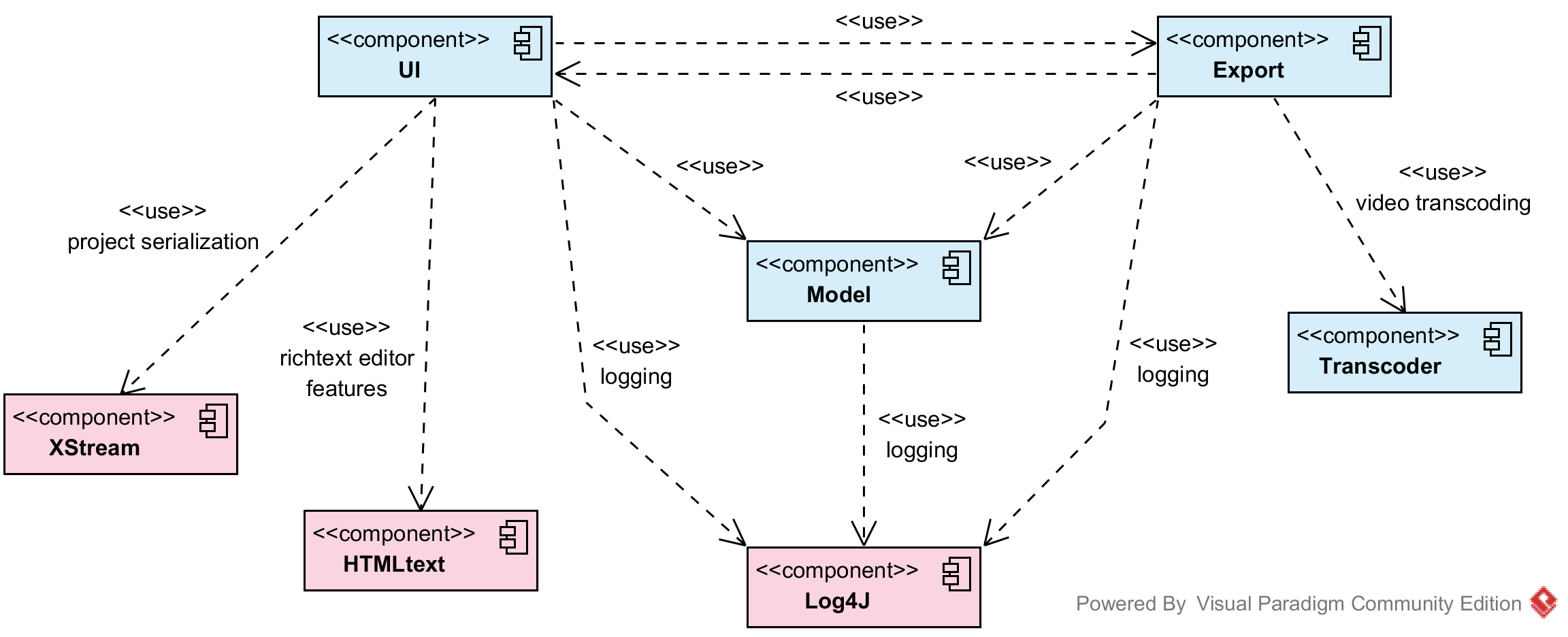
# SIVA Producer - Architectural documentation

This document offers an overview over the architecture of the SIVA Producer as well as implementation details about several hotspots of the software. Build and installation instructions can be found at the project's GitHub repository <https://github.com/SIVAteam/SIVA-Suite>. The SIVA Producer is based on Java 1.6, build upon the Eclipse RCP 3.8.2 framework and requires a 32bit runtime environment.

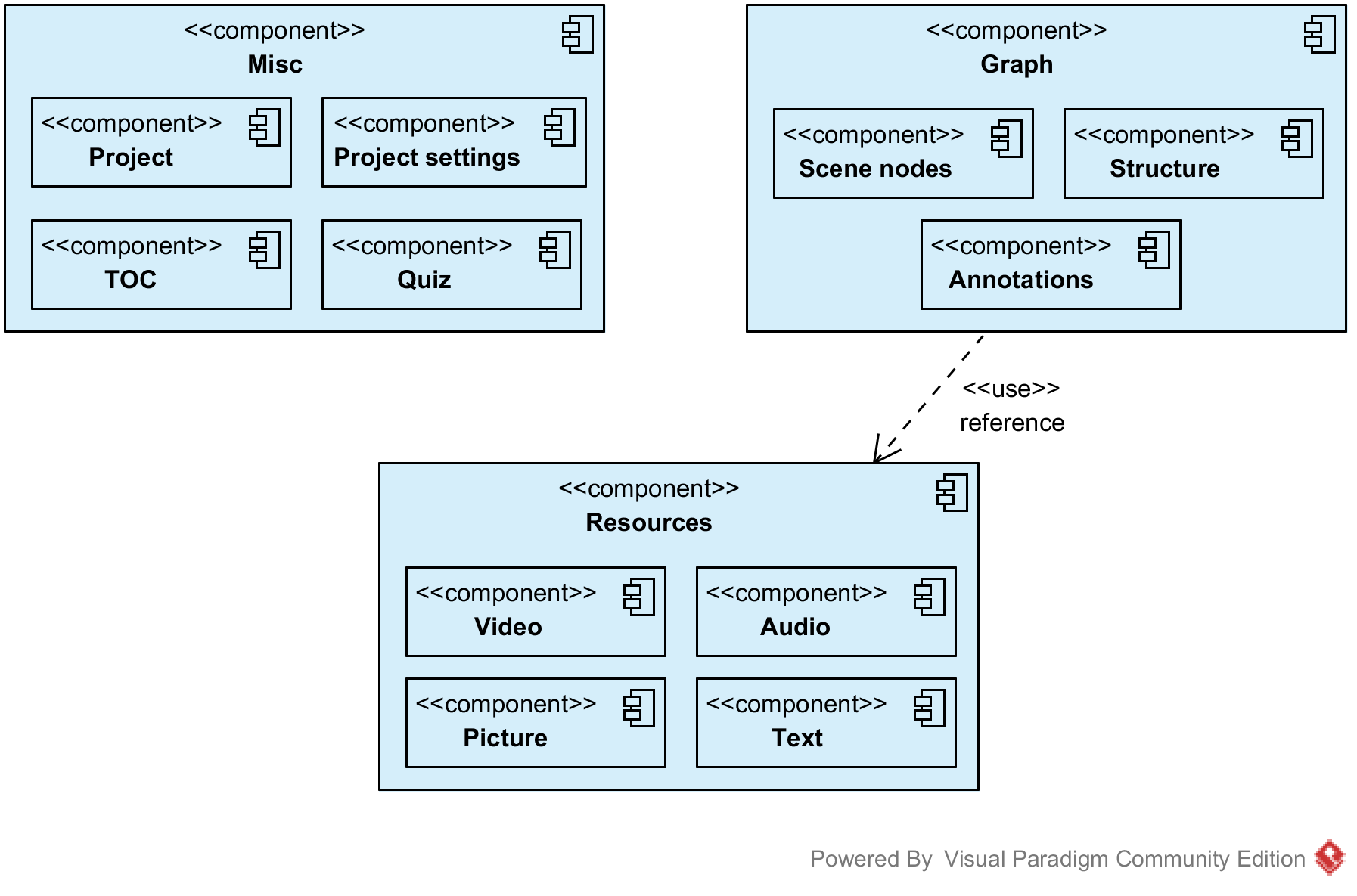
# Overview

The SIVA Producer consists of four components (blue) which are encapsulated in separate plug-ins. Additionally, third-party plug-ins (red) are used for logging, XML serialization and HTML editing.



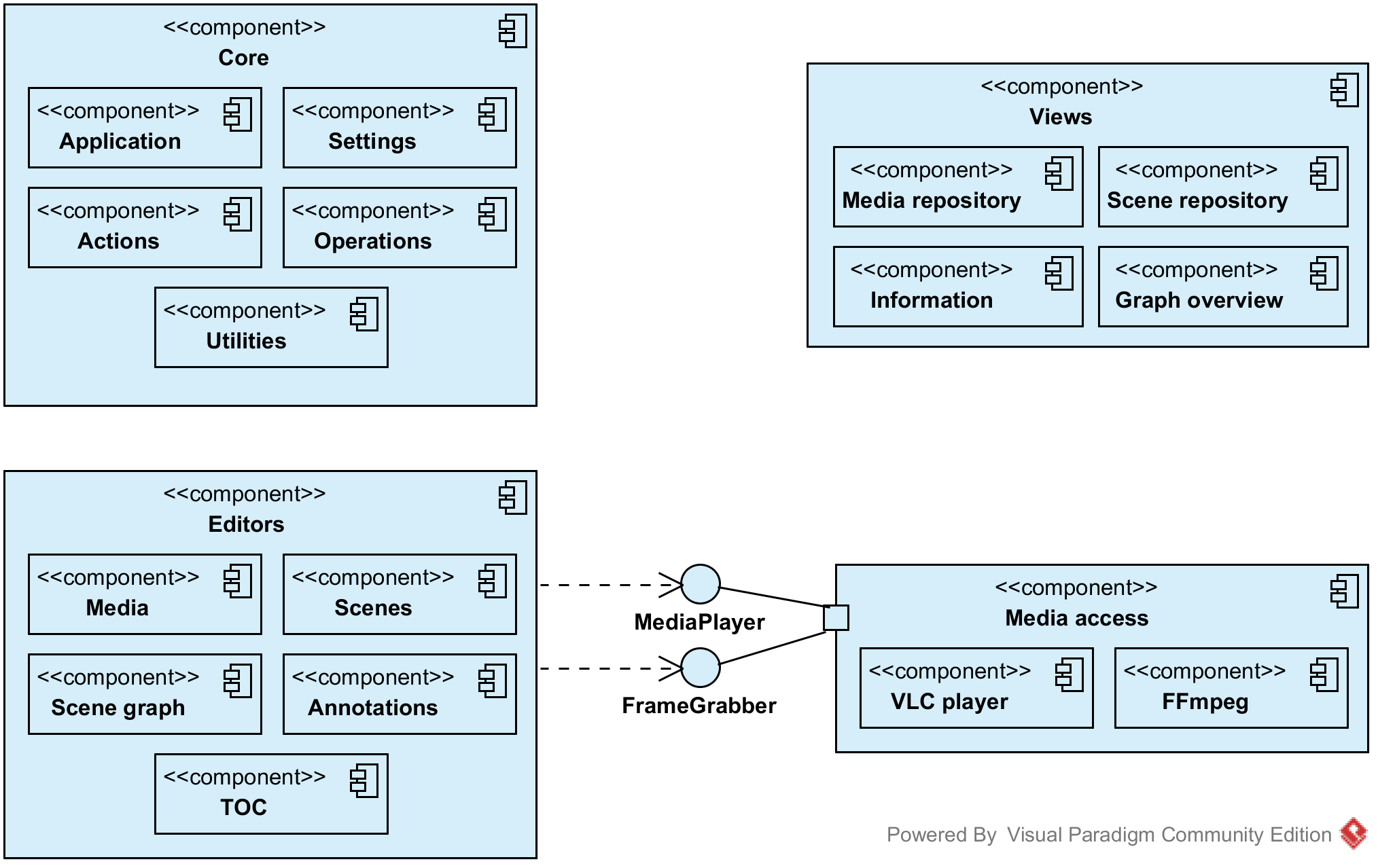
## Model

The model component contains bean classes for all elements which are used to define a hypervideo within the SIVA Producer. This comprises the project with its settings, the supported media resources, the table of contents, quiz elements, and all elements used in the scene graph. This also includes all annotation types, which are modeled as graph elements.



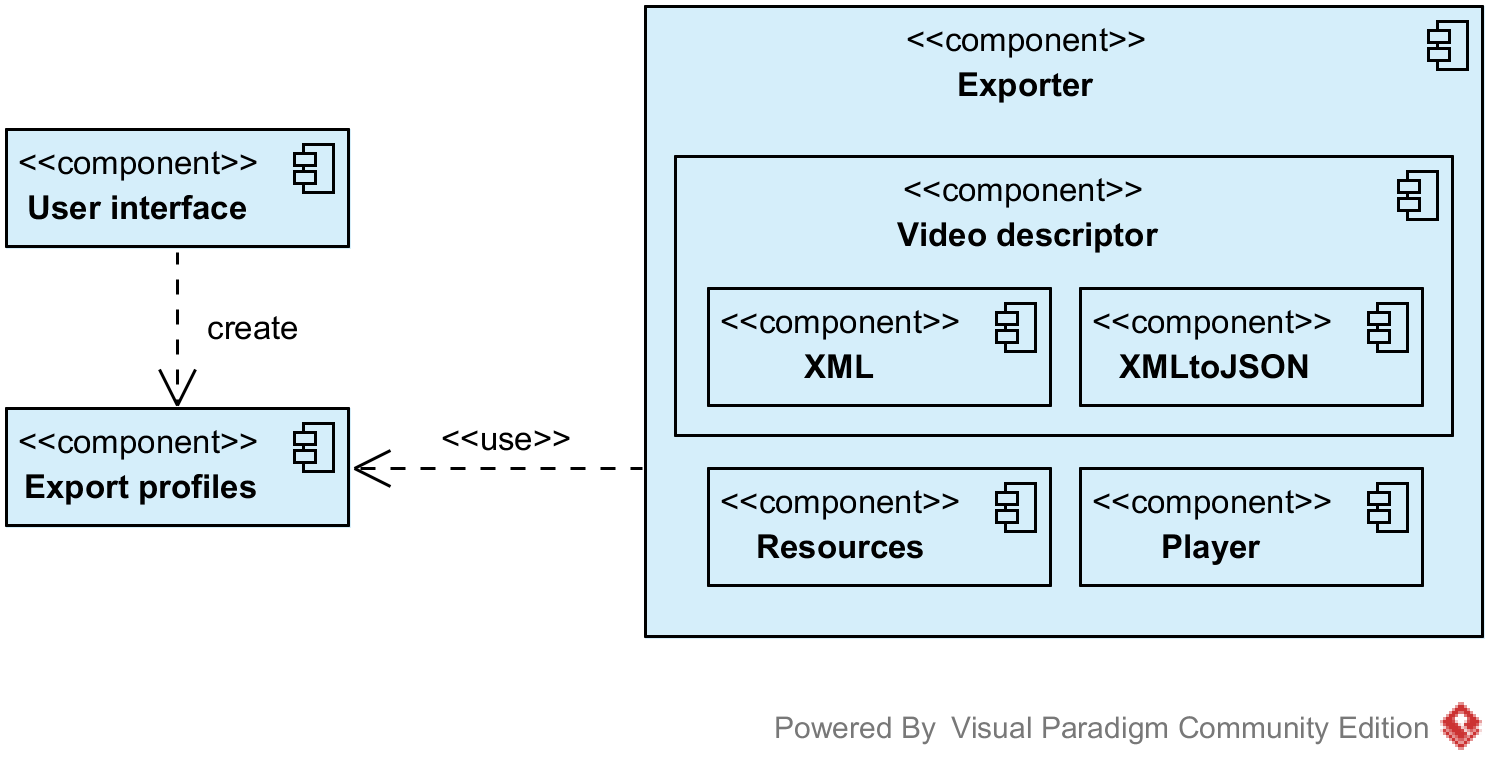
## UI

The UI component constitutes the main part of the application. Apart from the user interface it contains the core RCP application files and bindings to the VLC and FFmpeg media frameworks.



## Export

The export component comes with its own interface for the creation and application of export profiles. These determine the settings used throughout the export process. The export process is mainly split into three parts. The creation of descriptor files, the export of the hypervideo resources and adding of the video player.

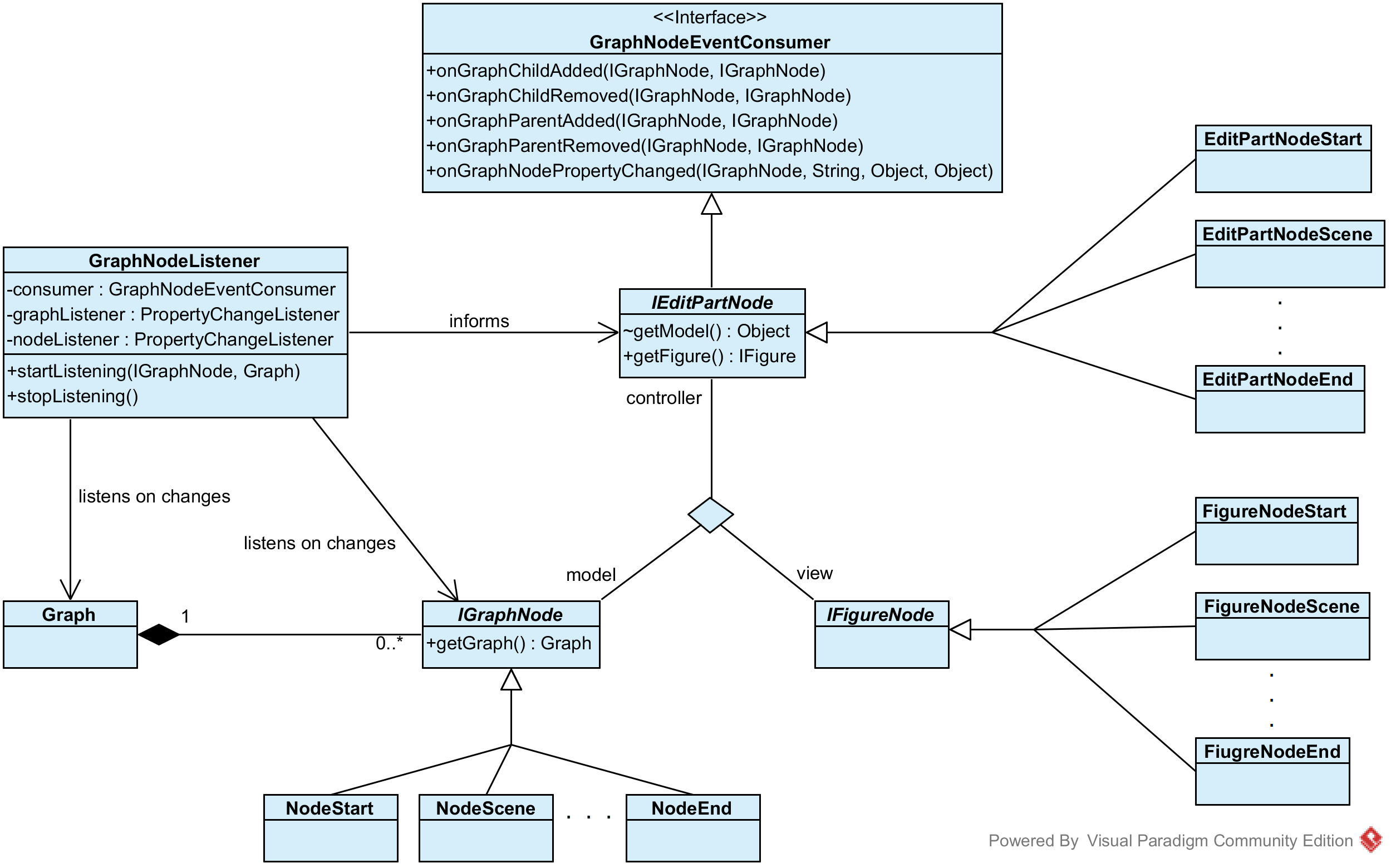


# Hotspots

From an architectural point of view, the locations in the SIVA Producer which are listed here are likely to be involved when extensions are needed or bugs need to be fixed.

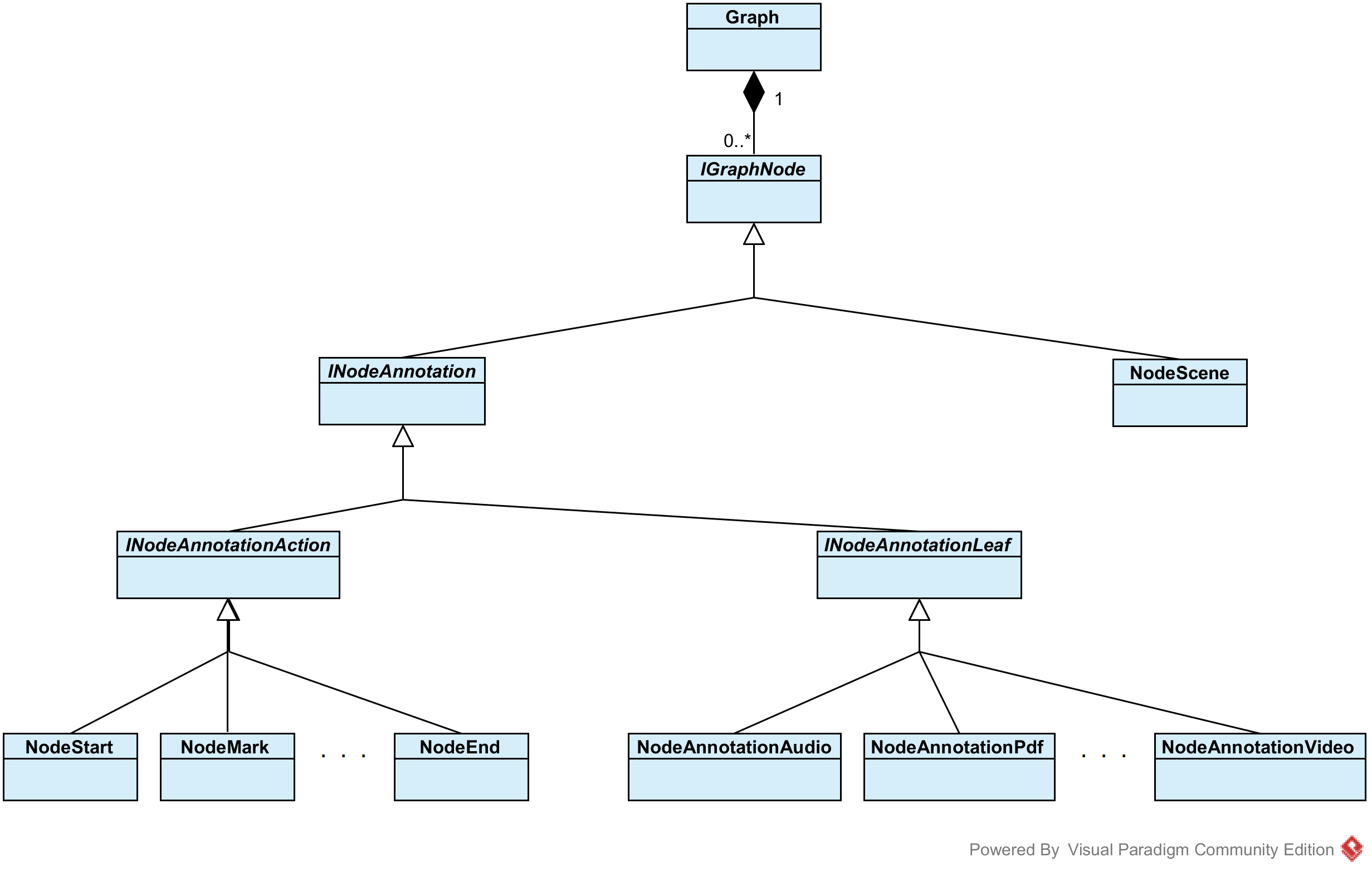
## Graph framework

The implementation of the scene graph is based on the Graphical Editing Framework (GEF). For a graph element of the model component there exists a visualization element (IFigureNode) and a control element (IEditPartNode) in the UI component. An IEditPartNode observes its model element via a GraphNodeListener and updates the view accordingly. On interaction with a graph figure, the underlying IEditPartNode is responsible for opening the appropriate editors and persisting changes to the model.



## Graph, graph nodes and annotations

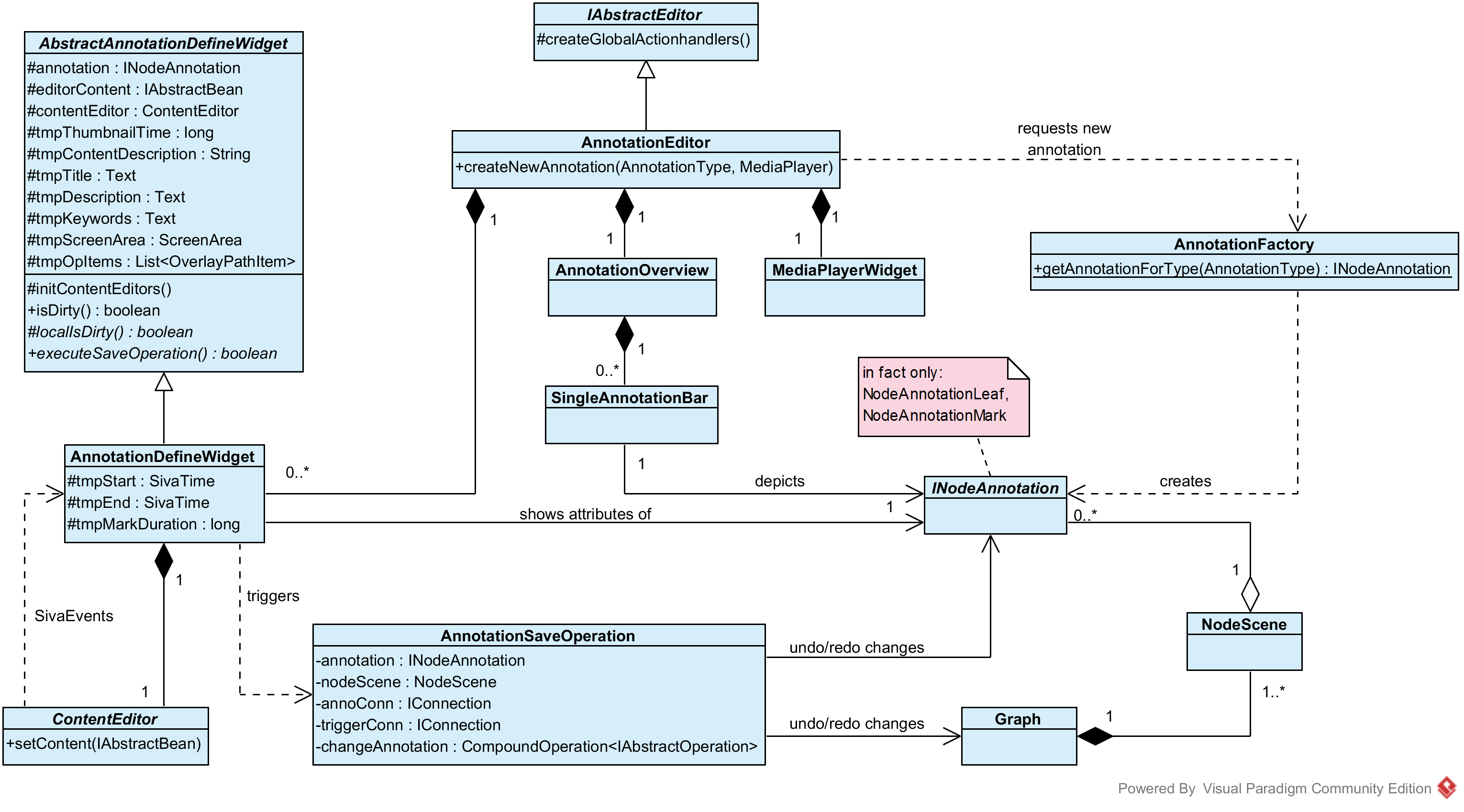
The type hierarchy of graph nodes shows two anomalies. First, all graph nodes, apart from the scene nodes, are subtypes of INodeAnnotation. This includes even elements used for the navigational structure. Second, while mark annotations are a real type of annotation, they use the same supertype as all of the structural graph nodes. Both situations may lead to problems in properly using polymorphism which already lead to undesired use of the instanceof operator. However, changing the type hierarchy would require a comprehensive check of existing instanceof locations and, therefore, has not been done yet.



## Annotation editor

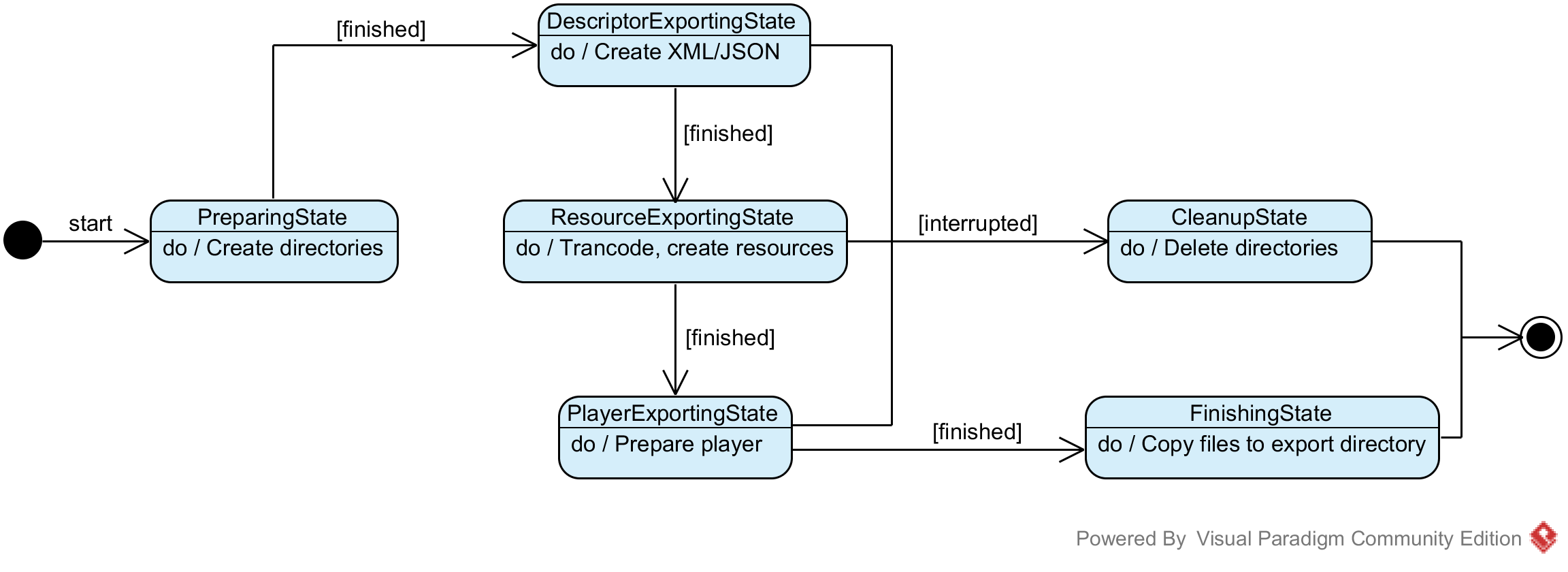
While the editors for scene annotations and global annotations show slight differences in their interface, they are similar in handling annotation editing. Both editors use CTabFolders grouping AbstractAnnotationDefineWidgets for editing the parameters and content of annotations. As a major difference, however, scene annotations (in contrast to global annotations) are part of the graph and connected to their respective NodeScene by DependentConnections.

The abstract class AbstractAnnotationDefineWidget encapsulates most functionality needed for manipulating the annotation content. It initializes the editor appropriate for the current type of annotation and listens to SivaEvents triggered by the content editor and PropertyChangeEvents caused by changing media objects in other editors or views. All changes to an annotation are persisted by calls to executeSaveOperation(), which creates and executes an AnnotationSaveOperation supporting undo/redo. For some annotation types (PDF, Video, ...) the changeable content also contains meta information (summary, video thumbnail, ...). This information as well as annotation type independent variables (title, description, ...) are stored in separate temporary fields in the widget which have to be passed to the AnnotationSaveOperation.



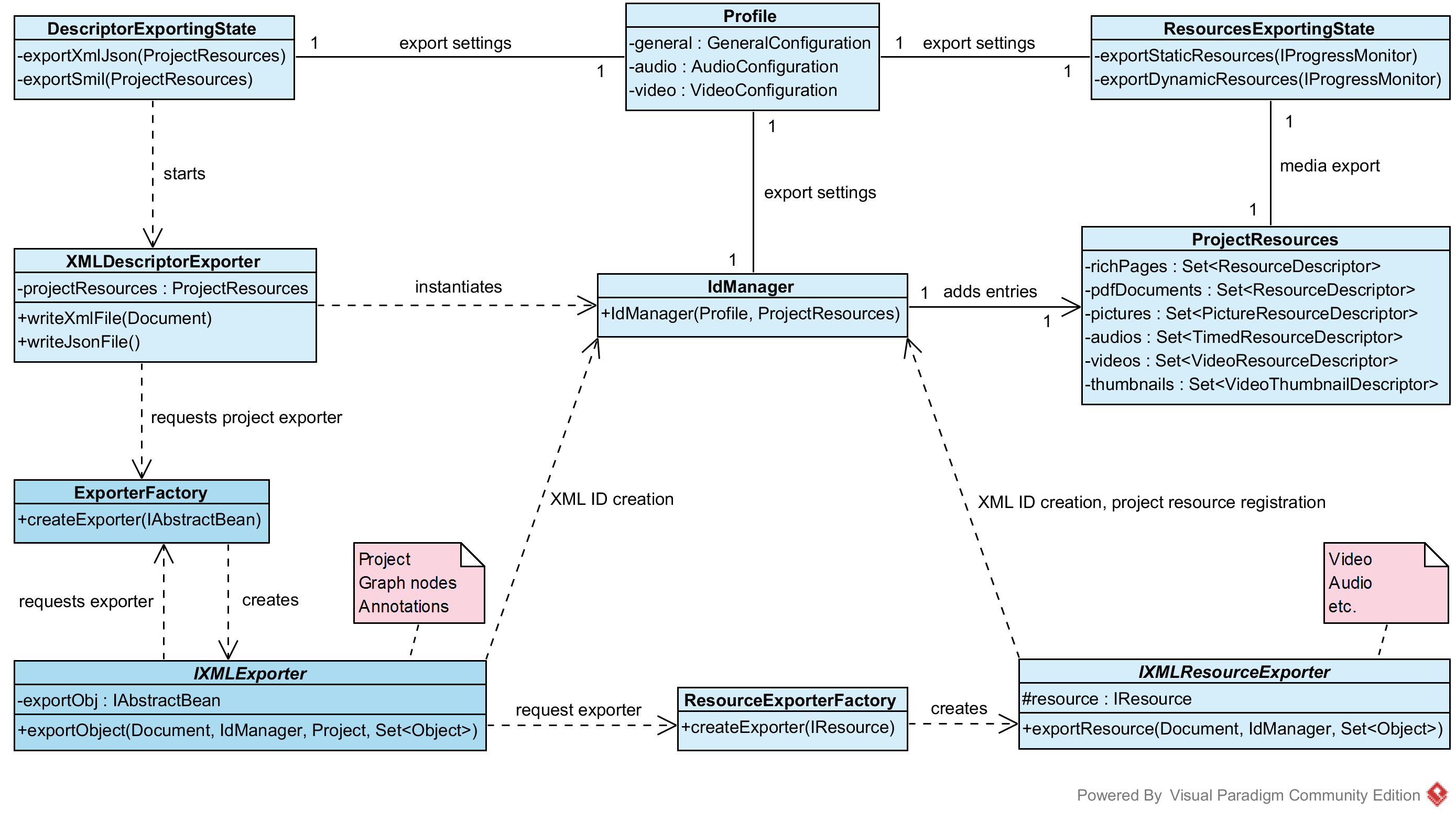
## Export process

When an export action is initiated, an Exporter is created and executed for each ExportProfile chosen by the video author. The export of each profile is performed in several steps reflected by state classes in the source code.



## Export structure

The most complex tasks during the export process are the creation of the descriptor (XML, JSON) files and the creation/transformation of the media files used in a hypervideo. After exporting global annotations and settings referenced by the Project itself, the scene graph is traversed and each node is exported using specialized IXMLExporters. In each exporter, the traversal continues by requesting and executing IXMLExporters for the successors of the current node. This "loop" (dark blue) makes up the main part of the XML export. The IdManager has a central role during this process. For all exported objects it creates unique IDs by which these objects can be referenced in the XML file. Additionally, it registers all media resources in the ProjectResources which are needed by ResourcesExportingState following the XML export.



## Project settings

Currently no global settings exists for customizing the SIVA Producer itself. However, there are several project specific settings affecting the hypervideo and the appearance of the player during playback. These settings have to be chosen whenever a new project is created (ProjectCreateWizard) and are also accessible via the menus (ProjectEditSettingsAction). Adding new player relevant settings usually comprises four steps:

- add necessary fields, getters and setters to ProjectsSettings  
 - adapt the user interface in PlayerSettingsComposite  
 - add tag values to SivaDefinition  
 - implement export of the setting in XMLExporterProject and SivaPlayerXML2JSONConverter

