**Model–view–viewmodel**



Model: representation of the problem domain, contains behaviour and data.

View: what the user sees – the look of the system.

Viewmodel: a view of the model. State of the data in the model. Behaviour of the model. Abstraction of the model.

Binder: crucial part of the pattern, allows declarative databinding to avoid boilerplate required to sync the view and the model.

Designer and developer may work in parallel. Eliminates most of the code-behind (code to support the view)

In WPF (Windows Presentation Framework), this pattern is implemented, and GUI developers can write only XAML in order to design GUI.

# Model–view–presenter

# https://upload.wikimedia.org/wikipedia/commons/d/dc/Model_View_Presenter_GUI_Design_Pattern.png

Model: data and behaviour.

View: passive interface -> display data, routes events to presenter

Presenter: format data, acts upon model and view.

Better separation of concerns.

Implemented in Windows forms.

Usually the view instantiates the presenter, giving a reference to itself.

You should define an interface of the view in order to replace it with a mock, when testing the presenter.

# Model–view–adapter

Mostly the same structure as in MVP, with the difference that the adapter has a reference to the view. This way the adapter observes the events of the view.