private) std::unique_ptr<QImage>
+ (private) FrameView* filterPreview
+ (private) QLabel* label_filter
+ (private) QPushButton* button_apply
+ (private) QPushButton* button_reset
```

Steuerung der Gui.

### 2.6.2 Allgemein erbende Klassen

`+ (protected) void updateUi()`

Geerbt aus der Basisklasse.

`+ (private) void createFilterOptions()`

Erstellt die Gui.

`+ (private slots) ...`

Slots fuer die Gui-Elemente.

`+ (private) ...`

Attribute zur Steuerung der Gui.

## 2.7 Filter

### 2.7.1 Basisklasse: Filter

```
+ (public) static std::unique_ptr<Filter> CreateFilter(QString filtername)
```

Fabrikmethode um einen Filter aufgrund des Namens zu erstellen.

```
# (public) virtual std::string getName()
```

```
std::string => QString
```

Bessere Kompatibilität mit den Qt Gui Elementen.

```
+ (public) virtual void restore(QString description)
+ (public) virtual QString getSaveString()
```

Zum einfachen Kopieren von Filtern, wenn die konkrete Klasse nicht bekannt ist und zum einfachen Laden und Speichern der Filter.

### 2.7.2 Allgemein erbende Klassen

```
+ (public) static const QString FILTERNAME
```

Um Filternamen nicht 'nackt' in den Quelltext zu schreiben.

```
+ (public) void QString getName()
+ (public) void restore(QString description)
+ (public) void QString getSaveString()
```

Geerbt aus der Basisklasse.

### 2.7.3 BlurFilter

```
- (public) bool getPreserveEdges()
- (public) bool setPreserveEdges()
- (private) bool preserveEdges
```

Die Option hatte keinen Effekt auf das Ergebnis.

### 2.7.4 BorderFilter

```
# (public) void setColor(QRgb color)
# (public) QRgb getColor()
# (private) QRgb color

QRgb => QColor
```

Komfortablerer Umgang mit QColor.

### 2.7.5 GridFilter

```
# (public) void setColor(QRgb color)
# (public) QRgb getColor()
# (private) QRgb color

QRgb => QColor
```

Komfortablerer Umgang mit QColor.

### 2.7.6 RectangleFilter

```
# (public) void setColor(QRgb color)
# (public) QRgb getColor()
# (private) QRgb color

QRgb => QColor
```

Komfortablerer Umgang mit QColor.

## 2.8 Mementos

### 2.8.1 AnalysisBoxContainerMemento

```
# (public) std::vector<AnalysisBoxMemento> getAnalysisBoxList()

std::vector<AnalysisBoxMemento>
=> std::vector<std::unique_ptr<AnalysisBoxMemento>>&

# (private) std::vector<AnalysisBoxMemento*> mementoList

std::vector<AnalysisBoxMemento*>
=>std::vector<std::unique_ptr<AnalysisBoxMemento>>
```

Unnötiges Kopieren vermeiden und nackte Pointer vermeiden.

```
- (public) void setAnalysisBoxList(std::vector<AnalysisBoxMemento> analysisBoxList)
+ (public) void addMemento(std::unique_ptr<AnalysisBoxMemento> memento)
```

Komfortablere Schnittstelle.

### 2.8.2 AnalysisBoxMemento

```
# (public) QString getVideoPath()
# (public) void setVideoPath(QString videoPath)
# (private) QString videoPath

VideoPath => Path

# (public) Model::Video* getMacroVideo()
# (public) void setMacroVideo(Model::Video* macroVideo)
# (private) Model::Video* macroVideo

Macro => MacroBlock

# (public) Model::Graph getPsnr()
# (public) void setPsnr(Model::Graph psnr)
# (private) Model::Graph psnr

Psnr => PsnrGraph

# (public) Model::Graph getBitrate()
# (public) void setBitrate(Model::Graph Bitrate)
# (private) Model::Graph bitrate

bitrate => bitrateGraph
```

Sauberere Namensgebung.

```
# (public) Model::Graph getPsnrGraph()
# (public) void setPsnrGraph(Model::Graph psnr)
# (private) Model::Graph psnrGraph
# (public) Model::Graph getBitrateGraph()
# (public) void setBitrateGraph(Model::Graph bitrate)
# (private) Model::Graph bitrateGraph

Model::Graph => Model::Graph*
```

Unnötiges Kopieren vermeiden.

```
+ (public) void setEncoder(QString encoder)
+ (public) QString getEncoder()
+ (public) void setAverageBitrate(QString bitrate)
+ (public) QString getAverageBitrate()
+ (public) void setFilename(QString filename)
+ (public) QString getFilename()
+ (public) void setFilesize(QString size)
+ (public) QString getFilesize()
+ (public) void setRedHistogram(Model::GraphVideo* video)
+ (public) Model::GraphVideo* getRedHistogram()
+ (public) void setGreenHistogram(Model::GraphVideo* video)
+ (public) Model::GraphVideo* getGreenHistogram()
+ (public) void setBlueHistogram(Model::GraphVideo* video)
+ (public) Model::GraphVideo* getBlueHistogram()

+ (private) QString encoder
+ (private) QString avergaeBitrate
+ (private) QString filename
+ (private) QString filesize
+ (private) Model::GraphVideo* redHistogram
+ (private) Model::GraphVideo* greenHistogram
+ (private) Model::GraphVideo* blueHistogram
```

Vergessene Attribute, die auch gespeichert werden müssen.

### 2.8.3 AnalysisTabMemento

```
# (public) int getCurrentVideoPosition()
# (public) void setCurrentVideoPosition(int currentVideoPosition)
# (private) int currentVideoPosition

CurrentVideo => Player
```

Sauberere Namensgebung.

```
- (public) int getCurrentlyShownAnalysisVideo()
- (public) void setCurrentlyShownAnalysisVideo(int currentVideo)
- (private) int currentlyShownAnalysisVideo
- (public) float getCurrentSpeed()
- (public) void setCurrentSpeed(float currentSpeed)
- (private) currentSpeed
```

Nicht realisierbar bzw. nicht mehr von nöten.

```
# (public) AnalysisBoxContainerMemento getAnalysisBoxContainerMemento()

AnalysisBoxContainerMemento => AnalysisBoxContainerMemento*

# (public) void setAnalysisBoxContainerMemento(AnalysisBoxContainerMemento memento)

AnalysisBoxContainerMemento => std::unique_ptr<AnalysisBoxContainerMemento>

+ (private) std::unique_ptr<AnalysisBoxContainerMemento> analysisboxMemento
```

Unnötiges Kopieren vermeiden.

```
+ (public) void setRawVideo(Model::YuvVideo* video)
+ (public) void setRawVideo(std::unique_ptr<Model::YuvVideo> video)
+ (public) std::unique_ptr<Model::YuvVideo> releaseVideo()
+ (public) Model::YuvVideo* getRawVideo()
+ (public) GUI::AnalysisGraph getAnalysisGraph()
+ (public) void setAnalysisGraph(GUI::AnalysisGraph graph)
+ (public) GUI::AnalysisVideo getAnalysisVideo()
+ (public) void setAnalysisVideo(GUI::AnalysisVideo video)
+ (private) Model::YuvVideo* rawVideo
+ (private) std::unique_ptr<Model::YuvVideo> ownedVideo
+ (private) GUI::AnalysisGraph graph
+ (private) GUI::AnalysisVideo video
```

Vergessene Attribute, die auch gespeichert werden müssen.

### 2.8.4 FilterTabMemento

```
# (public) bool getWasApplied()
# (public) void setWasApplied(bool wasApplied)
# (private) bool wasApplied

WasApplied => IsFilteredVideoShown

# (public) int getDisplayedFrame()
# (public) void setDisplayedFrame(int displayedFrame)
# (private) int displayedFrame

DisplayedFrame => CurrentFrame
```

Sauberere Namensgebung.

```
# (public) Model::FilterList getFilterList()
# (public) void setFilterList(Model::FilterList list)
# (private) Model::FilterList  filterList

Model::FilterList => std::unique_ptr<Model::FilterList>
```

Abstrakte Klassen können nur als Pointer übergeben werden.

```
- (public) std::string getLoadedFile()
- (public) void setLoadedFile(std::string loadedFile)
- (private) std::string loadedFile
```

Wurden ersetzt durch bessere Methoden.

```
+ (public) void setRawVideo(Model::YuvVideo* rawVideo)
+ (public) void setRawVideo(std::unique_ptr<Model::YuvVideo> video)
+ (public) std::unique_ptr<Model::YuvVideo> releaseVideo()
+ (public) Model::YuvVideo* getRawVideo()
+ (public) void setIsPreviewShown(bool isShown)
+ (public) bool isPreviewShow()
+ (public) void setCurrentFrame(int currentFrame)
+ (public) int getCurrentFrame()
+ (public) void setCurrentlySelectedFilter(int filter)
+ (public) int getCurrentlySelectedFilter()
+ (private) Model::YuvVideo* rawVideo
+ (private) std::unique_ptr<Model::YuvVideo> ownedRawVideo
+ (private) bool isPreviewShown
+ (private) int currentFrame
+ (private) int currentlySelectedFilter
```

Vergessene Attribute, die auch gespeichert werden müssen.

### 2.8.5 MainWindowMemento

```
# (public) AnalysisTabMemento getAnalysisTabMemento()
# (public) FilterTabMemento getFilterTabMemento()

AnalysisTabMemento => AnalysisTabMemento*
FilterTabMemento => FilterTabMemento*

# (public) void setAnalysisTabMemento(AnalysisTabMemento memento)
# (public) void setFilterTabMemento(FilterTabMemento memento)

AnalysisTabMemento => std::unique_ptr<AnalysisTabMemento>
FilterTabMemento => std::unique_ptr<FilterTabMemento>

# (private) AnalysisTabMemento* analysisTab
# (private) FilterTabMemento* filterTab

AnalysisTabMemento* => std::unique_ptr<AnalysisTabMemento>
FilterTabMemento* => std::unique_ptr<FilterTabMemento>
```

Unnötiges Kopieren und nackte Pointer vermeiden.

## 2.9 Undo Commands

### 2.9.1 AddFilter

```
# (public) AddFilter(GUI::FilterTab* filterTab,Model::Filter filter)

(...) => (GUI::FilterTab& filterTab, QString filternam)

- (private) Model::Filter filter
+ (private) QString filtername
+ (private) std::unique_ptr<Memento::FilterTabMemento> memento
```

Benötigte Attribute zum Rückgängig machen haben sich geändert.

### 2.9.2 AddVideo

```
# (public) AddVideo(GUI::AnalysisBoxContainer* container,Model::EncodedVideo video)

Model::EncodedVideo => QString

- (private) Model::EncodedVideo video
+ (private) QString filename
```

Benötigte Attribute zum Rückgängig machen haben sich geändert.

### 2.9.3 FilterReset

```
# (public) FilterReset(GUI::FilterTab* tab,Model::FilterList list)

(...) => (GUI::FilterTab& filterTab)

# (private) Model::FilterList filterList

Model::FilterList => std::unique_ptr<Model::FilterList>

+ (private) std::unique_ptr<Memento::FilterTabMemento> memento
```

Benötigte Attribute zum Rückgängig machen haben sich geändert.

### 2.9.4 LoadAnalysisVideo

```
# (public) LoadAnalysisVideo(GUI::AnalysisTab* tab,
Memento::AnalysisTabMemento memento, Model::YuvVideo video)

(...) => (GUI::AnalysisTab* tab, std::unique_ptr<YuvVideo> video)

# (private) Model::YuvVideo* video

Model::YuvVideo* => std::unique_ptr<YuvVideo>

+ (private) std::unique_ptr<Memento::AnalysisTabMemento> memento
```

Benötigte Attribute zum Rückgängig machen haben sich geändert.

### 2.9.5 LoadFilterConfig

```
# (public) LoadFilterConfig(GUI::FilterTab* tab,
Model::FilterList old,Model::FilterList new)

(...) => (GUI::FilterTab& tab,std::unique_ptr<Model::FilterList> newList)

# (private) Model::Filter:ist* newList

Model::FilterList* => std::unique_ptr<Model::FilterList>

- (private) Model::FilterList* oldList
+ (private) std::unique_ptr<Memento::FilterTabMemento> memento
```

Benötigte Attribute zum Rückgängig machen haben sich geändert.

### 2.9.6 LoadFilterVideo

```
# (public) LoadFilterVideo(GUI::FilterTab* tab,Model::YuvVideo video,
Memento::FilterTabMemento memento)

(...) => (GUI::FilterTab& tab,std::unique_ptr<Model::YuvVideo> video,
std::unique_ptr<Memento::FilterTabMemento> memento)

# (private) Memento::FilterTabMemento* memento
# (private) Model::YuvVideo* video

Memento::FilterTabMemento* => std::unique_ptr<FilterTabMemento>
Model::YuvVideo* => std::unique_ptr<Model::YuvVideo>
```

Benötigte Attribute zum Rückgängig machen haben sich geändert.

### 2.9.7 MoveFilterDown

```
# (public) MoveFilterDown(GUI::FilterTab* tab, int oldIndex, int newIndex)

(...) => (GUI::FilterTab& tab, int index)

- (private) int newIndex
- (private) int oldIndex
+ (private) int index
+ (private) std::unique_ptr<Memento::FilterTabMemento>
```

Benötigte Attribute zum Rückgängig machen haben sich geändert.

### 2.9.8 MoveFilterUp

Genau diesselben Aenderungen wie in der Klasse 'MoveFilterDown'.

### 2.9.9 RemoveFilter

```
# (public) RemoveFilter(GUI::FilterTab* tab,Model::Filter filter, int index)

(...) => (GUI::FilterTab& tab, int index)

# (private) Model::Filter* filter

Model::Filter* => std::unique_ptr<Model::filter>

+ (private) std::unique_ptr<Memento::FilterTabMemento> memento
```

Benötigte Attribute zum Rückgängig machen haben sich geändert.

### 2.9.10 RemoveVideo

```
# (public) RemoveVideo(GUI::AnalysisBoxContainer* container, Model::EncodedVideo video)

(...) => (GUI::AnalysisBoxContainer* container, GUI::AnalysisBox* box)

- (private) Model::EncodedVideo* video
+ (private) int index
+ (private) std::unique_ptr<Memento::AnalysisBoxMemento> memento
```

Benötigte Attribute zum Rückgängig machen haben sich geändert.

### 2.9.11 WriteComment

```
# (public) WriteComment()

() => (GUI::AnalysisBoxContainer* container, int index, QString oldComment,
QString newComment)

+ (private) GUI::AnalysisBoxContainer* container
+ (private) int index
+ (private) QString oldComment
+ (private) QString newComment
```

Benötigte Attribute zum Rückgängig machen haben sich geändert.

## 2.10 Model

### 2.10.1 AVVideo

```
std::unique_ptr<AVFrame> => AVFrame*
int index => std::size_t index
```

Änderungen aufgrund der Implementierung von AVFrame und std::vector.

```
# (public) AVVideo(int fps, int width, int height)
```

```
(...) => (int fps)
```

Die Dimension wird jetzt vom ersten Frame, dass in das Video gesteckt wird, bestimmt.

```
- (public) void insertFrames(int index, std::vector<std::unique_ptr<AVFrame>>& frames)
```

Wird nicht benötigt.

```
+ (public) void setFps(int fps)
```

Um beim Konvertieren die Fps korrekt zu setzen.

```
+ (public) void appendFrame(AVFrame* frame)
```

Komfortablere Schnittstelle.

```
+ (public) bool isComplete()
+ (public) void setIsComplete(bool isComplete)
+ (private) bool isComplete
```

Um feststellen zu können, ob das Video vollständig ist, oder noch auf Frames gewartet werden muss.

### 2.10.2 EncodedVideo

```
- (public) int getFileSize()
- (public) getNumberOfColors()
- (public) void setBitrate (Model::Graph graph)
- (public) void setPsnr (Model::Graph graph)
- (public) void setRedHistogramm (Model::Graph graph)
- (public) void setGreenHistogramm (Model::Graph graph)
- (public) void setBlueHistogramm (Model::Graph graph)
- (public) void setMacroblockVideo (GUI::Video video)
- (public) void setRgbDiffVideo (GUI::Video video)
- (private) int fileSize
- (private) int numberOfColors
- (private) QString codec
```

Wird nicht benötigt.

```
# (public) Model::Graph& getRedHistogram()
# (public) Model::Graph& getGreenHistogram()
# (public) Model::Graph& getBlueHistogram()
# (private) Model::Graph* redHisto
# (private) Model::Graph* greenHisto
# (private) Model::Graph* blueHisto

Model::Graph& => Model::GraphVideo&
Model::Graph* => std::unique_ptr<Model::GraphVideo>
```

Um Rgb Histogramme für das Video speichern zu können.

```
# (public) Model::Graph& getPsnr()

Model::Graph& => Model::Graph*
() => (Model::Video* reference)

# (public) Model::Video& getRgbDiffVideo(Model::Video* reference)

Model::Video& => Model::Video*
```

Wenn die reference null ist und nicht berechnet werden kann, muss ein nullptr zurück gegeben werden.

```
+ (public) int getAverageBitrate()
```

Zusätzliches Attribut zum Anzeigen.

```
# (private) Model::Video* video
# (private) Model::AVVideo* avVideo
# (private) Model::Video* rgbDiffVideo
# (private) Model::Video* macroBlockVideo
# (private) Model::Graph* bitrate
# (private) Model::Graph* psnr

Model::Video* => std::unique_ptr<Model::Video>
Model::AVVideo* => std::unique_ptr<Model::AVVideo>
Model::Graph* => std::unique_ptr<Model::Graph>

- (private) Model::Video* displayVideo
```

Nackte Pointer vermeiden.

```
+ (private) std::unique_ptr<Utility::VideoLoader> loader
+ (private)  std::unique_ptr<Utility::BitrateCalculator> bitrateCalculator
+ (private) std::unique_ptr<Utility::RGBHistogramCalculator> rgbHistoCalculator
+ (private) std::unique_ptr<Utility::PsnrCalculator> psnrCalculator
+ (private) std::unique_ptr<Utility::RGBDifferenceCalculator> rgbDiffCalculator
+ (private) std::unique_ptr<Utility::VideoConverter> macroblockConverter
+ (private) std::unique_ptr<Utility::VideoConverter> videoConverter
+ (private) std::unique_ptr<Utility::VideoLoader> macroblockLoader
+ (private) void calculateHistograms()
```

Attribute zur Berrechnung der einzelnen Videos und Graphen.

### 2.10.3 FilterList

```
- (public) Filter* getFilterByName(std::string name)
- (public) int getIndex(std::string name)
- (public) void addFilter(std::string name, int index)
- (public) void removeFilter(std::string name)
```

Schlechte Schnittstelle.

```
+ (public) std::unique_ptr<Filter> removeFilter(std::size_t index)
+ (public) Filter* appendFilter(QString name)
+ (public) void insertFilter(std::unique_ptr<Filter> filter, std::size_t index)
+ (public) std::size_t getSize()

# (public) void moveFilter(int oldPos, int newPos)

int => std::size_t
```

Bessere Schnittstelle.

```
# (private) std::vector<Filter*> filters

Filter => std::unique_ptr<Filter>
```

Nackt Pointer vermeiden.

### 2.10.4 Graph

```
- (public) void cut(int x)
```

Wird nicht benötigt.

```
# (public) void addValue(int x, double y)
# (public) double getValue(int x)
# (public) removeValue(int x)
# (public) int getLength()

addValue => setValue
getLength => getSize
int => std::size_t

+ (public) getBiggestValue()
```

Bessere Schnittstelle.

### 2.10.5 Project

```
# (public) void setMemento(Memento::MainWindowMemento memento)
# (private) Memento::MainWindowMemento* memento

Memento::MainWindoeMemento => std::unique_ptr<Memento::MainWindowMemento>
Memento::MainWindoeMemento* => std::unique_ptr<Memento::MainWindowMemento>
```

Unnötiges Kopieren vermeiden.

```
+ (public) void setName(QString name)
```

Um den Namen des Projektes setzen zu können, welches beim Start des Programms als leeres Projekt erstellt wird.

### 2.10.6 Video

```
# (public) Video(int fps, int width, int height)

(...) => (int fps)
```

Die Dimension wird jetzt vom ersten Frame, dass in das Video gesteckt wird, bestimmt.

```
- (public) void insertFrames(int index, std::vector<std::unique_ptr<QImage>>& frames)
```

Wird nicht benötigt.

```
# (public) QImage* getFrame(int index)
# (public) void insertFrame(std::unique_ptr<QImage> frame, int index)
# (public) void removeFrame(int index)
# (public) int getNumberOfFrames()

int => std::size_t
```

An die Schnittstelle von vector anpassen.

```
+ (public) void setFps(int fps)
+ (public) bool appendFrame(std::unique_ptr<QImage> frame)
```

Schnittstelle erweitern.

```
+ (public) void setIsComplete(bool isComplete)
+ (public) bool isComplete()
+ (private) bool isComplete
```

Um feststellen zu können, ob das Video vollständig ist, oder noch auf Frames gewartet werden muss.

### 2.10.7 YuvVideo

```
# (public) YuvVideo(QString path,Utility::PixelSheme type, int width, int height, int fps)

(...) => (QString path, Utility::YuvType type,
Utility::Compression compression, int width, int height, int fps)

# (private) Utility::PixelSheme yuvType

Utility::PixelSheme => Utility::YuvType
```

Compression wurde im Design vergessen und Pixelsheme und YuvType waren diesselben Klassen.

```
+ (public) int getWidth()
+ (public) int getHeight()
+ (public) int getFps()
```

Getter wurden im Design vergessen.

```
+ (public) GraphVideo& getRedHistogram()
+ (public) GraphVideo& getGreenHistogram()
+ (public) GraphVideo& getBlueHistogram()
+ (private) std::unique_ptr<GraphVideo> redHisto
+ (private) std::unique_ptr<GraphVideo> greenHisto
+ (private) std::unique_ptr<GraphVideo> blueHisto
+ (private) std::unique_ptr<Utility::RGBHistogramCalculator> histogramCalculator
+ (private) void calculateHistograms();
```

Zum Berrechnen von den Histogrammen.

```
- (public) Model::AVVideo& getAvVideo()
- (private) std::unique_ptr<Model::AVVideo> avVideo
```

Wird nicht benötigt.

```
+ (private) std::unique_ptr<Utility::YuvFileReader> fileReader
+ (private) void loadVideo()
```

Zum Laden des Videos.

## 2.11 Utility

### 2.11.1 BitrateCalculator

```
- (public) Model::Graph calculate()
+ (public) void calculate(Model::Graph* target)

+ (private) std::thread calculator
+ (private) bool isRunning
+ (private) Model::Graph* target
+ (private) void calculateP()
```

Threading.

### 2.11.2 FilterApplier

```
# (public) FilterApplier(Model::FilterList& list)

(...) => (Model::FilterList& list, int width, int height, int pixelFormat)
```

Mehr informationen für Ffmpeg benötigt.

```
# (public) void applyToVideo(Model::AVVideo& target, Model::AVVideo& source)

Model::AVVideo => Model::Video

+ (public) void applyToVideo(Model::Video& target, Model::AVVideo& source)

- (private) AVFrame applyToFrame(AVFrame& source)
+ (public) AVFrame* applyToFrame(AVFrame& source)
```

Bessere Schnittstelle.

```
+ (signals) void applyComplete(bool successful)
```

Um die stausbar updaten zu können.

```
+ (private) int width
+ (private) int height
+ (private) int pixelFormat
+ (private) AVFilterGraph* filterGraph
+ (private) AVFilterContext* buffersinkContext
+ (private) AVFilterContext* buffersourceContext
+ (private) std::string filterDescription
+ (private) std::thread applier
+ (private) Model::Video* source
+ (private) Model::AVVideo* source1
+ (private) Model::Video* target
+ (private) bool isRunning
+ (private) void createFilterString()
+ (private) void applyToVideoP()
+ (private) void applyToAVVideoP()
```

Threading und Attribute für Ffmpeg.

### 2.11.3 FilterConfigurationLoader

```
# (public) Model::FilterList getConfiguratin()

Model::FilterList => std::unique_ptr<Model::FilterList>
```

Unnötiges Kopieren vermeiden.

### 2.11.4 FilterConfigurationSaver

```
+ (private) Model::FilterList* filterList
```

Speichern der Liste, die im Konstruktor übergeben wird.

### 2.11.5 ProjectReader

```
# (public) Model::Project readProject()

Model::Project => std::unique_ptr<Model::Project>
```

Unnötiges Kopieren vermeiden.

```
+ (private) QFile file
+ (private) QTextStream dataStream

+ (private) void parseFilterTab(Memento::FilterTabMemento* memento,QString line)
+ (private) void parseAnalysisTab(Memento::AnalysisTabMemento* memento,QString line)
```

Mehr Lesbarkeit des Codes.

### 2.11.6 ProjectWriter

```
# (public) ProjectWriter(Model::Project project)

Model::Project => Model::Project*
```

Unnötiges Kopieren vermeiden.

```
- (public) void saveResults()
```

Wurde in eigene Klasse ausgelagert.

```
+ (private) Model::Project* project
+ (private) std::unique_ptr<QFile> file
+ (private) QTextStream dataStream
```

Attribute zum Speichern benötigt.

### 2.11.7 PsnrCalculator

```
# (public) PsnrCalculator(Model::AVVideo& reference, Model::AVVideo& video)
```

```
Model::AVVideo => Model::Video
```

Bessere Schnittstelle.

```
+ (private) double calculateMeanSquareError(QImage* frame1, QImage* frame2);
```

Wird zum Berrechen benötigt.

```
+ (private) std::thread calculator
+ (private) Model::Graph* target
+ (private) bool isRunning
```

```
+ (private) void calculateP()
```

Threading.

### 2.11.8 RgbDifferenceCalculator

```
# (public) void calculateVideo(Model::Video& target)
```

```
Model::Video& => Model::Video*
```

Bessere Schnittstelle.

```
+ (private) Model::Video* target
+ (private) std::thread calculator
+ (private) bool isRunning
```

```
+ (private) void calculateP()
```

Threading.

### 2.11.9 RGBHistogramCalculator

```
# (public) void calculate()

() => (Model::GraphVideo* targetRed,Model::GraphVideo* targetGreen,
Model::GraphVideo* targetBlue)

+ (private) std::thread calculator
+ (private) bool isRunning
```

Threading.

```
# (private) Model::Graph red
# (private) Model::Graph green
# (private) Model::Graph blue

Model::Graph => Model::GraphVideo*
```

Um Rgb Histogramm für ein ganzes Video berrechnen zu können.

### 2.11.10 VideoLoader

```
# (public) VideoLoader(QString path)

(...) => (QString path, AVDictionary* dict)

+ (private)  AVDictionary *dict_;
```

Um Macroblöcke laden zu könne.

```
- (public) std::unique_ptr<Model::AVVideo> loadVideo()
+ (public) void loadVideo(Model::AVVideo *target)

+ (private) std::thread loader
+ (private) bool isRunning

+ (private) void loadP()
```

Threading.

```
+ (public) QString getCodec()
+ (public) int getAverageBitrate()

+ (private) QString codec
+ (private) int averageBitrate
```

Attribute des Videos.

### 2.11.11 VideoConverter

```
+ (public) VideoConverter(Model::AVVideo* video)
+ (public) VideoConverter(Model::Video* video)

- (public) static std::unique_ptr<Model::AVVideo>
convertVideoToAVVideo(Model::Video& video)
- (public) static std::unique_ptr<Model::Video>
convertAVVideoToVideo(Model::AVVideo& video)

+ (public) void convertAVVideoToVideo(Model::Video* target)
+ (public) void convertVideoToAVVideo(Model::AVVideo* target)
+ (private) Model::Video* video
+ (private) Model::Video* videoTarget
+ (private) Model::AVVideo* avvideo
+ (private) Model::AVVideo* avvideoTarget

+ (private) std::thread converter
+ (private) bool isRunning

+ (private) void convertVideoP()
+ (private) void convertAVVideoP()
```

Threading.

```
# (public) std::unique_ptr<AVFrame> convertImageToAVFrame(Qimage& image)

std::unique_ptr<AVFrame> => AVFrame*

# (public) static std::unique_ptr<QImage> convertAVFrameToImage(AVFrame& frame, int width,

(...) => (AVFrame& frame)
```

AVFrames kann man nicht in Smartpointer packen und die übergebenen Attribute sind bereits in AVFrame enthalten.

```
+ (public) static std::unique_ptr<QImage> convertGraphToImage(
Model::Graph* graph,int width,int height, GUI::GraphCalculator* calculator=0);

+ (public) static std::unique_ptr<Model::Video> convertGraphVideoToVideo(
Model::GraphVideo* video,int width, int height,GUI::GraphCalculator* calculator);
```

Konvertieren Graphen zu Images.

## 2.12 YuvFileReaders

### 2.12.1 Basisklasse: YuvFileReader

```
# (public) YuvFileReader(QString filename, int width, int height)

(...) => (QString filename, int width, int height, int framesize)
```

Um den Buffer alloziieren zu können, in den die Datei gelesen wird.

```
- (public) virtual std::unique_ptr<Model::Video> read()
+ (public) void read(Model::Video* target)

+ (public) void stopReading();
+ (public) bool isRunning();


+ (private) bool isRunning
+ (private) std::thread reader

+ (private) void readP()
```

Threading.

```
- (protected) std::unique_ptr<QByteArray> binaryData
- (protected) std::unique_ptr<Model::Video> video

+ (protected) virtual std::unique_ptr<QImage> parseNextFrame()

+ (protected) unsigned char* binaryData
+ (protected) QDataStream dataStream
+ (protected) QFile file
+ (protected) Model::Video* video
+ (protected) int frameSize
+ (protected) bool* complete
```

Attibute, die zum Einlesen benötigt werden und Verhinderung von Codedopplung.

### 2.12.2 Allgemein erbende Klassen

```
- (public) std::unique_ptr<Model::Video> read()

# (private) std::unique_ptr<QImage> parseNextFrame()

private => protected
```

Geerbt von der Basisklasse.

## 2.13 YuvFileSavers

### 2.13.1 Basisklasse: YuvFileSaver

```
+ (signals) void saveComplete(bool successful,QString filename,int width,int height)
```

Um dem Nutzer bescheid sagen zu können.

```
+ (public) static int RgbToY(QRgb pixel)
+ (public) static int RgbToU(QRgb pixel)
+ (public) static int RgbToV(QRgb pixel)
```

Funktionen zur Konvertierung.

### 2.13.2 Allgemein erbende Klassen

```
+ (private) std::thread safer
+ (private) bool isRunning

+ (private) void saveP()
```

Threading.

### 2.13.3 Yuv420FileSaver

```
- (private) void saveFrame(int index)
```

Nicht benötigt.

### 2.13.4 Yuv422FileSaver

```
+ (private) Compression compression
```

Vergessen im Design.

### 2.13.5 Yuv444FileSaver

```
+ (private) Compression compression
```

Vergessen im Design.

## 2.14 GUI

### 2.14.1 AnalysisBox

```
# (public) Memento::AnalysisBoxMemento getMemento()

Memento::AnalysisBoxMemento => std::unique_ptr<AnalysisBoxMemento>

# (public) void restore(Memento::AnalysisBoxMemento memento)

Memento::AnalysisBoxMemento => Memento::AnanlysisBoxMemento*
```

Unnötiges Kopieren vermeiden.

```
- (public) void setRawVideo(Model::Video* video
- (private) Model::Video rawVideo*

+ (public) void setParentContainer(AnalysisBoxContainer* container)
+ (private) AnalysisBoxContainer* parentContainer
```

Hole das Rohvideo über den Container.

```
- (public) void setAnalysisVideo(Model::EncodedVideo video)
+ (public) void setFile(QString filename)

# (private) Model::EncodedVideo* origVideo

Model::EncodedVideo* => std::unique_ptr<Model::EncodedVideo>
```

Kapsle das Video in der box.

```
# (public) void setControlPanel(GlobalControlPanel* panel)
# (private) GlobalControlPanel* controlPanel

GlobalControlPanel* => std::shared_ptr<GlobalControlPanel>
```

Destructor Abhängigkeiten.

```
- (public) void showRGBDifferenceVideo()
- (public) void showMacroBlockVideo()

+ (public) void showGraph(AnalysisGraph graph)
+ (public) void showAnalysisVideo(AnalysisVideo video)
+ (public) void showAttributes()
```

Bessere Schnittstelle.

```
+ (public) QString getPath()
+ (public) QPlainTextEdit* getCommentBox()
+ (private) QString currentComment
```

Um den WriteComment UndoCommand erstellen zu können.

```
+ (public) void lockUi()
+ (public) void unlockUi()
```

Ui locken um Ergebnisse speichern zu können.

```
+ (private slots) void updateLabels()
+ (private) QTimer timer_updateLabels
```

Um labels so lange zu aktualisieren, bis alle Information angezeigt werden.

```
- (private) int currentlyPlayedVideo
+ (private) GraphWidget* graphWidget
+ (private) QLabel* label_title
+ (private) QLabel* label_filename
+ (private) QLabel* label_filesize
+ (private) QLabel* label_codec
+ (private) QLabel* label_averageBitrate
+ (private) std::unique_ptr<GraphPlayer> graphPlayer
+ (private) std::unique_ptr<GraphCalculator> calculator

# (private) VideoPlayer* plainVideoPlayer
# (private) VideoPlayer* analysisVideoPLayer

VideoPlayer* => std::unique_ptr<VideoPlayer>
```

Zusätzlich für die GUI benötigt.

```
- (private) GraphWidget psnrGraph
- (private) GraphWidget bitrateGraph
- (private) GraphWidget redHistogramm
- (private) GraphWidget blueHistogramm
- (private) GraphWidget greenHistogramm
```

Wird in EncodedVideo gespeichert.

### 2.14.2 AnalysisBoxContainer

```
# (public) Memento::AnalysisBoxContainerMemento getMemento()

Memento::AnalysisBoxContainerMemento =>
 std::unique_ptr<Memento::AnalysisBoxContainerMemento>

# (public) void restore(Memento::AnalysisBoxContainerMemento memento)

Memento::AnalysisBoxContainerMemento => Memento::AnalysisBoxContainerMemento*
```

Unnötiges Kopieren vermeiden.

```
+ (public) void setParentTab(AnalysisTab* parent)
+ (public) AnalysisTab* getParentTab()
+ (private) AnalysisTab* parent
```

Dass sich die Analysisboxen das Rohvideo holen können.

```
- (public) void addVideo(QString path)
- (public) void addVideo(Model::EncodedVideo video)
+ (public) AnalysisBox* appendBox(AnalysisBox* box)

# (public) void removeBox(AnalysisBox& box)

AnalysisBox& => AnalysisBox*
void => int

- (public) void setRawVideo(Model::Video* video)
- (private) Model::Video* video

- (public) void showMacroBlockVideo()
- (public) void showRGBDifferenceVideo()

+ (public) void showGraph(AnalysisGraph graph)
+ (public) void showAnalysisVideo(AnalysisVideo video)
+ (public) void showAttributes()
+ (public) AnalysisGraph getShownGraph()
+ (public) AnalysisVideo getShownVideo()
+ (private) AnalysisGraph currentGraph
+ (private) AnalysisVideo currentVideo

+ (public) void clear()
+ (public) AnalysisBox* insertBox(AnalysisBox* box, std::size_t index)
+ (public) AnalysisBox* getAnalysisBox(std:size_t index)
+ (public) int getIndex(AnalysisBox* box)
+ (public) std::size_t getNumberOfBoxes()
```

Bessere Schnittstelle.

```
+ (private) QVBoxLayout v_boxes
+ (private) QSpacerItem* spacer
+ (private) QPushButton* button_addVideo

+ (private slots) void addVideo()
+ (private) void updateUi()
```

Zusätzlich für die GUI benötigt.

```
# (public) void setControlPanel(GloablControlPanel* panel)
# (private) GlobalControlPanel* panel

GloablControlPanel* => std::shared_ptr<GlobalControlPanel>
```

Destructor Abhängigkeiten.

```
+ (public) void lockUi()
+ (public) void unlockUi()
```

Um Ergebnisse speichern zu können.

### 2.14.3 AnalysisTab

```
# (public) Memento::AnalysisTabMemento getMemento()

Memento::AnalysisTabMemento => std::unique_ptr<Memento::AnalysisTabMemento>

# (public) void restore(Memento::AnalysisTabMemento memento)

Memento::AnalysisTabMemento => Memento::AnalysisTabMemento*
```

Unnötiges Kopieren vermeiden.

```
# (public) void setRawVideo(Model::YuvVideo video)

Model::YuvVideo => Model::YuvVideo*

+ (public) Model::YuvVideo* getRawVideo()
```

Rawvideo wird im UndoCommand gespeichert.

```
+ (public) bool isRawVideoLoaded()
```

Für den Container, ob er ein Video hinzufügen darf.

```
+ (public) void setParentWindow(MainWindow* window)
+ (public) MainWindow* getParentWindow()
+ (private) MainWindow* parentWindow
```

Für die Statusbar.

```
+ (protected) void resizeEvent(QResizeEvent * event)

+ (private slots) void showBitrate()
+ (private slots) void showRedHistogram()
+ (private slots) void showBlueHistogram()
+ (private slots) void showGreenHistogram()
+ (private slots) void showPsnr()
+ (private slots) void showAttributes()
+ (private slots) void analysisVideoChanged(int index)
+ (private slots) void updateLabels()
+ (private slots) void resultSavingFinished()
- (private slots) void addVideo()

+ (private) QPushButton* button_attributes
+ (private)  QPushButton* button_redHistogram
+ (private)  QPushButton* button_blueHistogram
+ (private) QPushButton* button_greenHistogram
+ (private) QPushButton* button_bitrate
+ (private) QPushButton* button_psnr
```

```
+ (private) QVBoxLayout* v_rawVideo
+ (private) GraphWidget* graphWidget
+ (private) QString stylesheet_buttons
+ (private) QString stylesheet_buttonsSelected
+ (private) QLabel* label_filesize
+ (private) QLabel* label_resolution
+ (private) QLabel* label_framesize
+ (private) QPushButton* button_loadnewvideo
- (private) QPushButton button_addVideo
+ (private) std::unique_ptr<ForwardPlayer> forwardPlayer
+ (private) std::unique_ptr<GraphPlayer> graphPlayer
+ (private) std::unique_ptr<GraphCalculator> calculator
+ (private) FrameView* rawVideoView
+ (private) std::unique_ptr<Utility::ResultSaver> resultsSaver
+ (private) QTimer timer_labelUpdater

# (private) VideoPLayer* player

VideoPlayer* => std::unique_ptr<VideoPlayer>
```

Zusätzlich für die GUI benötigt.

### 2.14.4 ControlPanel

```
# (public) void addVideoPlayer(Player& player)
```

```
PLayer& => Player*
```

Bessere Schnittstelle.

```
# (private) std::vector<VideoPayer*>
```

```
VideoPlayer* => Player*
```

Dass auch GraphPlayer akzeptiert werden.

### 2.14.5 FilterContainerTab

```
# (public) void addFilter(Model::Filter filter)
```

```
Model::Filter => QString
```

Bessere Schnittstelle.

```
- (public) void unchecl(QString filtername)
```

Nicht mehr benötigt.

```
+ (private) container
+ (private) filterContainer
```

Zusätzlich für die GUI benötigt.

### 2.14.6 FilterTab

```
# (public) Memento::FilterTabMemento getMemento()

FilterTabMemento => std::unique_ptr<FilterTabMemento>

# (public) void restor(Memento::FilterTabMemento memento)

FilterTabMemento => FilterTabMemento*
```

Unnötiges Kopieren vermeiden.

```
+ (public) static constexpr int MAX_PREVIEW_COUNT
```

Magic numbers vermeiden.

```
# (public) void insertFilter(Model::Filter filter, int index)

Model::Filter =-> std::unique_ptr<Filter>
int => std::size_t

# (public) void removeFilter(std::string filter)

void => std::unique_ptr<Model::Filter>
std::string => std::size_t

# (public) void moveFilter(int pos1, int pos2)

int -> std::size_t

+ (public) void changeFilter(int index, QString newState)
+ (public) Model::Filter* appendFilter(QString filtername)

# (public) void setFilterList(Model::FilterList list)

Model::FilterList => std::unique_ptr<Model::FilerList>

+ (public) Model::FilterList* getFilterList()
```

Bessere Schnittstelle für Filter.

```
# (public) void setRawVideo(Model::YuvVideo video)

Model::YuvVideo => Model::YuvVideo*
```

Video wird in UndoCommand gespeichert.

```
+ (public) void updateFilterPreview()
+ (public) void showFilteredVideo()
```

Mehr Kontroll über den Tab.

```
+ (public) void setMainWindow(MainWindow* window)
+ (public) MainWindow* getMainWindow()
+ (private) MainWindow* mainWindow

+ (private slots) void notifyOnSaveComplete(bool successful,QString filename,int width,int
+ (private slots) void notifyOnApplyComplete(bool successful)
```

Für Statusbar.

```
- (private) QLabel* filterOptions
- (private) QFrame* filterContainer
+ (private) QVBoxLayout* v_player

# (private) VideoPlayer* player

VideoPlayer* => std::unique_ptr<VideoPlayer>

+ (private) QHBoxLayout* h_filterOptions
+ (private) QSpacerItem* spacer_filterOptions
+ (private) FilterConfigurationBox* currentFilterOptionsBox

# (private) Model::FilterList filterList

Model::FilterList => std::unique_ptr<Model::FilterList>

+ (private) std::unique_ptr<Model::AVVideo> originalPreviewFrames
+ (private) std::unique_ptr<Model::Video> filteredPreviewFrames
+ (private) std::unique_ptr<Utility::FilterApplier> previewCalculator
+ (private) std::unique_ptr<Utility::FilterApplier> filterApplier
+ (private) std::unique_ptr<Model::Video> filteredVideo
+ (private) bool isFilteredVideoShown
+ (private) std::unique_ptr<Utility::YuvFileSaver> safer
- (private) st::vector<FilterContainerTab> filterContainerTab

+ (private) void createButtons()
+ (private) void createListView()
+ (private) void createVideoPlayer()
+ (private) void createFilterTabs()
+ (private) void initPlayer()
+ (private) void initFilterList()
+ (private) void calculatePreviewFrames()
```

Zusätzlich für die GUI benötigt.

### 2.14.7 FilterView

```
# (public) void setFilter(Model::Filter filter)

Model::Filter => QString
```

Bessere Schnittstelle.

```
+ (private) static std::unique_ptr<QImage> defaultImage

- (private) void checkBoxStateChanged()
- (private) QCheckBox* checkbox
- (private) QLabel* preview

# (private) Model::Filter* filter
Model::Filter* => std::unique_ptr<Model::Filter>

+ (private) QPushButton* button_addFilter
+ (private slots) void buttonPressed()
+ (private) std::unique_ptr<QImage> filterImage
```

Für die GUI.

### 2.14.8 ForwardPlayer

```
+ (public) void setMasterVideoPlayer(Player* player)
+ (private) Player* player
- (private) ControlPanel controlPanel
```

Um alle Methoden implementieren zu können.

```
+ (public) std::size_t getNumberOfFrames()
```

Geerbt vom Player.

### 2.14.9 FrameView

```
# (protected) void repaintEvent(QPaintEvent* event)
repaintEvent => paintEvent
```

Falscher Name.

```
+ (private) QPixmap drawedImage
```

Unnötiges Konvertieren vermeiden.

### 2.14.10 GlobalControlPanel

```
+ (public) int getPosition()
```

War nötig.

### 2.14.11 GraphWidget

```
# (public) void drawGraph(Model::Graph g, bool filled)
(...) => (Model::Graph* graph)

+ (public) Model::Graph* getGraph()
```

Bessere Schnittstelle.

```
- (public) void setLineColor(QRgb color)
- (public) void setFillColor(QRgb color)
- (private) QRgb lineColor
- (private) QRgb fillColor

+ (public) void setGraphCalculator(GraphCalculator* calculator)
+ (private) GraphCalculator* calculator
```

Berechnen wird ausgelagert.

```
+ (public) void buildScene()
+ (protected) void resizeEvent(QResizeEvent* event)
+ (private) QTimer updater
+ (private) QGraphicsScene scene
+ (private slots) void updateView()
```

Zum Zeichnen.

### 2.14.12 MainWindow

```
# (public) Memento::MainWindowMemento getMemento()
Memento::MainWindowMemento => std::unique_ptr<Memento::MainWindowMemento>

# (public) void restore(Memento::MainWindowMemento memento)
Memento::MainWindowMemento => Memento::MainWindowMemento*
```

Unnötiges Kopieren vermeiden.

```
- (public) Model::Project& getProject()
```

Nicht benötigt.

```
# (private) Model::Project* loadedProject
Model::Project* => std::unique_ptr<Model::Project>
```

Nackte Pointer vermeiden.

```
+ (public) QStatusBar* getStatusBar()
```

Für Statusbar.

```
- (private) void createMenuBar()

- (private) QMenuBar* menubar_project
- (private) QMenuBar* menubar_edit

+ (private) FilterTab* filterTab
+ (private) AnalysisTab* analysisTab
+ (private) Ui::MainWindow* ui
```

Für GUI.

### 2.14.13 Player

```
# (public) virtual void setPosition(int position)
# (public) virtual int getPosition()

int => std::size_t
```

An vector anpassen.

```
+ (public) std::size_t getNumberOfFrames()
```

Um die ControlPanels korrekt aktualisieren zu können.

### 2.14.14 PlayerControlPanel

```
# (public) void updateUi()
public => public slots

+ (private) QTimer updater
```

Um die Gui dem Player anzupassen.

### 2.14.15 PreviewControlPanel

```
# (public) void updateUi()
public => public slots

+ (private) QTimer updater
- (private) void updateLabel()
```

Um die GUI dem Video anzupassen.

### 2.14.16 Timer

```
# (public) void addPlayer(VideoPlayer& player)
# (public) void removePlayer(VideoPlayer& player)

VideoPlayer& => Player&

+ (private) std::vector<Player*> players
```

Um auch GraphPlayer reinpacken zu können.

### 2.14.17 VideoPlayer

```
# (public) void setVideo(Video& video)

(...) => (Video* video,bool updateTimer)
```

Um Videos hinzuzufügen ohne den Timer zu stoppen.

```
# (public) void setPosition(int position)
# (public) int getPosition()
# (private) int position

int => std::size_t

+ (public) std::size_t getNumberOfFrames()
```

Geerbt von der Basisklasse.

```
+ (private) QTimer viewUpdater
+ (private slots) void updateViews()

# (private) Timer timer
Timer => std::shared_ptr<Timer>
```

War nötig.

### 2.14.18 YuvInfoDialog

```
# (public) PixelSheme getPixelSheme()
PixelSheme => Utility::YuvType
```

PixelSheme und YuvType waren doppelt.

```
- (public) void show()
- (public) bool wasSuccessfull()
```

Wird von QDialg übernommen.

```
+ (private) QStringList pixelShemeList
+ (private) QStringList compressionList
```

Zusätzlich für GUI benötigt.

### 2.14.19 YuvFileOpenDialog

```
+ (public) static constexpr int MAX_SAVED_ENTRIES
+ (public) static const QString SAVE_FILENAME
```

Magische zahlen vermeiden.

```
+ (public) int getFps()
+ (public) int getWidth()
+ (public) int getHeight()
+ (public) Utility::Compression getCompression()
+ (public) Utility::YuvType getPixelSheme()

+ (private) int fps
+ (private) int width
+ (private) int height
+ (private) Utility::Compression compression
+ (private) Utility::YuvType type

+ (private) void askAttributes()
+ (private) bool parseAttributes()
```

Speichere auch Attribute der Yuv datei.

```
- (public) void show()
- (public) bool wasSuccessfull()
```

Wird von QDialg übernommen.

```
+ (private slots) void chooseFile()
+ (private slots) void selectionChanged(const QItemSelection& selection)
+ (private slots) void hasFinished(int result)

+ (private) QLineEdit* lineEdit_selectedFile
```

Zusätzlich für GUI benötigt.

```
- (private) static void loadRecentlyUsed()
- (private) static QListViewModel model_recentlyUsed

+ (private) static QStringListModel* getListModel()
+ (private) static void saveListModel(QString selectedFile, QString attributes)
+ (private) static QStringListModel* model_recentlyUsed
```

Für recently used Dateien.

# 3 Ergebnis

## 3.1 Verlauf der Phase

Unser Zeitplan war auf 4 Tage pro Woche mit einer Gesamtdauer von 4 Wochen ausgelegt. Ziemlich am Anfang wurde klar, dass wir uns bei der Dauer der einzelenen Implementierungszeiten sehr verschätzt hatten. Bei den GUI-Klassen war zu wenig Zeit eingeplant, während bei den Model- und Utility-Klassen zu viel Zeit war. Aufgrund dieser Schwierigkeiten hatten wir die Klausurpause innerhalb der Implementierungsphase gewählt, um genügend Zeit zu haben, falls es zeitlich knapp wird. Allerdings konnten wir unerwarteterweise die Implementation trotzdem innerhalb von 29 Tagen abschließen.

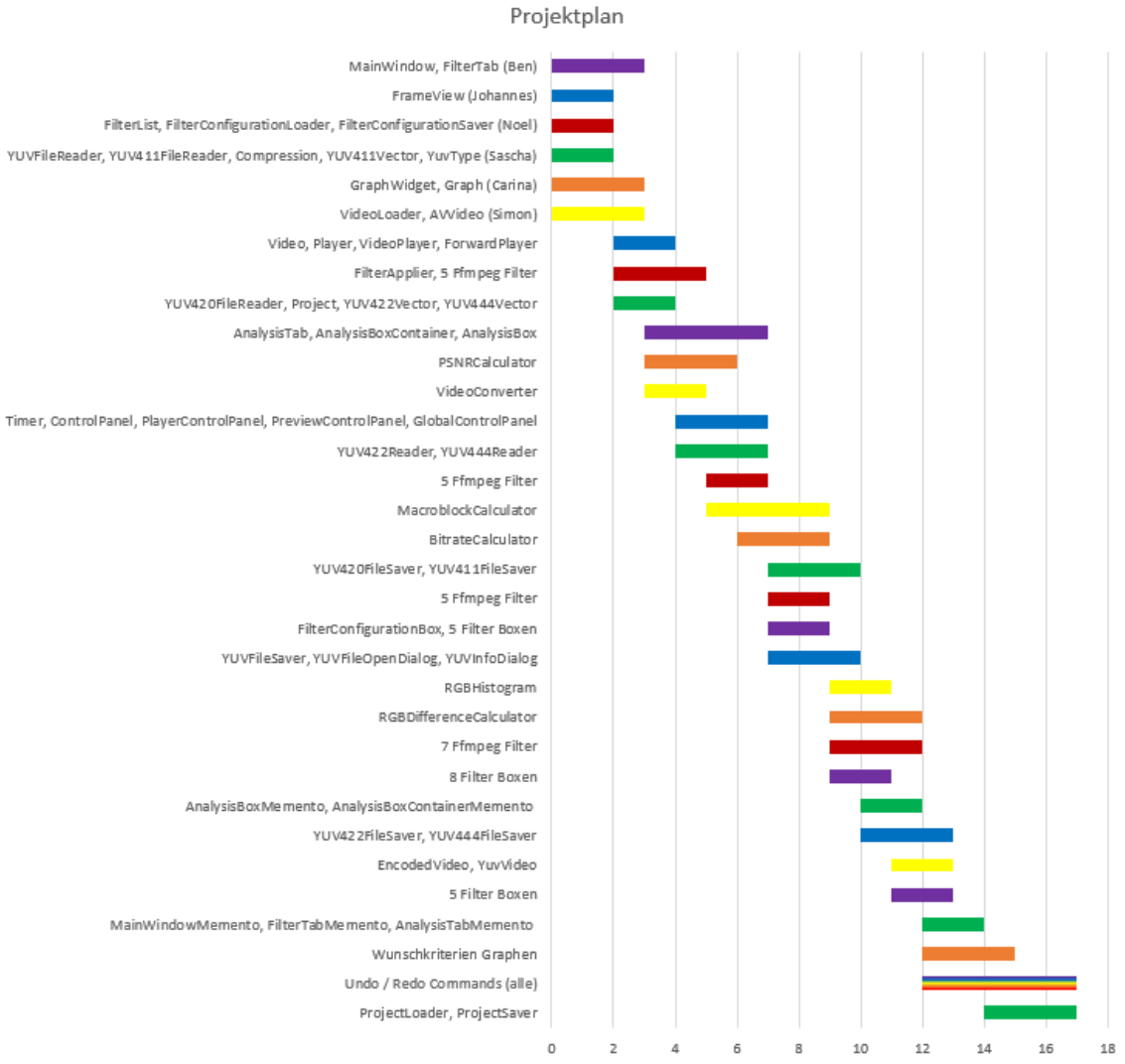


Abbildung 1: Ganttdiagramm der Implementierungsphase

## 3.2 Grobübersicht der Entwurfsänderungen

In diesem Abschnitt werden die größten Änderungen in den einzelnen Pakten kurz erlätert und begründet.

### 3.2.1 Allgemein

Auch wenn die Änderungen auf den ersten Blick groß und umfrangreich erscheinen, hat sich die grobe Idee, wie die Klassen miteinander kommunizieren und interagieren, nicht gändert.

**Smartpointer** Viele Änderung im gesamten Entwurf beziehen sich auf die Benutzung von Smartpointer anstatt von 'nackten' Pointern. Obwohl uns während des Entwurfs schon klar war, dass wir auf Smartpointer setzen werden, hatten wir keine Möglichkeit dies im Klassen diagramm korrekt darzustellen. Hauptproblem war dabei, dass Smartpointer häufig Attribute in den Klassen waren und wir diese Attribute mittels Aggregationen und Kompositionen dargestellt haben. Daher waren diese Aggregationen hinterher nur Pointer und keine Smartpointer.

**Abstrakte Klassen** Während der Entwurfsphase war uns nicht bewusst, dass es in C++ nur möglich ist, Objekte von abstrakten Klassen als Pointer zu übergeben. Deshalb mussten vorallem viele Methoden, die einen Filter als Parameter nehmen, geändert werden.

**Vergessenes** Ein kleiner Teil der Änderungen sind Attribute, die im Entwurf einfach vergessen wurden. Ein weiterer Faktor war auch, dass beim Ändern einer Klasse andere Klassen, die diese benutzten, nicht korrekt angepasst wurden.

### 3.2.2 Utility Paket

**Threading** Fast alle Änderungen in diesem Packet wurden gemacht, um das asynchrone Laden und Berrechnen von Videos und Graphen zu ermöglichen. Der Grund, weshalb wir Threading während des Entwurfs nicht mit einbezogen hatte, ist, dass wir nicht genau wussten, wie wir Threads einbinden und das Signalhandling entwerfen sollen.

### 3.2.3 Model Paket

**Bessere Schnittstelle** Die meisten Änderungen in diesem Paket wurden gemacht, um das Benutzen der Klassen einfacher zu machen. Teilweise passte die Schnittstelle, aufgrund von Änderungen an anderen Stellen, nicht mehr zu dem, was benötigt wurde.

### 3.2.4 UndoRedo Paket

Bei den Klassen in diesem Packet hatten wir schon während des Entwurfs große Schwierigkeiten einzuschätzen, was notwendig ist, um eine Aktion wieder rückgängig zu machen. Vorallem da Änderungen an einer Stelle im Entwurf sich sofort auf den entsprechenden Undo-command auswirken, hat dieses Packet schon im Entwurf viele Änderungen erfahren.

### 3.2.5 Memento Paket

**Pointer** Bei den Mementos hatten wir ursprünglich die Idee, dass sie die Daten, die sie halten, auch besitzen. Das impliziert, dass jedes mal, wenn ein Memento erstellt wird, viel kopiert wird. Das hat sich vorallem im Hinblick auf die Videos als eher unpracktikabel herrausgestellt. Daher wurden viele Attribute und Parameter zu Pointern gemacht.

**Vergessene Daten** Ebenfalls viele Änderungen beziehen sich auf Attribute, die wir einfach vergessen haben anzugeben.

### 3.2.6 GUI Paket

**Änderungen an der GUI** In diesem Packet sind,qie zu erwarten, die meisten Änderungen passiert. Das hängt vorallem damit zusammen, dass wir an manchen Stellen die GUI abgändert haben, weil sich Teile des geplanten Aufbaus als zu Umständlich in der Bedienung entpuppt haben, oder einfach nicht ins Gesamtbild passten. Allerdings lösen selbst kleine Änderungen ziemlich große Änderungen am Entwurf aus. Auch das Zusammenspiel der einzelnen GUI-Komponeneten hatten wir nicht richtig durchdacht, weshalb es in diesem Bereich auch etliche Änderungen gab.

## 3.3 Änderungen an der GUI

Es gab keine großen struktuellen Änderungen an der GUI. Das Look and Feel unterscheidet sich aber trotzdem deutlich von der erstellten Beispieloberfläche im Pflichtenheft. Sieht dort alles noch recht unorganisiert und gestaucht aus, setzt die fertige Oberfläche auf weiche Kanten, deutliche Übergänge zwischen den einzelnen Elementen und ein allgemein stimmiges Gesamtbild. In den folgenden Abschnitten werden die konkreten Unterschiede beschrieben und erläutert.

### 3.3.1 YuvFileOpen Dialog

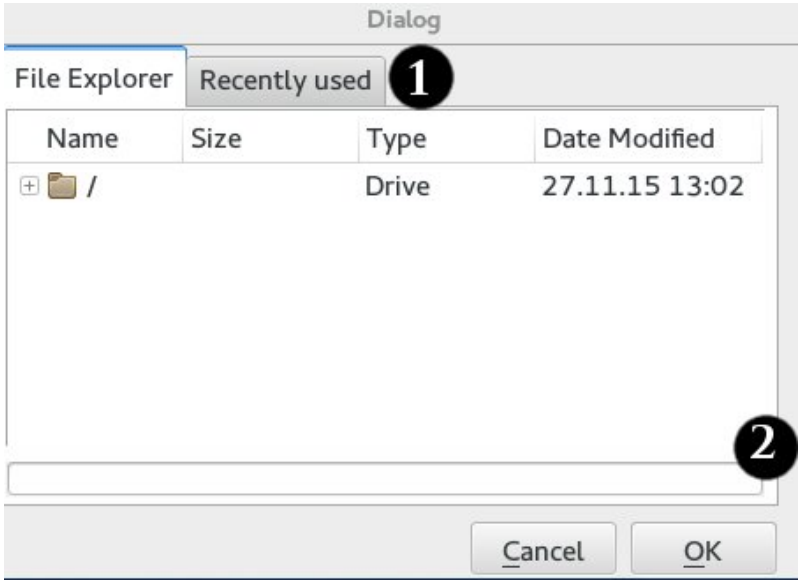


Abbildung 2: YuvFileOpenDialog aus dem Pflichtenheft

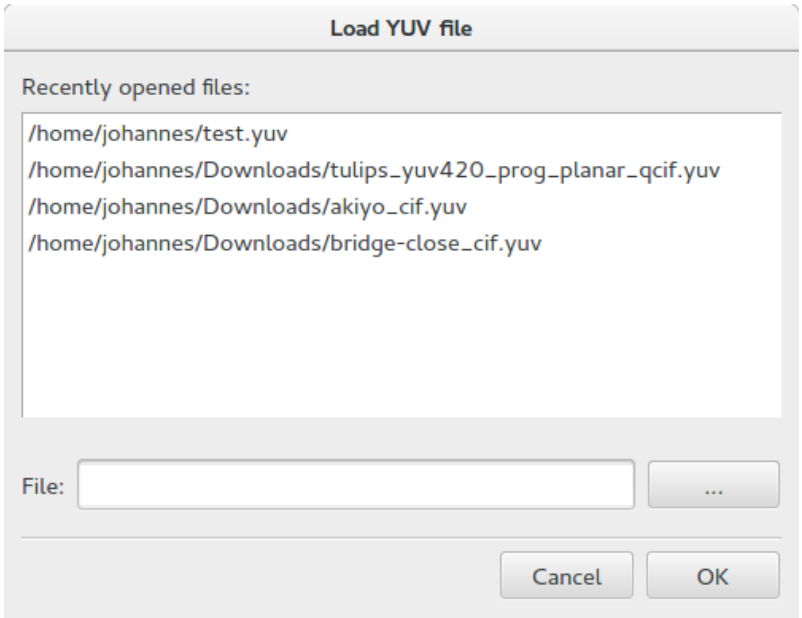


Abbildung 3: YuvFileOpenDialog im fertigen Pogramm

Die Tabs wurden zu einem verschmolzen und der Dateiexplorer ganz entfernt. Dieser ist nun über den Button (...) erreichbar. Wir haben uns für diese Änderung entschieden, da es nicht ohne weiteres möglich war, einen Dateiexplorer direkt einzubinden und haben daher eine einfachere Variante gewählt.

### 3.3.2 YuvFileInfo Dialog

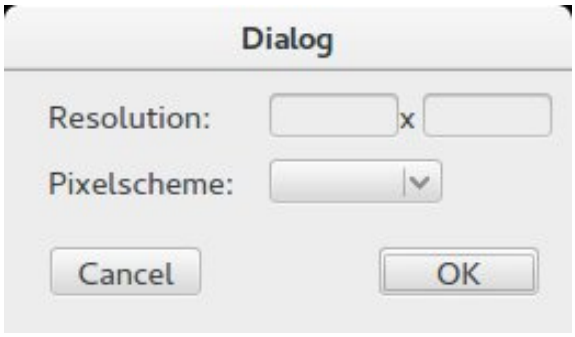


Abbildung 4: YuvFileInfoDialog aus dem Pflichtenheft

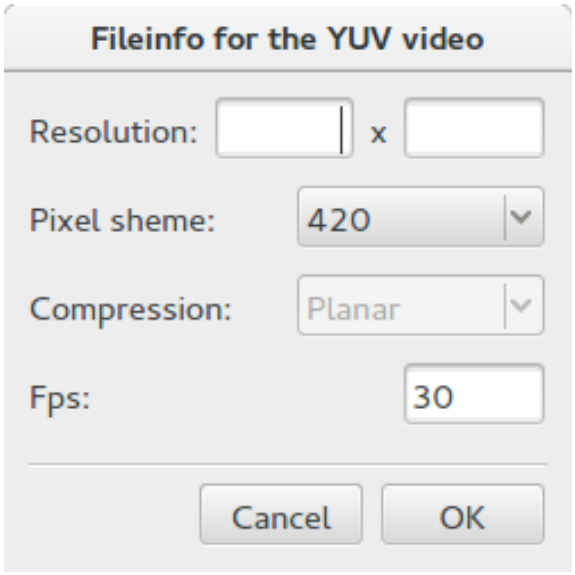


Abbildung 5: YuvFileInfoDialog im fertigen Pogramm

Neben kleinen Umordungen der Elemente sind außerdem noch weitere Eingabefelder für die Kompression und die Fps des Videos hinzugekommen. Die Kompression ist hinzugekommen, da wir jedes Pixelformat, außer 420, nun auch planar als auch packed unterstützen. Das Fps Feld ist dazu gekommen, um es dem Nutzer zu ermöglichen, Videos mit einer beliebigen Fps abzuspielen.

### 3.3.3 Filtertab

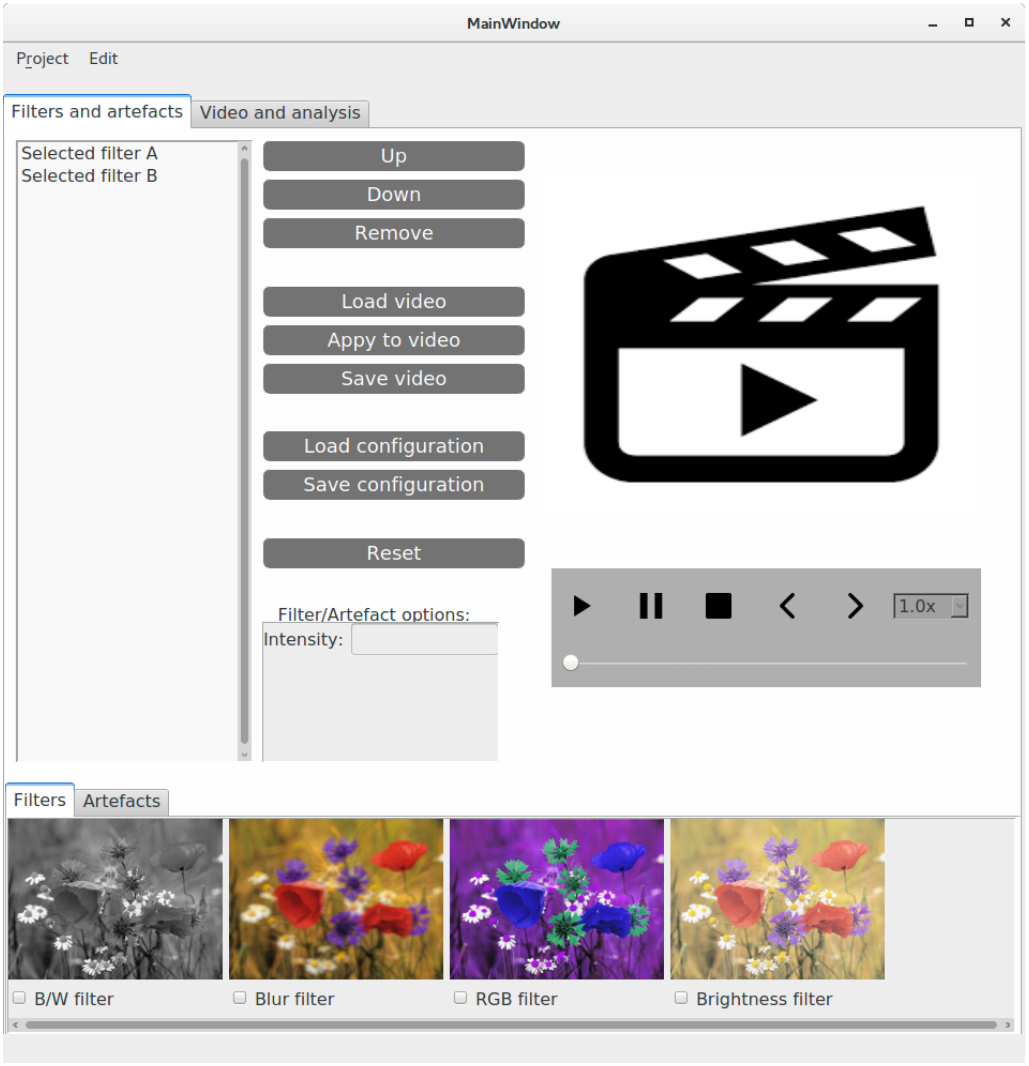


Abbildung 6: Filtertab aus dem Pflichtenheft

Im Filtertab gab es keine großartigen Anpassungen. Der deutlichste Unterschied ist, dass die Oberfläche im fertigen Programm farblich besser abgestimmt und wesentlich ansprechender ist. Einige kleine Änderungen gab es aber dennoch. Zum einem wurden die Checkboxen bei der Filterauswahl durch Buttons ersetzt, einfach um es dem Nutzer zu ermöglichen, einen Filter mehr als einmal anzuwenden. Die zweite und letzt Änderung am Filtertab, ist, dass die Filteroptionen einen eigenen Tab bekommen haben und nicht mehr mitten in der GUI hängen. Diese Änderung wurde rein aus Platzgründen gemacht.

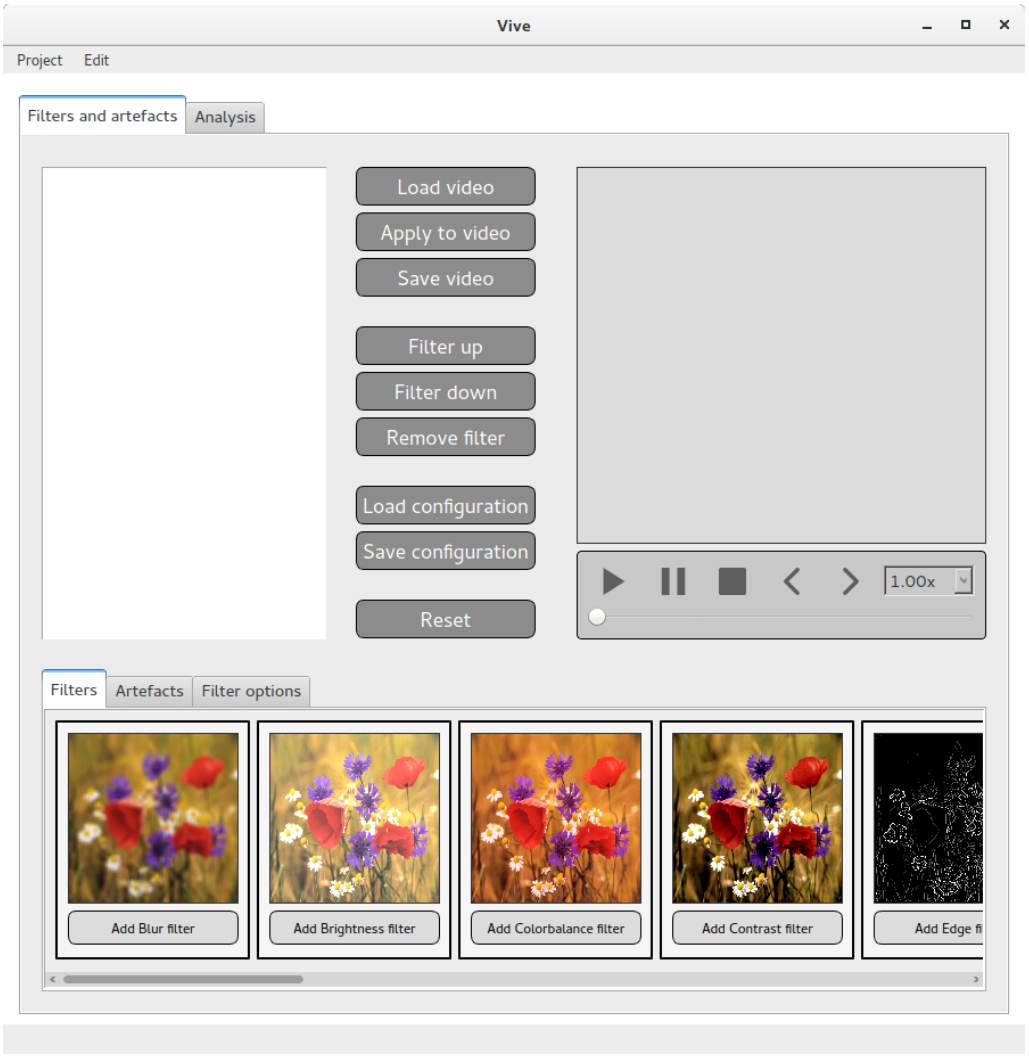


Abbildung 7: Filtertab im fertigen Pogramm

### 3.3.4 Analyisistab

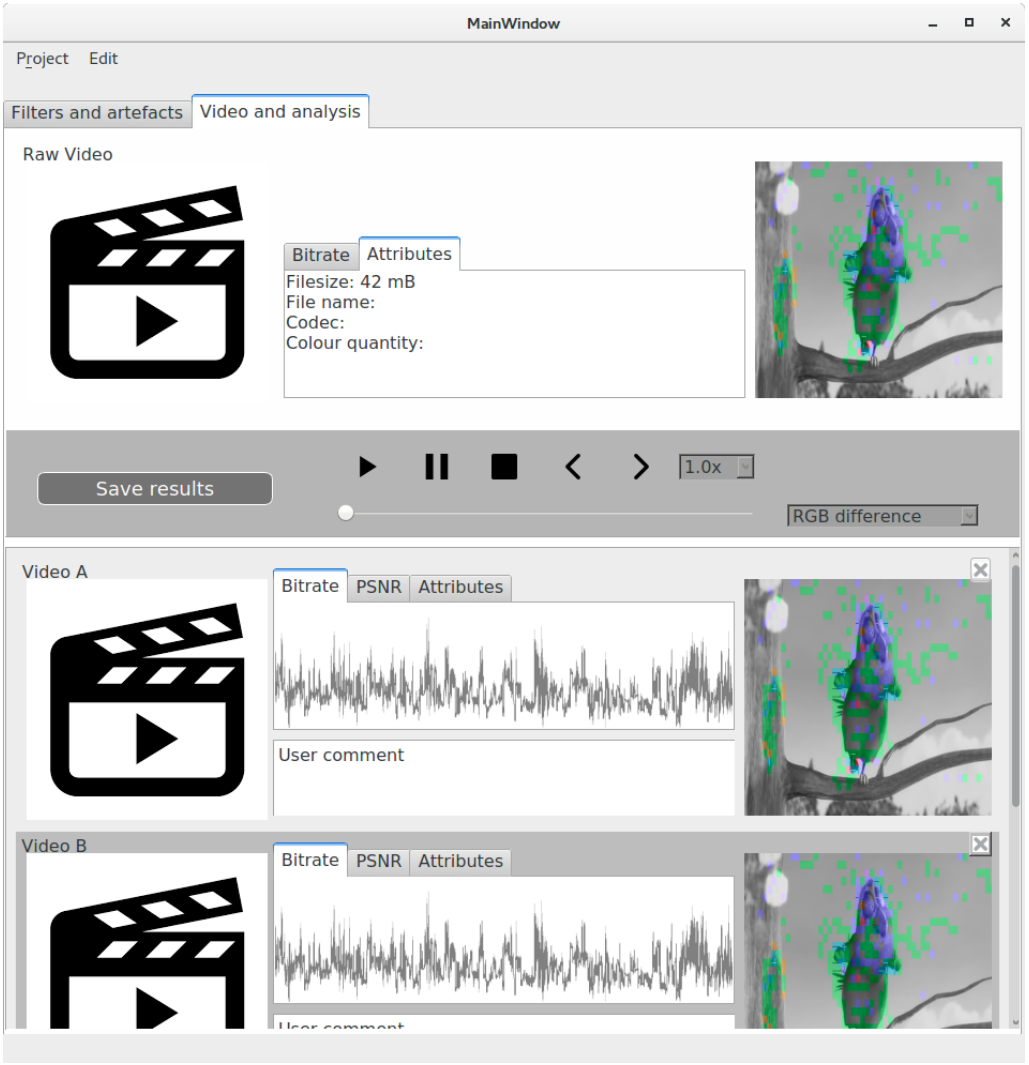


Abbildung 8: Analysistab aus dem Pflichtenheft

Ähnlich wie beim Filtertab ist auch hier die größte Änderung die optische Verschönerung des Gesamtbildes. Die einzige wichtige Veränderung, die passiert ist, ist, dass bei den encodeden Videos statt der einzelnen Tabs nun Buttons neben das Rohvideo gesetzt wurde, mit denen man auswählen kann, welcher Graph für alle Videos angezeigt werden soll. Diese Änderung erhöht den Komfort bei der Analyses immens, da nun nicht immer bei jedem Video einzeln der entsprechende Graph ausgeählt werden muss, um diese zu vergleichen.

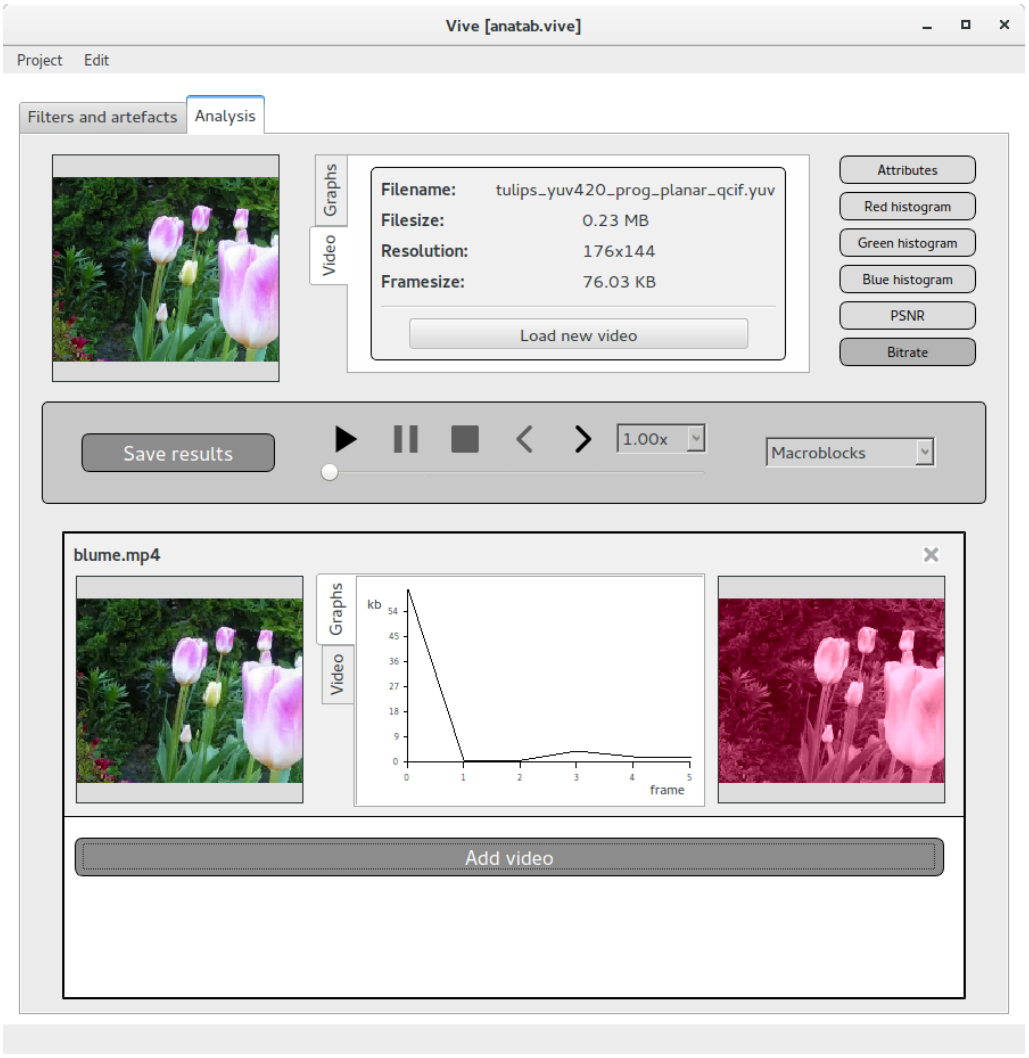


Abbildung 9: Analysistab im fertigen Pogramm

## 3.4 Entfernte Funktionalität

**Ergebnisse speichern beim Speichern des Projektes** Diese Funktionalität war weniger ein Implementierungstechnisches Problem als eine schlechte Entscheidung. Im Pflichtenheft hatten wir geplant, dass berechnete Videos und Graphen beim Projekt speichern mit abgespeichert werden und später, wenn das Projekt wieder geladen wird, auch wieder geladen werden. Allerdings geht das Neuberechnen dieser schneller, als das komplette Einlesen der Dateien. Deshalb wurde diese Funktionalität entfernt und berechnete Videos und Graphen können nur noch separat über die Ergebnisse speichern Funktionalität abgespeichert werden.

**Blenden Filter** Dieser Filter sollte bewirken, dass das Video entweder von einem schwarzen Bild aus einblendet oder auf ein schwarzes Bild ausblendet. Allerdings steht dieser Filter im Konflikt mit der Filtervorschau, da das Video komplett berechnet werden müsste, um die Vorschau korrekt darzustellen und wurde daher entfernt.

**Zoom Filter** Bei diesem Filter sind wir auf ein Bug in der libavfilter Bibliothek gestoßen, der das Bild bei der Ausgabe falsch skalierte. Um diesen Bug zu umgehen, hätten wir einige unschöne Änderungen am Entwurf machen müssen und haben den Filter daher nicht implementiert.

**Anzahl der Farben in encodierten Videos** Die Anzahl der Farben im Video sollte als Attribut bei den encodierten Videos angezeigt werden. Allerdings gibt es keinen effizienten Weg, diese Zahl zu berechnen. Allein um während der Berechnung zu speichern, welche Farbe bereits verwendet wurde, wären ca. 66 MB Arbeitspeicher notwendig, geschweige denn der CPU-Leistung. Daher steht der Aufwand dieses Attributs in keinem Verhältnis zu seiner Wichtigkeit und wurde daher entfernt.

## 3.5 Zusätzlich implementierte Funtionalität

**Filter mehr als einmal anwenden** Im Pflichtenheft war vorgesehen, dass ein Filter nur maximal einmal auf ein Video angewendet werden kann. Da diese Einschränkung implementierungstechnisch jedoch mehr Aufwand gewesen wäre, haben wir sie weggelassen.

**Attribute für das Rohvideo im Analysistab** Folgende Attribute werden für das Rohvideo angezeigt: Dateiname, Dateigröße, Auflösung und die Größe eines Frames im Arbeitsspeicher.

**Zusätzliches Attribut für encodierte Videos** Folgendes Attribut wird zusatzlich für encodierte Videos angezeigt: Durchschnittliche Bitrate.

**Bei Klick auf Bitrategraph im Video springen** Diese Funktionalität wurde nur für den Psnrgraphen spezifiziert. Allerdings war es kein großer Aufwand, diese Funktionalität auch dem Bitrategraphen zu geben, weshalb man nun auch bei einem Klick auf den Bitrategraphen im Video springen kann.

**Anzeigen von Motionvektoren** Zusätzlich zu den Macroblöcken werden im selben Video auch noch die Motiovektoren. Der Grund für diese Änderung ist allerdings ein Bug in der libavcodec Bibliothek.

**Zusätzliche Angaben im YuvFileInfoDialog** Folgende Zusätzliche Angaben können gemacht werden: Fps, Kompression.

# 4 Testüberdeckung

Während der Implementierung haben wir bereits damit begonnen, Testcode zu schreiben.

**Getestete Klassen** Es sind Unit Tests für alle Klassen der Packete Model und Memento vorhanden, sowie für die Klasse VideoConverter aus dem Packet Utility. Das ergibt eine insgesamte Anzahl von 35 Klassen.

**Testüberdeckung** Alle Testfälle ergeben eine Zeilenanzahl von 1900, was einer Testüberdeckung von 23% entsprichht.