Abstract view, to get callbacks when model changes. Subclasses of the view are used to interact with the model. This locks who does what. You have to be view and we can lock which view does what. Another rule, when you react to callback. You are not allowed to change model from a callback. View is most of the time, what is called model view. It takes huge model and translates the model to view specific data structure.

Graphics scene and layout of the items.

QML2PUPPET

3D Node Editor View

View Model

2D editing has two scene graphs. We synch the scene graph. As long as you only move items, it works pretty well. If you move item, we move it in the Qt Creator scene graph. You feel the lag in two cases. If there is side effect when you something that affects something else. E.g. anchors. If you move item A, then item B that is anchored to item A moves with lag.

If you anchor item with fill. If you ask instance "what is your size" as you'd have to wait for everyhitng to go through the puppet and things to settle. You do action, the results show up in 20 ms.

Rewriter creates QML code and reads it. Bridges QML code to model. Unit tested a lot.

QML File

If you want to get data of instances you use QML Object node.

Formeditor View

View Model

Navigator View

View Model

Propertyeditor View

View Model

Node Instance View (objects in QML2PUPPET)

Timeline View

View Model

Rewriter View

View Model

Meta System

Meta system is very close to the rewriter. Try to keep meta system separete. It's a huge proxy on QML code model. That simplifies the data in the QML. Types are not normalised in the model. So this handles that.

QQuickItem has C++ name QQuickItem, in model has two names. Both are types that can come up in the code model. The glue code on top cleans up. This ensures the mess only stays in the meta system.

Model

Model is at the core of everything. It is not "intelligent" intentionally. Basic QML DOM in memory. Nodes, children, with properties.

Model recreates the QML semantics, which can have bugs...

QmlModelNodeFacade

-m_modelNode

ModelNode

QmlObjectNode

Without ANY QML semantics. Lowest level that HAS TO work. If model doesn't work, then there is bug in the how the in memory data struct works. If model doesn't work, then there is bug in the how the in memory data struct works

Basic QML syntax implemented here (not even Qt Quick yet). All of Qt Quick 3D kind of works. We don't know they have mesh or material or matrix. It breaks couple of things. They are forced to show as visible items in the navigator.

Integrating all the views.

Viewmanager

adds 3d node edit view

Integration to rest of Qt Creator

Documentmanager

Doc changes, save, undo, redo, copy, paste etc.

3D Editor Plugin

QML Designer Plugin

Actionmanager

Actions on 3D Nodes

3D Select Tool

3D Move Tool

3D Rotate Tool

Created to easily define actions that can be shown in the menu and toolbar etc.

Qt Creator has it's own action manager. Qt Creator has it's actions spread over it's plugins. There is no architecture there. So we disable plugins, e.g. QML code model depends on C++ code model.