Cloud Provision System

Edwin Malek, Dionny Santiago, Xiayu Dong

Facilitator, Minute Taker, Leader

Overview of the System

Cloud Provision system:

A graphical Editor, capable of generating a CPS model, and transform its models to different XML types files and well as generating a series of cloud Calls for CPS deployment

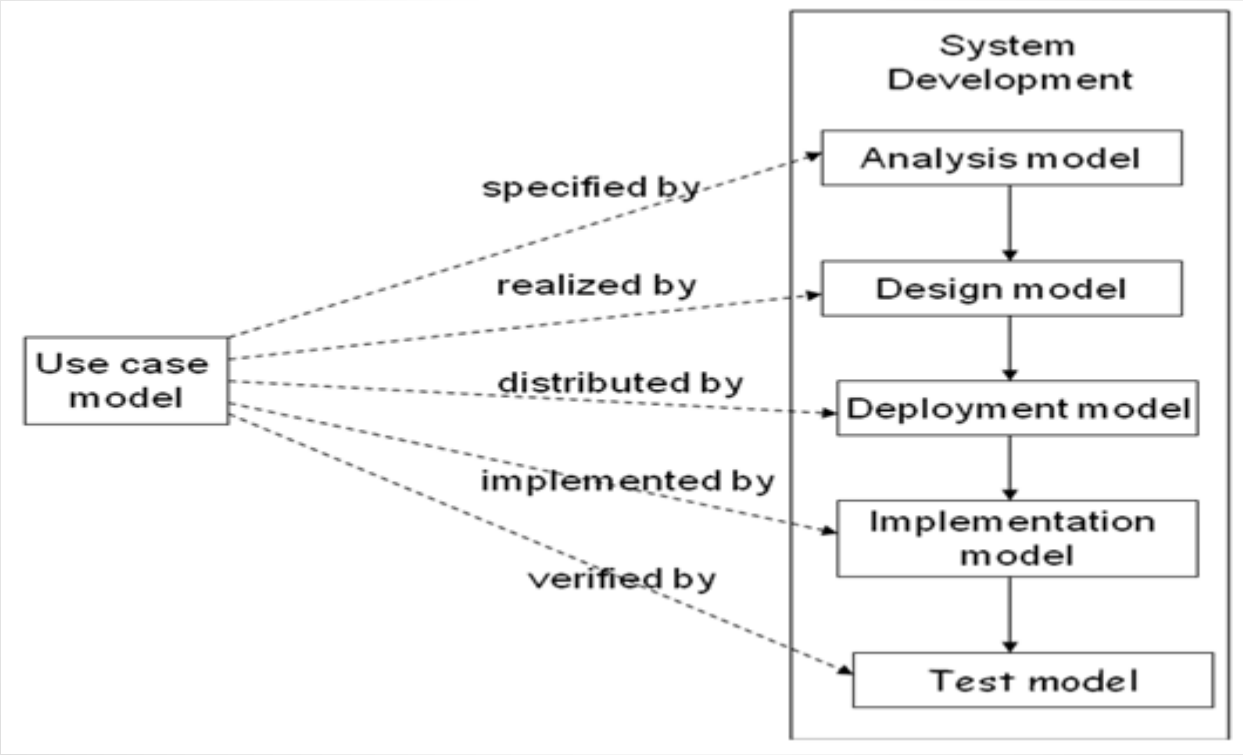
System Requirements (GVE)

The system shall allow the user to create a model by adding different types of Nodes to a canvas representing the possible deployment options available today among most of the cloud provisioning providers.

System requirement con’t (EE)

The system shall be able to analyze a user created CPS model and transform the model to different XML files, and finally generate appropriate cloud calls to according to the CPS provider.

Design Methodology



Software Architecture

Primary Pattern

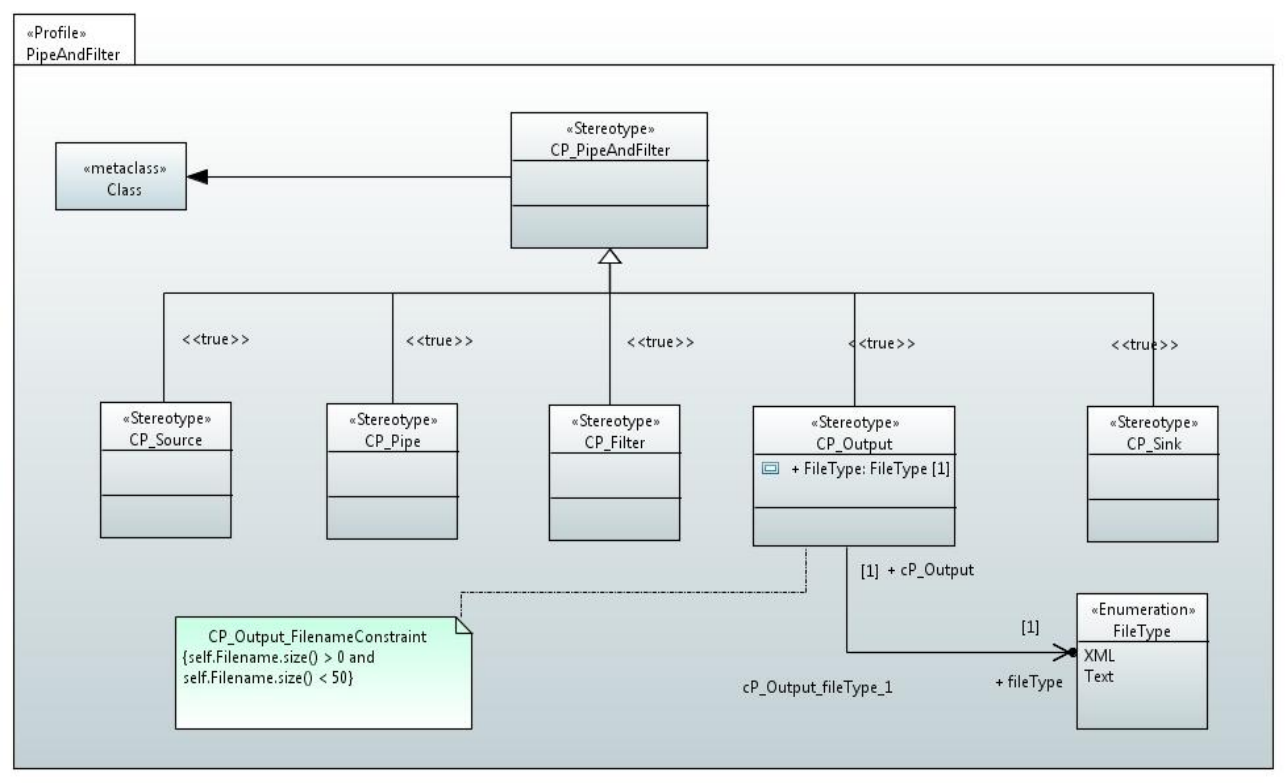
Pipe And Filter: Each Transformation is achieved through a Filter. Much like in a Compiler going Through its various transformation before execution.

Software Architecture con’t

Secondary Pattern

Model-View-Controller since the editor is primarily a visual software, and the multiple views for the model. Delegate controllers update and refresh the views.

Main Package Profile



Generative Architecture

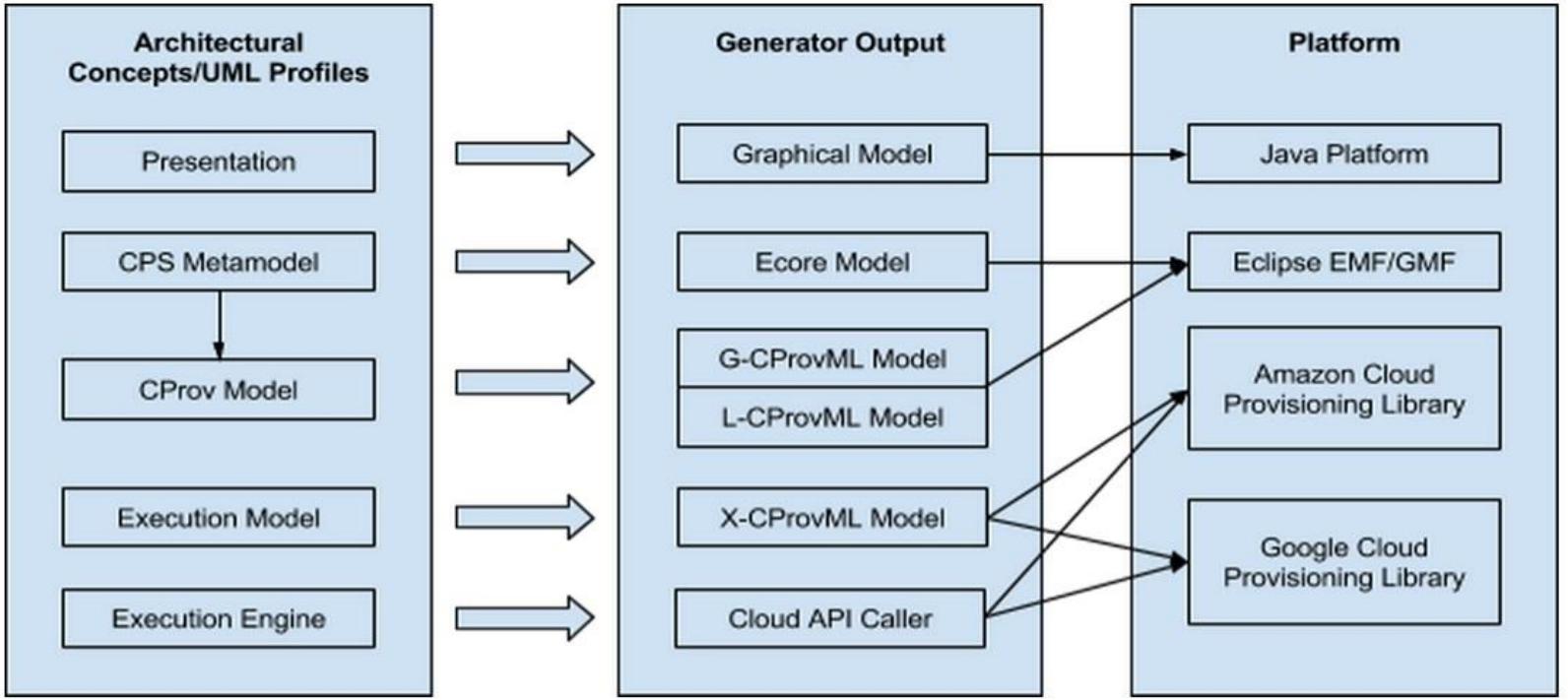
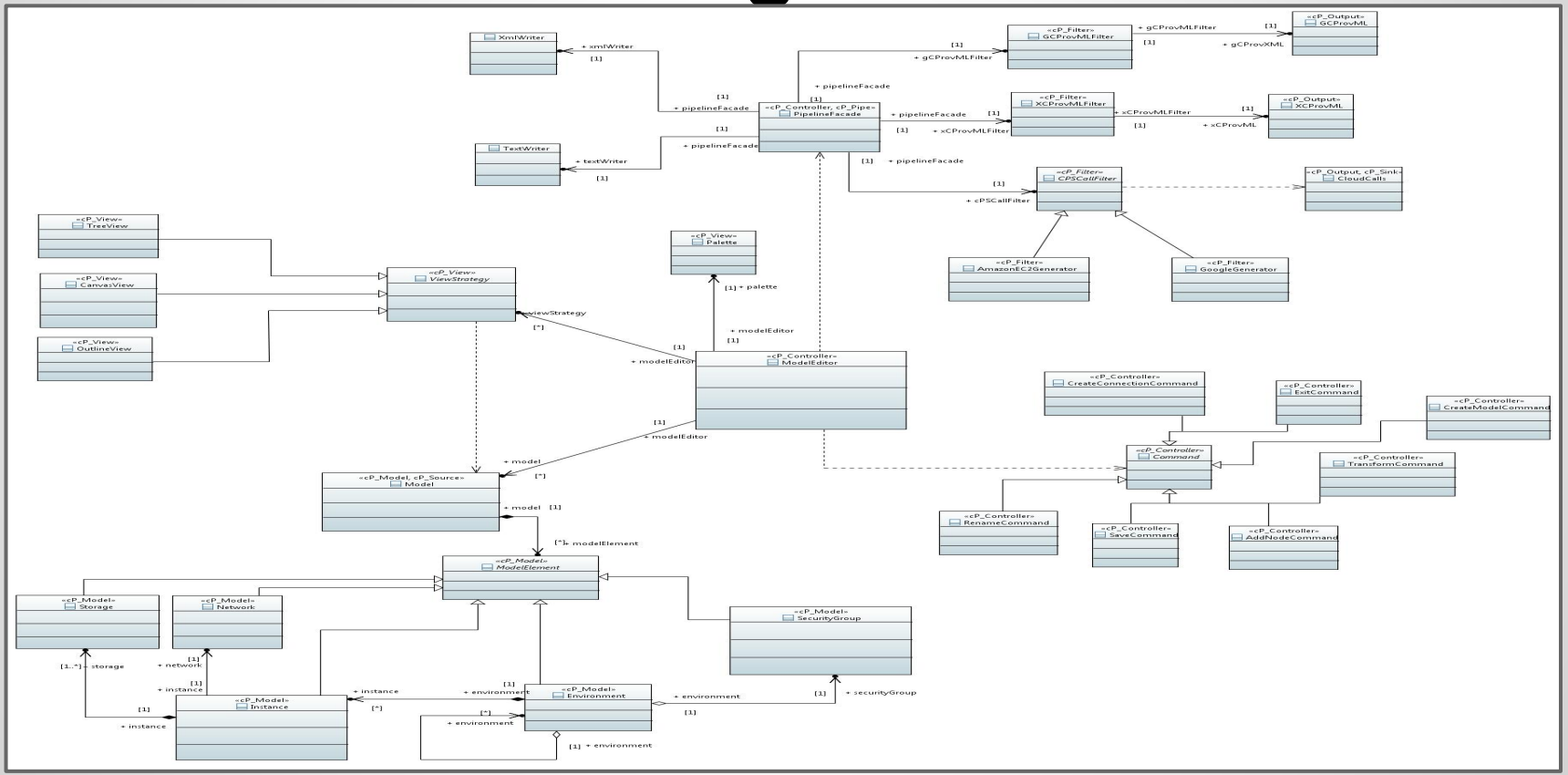


Figure 2-4 Generative Architecture Overview

Generative Architecture (con’t)

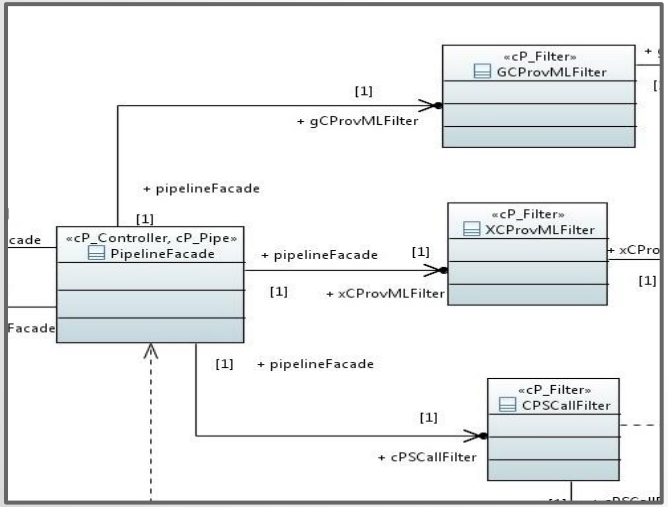
- A major subset of our graphical model is generated, but there are still pieces that we will customize using the java platform.
- CProv model generated is represented with 2 generated XML files: graphical model (G-CProvML) as well as a layout model (L-CProvML)
- X-CProvML model and is realized by several cloud provisioning libraries for Amazon and Google.
- E.E. is realized by the Cloud API Caller which uses the cloud provisioning platforms previously mentioned.

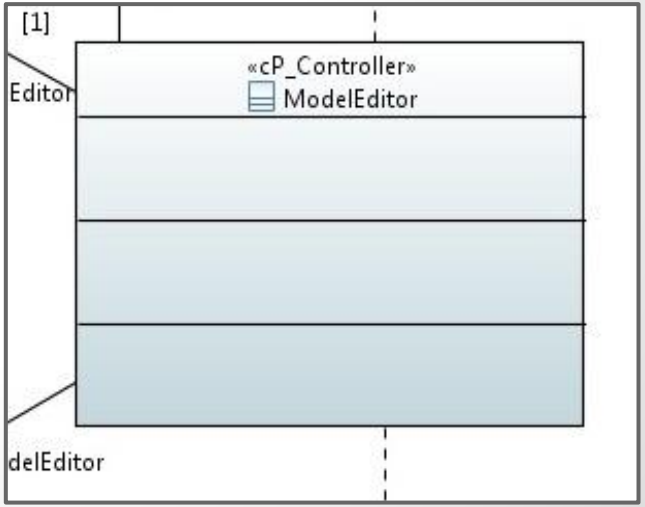
Minimal Class Diagram



Minimal Class Diagram con’t

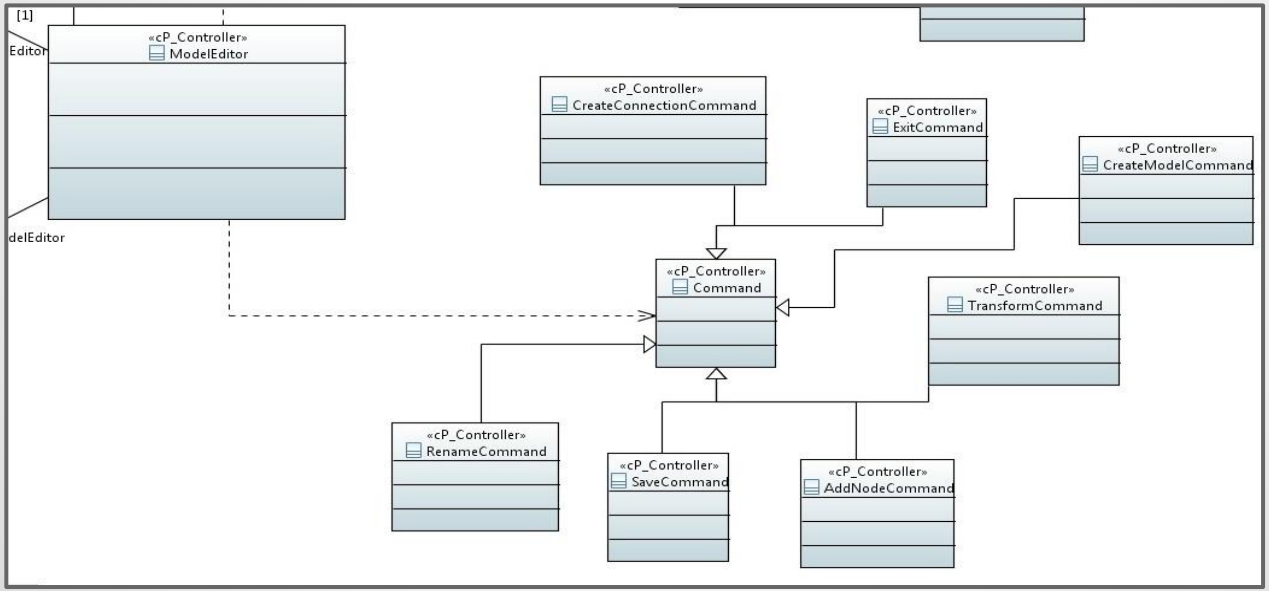
Singleton and Facade: Singleton only allows one instance of the ModelEditor class, and the Facade promotes low coupling between the subsystems.





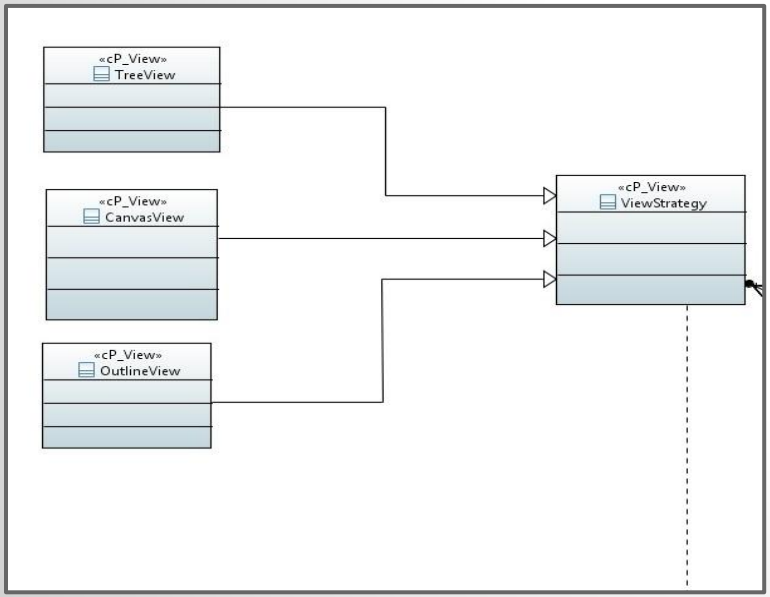
Minimal Class Diagram con’t

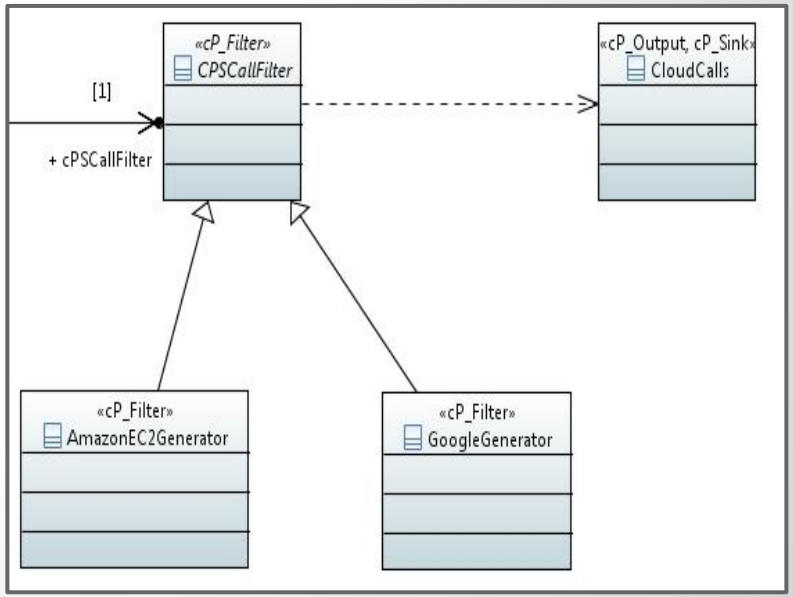
Command Pattern: Keeps track of certain states, in order to allow the user undo/redo functionalities.



Minimal Class Diagram con’t

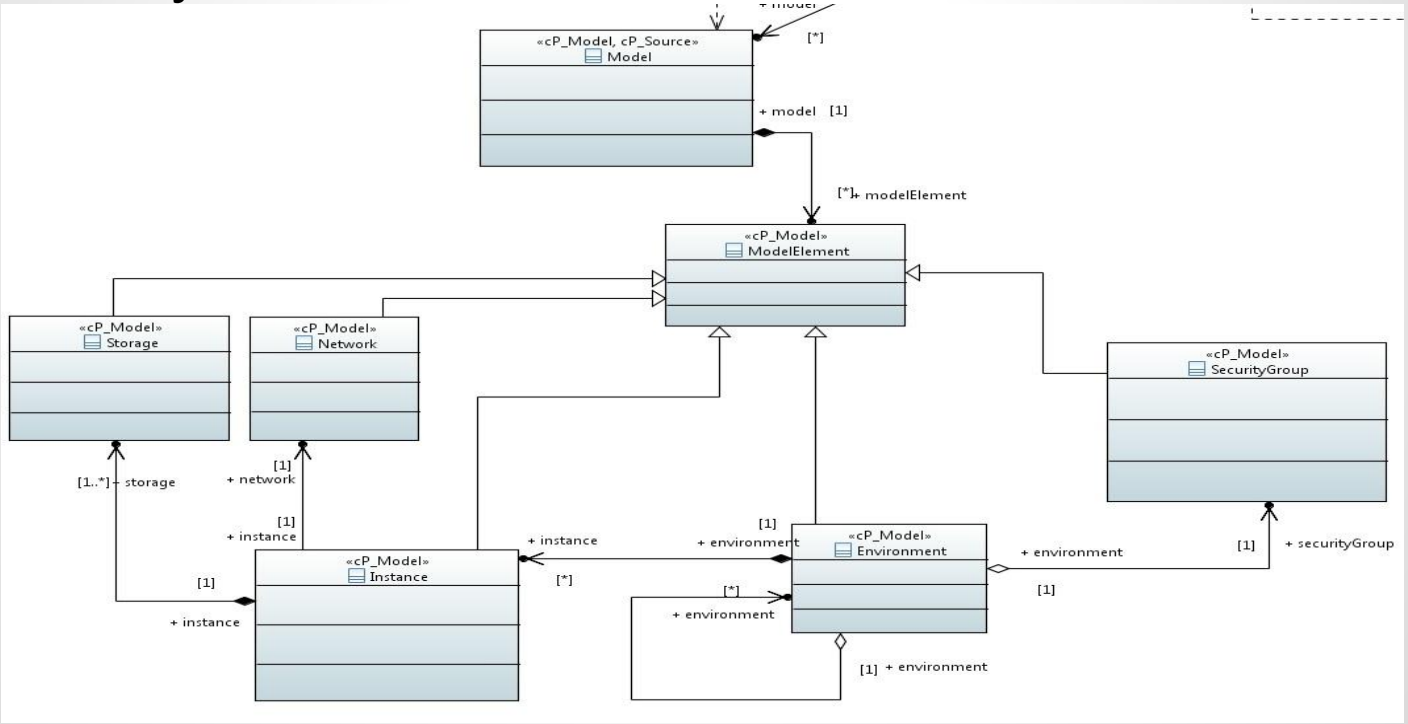
Strategy Pattern: used here to invoke different algorithms to render diverse views, here most views are rendered together.





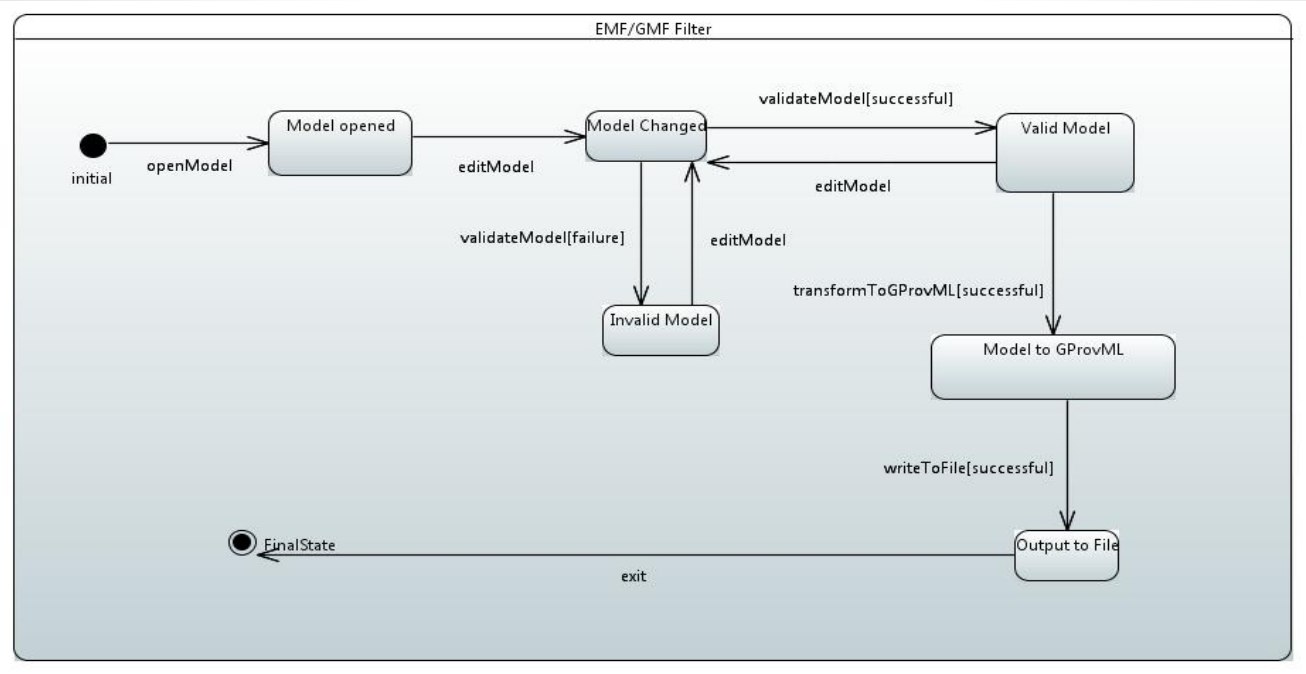
Minimal Class Diagram con’t

Model Subsystem:



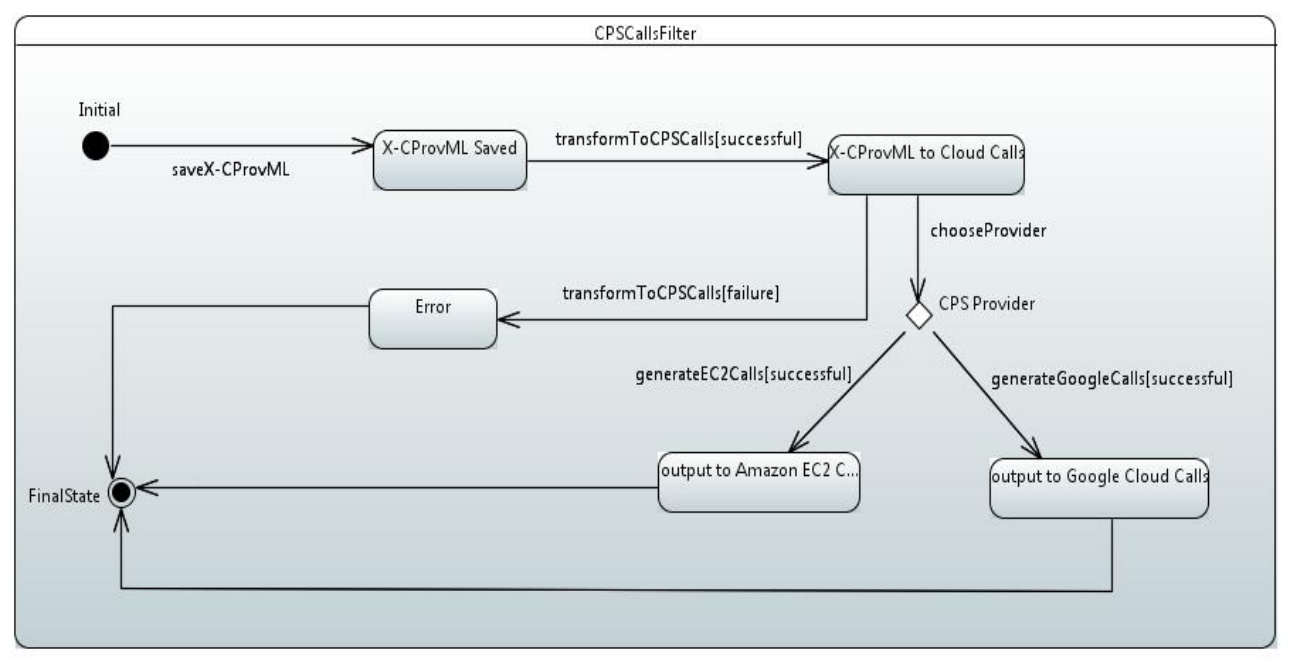
Object Interaction

State Machine Diagram (EMF/GMF filter)



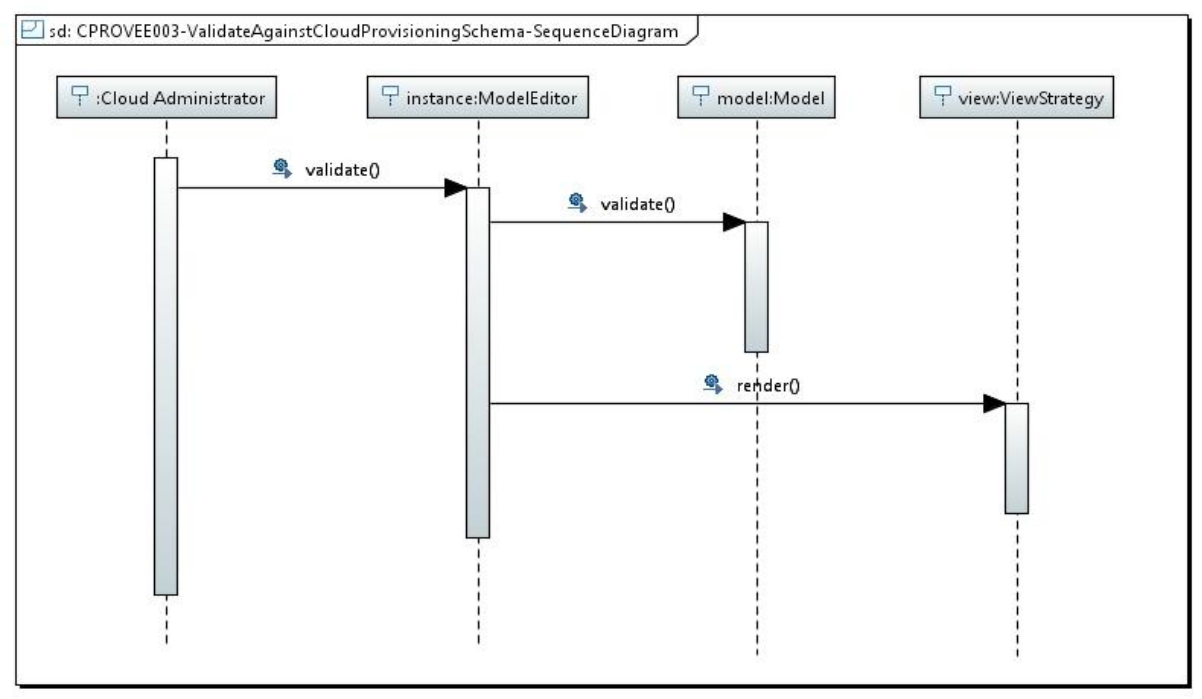
Object Interaction con’t

State Machine Diagram (CPSCalls Filter)



Object Interaction con’t

Sequence Diagram (Validate)



Object Interaction con’t

Sequence Diagram (AddConnection)