PSE 2012

OQAT

Objective Quality Assessment Toolkit

Praxis der Softwareentwicklung
WS 2012

E n t w u r f s d o k u m e n t

Auftraggeber
Karlsruher Institut für Technologie
Institut für Technische Informatik
CES - Chair for Embedded Systems
Prof.Dr.J.Henkel

Betreuer: S. Kobbe

Auftragnehmer

Name	E-Mail-Adresse
Eckhart Artur	artur.eckhart@gmail.com
Ermantraut Georg	georg.ermantraut@gmail.com
Leidig Sebastian	sebastian.leidig@gmail.com
Monev Alexander	bcclan@mail.bg
Sailer Johannes	johsailer@gmail.com

Karlsruhe, 20.5.2012

Inhaltsverzeichnis

1 Einleitung

Die Anwendung Öbjective Quality Assessment Toolkit", welche im Auftrag des CES hergestellt wird, wird wie im Pflichtenheft angekündigt nach dem "Modell View ViewModellËntwurfsmuster angefertigt. Hierbei ist das Ziel eine durchgehende Trennung der Aufgaben zu erreichen.

Das Subsystem Modell enthält alle statischen Daten, wie Videos und Projekte. Darüber hinaus wird die Möglichkeit diese auf die Festplatte zu schreiben realisiert. Auserdem werden die Plugins, was Filter, Metriken sowie die Komponenten zum Umgang mit Videos beinhaltet im Modell abgelegt.

Das ViewModell ist die Schnittstelle zwischen Modell und View. Es ist dafür zuständig erhaltene Eingaben des Benutzer zu verarbeiten und diese gegebenenfalls an das Modell weiterzureichen. Jeder View-Komponente ist ein ViewModell zugeordnet.

Die View zeigt den Aktuellen Zustand des Modells an. Da sich die View-komponenten nicht Sinnvoll im Entwurf darstellen lassen, weil sie im XAML Code stehen, wurde auf sie im Klassendiagramm verzichtet.

2 Gesamtentwurf

2.1 Model

2.1.1 Private Klassen

Project : Klasse
Alle für das Projekt relevanten Daten befinden sich hier. Dies beinhaltet einen Pfad, einen Namen und eine Beschreibung. IMemorizable wird implementiert.

OqatProperties : Klasse
Dies sind die globalen Einstellungen der Anwendung: Sprache sowie Pfad zum Pluginverzeichnis.

Caretaker : Klasse
Der Caretaker ist zuständig für das Laden und Speichern von Mementos von und auf die Festplatte. Er stellt Methoden bereit, damit andere Klassen Zugriff auf ihre Mementos haben und wird als Singelton realliesiert. Der Caretaker verwaltet eine Tabelle von Strings und dazu gehörigen Listen von Mementos.

SmartNode : Klasse
Videos eines Projektes werden durch die SmartNode in einer Baumstruktur zusammengestellt, um sie dann dem SmartTree der View zur Verfügung zu stellen. Eine SmartNode kann mehrere Kinder-SmartNodes haben.

2.1.2 Oqat Public Ressources

Video : Klasse
Implementiert IMemorizable. Im Video werden ein Pfad zu einer Videodatei, ein Videoinfo-Objekt sowie mögliche Extraressourcen gelegt. Falls es sich um ein Analysevideo handelt, werden zusätzlich entsprechende Daten hier abgelegt.

VideoEventArgs : Klasse
Enthält ein Video und ein Boolean, der TRUE ist, falls es sich um ein Referenzvideo handelt. Wird von Events als Argument benutzt.

Memento : Klasse
Ein Memento speichert den Zustand eines Objektes im Arbeitsspeicher. Es werden ein Name und gegebenenfalls auch ein Pfad zum Speichern auf die Festplatte bereitgestellt.

MementoEventArgs : Klasse
Enthält zwei Strings: Name eines Mementos und Referenz zu einem Plugin. Wird von Events als Argument benutzt.

IMemorizable : Interface
Es wird ermöglicht, den Zustand einer Klasse als Memento zu speichern und vorherige Zustände zu laden.

IvideoInfo : Interface
Falls ein Videoformat Voreinstellungen braucht bietet IVideoInfo die Möglichkeit diese abzulegen.

2.2 View

2.3 ViewModel

2.3.1 Oqat Organisation

Alle sichtbaren VM implementieren eine Methode onToggleView, die für den Ansichtswechsel zuständig ist und eventgesteuert funktioniert.

OqatApp : Klasse
Die Klasse wird beim starten des Programms als erstes angesprochen und initialisiert die Anwendung und den Pluginmanager.

VM Oqat : Klasse
Diese initialisiert die ViewModel Komponenten. Dazu gehören Willkommensbildschirm, das Projekt, Menü, Presentation und die Pluginlisten. Die Klasse stellt auch eine Delegatemethode zur Ansichtswechsel bereit.

VM ProjectExplorer : Klasse
Besteht aus einem SmartTree und einem Dateiexplorer, um diese in der GUI bereitzustellen. Er kann durch den Dateiexplorer Events zum Auswählen und Laden von Videodateien ins Projekt abfangen.

PluginManager : Klasse
Der Pluginmanager ist die Schnittstelle zwischen Oqat und den Plugins. Er stellt Methoden zum Abrufen von Plugins und den zugehörigen Mementos. Außerdem werden Events mit deren Handlern bei ihm registriert. Er ist als Singelton realisiert. Der Pluginmanager enthält einen Pfad zum Pluginordner sowie eine Tabelle, die den Plugintypen die zugehörigen Plugins zusammen mit deren Namen zuordnet.

VM Pluginlists : Klasse
Die Pluginlist verwaltet die hinzugefügten Metriken und Filter, sowie die anderen Plugins. Er kann die Events entryClick und entrySelect abfangen. Bei entryClick wird ein Filter oder eine Metrik vom Container ausgewählt und die zugehörigen Einstellungen angezeigt. Bei entrySelect wird der ausgewählte Filter oder Metrik zu einer Makrowarteschlange hinzugefügt. VM Pluginlists stellt eine Delegatemethode bereit, die neuerstellte Mementos nach dem Abfangen des jeweiligen Events zu dem jeweiligen Plugincontainer hinzufügt.

VM Menu : Klasse
Dieses ViewModel ist für die Verwaltung der Menüleiste und der Menüitems zuständig.

VM Welcome : Klasse
Ist der Willkommensbildschirm, der beim Programmstart die zuletzt benutzten Projekte anzeigt.

VM OptionsDialog : Klasse
Optionsfenster für allgemeine Einstellungen von Oqat.

VM Presentation : Klasse
Der zentrale Visualisierungsbereich wird hiermit verwaltet. Dazu nutzt es die Presentationplugins und bietet die Möglichkeit, den angezeigten Bereich in den Originalzustand zu versetzen.

ViewTypeEventArgs : Klasses
Enthält ein Objekt vom Typ ViewType. Wird von Events als Argument benutzt.

projectEventArgs : Klasse
Enthält ein Objekt vom Typ Projekt. Wird von Events als Argument benutzt.

ViewType : Enumeration
In dieser Enum sind die verschiedenen Typen von Ansichten aufgelistet.

VM ProjectOpenDialog : Klasse

VM VidImportOptionsDialog : Klasse

2.3.2 Macro

VM Macro : Klasse
Legt fest, was bei einem Macro-bezogenen Event passiert. VM Macro kann Events vom Typ onEntrySelect, onMacroSave, onStartProcess und onToggleView abfangen. OnEntrySelect dient zur Auswahl von Filtern oder Metriken, die zum Macro zugefügt werden. OnMacroSave dient zum Speichern des Macrofilters. OnStartProcess startet den Macrofilter oder –analysevorgang. VM Macro enthält als Attribute ein Objekt vom Typ Macro, ein Objekt vom Typ Viewtype sowie Referenzen zu Videoobjekten.

Macro : Klasse
Enthält eine Liste von MacroEntry-Objekten und implementiert IMacro.

PF MacroMetric : Klasse
Erbt von Macro, implementiert IMacro und IMetricOqat. Diese Klasse bietet Methoden zur Analyse von Frames bzw. Videos an. Die Liste von MacroEntry-Objekten enthält in diesem Fall Referenzen zu Metriken, die durch diese Methoden angewandt werden können.

PF MacroFilter : Klasse
Erbt von Macro, implementiert IFilterOqat. Die Liste von MacroEntry-Objekten enthält in diesem Fall Referenzen zu Filtern, die von einem MacroFilter durch die jeweiligen Methoden der Klasse PF MacroFilter auf Bilder bzw. ganze Videos angewandt werden können.

ExtraResourcesEventArgs : Klasse
Enthält Pfad zu Extraressourcen sowie deren Typ. Wird von Events als Argument benutzt.

2.4 Plugins

2.4.1 Präsentation

PP Player : Klasse
Stellt einen Container dem VideoSourcePlayer zur Verfügung. Lädt und entfernt die Videos, die von dem VideoSourcePlayer gezeigt werden sollen.

PP Diagramm : Klasse
Stellt Oxyplot einen Container zum Zeichnen von Diagrammen zur Verfügung. Lädt und entfernt Videos, deren Attributen dann im Diagramm dargestellt werden können.

VideoSource : Klasse
Implementiert IVideoSource von AForge. Implementiert Methoden zum Abspielen eines Videos.

PP DatPlayer : Klasse
Erbt von PP Player und bietet dem VideoSourcePlayer die Möglichkeit, MotionVektoren als Overlays darstellen.

PS YuvVideoHandler : Klasse
Erbt von IVideoHandler. Der YuvVideoHandler holt sich die Frames von einem YUV-Video bzw. schreibt sie wieder in das Video.

YuvVideoInfo : Klasse
Enthält Informationen, die benötigt werden, um ein YUV-Video abzuspielen: Anzahl Bilder und Auflösung.

VM Properties YuvVideoHandler : Klasse

2.4.2 Filter

Die Plugins vom Typ Filter enthalten die jeweilige Arithmetik für den jeweiligen Filter. Spezielle Filter wie z.B. Blur werden als Memento abgelegt. Die dazugehörige VM kontrolliert das Einstellungsfenster der Mementos. Es wird IFilterOqat implementiert.

2.4.3 Metric

Die Plugins vom Typ Metric enthalten die jeweilige Arithmetik für die jeweilige Metrik. Es wird IMetricOqat implementiert.

2.4.4 Oqat Public Ressources

IPlugin : Interface
Jede Klasse, die ein Plugin ist, muss dieses Interface Implementieren. Namenattribut, Plugintypattribut und Methoden zur Verwaltung von Events (mittels Tabellen von EventTypen und den zugehörigen Delegatemethoden bzw. Eventobjekten) werden dann von Iplugin geerbt. IPlugin implementiert IMemorizable.

IPresentation : Interface
Implementiert das IPlugin Interface. Plugins vom Typ Presentation müssen dieses Interface implementieren. IPresentation bietet Methoden, um ein Video zu laden oder entfernen, und legt fest, was beim Entladen von Presentationplugins passiert.

IMetricOqat : Interface
Implementiert das IPlugin Interface. Metriken müssen dieses Interface umsetzen. Es bietet eine Methode an, die für zwei Bitmap-Objekte ein IAnalysisInfo-Objekt zurückgibt.

IFilterOqat : Interface
Implementiert das IPlugin Interface. Filterplugins müssen dieses Interface umsetzten. Es bietet eine Methode zur Bearbeitung von Bildern an, die ein Bitmap annimmt und wieder ein Bitmap zurückgibt.

IVideoHandler : Interface
Implementiert das IPlugin Interface. Das Interface stellt Methoden zur Verfügung, die mit Frames in einem Video umgehen, indem man den Methoden eine Framenummer übergibt. Es bietet eine Methode an, mit der man sich einen bestimmten Frame holen kann, sowie eine Methode, an die man eine Framenummer und ein Offset übergibt und sich somit einen Array von Frames (Bitmaps) aus einem Video holt. Analog funktionieren die Methoden zum überschreiben von Frames.

IMacro : Interface
Implementiert das IPlugin Interface. Ein Objekt vom Typ Macro muss dieses Interface implementieren. Es verwaltet eine Liste von Objekten vom Typ MacroEntry und bietet eine Methode, die aus so einer Liste und einem beliebigen Namen ein Memento-Objekt erstellen kann.

MacroEntry : Klasse
Enthält zwei Stringattribute: eine Referenz zu einem Memento sowie eine Referenz zu einem Plugin.

MacroEntryFilter : Klasse
Erbt von MacroEntry. Enthält zwei Double-Attribute: startFrameRelative und endFrameRelative. Ein MacroFilter kann auf Teile von Videos, die eine verschiedene Anzahl Frames besitzen,

angewandt werden. Daher ist es sinnvoll, die Start- und Endframes für den Filtervorgang nur relativ zur gesamten Anzahl Frames anzugeben.

MacroEntryMetric : Klasse
Enthält zwei Attribute vom Typ Video. Das sind die Videos, auf die eine Macrometric angewandt wird.

AnalysisInfo : Klasse
Eine Metrik schreibt ihre Ergebnisse zuerst in ein AnalysisInfo-Objekt, damit dann später in das neue Video die Daten eingetragen werden können.

PluginType : Enumeration
Typen von Plugins

PresentationPluginType : Enumeration
Typen von Presentationplugins

EventType : Enumeration
Typen von Events

2.4.5 Externe Klassen

System EventArgs : Klasse
Die verwendeten Events nutzen zur Argumentübergabe jeweils eine Klasse die von EventArgs erbt.

VideoSourcePlayer : Klasse
Der bereitgestellte Videoplayer von aForge.

IVideoSource : Interface
Videoobjekte de vom aForge Player wiedergegeben werden sollen müssen dieses Interface implementieren.

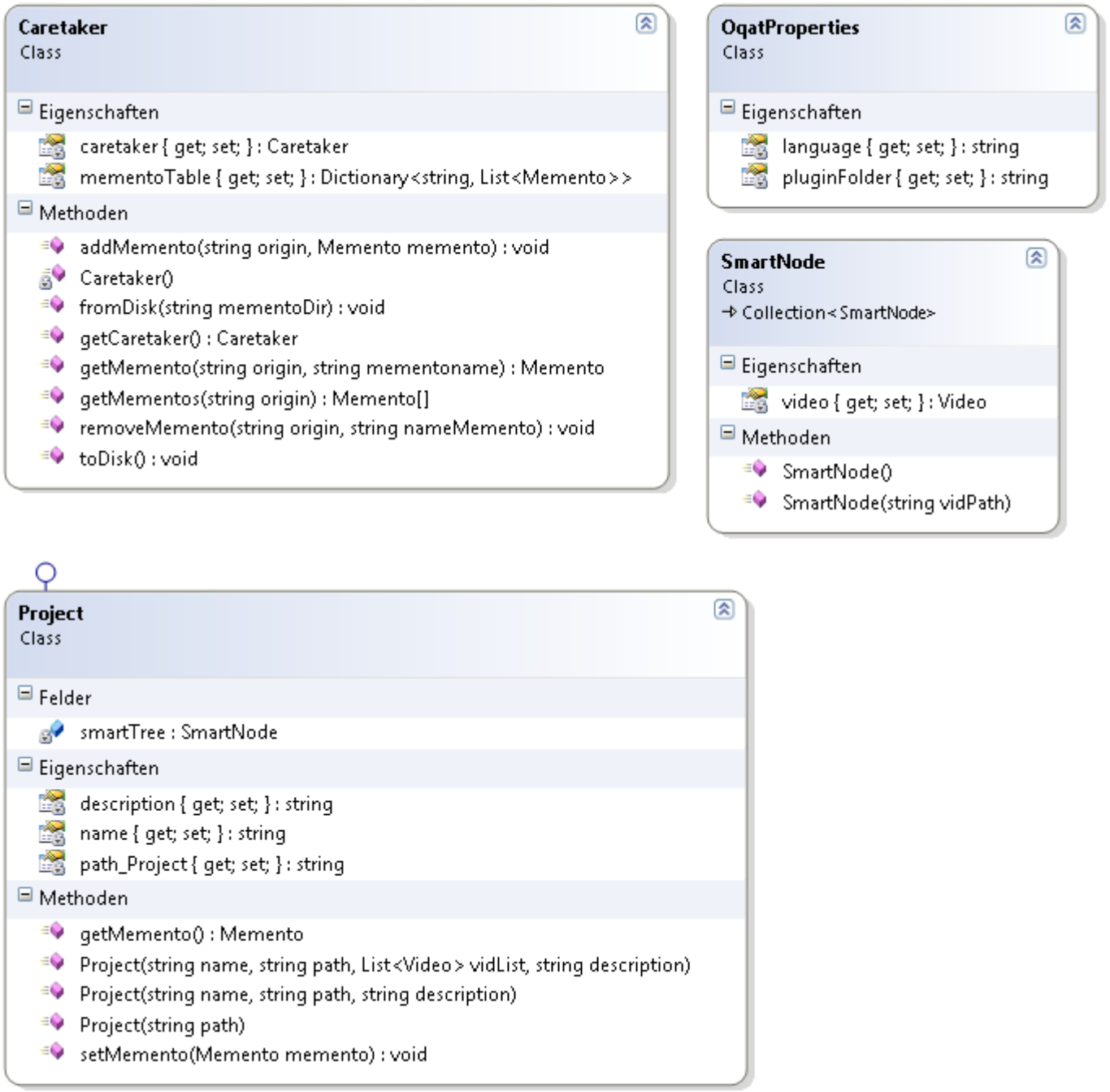

Abbildung 2.1: Model

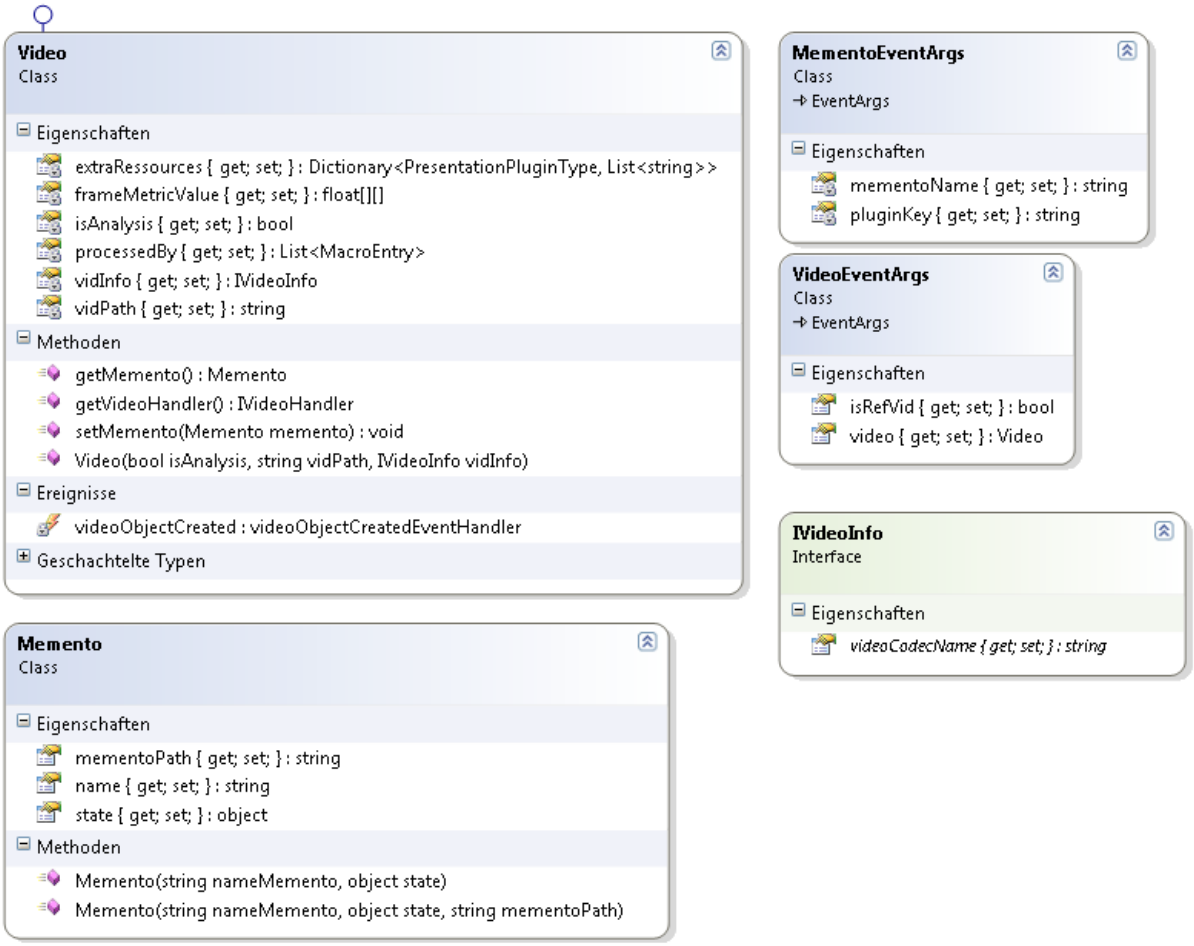

Abbildung 2.2: Model

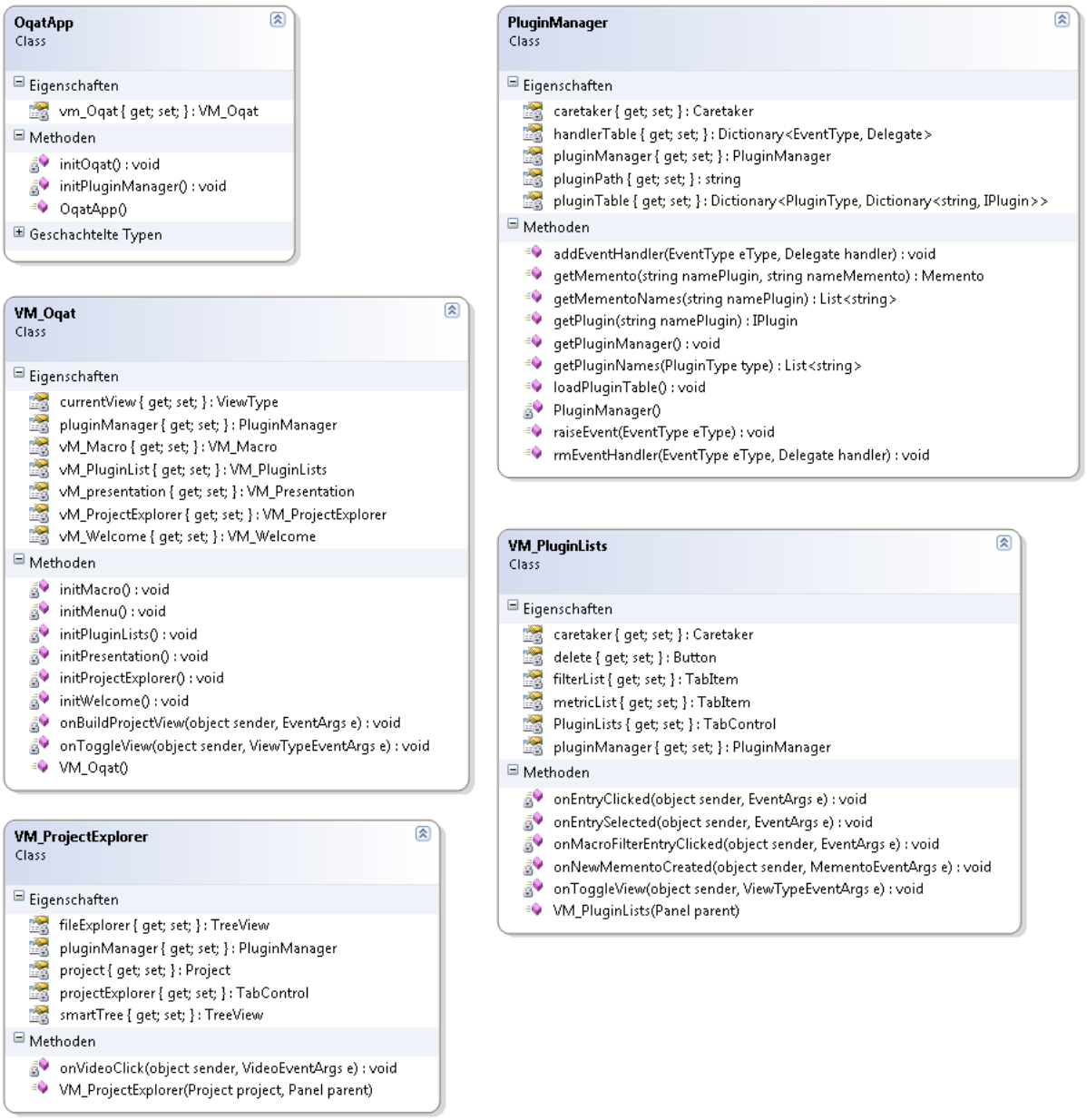

Abbildung 2.3: Oquat Organisation

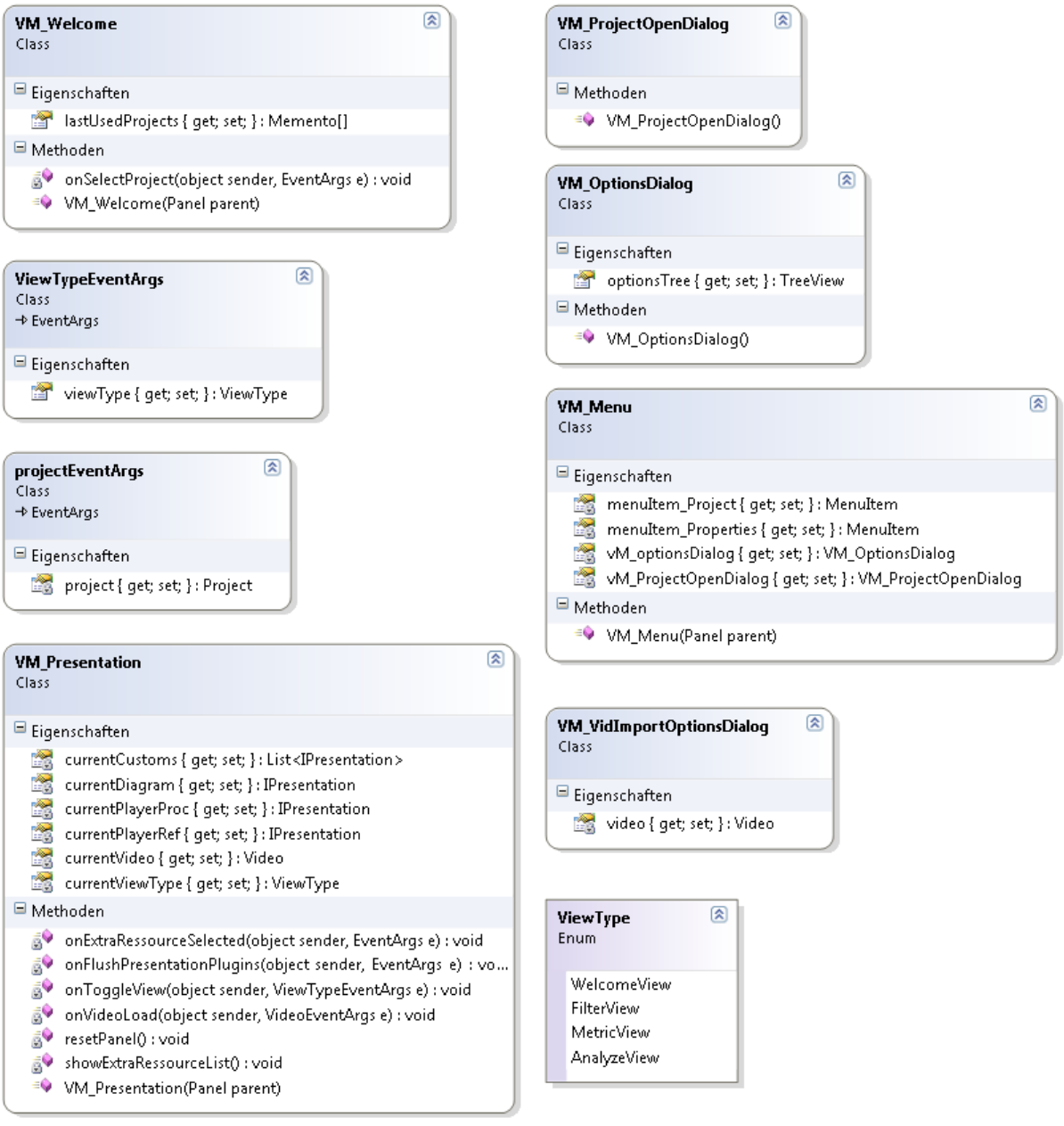

Abbildung 2.4: Oquat Organisation

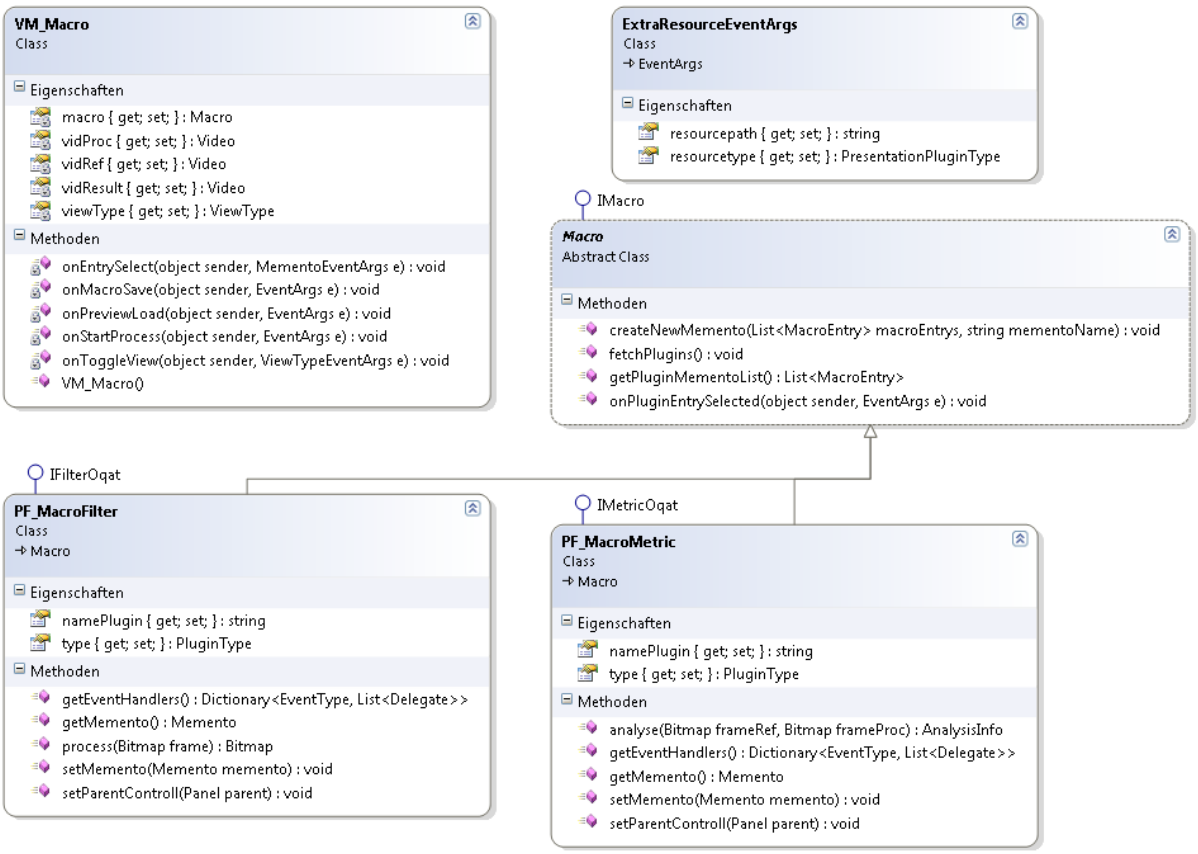

Abbildung 2.5: Macro

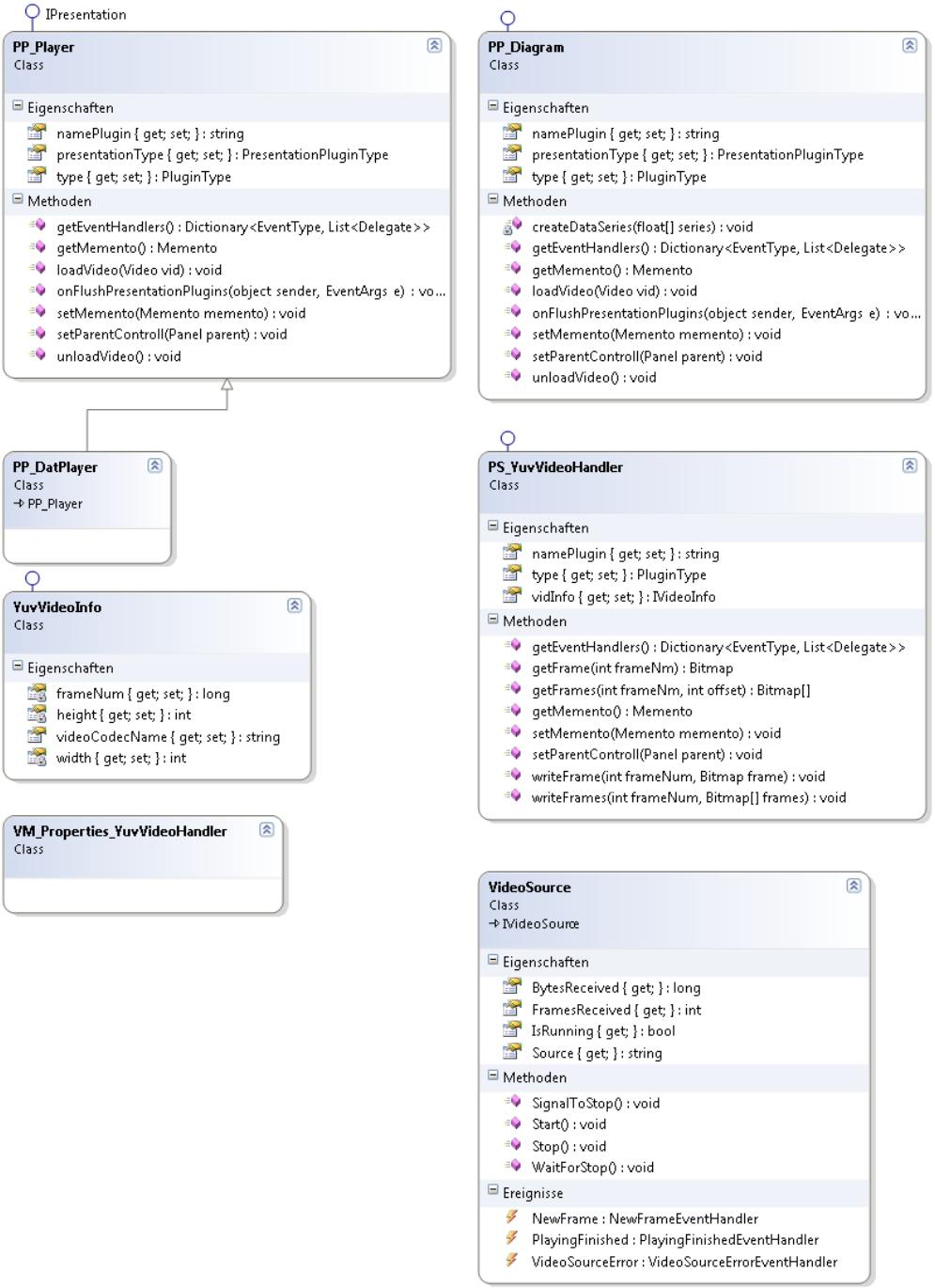

Abbildung 2.6: Presentation

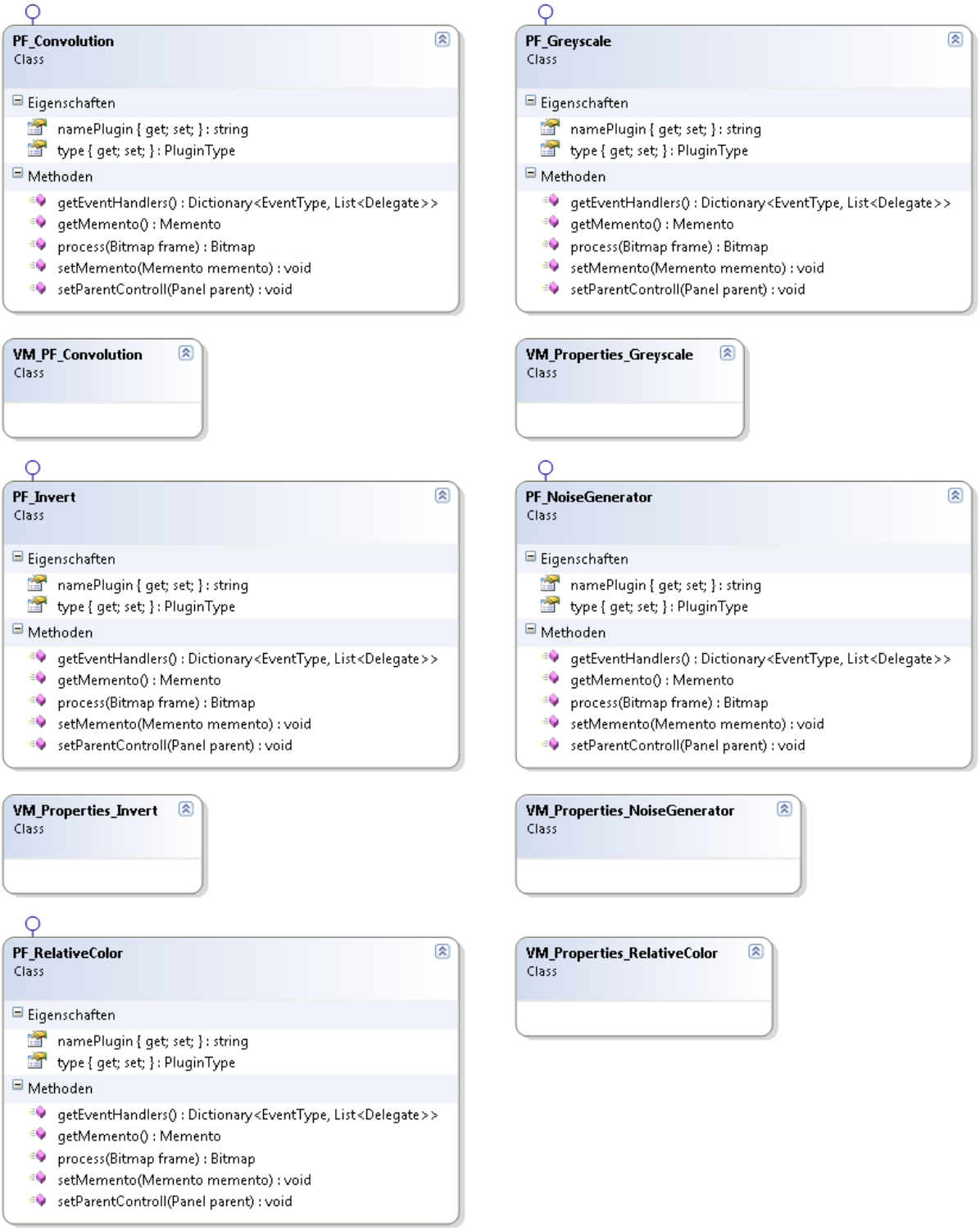

Abbildung 2.7: Filter

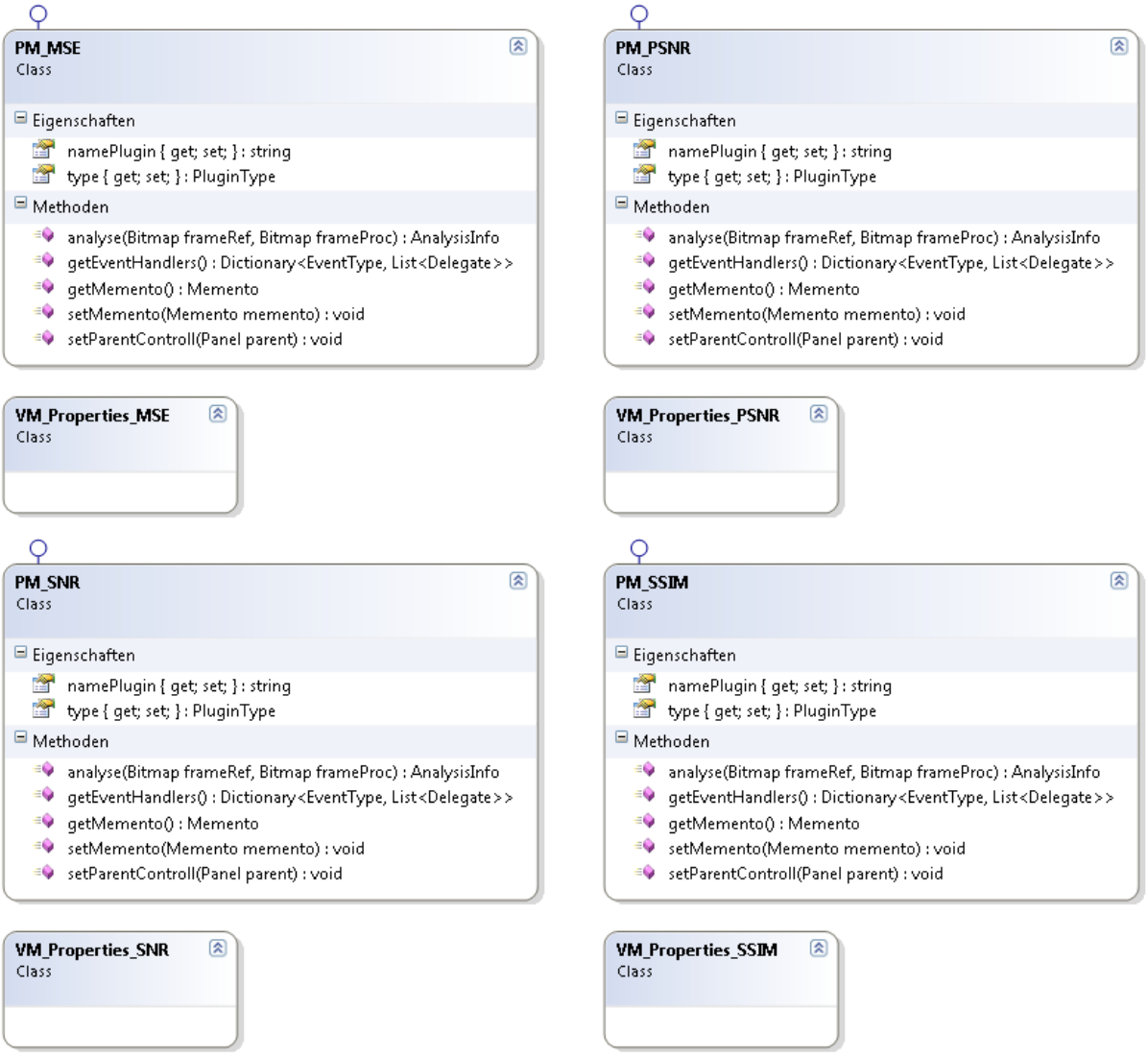

Abbildung 2.8: Metric

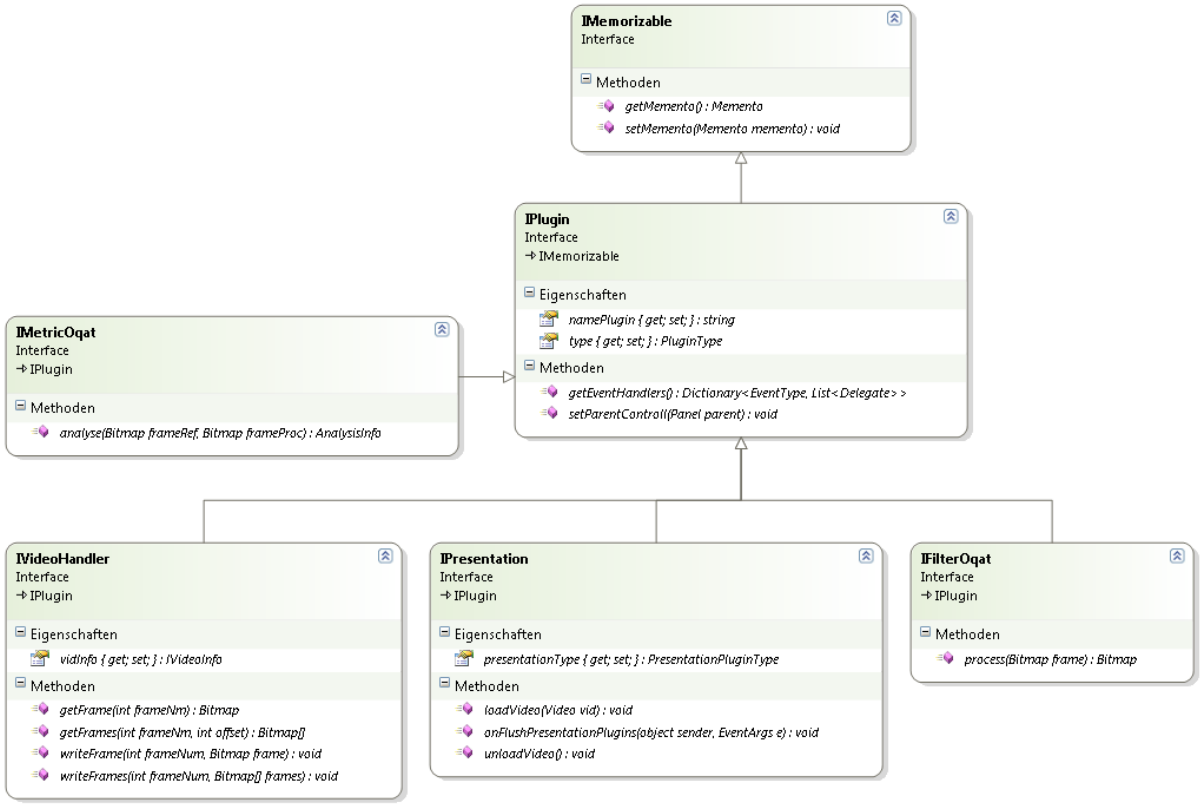

Abbildung 2.9: Public Plugins

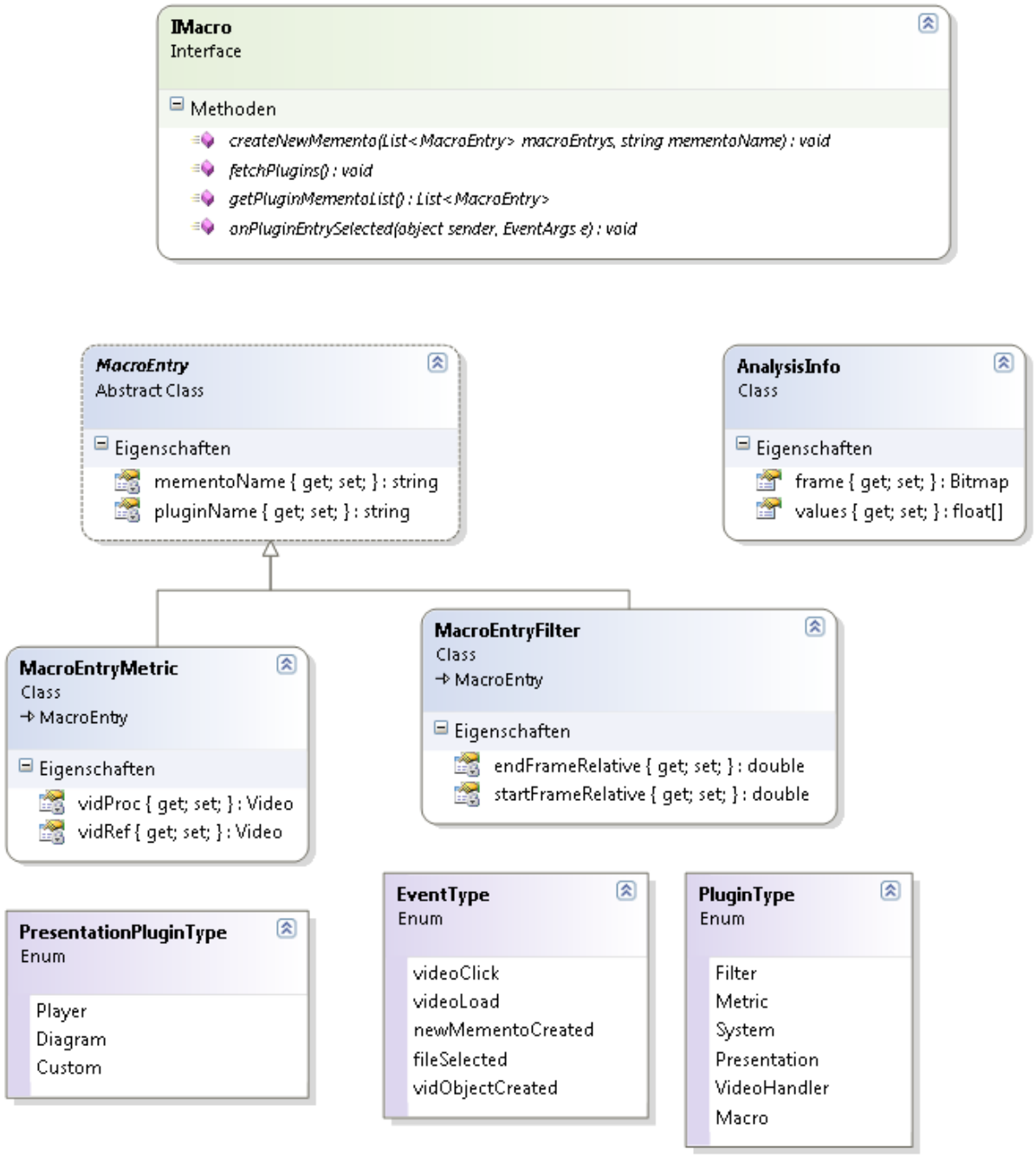

Abbildung 2.10: Public Plugins

3 Sequenzdiagramme

3.1 Initialisierung von OQAT

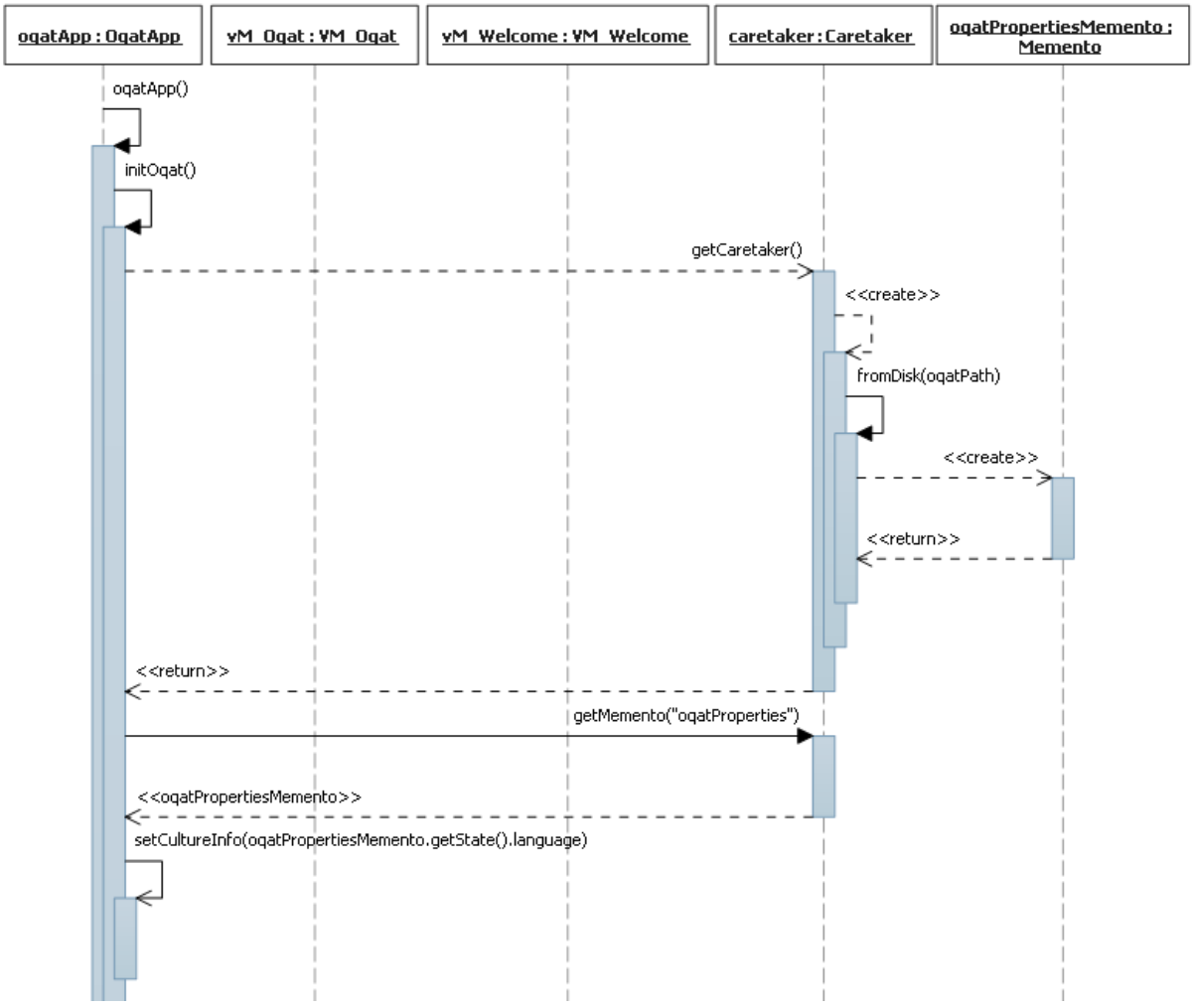

Abbildung 3.1: initOqat1

3.2 Initialisierung eines Projekts

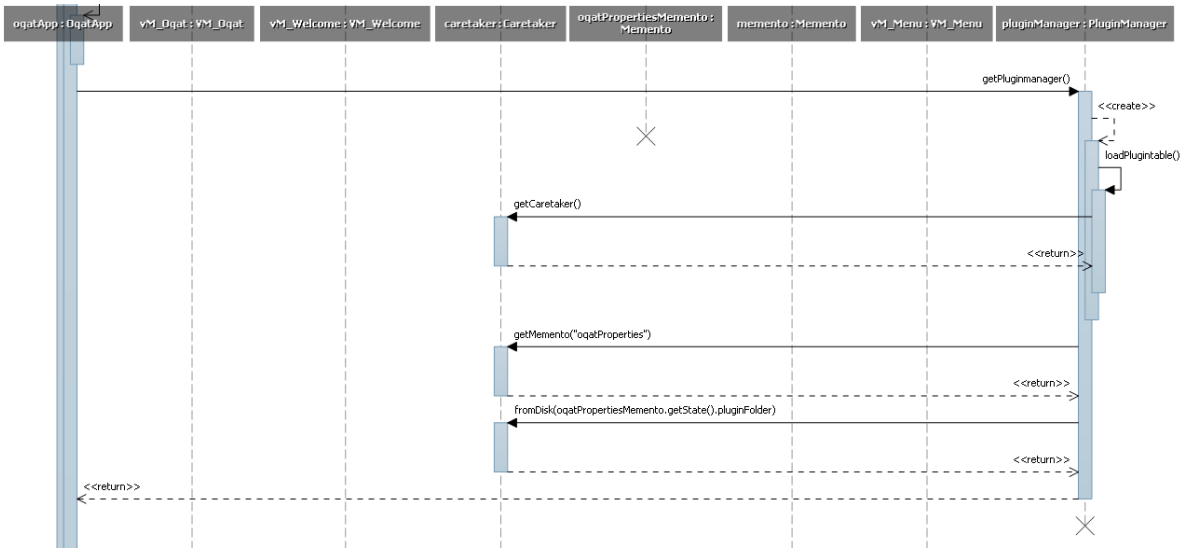

Abbildung 3.2: initOqat2

3.3 Macro

3.3.1 Macro Filter

3.3.2 Macro Metric

3.4 Video Load

3.5 Video Extra Ressource

3.6 vidImport

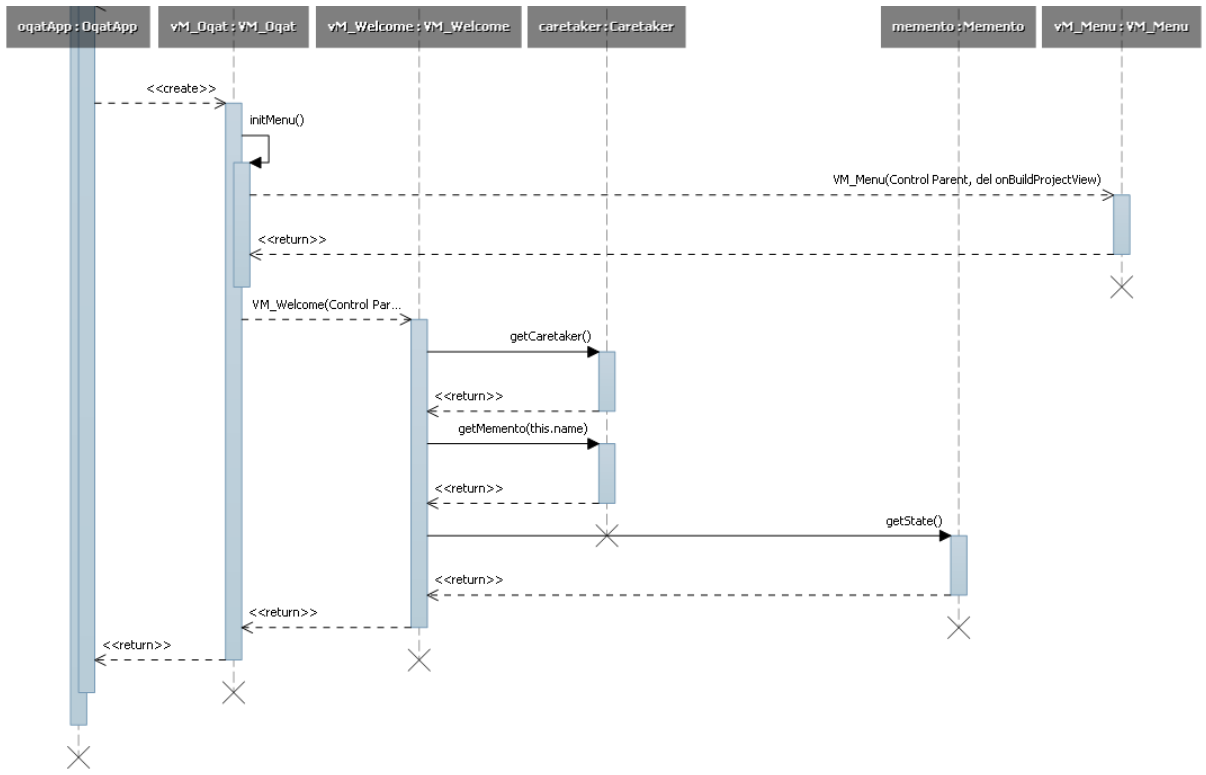

Abbildung 3.3: initOqat3name

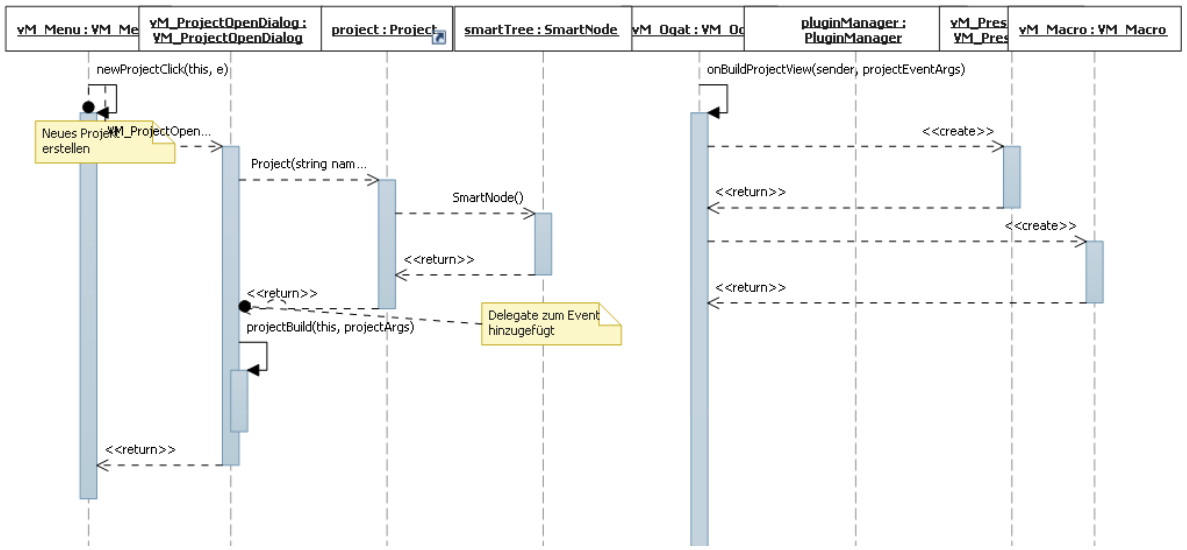

Abbildung 3.4: initProject1

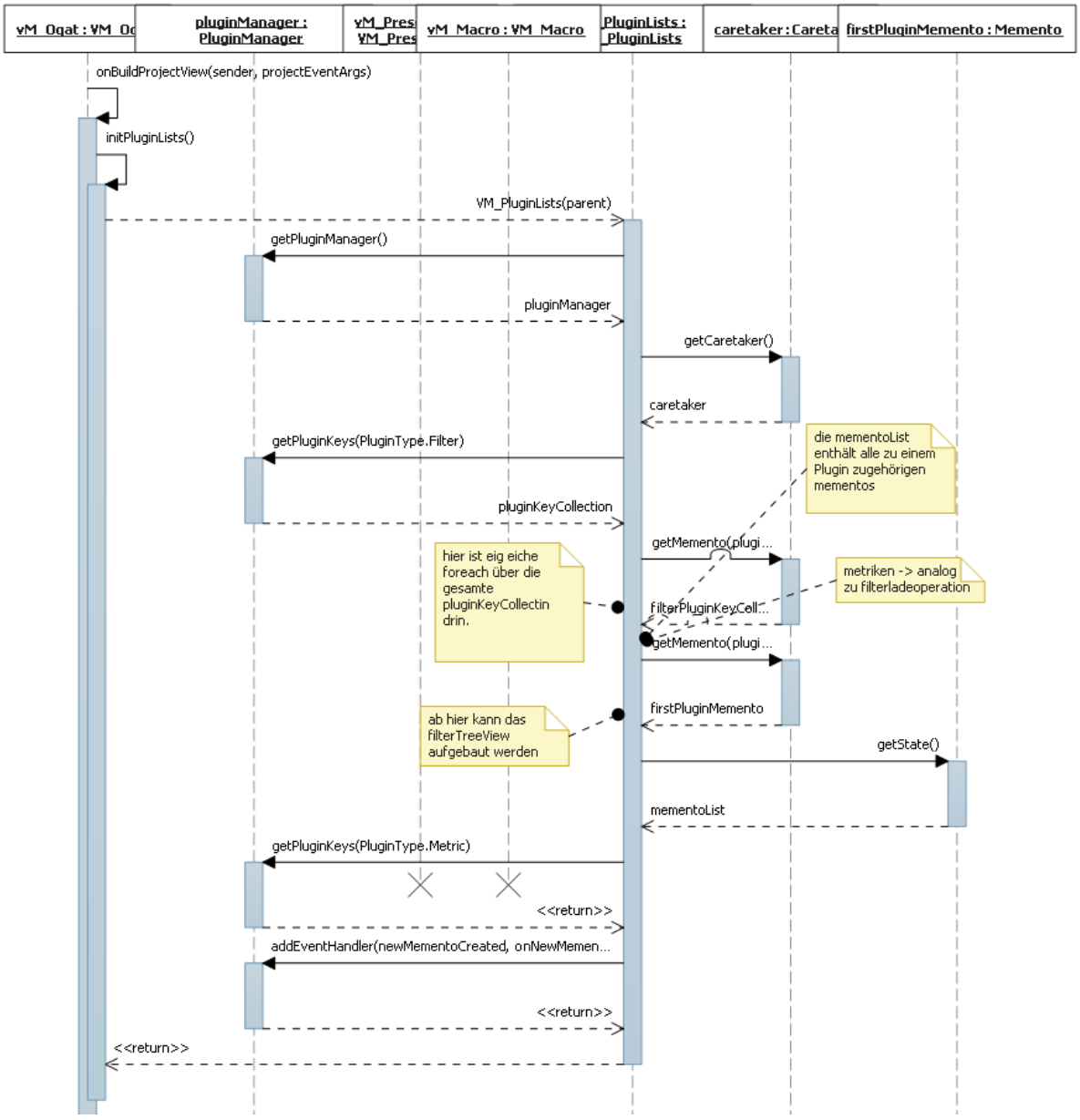

Abbildung 3.5: initProject2

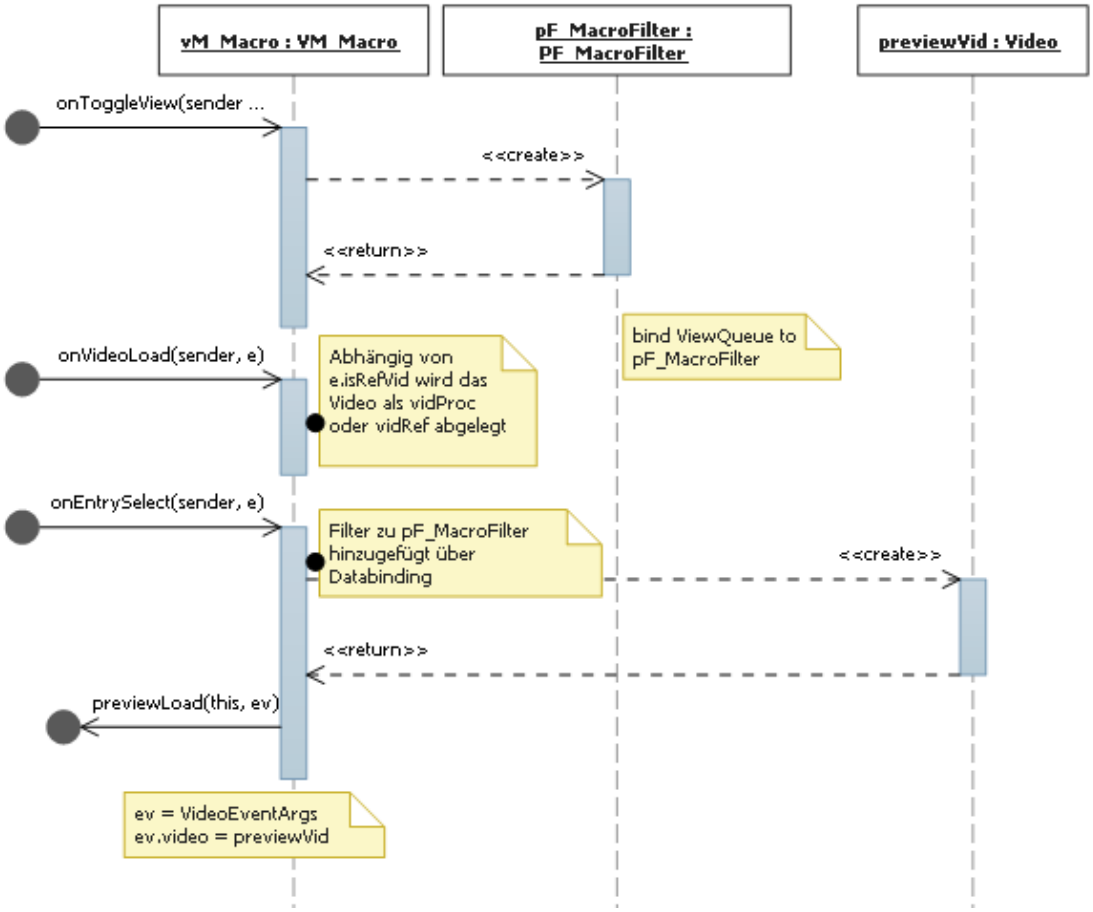

Abbildung 3.6: MacroFilter1

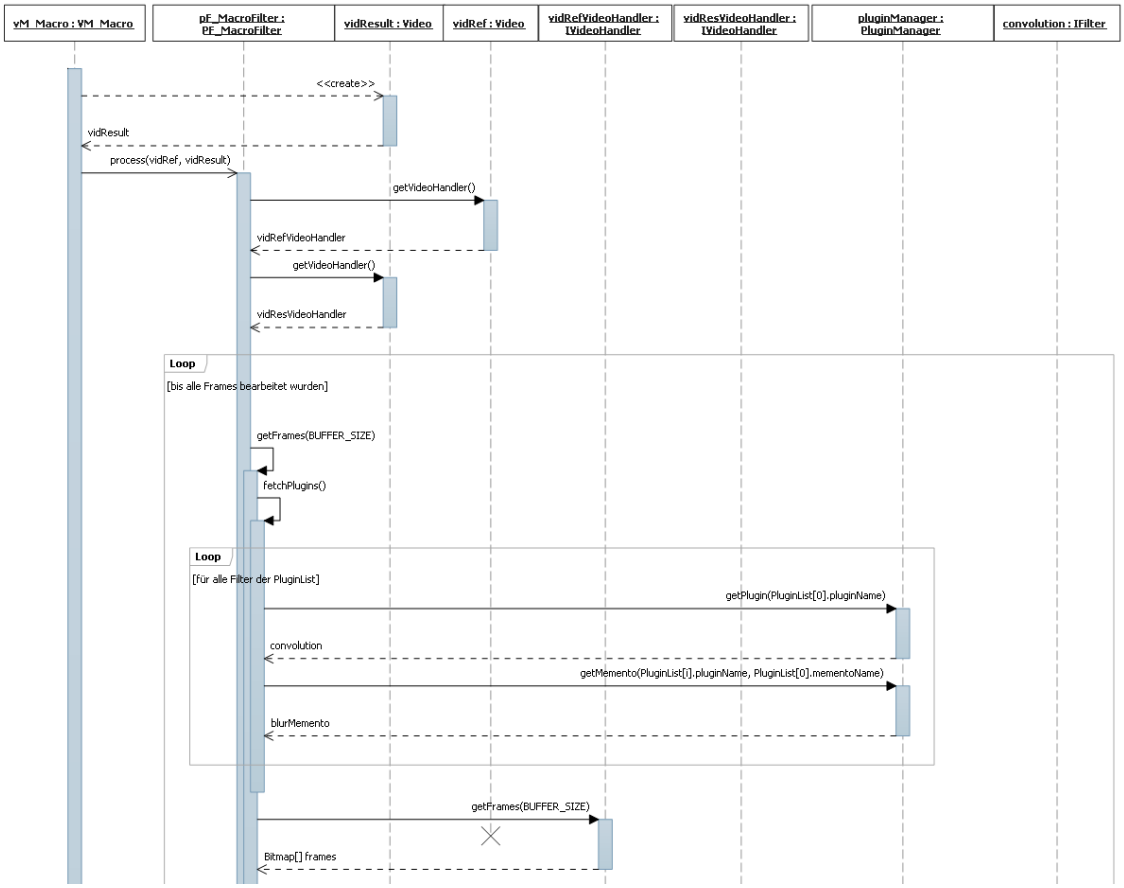

Abbildung 3.7: MacroFilter2

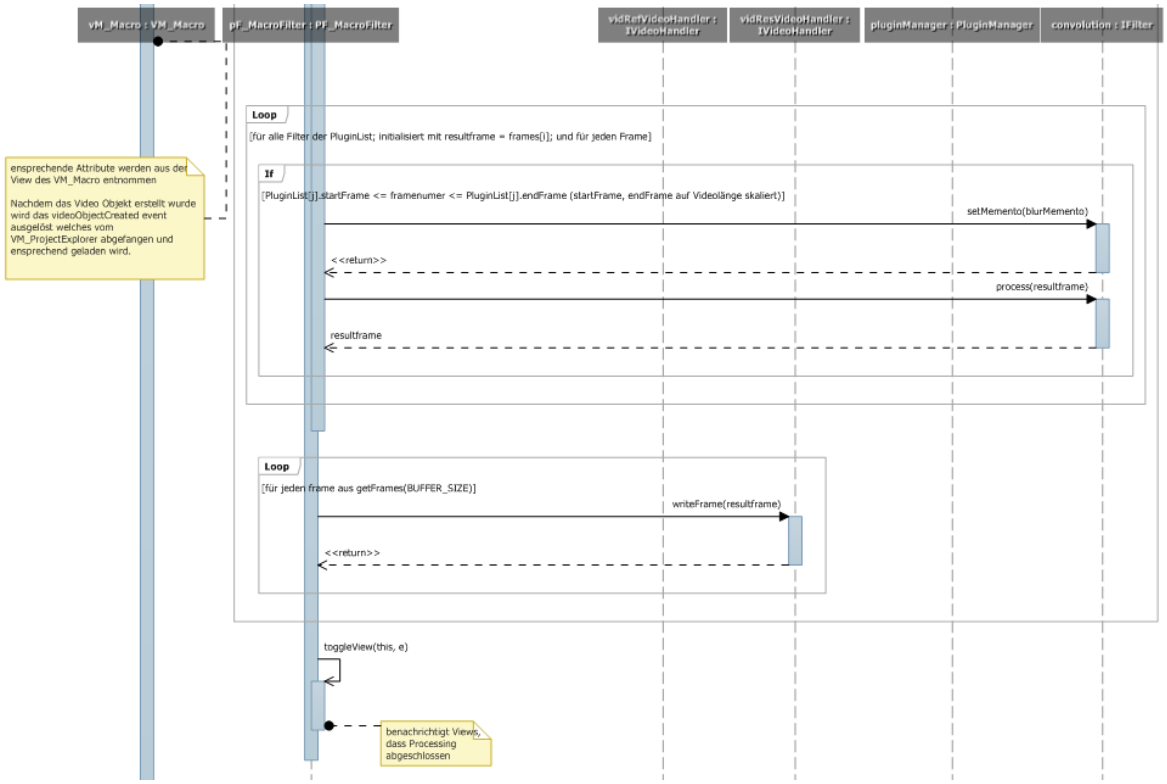

Abbildung 3.8: MacroFilter3

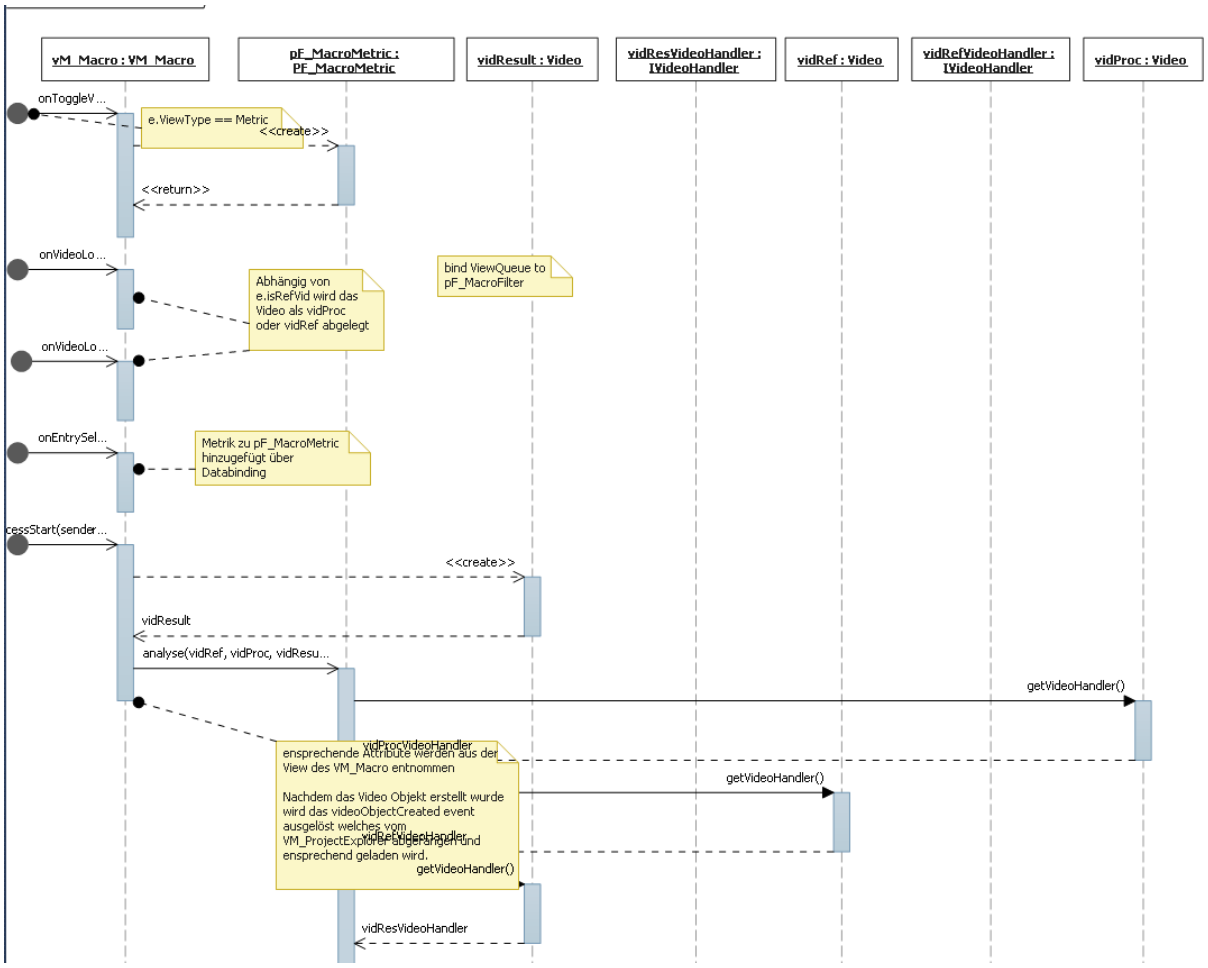

Abbildung 3.9: MacroMetric1

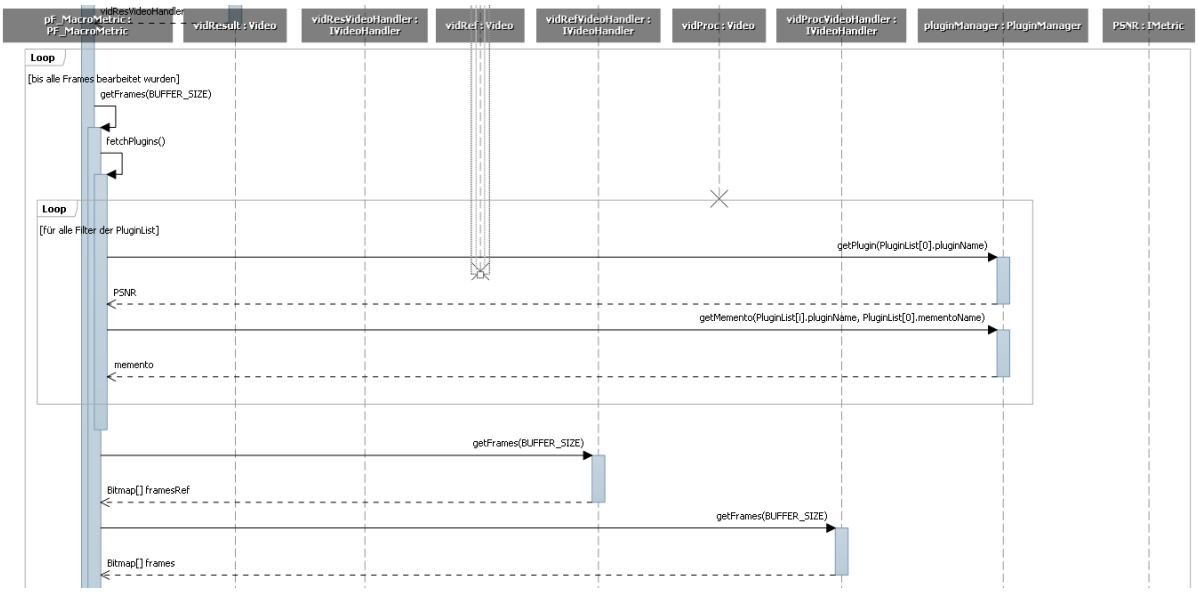

Abbildung 3.10: MacroMetric2

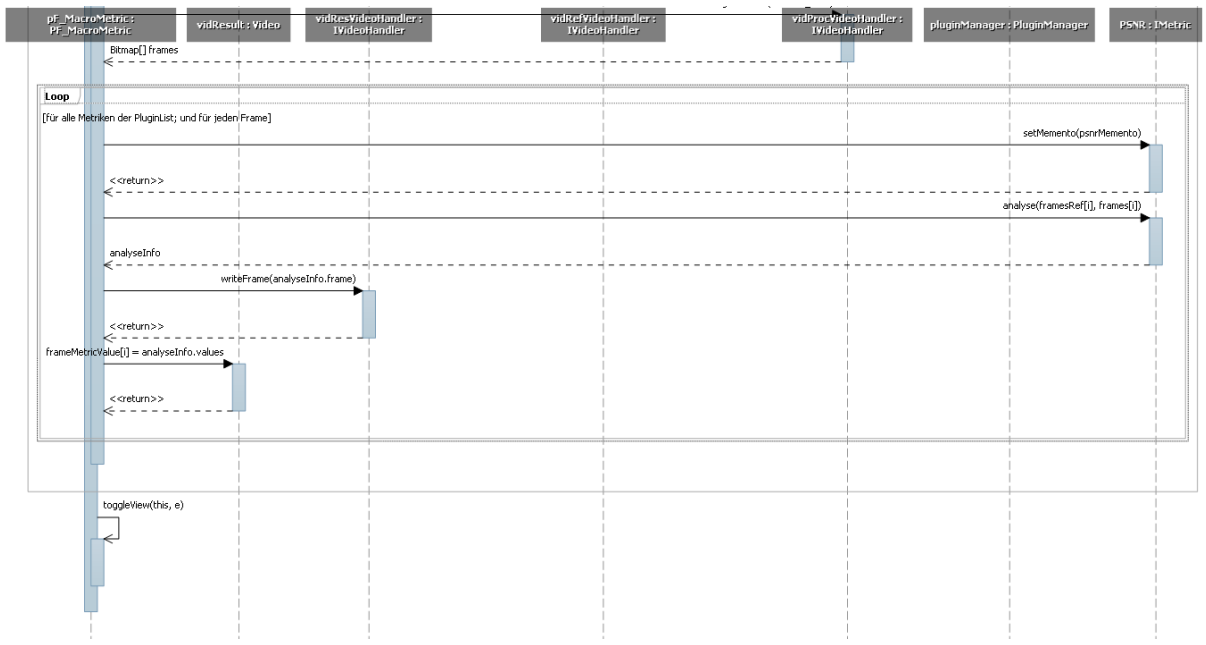

Abbildung 3.11: MacroMetric3

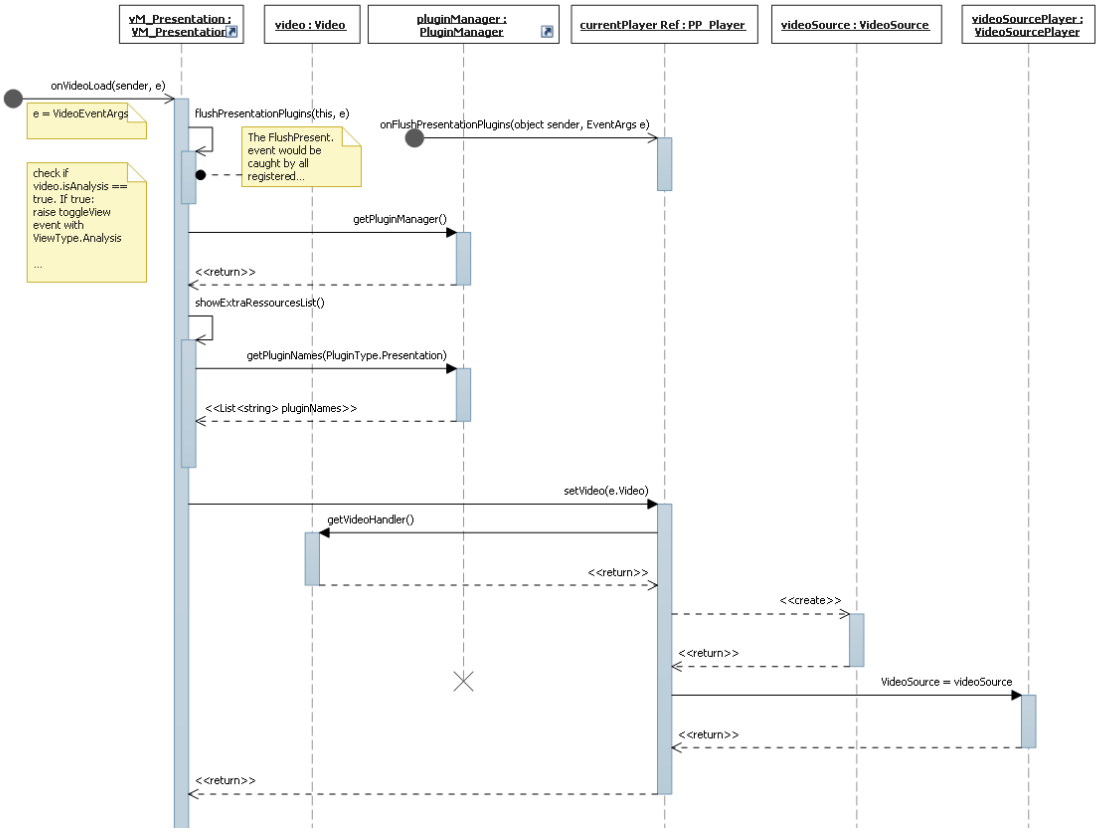

Abbildung 3.12: videoLoad1

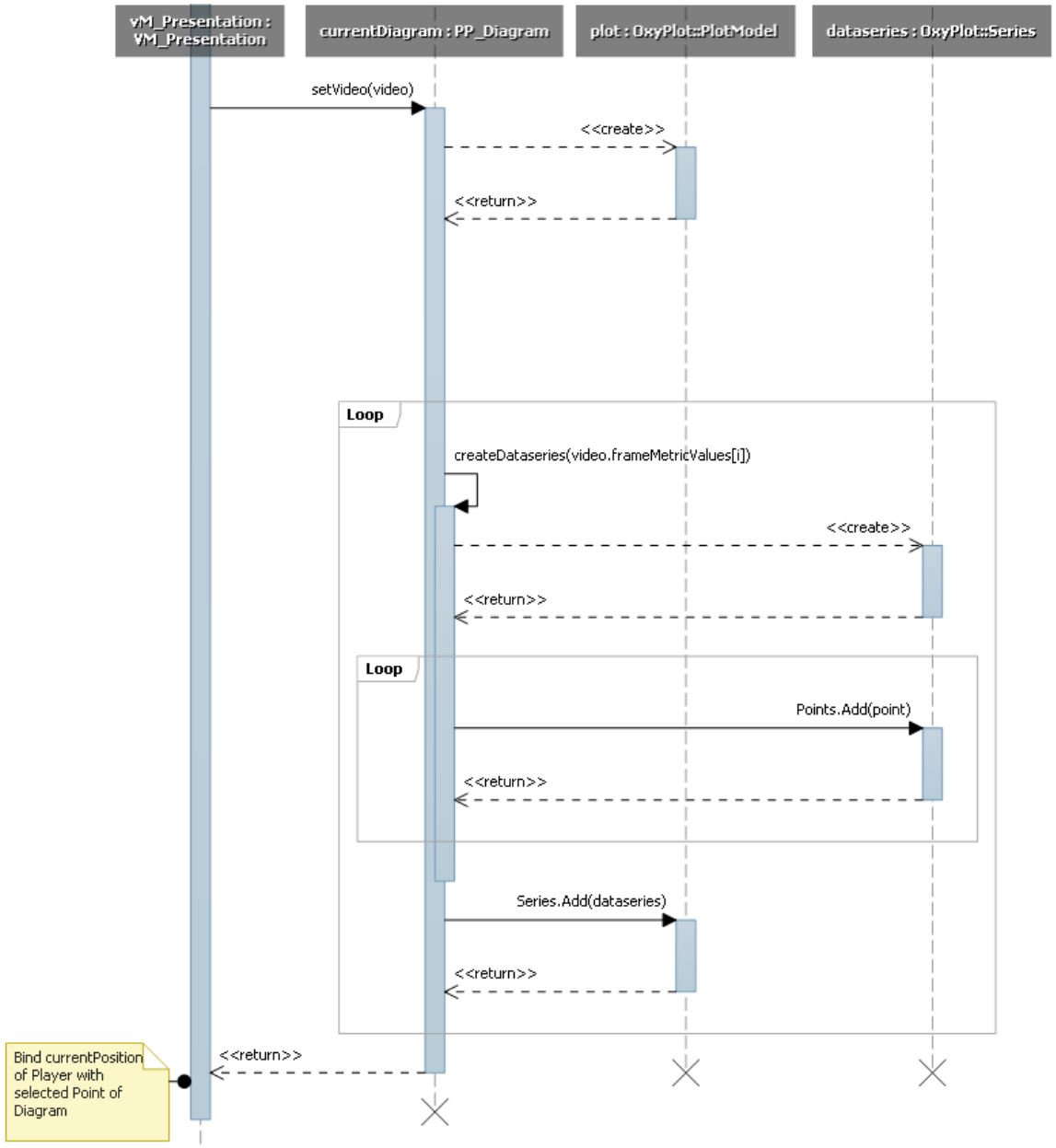

Abbildung 3.13: videoLoad2

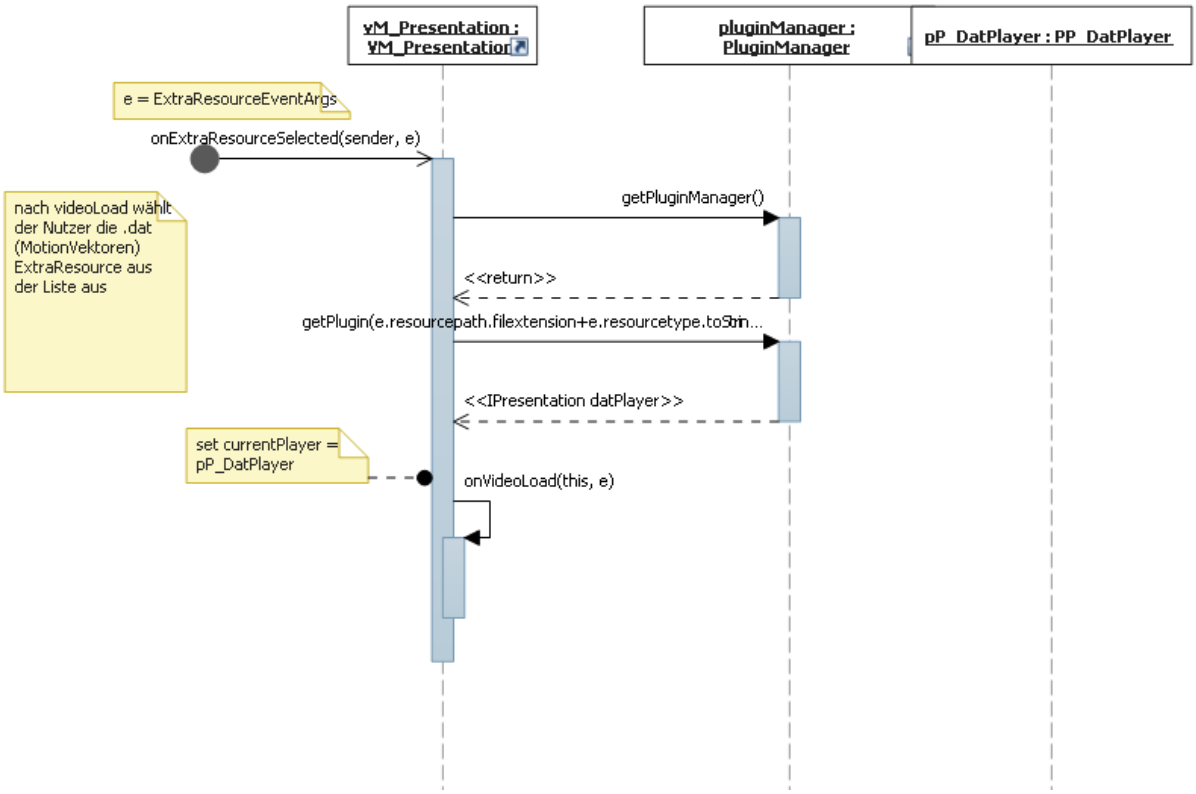

Abbildung 3.14: extraResourcen

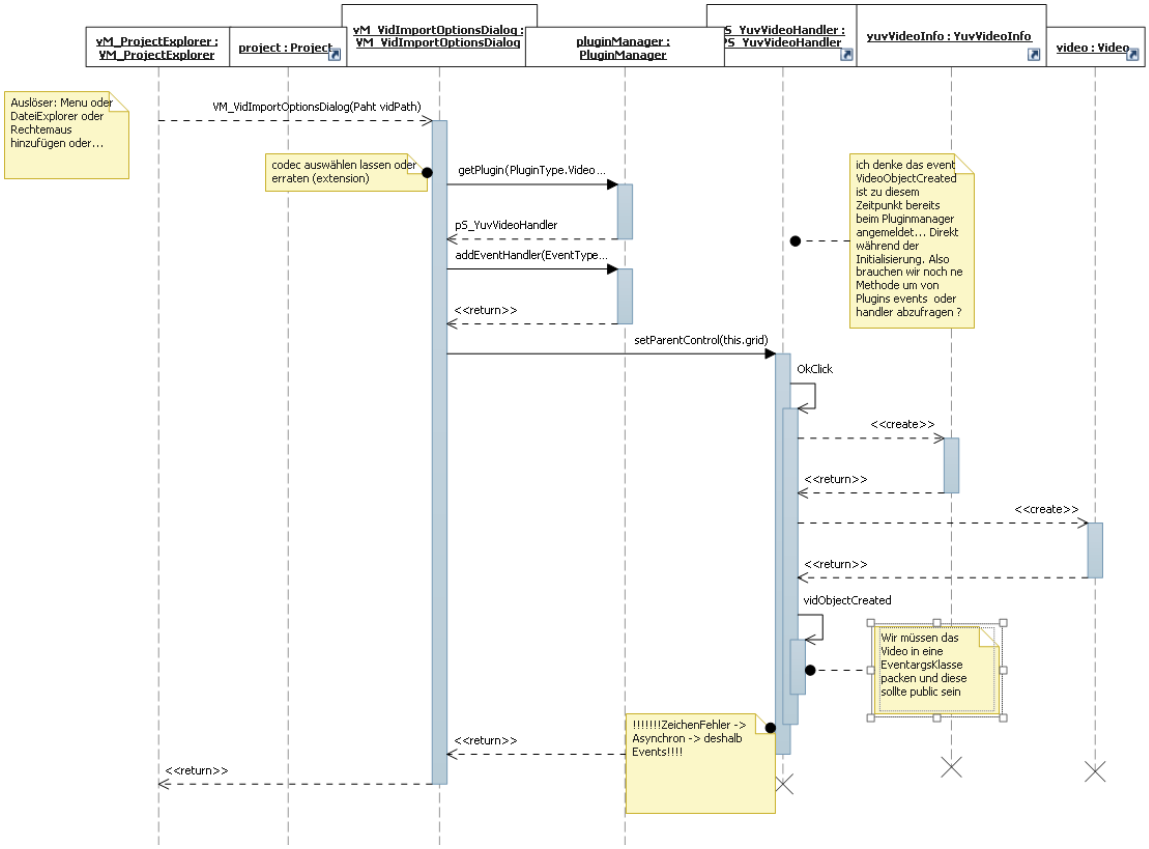

Abbildung 3.15: videoImport

Abbildungsverzeichnis