A screenshot of a computer

Description automatically generated**MVVM Architechture**

A blue square with white text

Description automatically generated with medium confidence

- Model: Models encapsulate the application’s data and business logic, having absolutely no relation to visual aspects

(Engines)

In our project, the “Models” are more like “VM”

🡪 Material Properties have SetHelper() to send signals to View & raise changes

- View model: Contains all application logics (get data from Model, return data, communicate with UI – View)

- View: Receiving User inputs and display UI (never know about Model)

\*\* VM communicates with View through **DataBinding**

* Databinding can be implemented with callbacks (**sending signals**)
* The VM is sometimes exposes Model classes directly to the view, in which case the Model class would need to support data binding and change notification events.

**Project Walkthrough**

* Values are entered from UI: either by entering on keyboard (needs to handle the data change as well as UI change), importing from .dll files (almost never), opening up a saved project file (copying the contents to the tree), or directly choosing the copy option.
* RSDataModel
* (partly) RSData

A screenshot of a computer program

Description automatically generated with low confidenceA screenshot of a computer program

Description automatically generated with low confidence

**RSDataModel**

**Definitions**

1. **Definitions** (static)
   1. 6 TestTypes (Uniaxial, Triaxial, SimpleShearTest, …)
   2. SeepageConditions, LoadingDirections, LoadingPlaces… (Selectable attributes)

**Models**

1. **RSDataPropFile** 
   1. Units: units used in file
   2. List<MaterialModel>
   3. fromXML() initializes the Units and Materials properties by deserializing the Units object and a list of MaterialModel objects from the RFC.XObjectElement.
   4. toXML serializes the Units object and each MaterialModel object in the Materials list to an RFC.XObjectElement
2. **RSDataPropFileManager** (static)
   1. LoadRSDataPropFile() Loads file from user 🡪 RSDataPropFile
   2. SaveRSDataPropFile() new RSDataPropFile 🡪 save to file (doc.save())

**Materials**

1. **Failure Range** Defines properties for the failure range of material
   1. FailureRange (MaterialModel)
   2. Numpoints, Signma3Max, unitWeight…..
2. **StrengthResult**
   1. ResultValid, OnPropertyChanged
   2. HB Classification & Criterion, MohrColumbFit, Rock Mass Parameter, …
3. **MaterialModel**