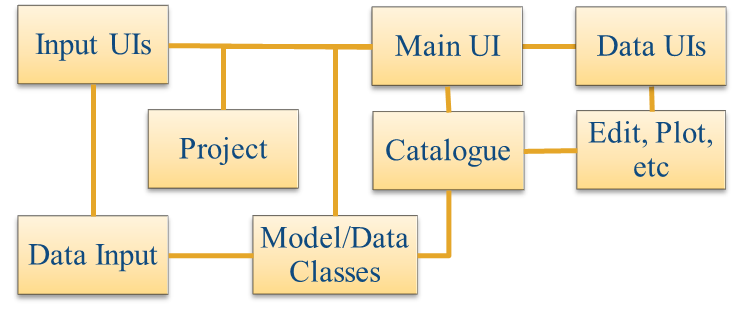
User Manual

Developers Manual

Adapting the default UI

Creating a bespoke UI

Figure 1 -schematic of ModelUI architecture (see Appendix A and B for further details)



# Appendix A – muitoolbox

*muitoolbox* is a collection of classes used to create bespoke UIs for models and data analysis. These include:

**muiModelUI** – an abstract class that defines the requirements for a user interface and provides several methods that are implemented unless overloaded in the implementing class.

**muPropertyUI** – an abstract class the provides the methods for loading and displaying input parameters, minimising the effort to set-up interactive data input.

**muiCatalogue** – manages the storing and access to model Cases (imported data or model outputs).

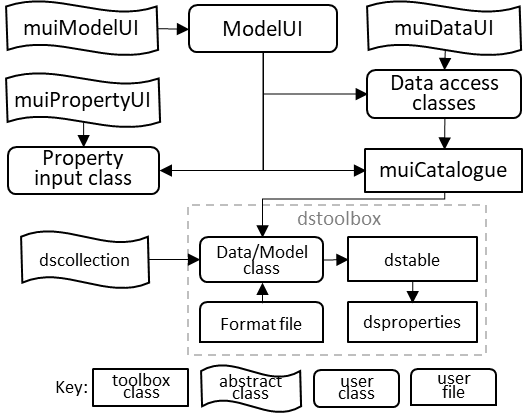
**muiDataUI** – an abstract class that defines the requirements for user interfaces to access the data.

**muiProject** – holds path and file of current project (working model or data set) and project details.

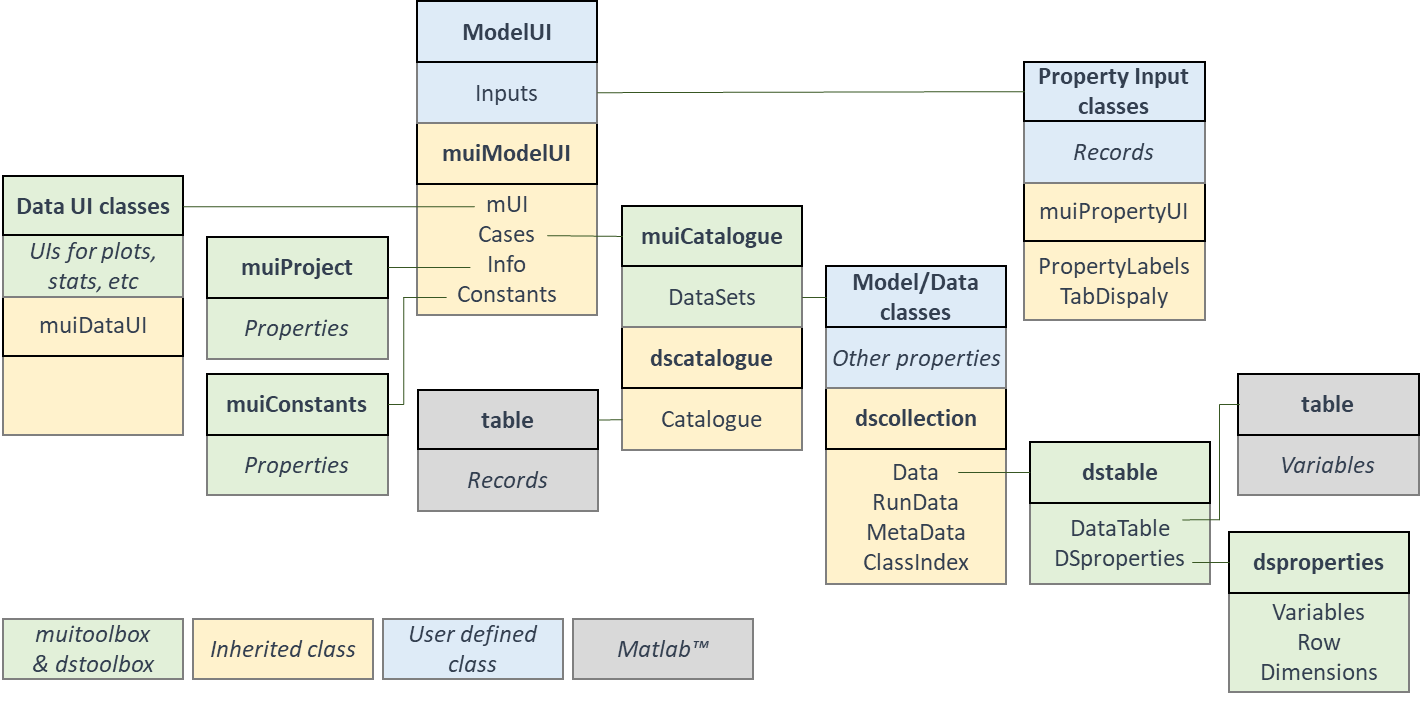
**muiConstants** – standard physical values (acceleration due to gravity, densities, year to seconds, etc).

The *muitoolbox* integrates with the *dstoolbox,* which stores and manages access to multi-dimensional data sets (see Appendix B).

These classes can be used together as illustrated in the following figure, where **ModelUI** is the class that defines the bespoke UI :



The purpose of the muitoolbox is to minimise the effort in creating or prototyping an interface for a model or data analysis tool. Creating a new model requires 3 components to be defined, namely the interface (ModelUI in the above illustration), one or more classes to manage the input of model parameters (if required) and the classes to hold imported data, or running a model and storing the output. Central to this is the holding of input data in the Inputs property and accessing the data via the Cases property. In this context, Cases comprise a record of each Case and a dataset. The records are held in the Catalogue property and the dataset (an instance of the data or model class) in the DataSets property of muiCatalogue. Each data or model class stores the dataset in the Data property, with additional information held in the RunData property (e.g. holding input parameters of a model run). Any type of dataset can be stored in the Data property but when using the dstoolbox multidimensional data sets can be stored using dstable and a full set of meta-data attached using dsproperties. The overall architecture and the properties that provide the links between one class and another are shown in the flow chart below.



## Usage

????

# Appendix B – dstoolbox

*dstoolbox* is a collection of classes used to store and manage access to multi-dimensional data sets. These include:

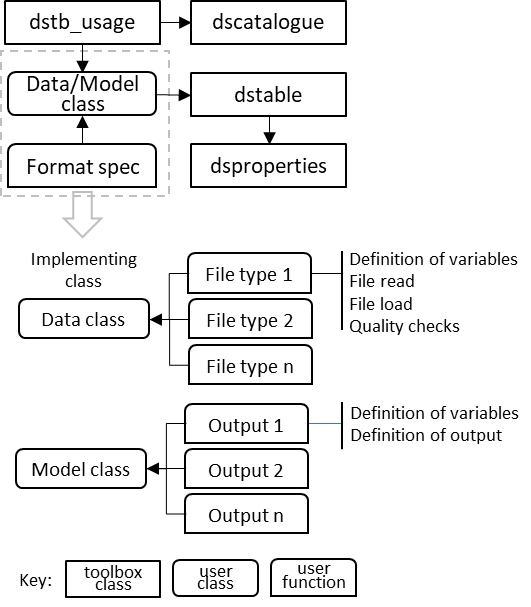
**dstable** - holds a collection of one or more datasets with one or more common dimension vectors.

**dsproperties** - defines the object used to assign metadata to a *dstable.*

**dscatalogue** - manages a catalogue of data sets

that handle all types of data set (eg imported or model data) which are loaded into *dstables* and catalogued using *dscatalgue*.

These classes can be used together as illustrated in the following figure:



## Usage

The toolbox is designed to store and manage multi-dimensional data sets, including meta-data of the variables and all dimensions and manage access to a collection of classes that hold data sets using a catalogue. In the schematic outline *dstb\_usage* is a class to illustrate how *dstable*, *dsproperties* and *dscatalgue* are used. Data are loaded into a *dstable* with relevant metadata added to the table and made accessible using *dsproperties*. Each time a class adds data a record is added using *dscatalogue*. The ‘Format Spec' user functions, shown in the upper part of the figure, are implemented with functions, indicated by 'File Type' and 'Output Type', shown in the lower part of the figure. These define the meta-data of the data set being saved (and any input parameters, or details needed to read and load data from a file, depending on the application).