Deviser tooltips

A couple of notes:

1. I have used SBML and been specific about packages for now as really that is what it is designed for. In some cases it is hard to actually explain without being that specific. I know we will have to revise this but not in a hurry ☺

2. In some places I have given a different tip for the same field but on different pages. Eg the Class attributes and child elements and the additional files stuff. If this is not possible let me know and I will revise as necessary.

3. For each field that needs a tip I have stated the field and then the sentence underneath is the suggested tip. Hopefully you should be able to copy paste.

# Front page

Name:

The short **name** or **label** given to this package. This will be used in the associated URL and as the package XML prefix.

Full Name:

The full title of the package as it would be written in a specification document.

required:

All SBML Level 3 packages must state whether the package may change the mathematical interpretation of the core model. This is the value given to the ***required*** attribute on the <*sbml>* element. Check this box if this package is required.

requires additional code:

In some cases it may be necessary to add code that is very specific to a package and will not be generated by deviser. Check this box if you wish to specify the location of such code.

Declaration:

The location of an additional **header** file that should be included within the generated code.

Implementation:

The location of an additional **code** file that should be included within the generated code.

NOTE FROM SARAH: I think the Number and offset text say enough and don’t need a tooltip.

# Version page

Core level:

The **SBML Core *level*** being targeted by Deviser. Currently only SBML L3V1 is supported.

Core version:

The **SBML Core** ***version*** being targeted by Deviser. Currently only SBML L3V1 is supported.

Pkg Version:

The ***version*** of the **SBML Level 3 Package** being described.

Classes:

Classes added to the package definition will be listed here. The order will determine the order in which Deviser generates the code.

# Add Class page

Name:

The name of the class that will be used by the code generation. The XML output may use a different name (see **XML ElementName**).

BaseClass:

The name of the base class from which this class is derived (if such exists). Note it need not be a class that is being described within the package; for most SBML L3 packages this will be ***SBase***.

TypeCode:

The value that will be used in an enumeration of types for this package. Typically a TypeCode takes the form SBML\_PACKAGE\_CLASS.

XML ElementName:

An optional name for this class to be used in the XML output where the XML output does not use the **Name** supplied. If blank, this defaults to **Name**.

hasListOf:

Check this box if this class is expected to be found within a corresponding listOf class.

ListOfName:

An optional name for the listOf class to be used in the XML output where the XML output does not use the default *listOfBars*, where Bar is the **Name** of the class (see ListOfClassName).

ListOfClassName:

An optional name for the ListOf class to be used by the code generation where this does use the default *ListOfBars*, where Bar is the **Name** of the class.

NOTE FROM SARAH: I would swap ListOfName and ListOfClassName around just as it makes more sense to me that one comes before the other.

Minimum number of children:

The minimum number of child elements expected when a listOf class is present. Note this needs to be ***1*** if the listOf cannot be empty. A missing entry implies there is no minimum.

Maximum number of children:

The maximum number of child elements expected when a listOf class is present. A missing entry implies there is no maximum.

isBaseClass:

Check this box if this class is the base class for further derived classes.

requires additional code:

In some cases it may be necessary to add code that is very specific to a package and will not be generated by deviser. Check this box if you wish to specify the location of such code.

Declaration:

The location of an additional **header** file that should be included within the generated code for this class.

Implementation:

The location of an additional **code** file that should be included within the generated code for this class.

Class attributes and child elements:

Each attribute and child element of the class being defined should be listed here.

ListOf attributes:

Each attribute of the ListOf class should be listed here. If the ListOf class contains elements other than the class being specified these should also be added here.

Attributes/elements table headers

Name:

The name of the attribute/element to be used by code generation. The XML output may use a different name (see **XML Name**).

Type:

The type of the attribute/element. Allowed values are: Sid, SIdRef, string, bool, doube, int, unisigned int, UnitSId, UnitSIdRef, enum, element, lo\_element, inline\_lo\_element.

Element:

This field provides additional information depending on the **Type** of the attribute/element. It may be the name of the element, enumeration or object being referenced.

Required:

States whether the attribute or element is mandatory. This should be ***true*** if the attribute/element is mandatory; ***false*** if not.

isBaseClass:

States whether this element is a base class. This should be ***true*** if the element is a base class and therefore not instantiated directly; ***false*** if not.

XML Name:

The name of the attribute/element as used by the XML output. If blank, this defaults to the **Name** given.

Instantiations:

Any classes derived from this class should be listed here.

Instantiations table headers

XML Name:

The name of the instantiation of this class to be used by the XML output.

Element:

The **Name** of the class that will be derived from this base class. Note this class should also be defined within this package.

Min No Children:

The minimum number of child elements that may be present on this element. A missing value implies there is no minimum.

Max No Children:

The maximum number of child elements that may be present on this element. A missing value implies there is no maximum.

NOTE FROM SARAH: I’m not sure what else to say about min/max numbers but I don’t think this is enough. Also should I use ‘on this element’ or ‘on this instantiated element’.

# addPlugin page

ExtensionPoint:

The **Name** of the class being extended by this plugin element. Note this can come from Core or another Level 3 package.

elementFromCore:

Check this box if the element being extended is part of SBML L3 Core.

Package:

The short **name** or **label** of the package being extended by this plugin object.

TypeCode:

The TypeCode of the specific class being extended by this plugin object.

has Attributes:

Check this box if this plugin adds attributes to the element being extended.

Class attributes and child elements:

Each attribute and child element of the plugin object being defined should be listed here.

Defined Classes:

This is a list of all the classes that have been defined by this package object. Highlight the classes used by this plugin and use the arrows to add them to the **Child Classes** list.

Child Classes:

This is a list of the child classes that will be present on this plugin object. Populate this list by highlighting the relevant class in the **Defined Classes** list and using the arrows to move it.

requires additional code:

In some cases it may be necessary to add code that is very specific to a package and will not be generated by deviser. Check this box if you wish to specify the location of such code.

Declaration:

The location of an additional **header** file that should be included within the generated code for this plugin.

Implementation:

The location of an additional **code** file that should be included within the generated code for this plugin.

# addEnum Page

Name:

The name of the enumeration type. Note the code generator will add ‘\_t’ to this name.

Enum table

Name:

The name that will be used for this enumeration entry. Note typically the enumeration entry names are given as CLASS\_ENUM\_VALUE.

Value(string):

The string value of this enumeration entry. This is the text that will appear in the XML output for this enumeration value.

# Mappings page

Package:

This is the short **name** or **label** of the package in which the listed class originates.

NOTE FROM SARAH: I think the rest of this is self-explanatory.

# Generate window

NOTE FROM SARAH: I’m not 100% sure I am correct in some of these – so please do correct as necessary.

Output Dir:

The directory where any output will be placed.

Package File:

The file corresponding to the package being defined. This field will be populated with the file that has been saved before Generation is possible.

Package Name:

The short **name** or **label** for the package being defined.

Open Output Dir:

Opens the directory specified in the **Output Dir** field.

Generate Package Code:

Creates the directory structure and generates the code based on the package defined.

LaTeX generation tab:

Generate Latex:

Generates the tex files necessary to create a minimal specification of the SBML L3 Package described. This includes definitions of classes and attributes and a set of validation rules.

Compile TeX:

Invokes the MikTex application to generate a pdf of the specification files. Note this is based on the template for SBML Level 3 Package specifications.

Integration and Testing tab:

Compile Dependencies:

Invokes the compiler to generate the necessary dependencies for libSBML.

Add Code to Source Dir:

Overlays the code generated for this package onto the main libSBML source tree. Note the code generated includes all the necessary CMake files to allow the package code to be integrated into the build.

Remove Code From Source Dir:

Removes the generated code from the main libSBML source tree.

Compile libSBML with package:

Invokes the compiler to generate libSBML with the newly defined package enabled. Note: You need to have added the code to the source tree prior to this step.

Test libSBML:

Opens an interactive python session with the newly compiled libSBML loaded to facilitate manual testing.

NOTE FROM SARAH: Please change the name of this from ‘Test libSBML’ to ‘Run libSBML Python session’