

TECHNISCHE FACHHOCHSCHULE BERLIN
University of Applied Sciences

Software-Projekt II

Prof. Christoph Knabe
SS 2005



Dokumentation

Projektgruppe:

Marco Kraus	s717586	marco@klear.org
Omar El-Dakhloul	s718254	omar@klear.org
Manuel Habermann	s718468	manuel@klear.org
Patric Bico Sherif	s717094	patric@klear.org

www.klear.org

Berlin, 01. Juli 2005

Inhaltsverzeichnis

1. Einleitung	3
2. Was ist „Klear“?	3
3. Hauptarbeitsfelder der Gruppenmitglieder	4
4. Softwareentwicklungsprozess	4
5. Externe Vorgaben	5
5.1 Technische Vorgaben	5
5.2 Inhaltliche Vorgaben	5
6. Verwendete Technologien	5
7. Technische Spezifikationen	6
7.1. Beschreibung	6
7.1.1. Muss-Kriterien	6
7.1.2. Kann-Kriterien.....	7
7.2. Grafische Darstellung.....	8
8. Abstraktes Klassendiagramm	9
8.1. Schichten-Modell	9
8.2. Engine	10
8.3. Exception	11
9. Klassendiagramme	12
9.1. Controller-Paket	12
9.2. App-Paket	13
9.3. Exception-Paket	14
10. Unittests	15
11. Installationsanleitung	16
11.1. Systemvoraussetzungen	16
11.2. Installation	16
12. Bedienungsanleitung	17
12.1. Hauptfenster	17
12.2. DVB Settings.....	18
12.3. General	19
12.4. Recording.....	20
12.5. Video	21
12.4. Recording Overview	22
12.5. Add new / Edit Klear record	22
13. Erfahrungen und Probleme	23
14. Daten und Fakten	24

1. Einleitung

Die soziale und wirtschaftliche Bedeutung der Informationstechnologie wächst rasant. Die Indikatoren zur Informationsgesellschaft sind Referenzwerte, welche diese Entwicklungen nachzeichnen und fassbar machen.

Bei der Digital Video Broadcasting (DVB) handelt es sich um ein Konsortium mit über 200 Mitgliedern aus den Sektoren Rundfunkveranstalter, Gerätehersteller, Netzbetreiber und Regulierungsbehörden aus über 30 Ländern. Dieses DVB-Konsortium hat sich für die Entwicklung eines international harmonisierten Standards zur Übertragung von digitalem Fernsehen verpflichtet.

Durch den Übergang vom analogen zum digitalen Fernsehen, stand die Gruppe Klear vor der Herausforderung eine DVB-Applikation zu entwickeln.

2. Was ist „Klear“?

Klear ist eine OpenSource-Anwendung für digitales Fernsehen (DVB im Speziellen) unter Linux. Im Rahmen der Software-Projekt Vorlesung an der Technischen Fachhochschule Berlin, hat die Klear-Gruppe, bestehend aus vier Studenten, sich das Ziel gesetzt, eine DVB-Applikation für das Betriebssystem Linux zu programmieren.

Die Kernfunktionen sollen neben normalem Fernsehen auch die Nutzung der digitalen Funktionen sein. Darunter zählt unter anderem auch die direkte Aufzeichnung von Sendungen in verschiedenen digitalen Formaten am Computer, das zeitgesteuerte Aufzeichnen, Videofilter, Audiosteuerungen, die Integration in die Betriebssystemumgebung und vieles mehr. Neben den sichtbaren Komponenten muss eine DVB-Anwendung auch die Hardware ansprechen, Kanaleinstellungen und Tuner steuern (verschiedene Steuerungen für DVBT/S/C) und eine persistente Konfiguration beinhalten.

Die Applikation ermöglicht es, Fernsehen in bestechender digitaler Qualität zu erleben. Im Zusammenspiel mit der TerraTec Cinergy T2 wird es realisierbar, DVB-T konformes Fernsehsignal anzuzeigen und per Hard Disk Video in Originalqualität aufzuzeichnen.

3. Hauptarbeitsfelder der Gruppenmitglieder

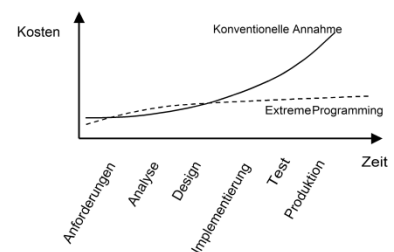
	Marco Kraus [Projektkoordination und Releasemanager] Programmiersprachen: C/C++, Java, C#, PHP, Pearl	marco@klear.org
	Omar El-Dakhloul [GUI-Design, Dokumentation, Klassendiagramme] Programmiersprachen: C/C++, Java, C#, PHP, Pearl	omar@klear.org
	Manuel Habermann [Live/ScheduledRecording, SystrayIntegration, KlearXineEngine] Programmiersprachen: Java, C#, C/C++, Pearl	manuel@klear.org
	Patric Bico Sherif [Live/ScheduledRecording, SystrayIntegration, KlearXineEngine] Programmiersprachen: Java, C#, C/C++, Opal, PHP, Pearl	patric@klear.org

Hinweis: In der API-Dokumentation steht zu jeder Klasse und Funktion der zuständige Autor, der den jeweiligen Code geschrieben hat und Verantwortung trägt.

4. Softwareentwicklungsprozess



Klear hat es von vornherein vermieden, ein überreguliertes Phasenmodell (siehe Aufwandskurve) zu erstellen und anhand dessen zu entwickeln und entschied sich eher für das Extreme Programming (XP). Daher entfallen ausgedehnte Dokumentationen wie ein erweitertes Pflichtenheft, StyleGuide, Softwareentwurfsskizze und Prüfprotokoll im Sinne des VModell97. Um dennoch eine begleitende schriftliche Dokumentation zu gewährleisten, haben wir die wichtigsten Punkte zusammengefasst.

XP ist eine relativ neue Vorgehensweise in der Softwaretechnik. Sie ist der bekannteste Prozess aus der Klasse der agilen Prozesse der Softwareentwicklung. Agile Softwareentwicklung zeichnet sich durch geringen bürokratischen Aufwand und wenigen flexiblen Regeln aus. In XP wird auf einen strikten Anforderungskatalog des Kunden verzichtet, dafür werden Kundenwünsche berücksichtigt, die sich noch während der Softwareentwicklung ergeben. Statt des klassischen Wasserfallmodells durchläuft der Entwicklungsprozess immer wieder in kurzen Zyklen (typ. eine Woche) sämtliche Disziplinen der klassischen Softwareentwicklung (Anforderungsanalyse, Design, Implementierung, Test). Nur die im aktuellen Iterationsschritt benötigten Merkmale werden implementiert.



5. Externe Vorgaben








5.1 Technische Vorgaben

	<p>Ein TerraTec Cinergy T2 wird für den Empfang des DVB-T Signals benutzt. Dieses Gerät hat eine USB-Verbindungsfläche um die Zugänglichkeit zum PC zu garantieren. Der Treiber des Geräts muss mit dem Betriebssystem Linux kompatibel sein. Übrigens deckt Linux ca. 70% aller DVB-Devices ab, so dass man auch andere Hardware ansprechen kann, die das System unterstützt.</p>
	<p>Als Betriebssystem kann eine willkürliche Linuxdistribution (Debian, Knoppix und SuSE) verwendet werden, die zumindest den KDE 3.2 unterstützt.</p>

5.2 Inhaltliche Vorgaben

Die Applikation soll im Open-Source-Mainstream liegen. Die sich hieraus ergebende Zielgruppe sind gleichzeitig Nutzer als auch Kunden. Daraus ergeben sich zu Weilen, geänderte inhaltliche Vorgaben. Mittels „Extreme-Programming“, kurz xP, als Vorgehensmodell soll den daraus resultieren Problemen entgegengewirkt werden. Um schnellstmöglich mit dem Kunden in Interaktion treten zu können wird die Applikation bereits sehr früh zu einer möglichen Freigabe ausgearbeitet. Ein weiterer Grund, weshalb hier auf xP als VM zurückgegriffen wird. Als Kommunikationsplattform und Präsentation von Klear wurde eine Internetpräsentation erstellt, die unter www.klear.org zu finden ist.

6. Verwendete Technologien

	<p>Als Umgebung für die Programmierung wird KDevelop benutzt. Programmiersprache: C++ / gcc 3.3.x als C++ Compiler</p>
	<p>QT / QTDesigner von Trolltech als GUI-Framework und C++Toolkit (qmake als BuildTool)</p>
	<p>DVB Kernel-API für die DVB-Hardware und Tuner-Ansteuerung</p>
	<p>Xine als Wiedergabe-API für die Video-Engines</p>
	<p>MPlayer für die alternative VideoEngine</p>
	<p>DOXYGEN wird für die Vorbereitung der API verwendet.</p>
	<p>CCCV wird als Metrik-Software benutzt</p>

7. Technische Spezifikationen

7.1. Beschreibung

7.1.1. Muss-Kriterien

Folgende Bauelemente wurden im Laufe des Projekts entwickelt, nachdem die Systemintegration durch den Kernel des Betriebssystems eingestellt und die nötige Firmware angepasst wurde:

Streaming:

Wie gelangen die digitalen Ströme von der Settop-Box auf den Computer? Alle Fernsehsender senden ihre Audiodaten im MPEGLayer2-Format (MP2 Stream). Bei der Auflösung der Videodaten gibt es jedoch Unterschiede. Die Video-Signale werden digitalisiert nach MPEG-2. Besonderes Kennzeichen von digitalen Video-Netzen ist, dass die Daten in Blöcken oder Containern übertragen werden. Dazu dienen die MPEG-2-Transport-Ströme.

Recording:

Wie zeichnet man die digitalen Ströme auf die Festplatte? Die digitalen Ströme, die empfangen werden, sollen nach Wunsch auf die Festplatte geschrieben werden. Zusätzlich soll die Möglichkeit bestehen, Aufnahmen nach einer vom Benutzer bestimmten Uhrzeit festzuhalten.

Tuner für DVB-T/S:

Es gibt mehrere technische Unterarten von DVB für die unterschiedlichen Übertragungswege, die sich hauptsächlich im Modulationsverfahren, dessen optimale Wahl entscheidend vom Frequenzbereich abhängt, und bei der Fehlerkorrektur unterscheiden:

DVB-T ist für die Übertragung durch terrestrische Senderketten im VHF- bzw. UHF-Bereich. DVB-S ist für die Übertragung durch direktstrahlende Satelliten. Der Benutzer kann zwischen Digital Video Broadcasting-Terrestrial und Digital Video Broadcast – Satellite auswählen.

DVB Wiedergabe-System:

Zur DVB Wiedergabe wird MPEG2 verwendet. Bei vielen Budgetkarten wird nur der MPEG Datenstrom zum Computer geleitet, da kein Hardwaredeko-der in der Hardware vorhanden ist. Dieser DVB Datenstrom muss also softwaretechnisch dekodiert werden (MPEG-Dekodierung) und zur Anzeige gebracht werden. Hierzu muss ein Videowiedergabesystem mit MPEG-Unterstützung erstellt werden. Idealerweise wird ein flexibles System mit verschiedenen Wiedergabeschnittstellen entwickelt, so dass der Endanwender aus verschiedenen Wiedergabetechniken wählen kann. Geplant sind Systeme mit Xine. MPlayer und VLC-Systeme können später entwickelt werden.

Sonstige Features:

Weitere folgende Features, die man standardmäßig in ähnliche Applikationen findet wurden implementiert: Deinterlace, Fullscreen, Screenshot, Volume-Control, Settings (für diverse Einstellungen, die der Benutzer tätigen kann).

7.1.2. Kann-Kriterien

Weitere Bauelemente werden auch nach dem Software-Projekt implementiert, unter anderem:

OSD:

Das On Screen Display (kurz: OSD) ist ein Menü, das über das momentane Bild (z. B. bei einem Fernseher oder Computerbildschirm) eingeblendet wird. Es dient zur Bedienung des Geräts, um Einstellungen vorzunehmen oder um im Teletext zu navigieren. Über Tasten an der Fernbedienung oder am Monitor kann man sich im Menü bewegen. Das OSD ist in modernen Geräten mehrsprachig.

EPG:

Der Electronic Program Guide (EPG), zu deutsch elektronische Programmzeitschrift, ist die digitale Variante einer gedruckten Programmzeitschrift. Mit Hilfe eines EPG kann man sich das Fernsehprogramm aller registrierten Fernsehsender ansehen. Zugeordnet zu jedem Sender erhält man eine Übersicht der Sendungen. Diese Übersicht beinhaltet den Titel und die Uhrzeit (von wann bis wann). Zusätzlich werden zu den einzelnen Sendungen kurze Beschreibungen des Inhalts, teils mit Bildern der Sendungen, angezeigt.

Videotext:

DVB-T ermöglicht durch die erweiterte Übertragung zusätzlicher Daten sogar einen "Videotext der nächsten Generation" mit erweiterter Interaktion und hoch auflösenden Grafiken.

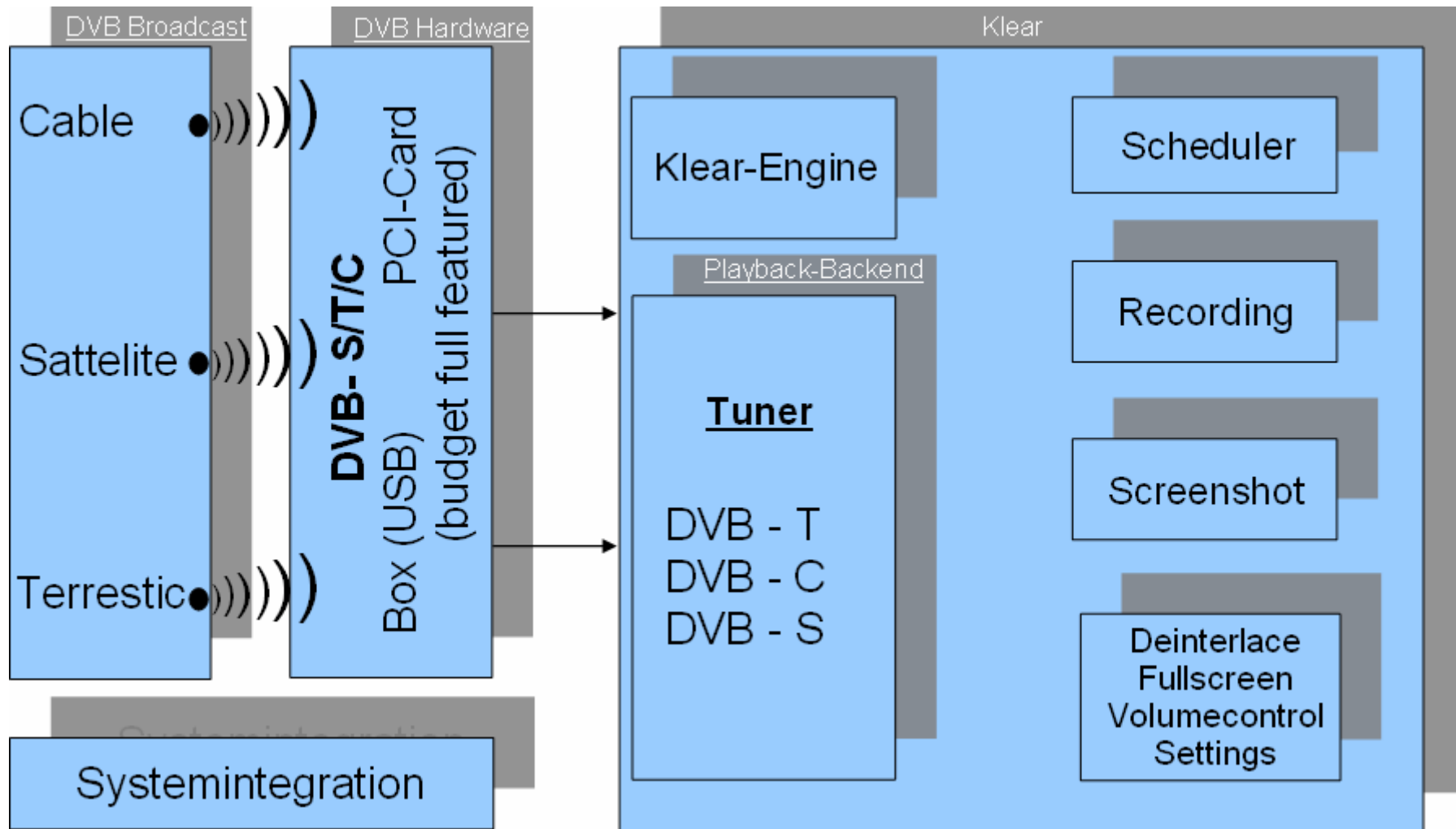
Tastenkombination:

Als Tastenkombination (auch Tastaturkombination, short cut, Tastensequenz) bezeichnet man das gleichzeitige oder aufeinander folgende Drücken mehrerer Tasten in einer bestimmten Abfolge bei Computersystemen.

Durch den Einsatz solcher Kombinationen können zum einen bestimmte Steuerbefehle an das Programm gegeben werden (beispielsweise für „Datei öffnen“, „Fenster schließen“, „Programm beenden“ usw.), aber auch bestimmte Funktionen des Betriebssystems direkt abgerufen werden. (beispielsweise „Starte neues Programm“ usw.)

Der Benutzer soll die Möglichkeit haben, eigene Tastenkombinationen zu definieren.

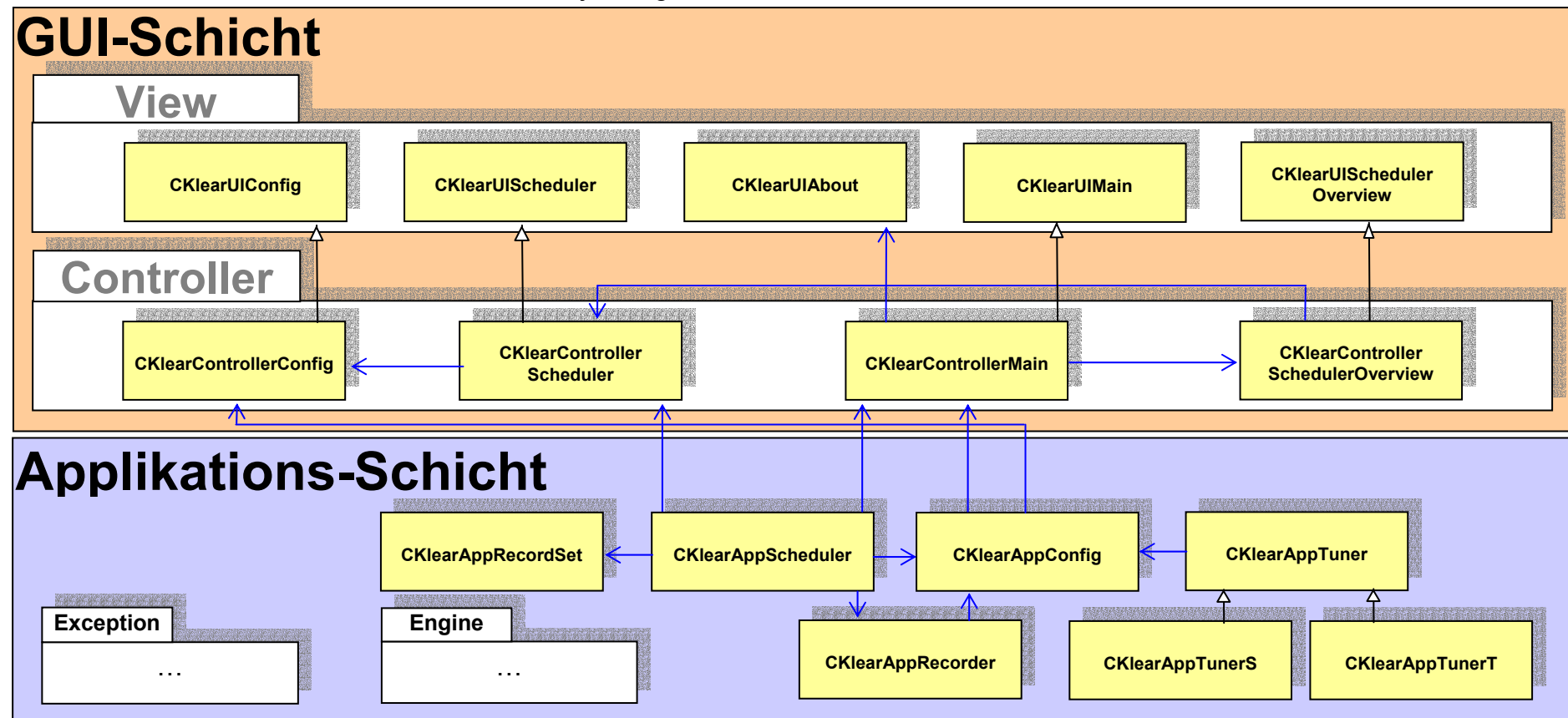
7.2. Grafische Darstellung



8. Abstraktes Klassendiagramm

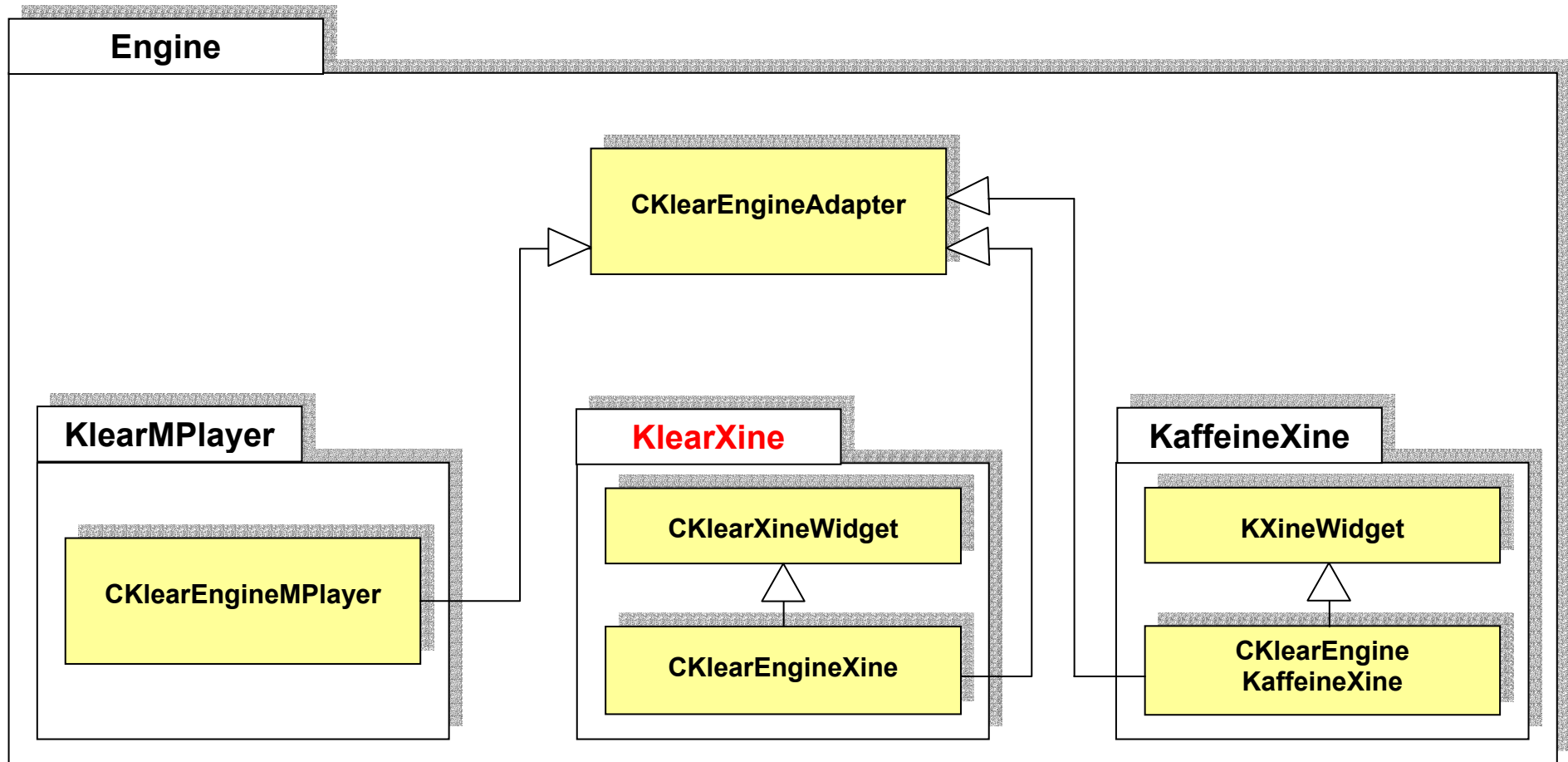
8.1. Schichten-Modell

Das abstrakte Klassendiagramm soll uns einen Überblick verschaffen welche Klassen und Pakete in welcher Schicht auftauchen. Darüber hinaus erkennt man, was für Beziehungen (Assoziationen / Relationen) die jeweiligen Klassen zueinander haben. Wichtig ist hier zu erwähnen, dass die GUI-Schicht, eine Unterteilung durch zwei Pakete aufweist. Das Paket „View“ beinhaltet reine XML-Dateien, die wie schon der Name sagt, für die Ansicht der GUI-Elemente verantwortlich sind. Das Paket „Controller“ bildet eine Schnittstelle zwischen der GUI-Schicht und der Applikations-Schicht. Es gehört aber dennoch zur GUI-Schicht, da alle Controller-Klassen von der jeweiligen UI-Klasse aus der View ableiten.



8.2. Engine

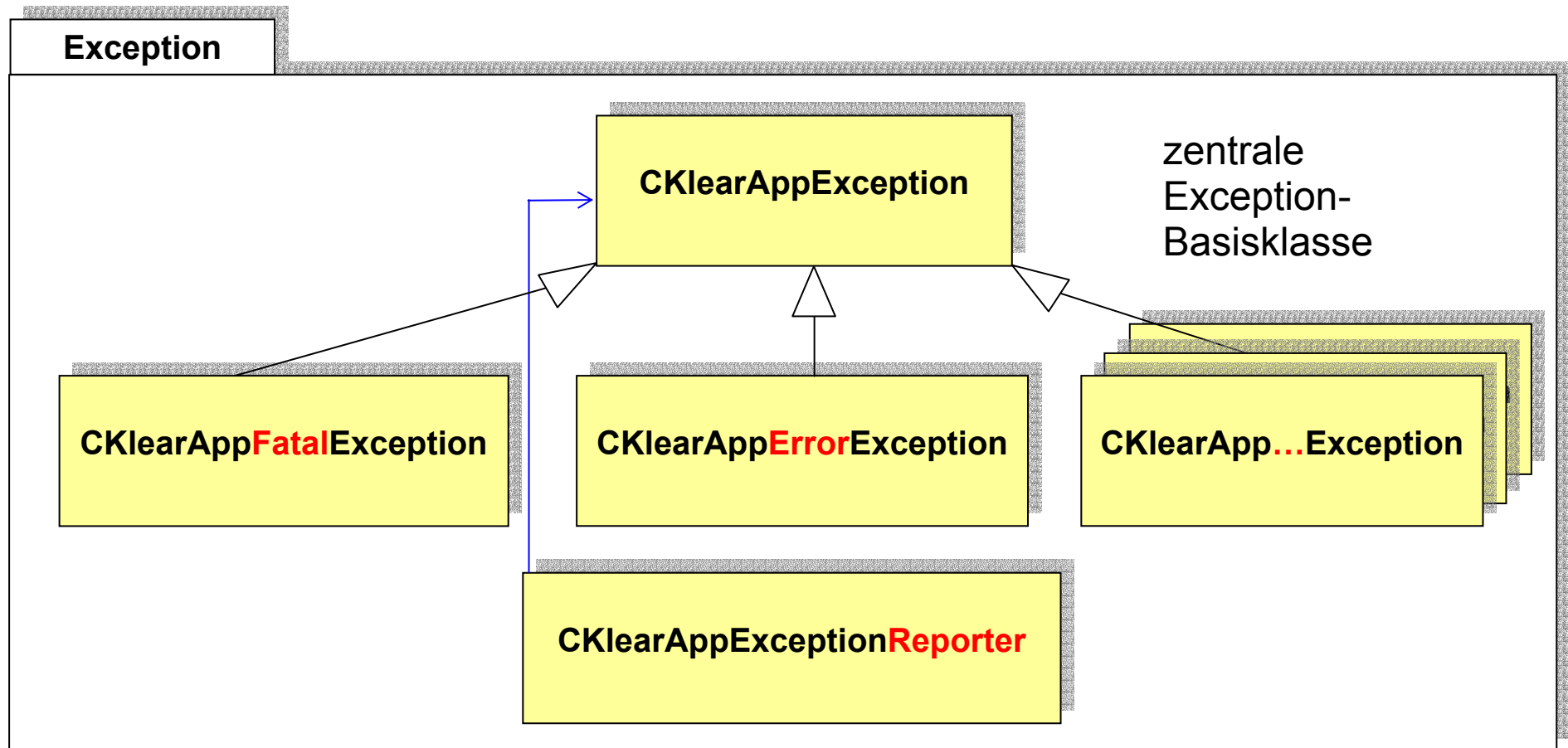
Anhand des abstrakten Klassendiagramms können wir erkennen, dass die Applikations-Schicht mit zwei Paketen bestückt ist. Ein Paket beschreibt die Engine. KlearMPlayer, KlearXine und KaffeineXine sind weitere Pakete, die das Paket Engine beinhaltet. Diese sind die Engines, die wir im Rahmen unseres Projekts benutzt bzw. entwickelt haben. Von jedem dieser Pakete erbt eine Klasse von CKlearEngineAdapter, die als Oberklasse der Engines dient.



8.3. Exception

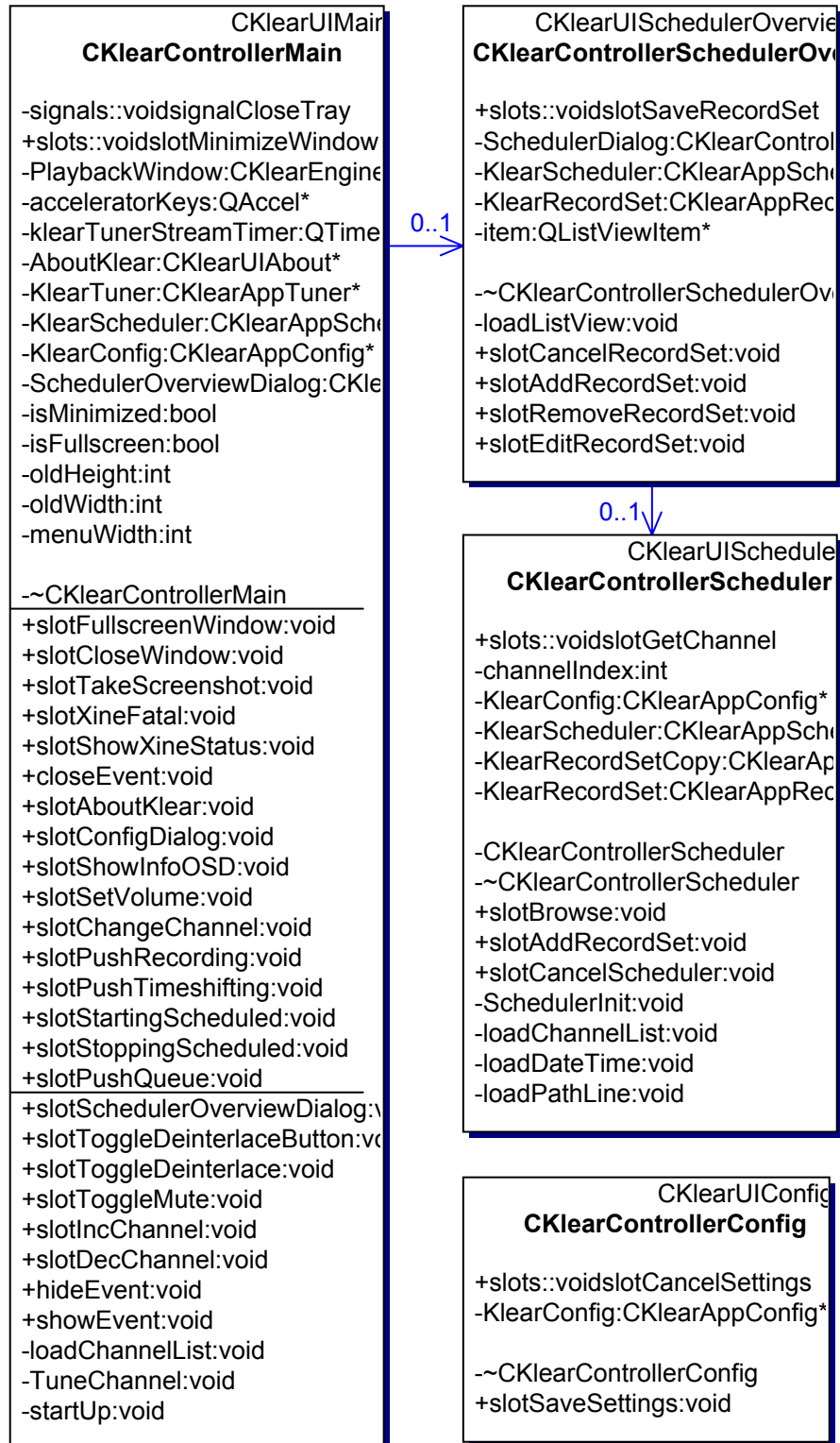
Exception ist das zweite Paket der Applikations-Schicht. Das Paket beinhaltet eine Exception-Basisklasse von der mehrere definierte Exception-Klassen ableiten. „**CKlearApp...Exception**“ kennzeichnet weitere Exception-Klassen.

CKlearAppExceptionReporter ist die einzige Klasse, die nur eine Assoziation mit der Basisklasse bildet. Bei einer Ausnahme würde diese Klasse Ihre eigene GUI unabhängig von der GUI-Schicht starten bzw. anzeigen.

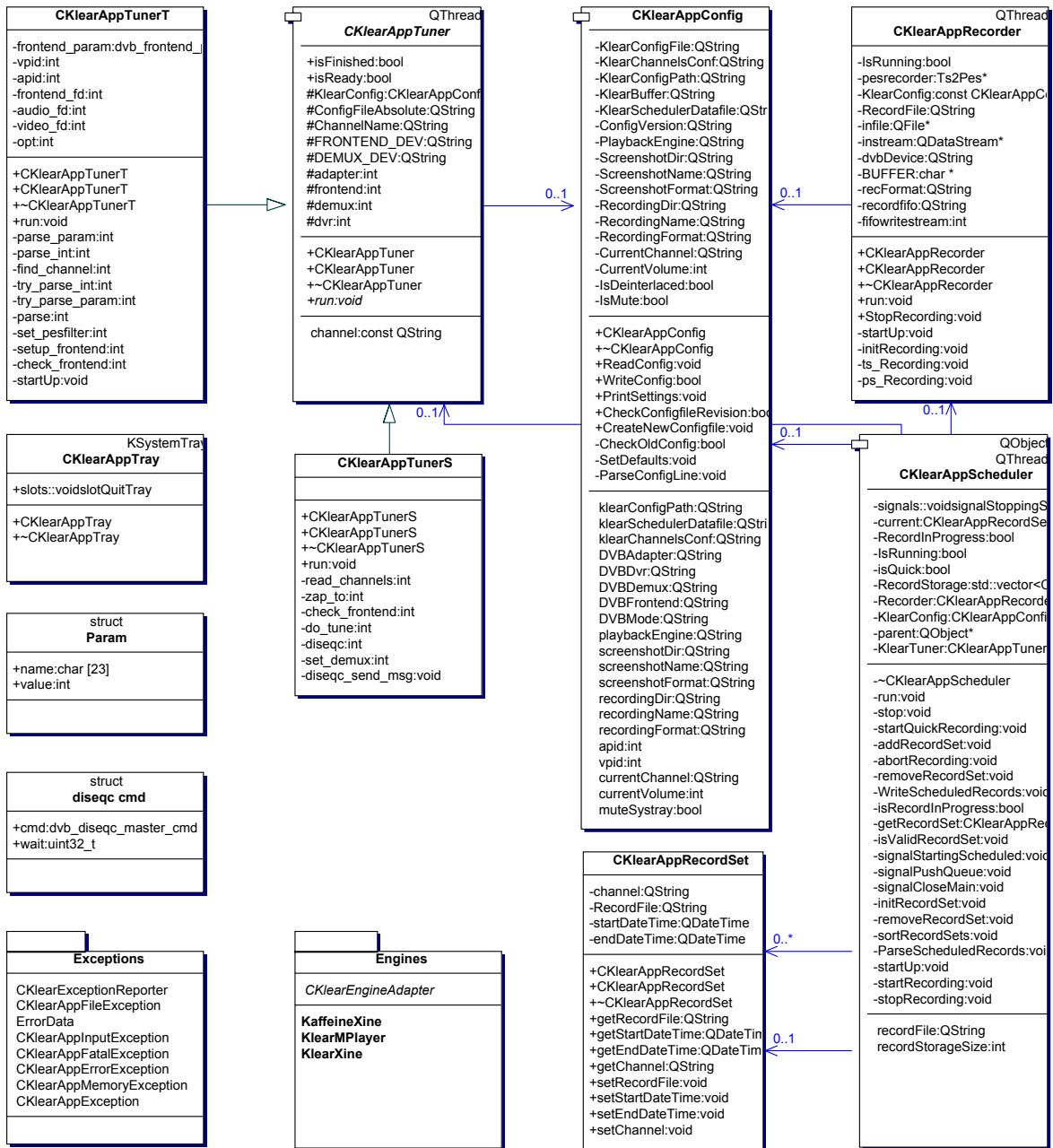


9. Klassendiagramme

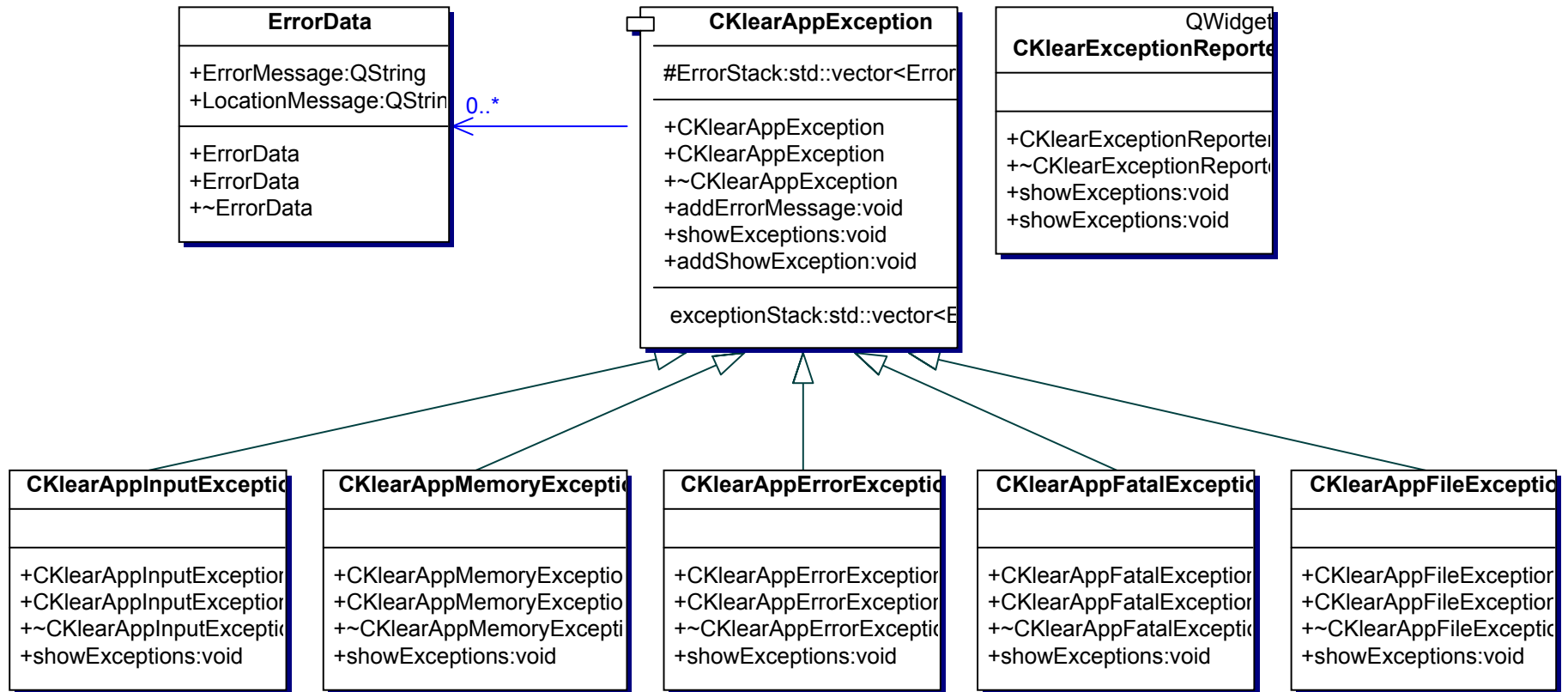
9.1. Controller-Paket



9.2. App-Paket



9.3. Exception-Paket



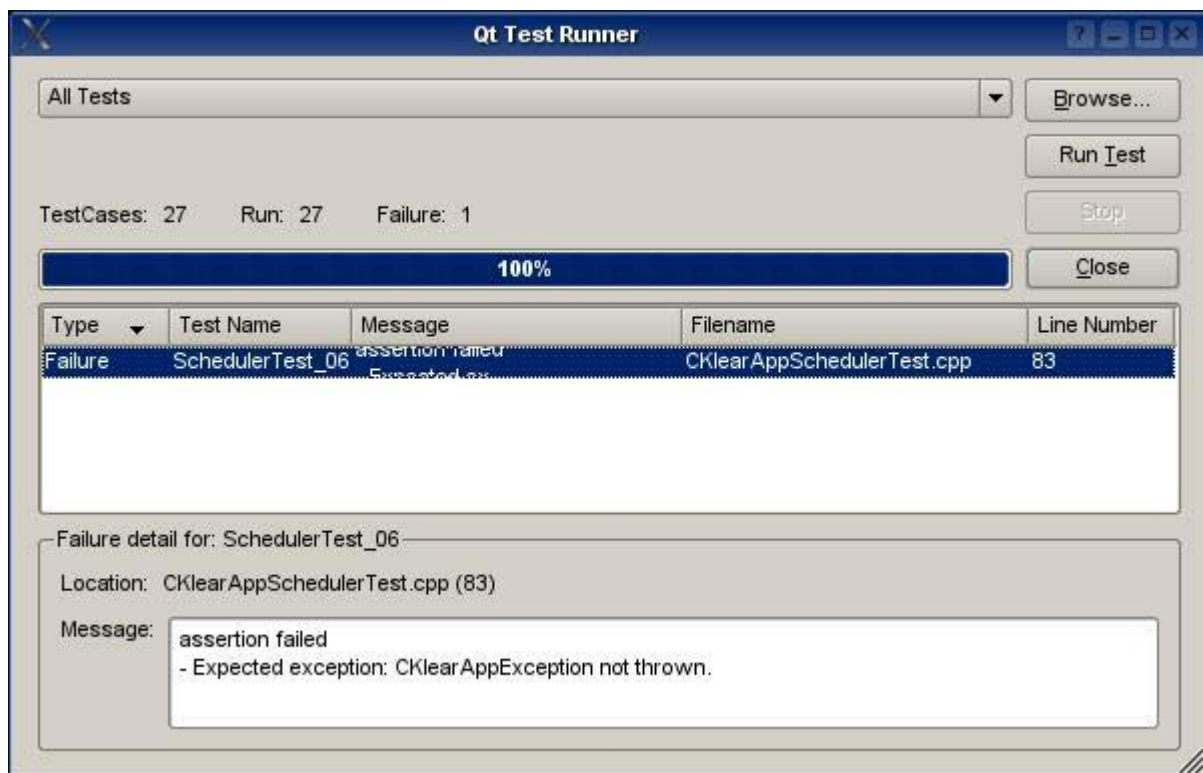
10. Unittests

Unit-Tests (auch Komponententests) sind Teil eines Softwareprozess-Vorgehensmodells (z.B. Extreme Programming). Sie dienen zur Validierung der Korrektheit von Modulen einer Software, z.B. von einzelnen Klassen. Als Voraussetzung für Refactoring kommt ihnen besondere Bedeutung zu. Nach jeder Änderung sollte durch Ablaufenlassen aller Testfälle die Fehlerfreiheit überprüft werden. Beim testgetriebenen Programmieren, auch TestFirst-Programmieren genannt, werden die Unit-Tests parallel zum eigentlichen Sourcecode erstellt und gepflegt. Dies ermöglicht bei automatisierten, reproduzierbaren Unit-Tests, die Auswirkungen von Änderungen sofort nachzuvollziehen. Der Programmierer entdeckt dadurch sicher ungewollte Nebeneffekte oder Fehler durch seine Änderung. Ein Komponententest ist ein ausführbares Codefragment, das das sichtbare Verhalten einer Komponente (z.B. einer Klasse) verifiziert und dem Programmierer eine unmittelbare Rückmeldung darüber gibt, ob die Komponente das geforderte Verhalten aufweist oder nicht. Durch diese Rückmeldung wird die Wartbarkeit z.B. durch Refactoring vereinfacht bzw. erst ermöglicht. Komponententests sind ein wesentlicher Bestandteil der Qualitätssicherung in der Softwareentwicklung.

Unsere Unittets basieren auf CppUnit.

CppUnit ist die Portierung von JUnit auf C++. Ursprünglich wurde es von Michael Feathers geschrieben, ist jetzt aber ein offenes Projekt unter SourceForge.

Zusätzlich nahmen wir den Qt Test Runner (siehe Bild), als grafische Oberfläche für die Unittests.



11. Installationsanleitung

11.1. Systemvoraussetzungen

Perl – um die Applikation laufen zu lassen

qmake – für das starten der “automake-system”

libxine-devel – um die Xine als Video-Backend benutzen zu können

qt3-devel – für die Darstellung der GUI-Oberfläche und das benutzen von QT

Sie können qmake auch ohne Perl starten. Zu beachten wären aber die nötigen Includes und Libraries, die angebunden werden müssen..

11.2. Installation

Extrahieren Sie folgende Datei:

```
tar xvzf klear-0.x.tar.gz
```

und starten Sie folgenden Befehl:

```
./configure
```

Falls der Befehl nicht ausgeführt wird, so installieren Sie die nötigen Includes unter libs/includes oder geben Sie folgenden Befehl an:

```
./configure --extra-path=/path/to/libs
```

Nach erfolgreicher Konfiguration starten Sie:

```
make
```

und installieren die jeweiligen Binaries:

```
make install
```

12. Bedienungsanleitung

12.1. Hauptfenster

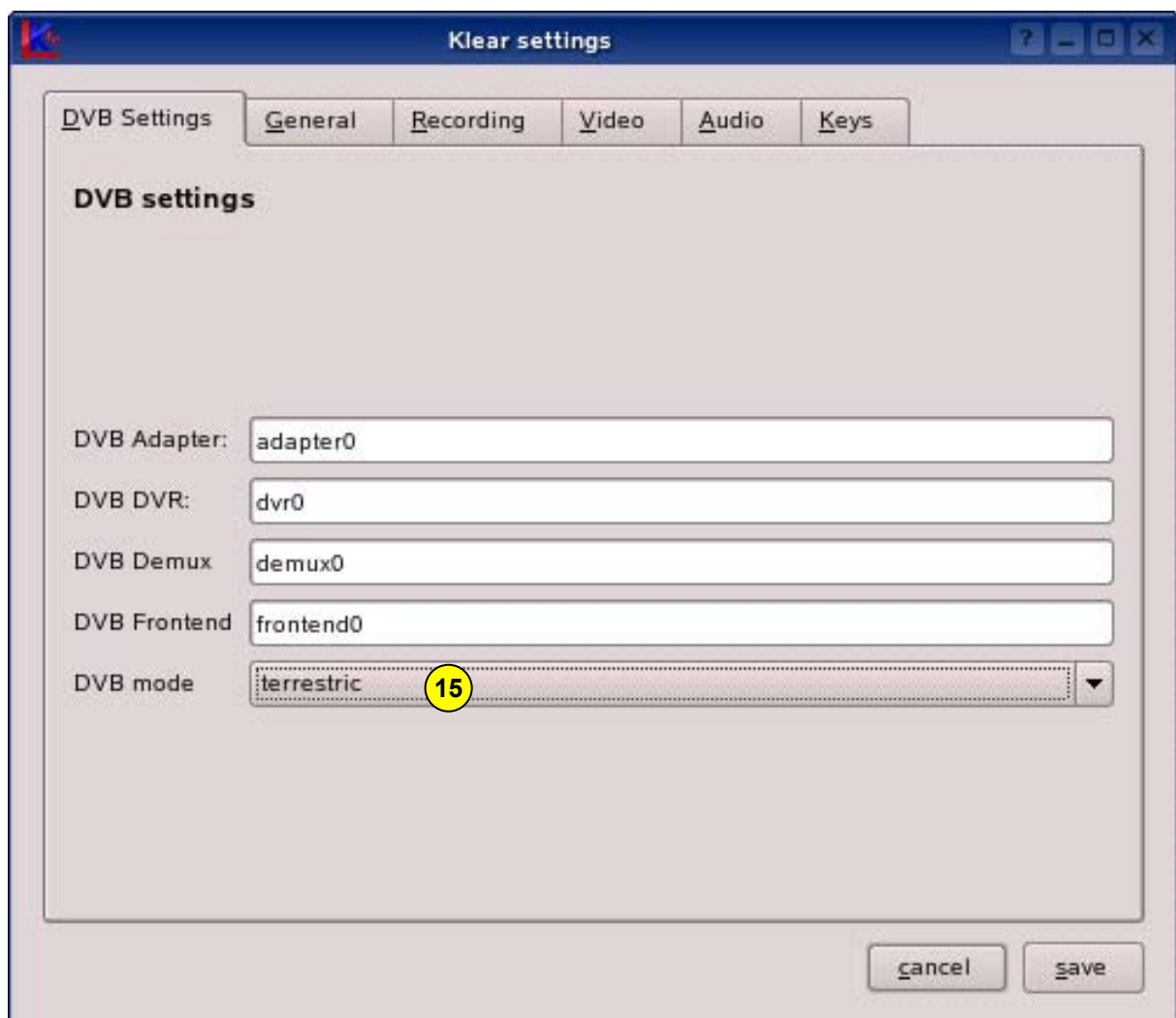


(1)	Das großzügig angelegte Display ist der Blickfang des Hauptfensters.
(2)	Durch Tatigung dieses Buttons gelangt man zu den Klear-Einstellungen.
(3)	Der Benutzer hat die Moglichkeit Informationen zu beziehen.
(4)	Hilfestellungen stehen mit diesem Button zur Verfugung.
(5)	Die vorhandenen Kanale werden auf der Kanalliste angezeigt.
(6)	Durch den Regler lasst sich die Lautstarke einstellen.
(7)	Anhand des Lautsprechers kann man auf Stumm umschalten.
(8)	Mithilfe des Quick-Recording-Buttons lasst sich das empfangene Signal aufzeichnen.
(9)	Durch diesen Button besteht auch die Moglichkeit des zeitgesteuerten Aufzeichnens.
(10)	Ermoglicht die Aufnahme eines Standbildes (Screenshot).
(11)	Das Zeilensprungverfahren lasst sich durch den Deinterlace-Button besser erzielen.
(12)	Vollbilddarstellung des Playbacks.
(13)	Minimieren des Fensters. Kanalliste, sowie alle Buttons werden ausgeschaltet.
(14)	Beenden der Applikation

12.2. DVB Settings

Durch das Anklicken des Schraubenziehers gelangt man zu den Klear-Settings. Es stehen verschiedene Reiter zur Verfügung, die zur Einstellung der Applikation gedacht sind. Unter dem Reiter DVB Settings lassen sich interne Einstellungen für das ansprechen der Settop-Box ändern bzw. regeln. So kann man auch unter DVB mode (15) den entsprechenden Tuner aussuchen:

- DVB-S für die Übertragung durch direktstrahlende Satelliten
- DVB-C für die Übertragung über Kabelnetze
- DVB-T für die Übertragung durch terrestrische Senderketten



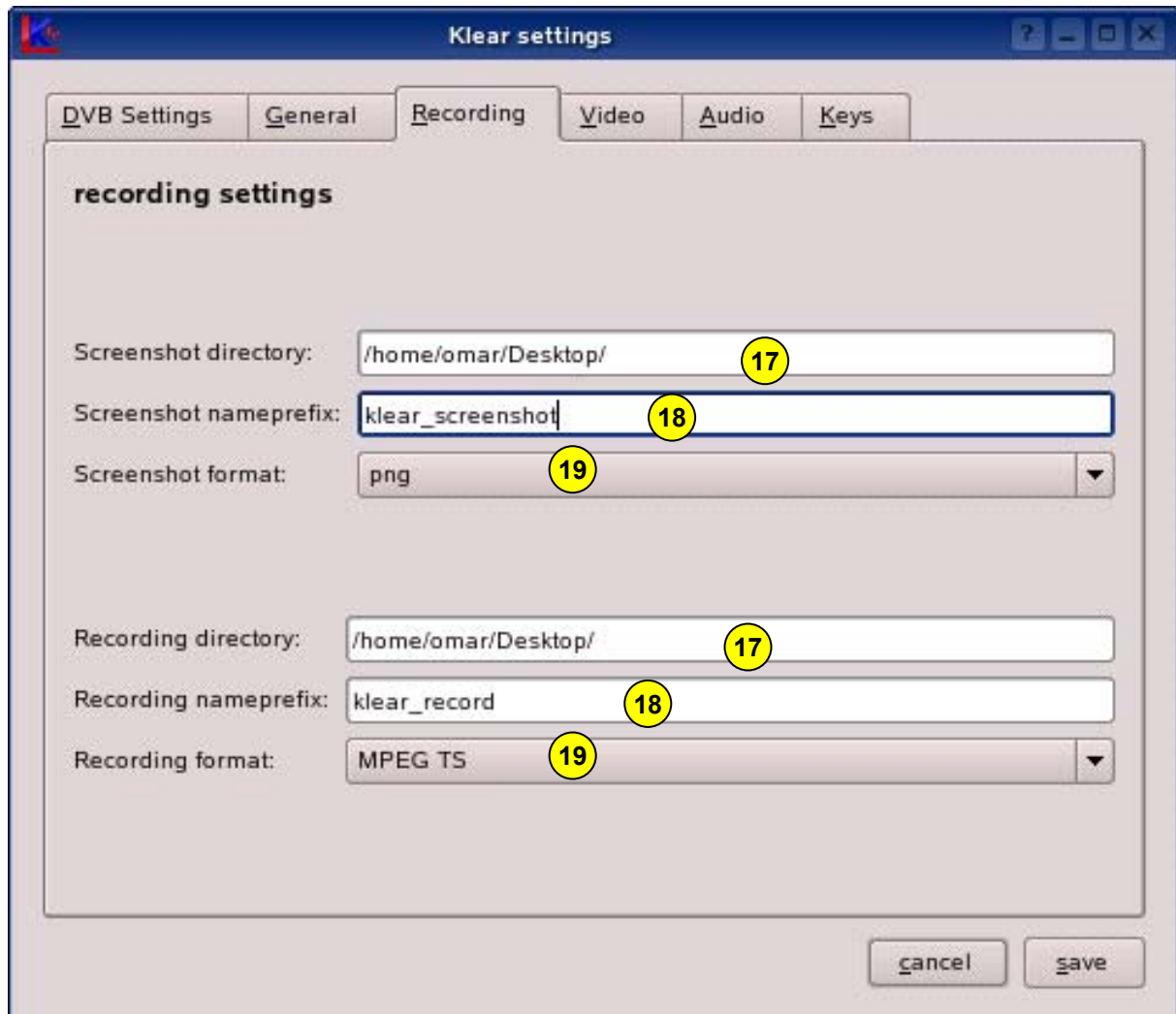
12.3. General

Der Reiter General steht für allgemeine Einstellungen. Momentan sieht man lediglich ein Choice-Feld (16), das die Stummschaltung bewirkt, sobald man die die Applikation bzw. das Fenster minimiert.



12.4. Recording

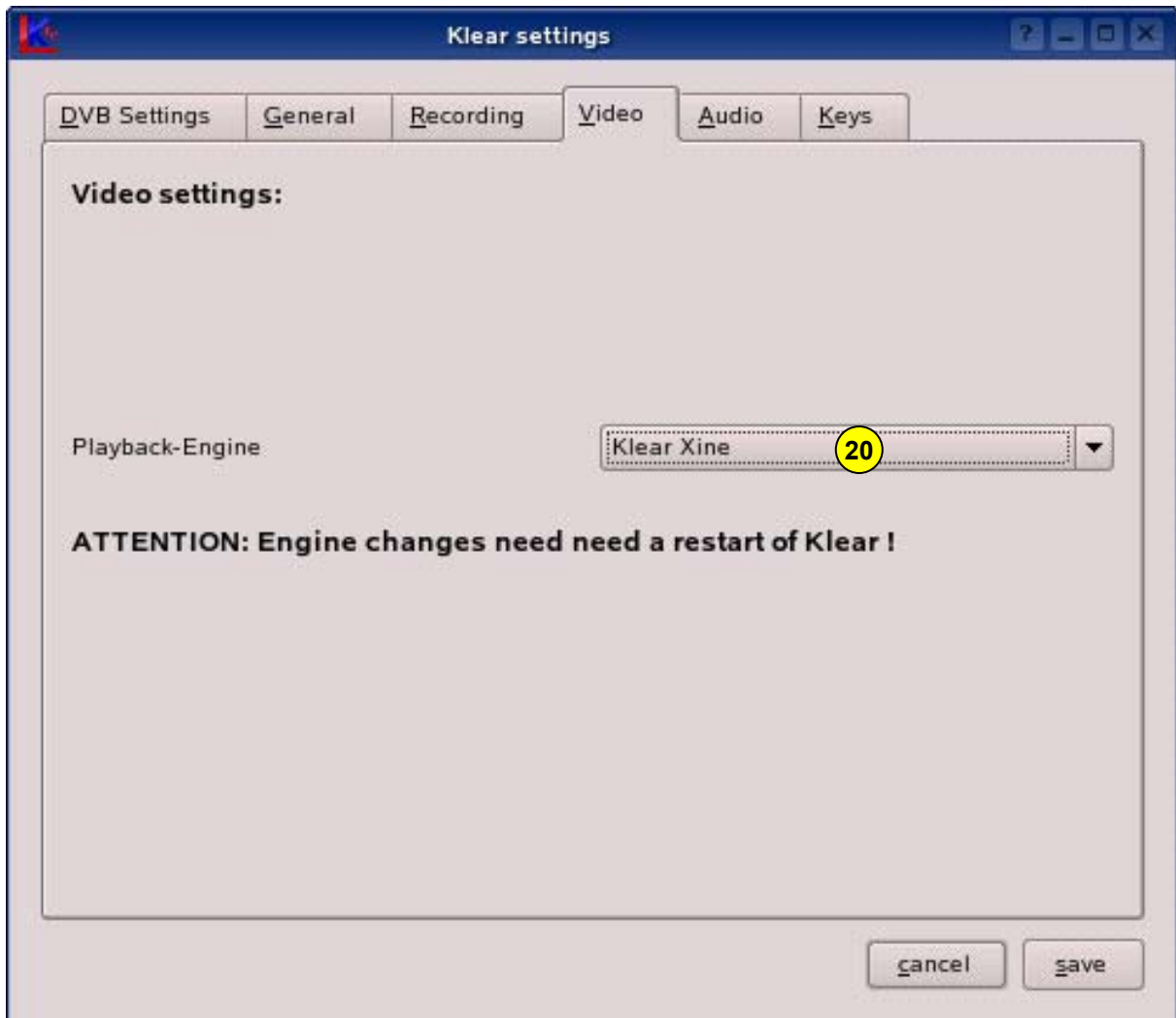
Unter Recording werden die Einstellungen eingegeben, die zum abspeichern von Screenshots oder Aufnahmen benötigt werden. So kann der Benutzer sowohl das Zielverzeichnis (17) als auch den Namen (18) der Datei selber angeben. Ihm steht auch die Möglichkeit zu entscheiden, in welches Format (19) sein Screenshot bzw. Aufnahme abgespeichert werden soll.



12.5. Video

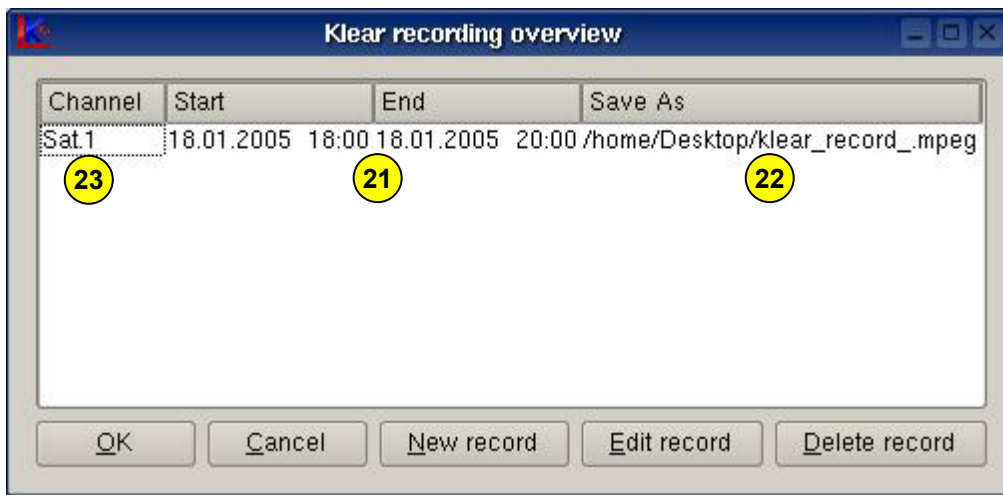
Durch den Reiter „Video“ hat man die Möglichkeit eine Playback-Engine (20) auszusuchen. Anhand der Klassendiagramme erkennt man, dass Klear zu drei Engines verfügt (KlearXine, KaffeineXine und KlearMPlayer).

Nach entsprechender Auswahl muss man jedoch die Applikation neustarten.



12.4. Recording Overview

Durch drücken auf den Button mit der Uhr, der für den Scheduler gedacht ist, gelangt man auf dieses Fenster. Hier lassen sich Aufzeichnungen speichern, editieren oder löschen. Auf einen Blick sieht man eine Liste aller Zeitgesteuerten Aufnahmen, die noch nicht die entsprechende Startzeit bzw. Endzeit (**21**) erreicht haben. Gleichzeitig erkennt man, wo die entsprechende Aufnahme abgespeichert (**22**) wird und von welchem Kanal (**23**) aus aufzunehmen soll.



12.5. Add new / Edit Klear record

Um die zeitgesteuerte Aufnahme zu speichern, muss das entsprechende Kanal ausgesucht werden. Gleichzeitig wird ein Standardzielverzeichnis angegeben, dass man manuell ändern kann bzw. Mithilfe des Durchsuchen-Buttons (**24**) zum Zielordner navigieren kann. Anzugeben muss dementsprechend auch die Start- und Endzeit, die natürlich in der Zukunft liegen müssen.



13. Erfahrungen und Probleme

Wir wählten zu Beginn des Semesters *Extreme Programming* als Vorgehensmodell, da wir uns noch nicht über die technischen Details unsere Projekts im Klaren waren. Die Entwicklung sollte daher auch komplett neue Erfahrungen in Hard und Software mitbringen. Wir starteten mit insgesamt fünf Entwicklern. Hiervon hatte aber nur ein Student praktische Erfahrung mit QT und C++ unter Linux. Am Ende war dies aber kein grosser Nachteil. Die Einarbeitungszeit in QT war recht kurz und die Probleme mit C++ konnten in Teamarbeit immer recht schnell gelöst werden.

Durch das “Extreme Programming” entstanden in den ersten Entwicklungsmonaten mehrere tausend Zeilen an Code. Zusätzlich wurden aber auch viel Code wieder verworfen, da die Tests zeigten, dass die Technologien ungeeignet waren (beispielsweise die direkte Verwendung von ffmpeg zur Videotranskodierung). Die Nutzung von XP als Vorgehensmodell verursachte aber auch, dass im Laufe des Projekts generelle Änderungen (zum Beispiel im Styleguide, Codingguide oder der Architektur) weitere Änderung an sehr vielen Stellen im Code mit sich brachte. Vor allem im SWP2 beschäftigten wir uns viel mit Refactoring, weshalb wir deutlich weniger neuen Funktionen und Versionen erarbeiten konnten. Auch vernachlässigten wir zu Beginn des Projektes ein wenig die Qualitätssicherung (Exceptions, Unittests). Dies mussten wir durch Mehrarbeit im zweiten Projektteil dann aufarbeiten. Zudem verließ uns ein Entwickler, der primär für die Dokumentation und Klassendiagramme zuständig war nach SWP1, so dass wir auch diese Aufgaben intern noch verteilen mussten.

Auch stellten wir Klear als OpenSourceSoftware früh und oft der Öffentlichkeit vor. Das Feedback aus aller Welt konnten wir wieder in unser Projekt einfliessen lassen. So haben wir auf Grund der großen Nachfrage die Unterstützung für DVBS (Satellitenfernsehen) in Klear eingebaut. In unserem ursprünglichen Plan war dies nicht vorgesehen. Durch die Vielzahl an Testern, die wir so erreichten konnten wir auch Probleme lokalisieren, die bei uns auf den Entwicklungssystemen nicht aufgetreten sind. Viele Tester und viel Feedback sind daher ein deutliches Plus in der Entwicklung.

Durch das Vorgehensmodell entstanden uns weiter Vor, aber auch Nachteile in der Lehrveranstaltung. Da nur zwei Gruppen XP wählten, war die Lehrveranstaltung natürlich sehr auf die klassischen Entwicklungsmodelle ausgerichtet. Die für uns nützlichen Unittestund MocktestVorlesungen kamen zu spät. Gewünschte CodeRichtlinien und Spezifikationen wurden manchmal erst im SWP2 bekannt gegeben, wenn die anderen Gruppen in die Implementierungsphase eintraten. Da hatten wir aber bereits mehrere tausend Zeilen Code erzeugt und mussten diese nachbearbeiten. Wir trafen uns jede Woche zur Rücksprache mit Herr Prof. Knabe und konnten aber so die Entwicklungsrichtung zusammen bestimmen und Fehler frühzeitig beheben. Auch konnten wir etwas dem Druck der Meilensteine entgehen, was für uns ein etwas entspannteres Arbeiten bedeutete.

Zusammenfassend ist daher zu sagen, dass die Anwendung im SWP1 also die grobe Struktur und vor allem den nötigen Funktionsumfang und im SWP2 die verbesserte Architektur, einen deutlichen Qualitätsanstieg und die Stabilität bekommen hat. Neben dem enormen Wissenszuwachs in den verwendeten Technologien war vor allem die Teamarbeit und die Teamorganisation eine wichtige Erfahrung. Auch "extreme programming" war eine wichtige Erfahrung. Fehler, die man in klassischen Vorgehensmodellen vermeidet, treten hier doch wieder auf. Sie lassen sich jedoch durch viel praktische Arbeit und Erfahrung minimieren. Diese muss man jedoch erst Sammeln. Den ersten Schritt haben wir somit getan.

14. Daten und Fakten

- Vier Projektmitglieder
- Ca. 25 000 LOC inklusive GUI Komponenten und Unittests
- Neun öffentliche Releases
- Ca. 5 000 Downloads der PreviewVersionen
- Ca. 50 User auf der Mailingliste
- Binärpakete u.a. für Debian und Mandriva verfügbar
- Über 100 eMails an Feedback aus der ganzen Welt