# C++ Coding Guidelines for COPASI

# Under Development

Stefan Hoops on October 29, 2007

# Contents

# List of Tables

# Chapter 1

# Naming Conventions

The intent of naming conventions is to allow programmers, which are not familiar with the code to easily grasp the meaning and scope of symbols in the source code. Each programmer of COPASI should adhere for its own benefit and to the advantage of the project to the following conventions.

## 1.1 Class Names

Class names must all start with a capital letter `C`. This is followed by a descriptive name. This name might be composed by different words. These words must all start with capital letters and are concatenated without underscores. Good examples for class names include: `CCopasiXMLParser` , `CExpatTemplate`, and `CMathModel`.

## 1.2 Variable Names

In general a variable name should be descriptive. This name might be composed by different words. These words must all start with capital letters and are concatenated without underscores. In addition the following standards should be followed:

| | |
|---|---|
| **Counters** | might be used such as `i`, `k`, and `l`, which may be used in loops. |
| **Iterators** | might be used such as `it` and `end`, which may be used in loops. |
| **Pointers** | are prepended with a lower case letter `p`. |
| **Method Parameters** | must start with a lower case letter. |
| **Class Member Variables** | are prepended with a lower case letter `m`. |
| **Class Member Pointers** | must have the prefix `mp`. |

## 1.3 Method Name

Method names should have a descriptive name starting with a lower case letter. This name might be composed by different words. These words beginning with the second must start with capital letters and are concatenated without underscores. Good examples for method names include: `createMetabolite`, `compileIfNecessary`, and `buildMoieties`

In addition the following standards should be followed:

| | |
|---|---|
| **Retrieval Methods** | must start with `get` followed by the member variable name without the prefix. |
| **Set Methods** | must start with `set` followed by the member variable name without the prefix. |
| **Boolean Query Functions** | should start if applicable with `is`. |

# Chapter 2

# Program Code Guidelines

## 2.1 Use of #include

To minimize compilation problems on the different platforms and with the libraries COPASI depends on it is necessary to define some rules, which need to be followed when specifying include files in the COPASI code.

The #include statement distinguishes between two different type of include files.

- **Global Includes:** Global include files are indicated by surrounding the filename with < and >, i.e. the statements looks like: `#include <filename>`. All files not located in the COPASI source tree are considered global includes.
- **Local Includes:** Local include files are indicated by surrounding the filename with double quotes, i.e. the statements looks like: `#include "filename"`. All files located in the COPASI source tree are considered local includes.

### 2.1.1 Order of #include in Header Files

The number of include statements in header files should be kept to the minimum necessary to compile the header file, i.e., a COPASI code file just containing the following two lines of code must compile successfully:

```
#include "copasi.h"
#include "filename"
```

To achieve the minimal list of includes one should replace all not needed includes with forward declarations as often as possible. Please note that it should never be necessary to include `copasi.h` in a header file.

1. **Qt Header Files:** Qt header files must appear first. Please note that the Qt header file names are all lowercase.
2. **Global Header Files:** Other global header files must appear next.
3. **COPASI Header Files:** All COPASI header files must be included with the full filename starting with the `copasi` directory, i.e., the include statement looks like `#include "copasi/.."`

### 2.1.2 Order of #include in Code Files

1. **Qt Header Files:** Qt header files must appear first. Please note that the Qt header file names are all lowercase.
2. **Global Header Files:** Other global header files must appear next.
3. **copasi.h:** This file must precede all other local includes since they depend on the defines of `copasi.h` to be present.
4. **COPASI Header Files:** COPASI header files must be included after `copasi.h`.
5. **blaswrap.h or clapackwrap.h:** These two wrapper files must appear last as their defines often conflict with other includes.

## 2.2 Loops

Counter or iterators used in loops must not be declared in the `for` statement as the scope of the variables declared within is not well defined in C and C++ standard and thus leads to compile problems. The solution is to define those variables prior to the `for` statement.

# Chapter 3

# Code Documentation

## 3.1 Class Documentation

The COPASI team uses doxygen to generate API documentation. It is therefore necessary that the classes, methods, and variables or documented completely.

## 3.2 Variable Documentation

## 3.3 Method Documentation

## 3.4 Incline Code Documentation

# Chapter 4

# COPASI Object Structure

COPASI has an object structure which is used to access all objects which can be calculated, plotted, or printed. Objects can be grouped to larger objects called containers. COPASI itself functions as a root container, i.e., it contains all objects which might be of interest for calculation or output. The access to any object is provided with an LDAP like common name (CN) which can be resolved starting from any container.

## 4.1 Object

The object (`class CCopasiObject`) is the main building block of the object structure. Each object has the following main attributes:

ATTRIBUTE LIST

**`std::string mObjectName`** The name of the object.

**`std::string mObjectType`** The type of the object, e.g. Compartment.

**`CCopasiContainer * mpObjectParent`** A pointer to the objects parent container.

**`unsigned int mObjectFlag`** A flag indicating object properties.

In addition to the expected `get` and `set` methods the objects provides the following methods:

METHOD LIST

**`CCopasiContainer * getObjectAncestor(const std::string &type) const`** This method retrieves the closest ancestor of the object of the specified type.

**`virtual CCopasiObjectName getCN() const`** This method retrieve the common name (CN) of the object.

## 4.2 Container

## 4.3 COPASI Vector

## 4.4 Parameter

## 4.5 Parameter Group

## 4.6 Dependencies

# Chapter 5

# The Model State

## 5.1 State Template Order

This section describes whether and in which order the above calculated simulation objects appear in the initial and current state of the model. The order is described by the class CStateTemplate. Please note that all objects with the type "Fixed" are marked "Unused" as there is no need to calculate them during simulation.

| Object | Type | Status |
|---|---|---|
| Model | Time | Used |
| Model Value | ODE | Used |
| Compartment | ODE | Used |
| Metabolite | ODE | Used |
| Metabolite | Reaction | Independent |
| Metabolite | Reaction | Dependent |
| Metabolite | Assignment | Used |
| Compartment | Assignment | Used |
| Model Value | Assignment | Used |
| Any | Any | Unused |

Table 5.1: State Template Order

## 5.2 Setting the Initial State

The initial state of a COPASI model is due to the introduction of assignments for all model entities no longer determined through fixed values. To calculate the current initial values those assignments have to be evaluated at $t = T_0$. These evaluation will effect in many cases other initial values. For example the initial particle numbers of metabolites will change when a compartment's initial volume changes. To detemine the initial values, which need to be recalculated as well as the functions used for their calculation COPASI uses a dependecy system. In this system each initial value stores the information on which values it depends and which method to call to recalculated its value.

Since the change of one initial value may now result in a complicated chain of updates for other initial values it no longer suffices to use its update method to set its value. We need to add an additional step which will update all values effected by the change. To retrive the list of needed refreshes in the proper order the class CModel provides the following method.

**`vector<Refresh *> buildInitialRefreshSequence(set<const CCopasiObject *> &changedObjects);`**

This method retrieves a vector of refresh methods, which must be called to update all other initial values given a set of changed objects. If the set of changed objects is empty the method will return a vector of refresh method, which suffices to synchronize all initial values.

## 5.3 Setting the State

## 5.4 Applying Assignments

## 5.5 Calculating Current Values

## 5.6 Dependency Specifications

### 5.6.1 Initial Values

#### 5.6.1.1 Metabolite with Assignment

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Initial Particle Number | Initial Concentration, parent compartment Volume | refreshInititalValue |
| Initial Concentration | all objects in implicit expression | refreshInitialConcentration |

Table 5.2: Metabolite with Assignment

#### 5.6.1.2 Metabolite without Assignment

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Initial Particle Number | initial concentration, parent compartment volume | refreshInitialValue |
| Initial Concentration | parent compartment Volume | refreshInititalConcentration |

Table 5.3: Metabolite without Assignment

The initial concentration is of course dependent on the initial particle number. This must be taken into account manually to avoid circular depencies. However, this is straight forward as the initial particle number may only be set directly, in which case no further dependencies are encountered or calculated from the initial concentration in the case the concentration is given the normal dependecies take care of the problem.

#### 5.6.1.3 Compartment, Global Quantity with Assignment

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Initial Value | all objects in implicit expression | refreshInitialValue |

Table 5.4: Compartment, Global Quantity with Assignment

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Initial Value | none | none |

Table 5.5: Compartment, Global Quantity without Assignment

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Initial Number | Particle numbers of all involved metabolites | refreshInitialValue |

Table 5.6: Moiety

#### 5.6.1.4 Compartment, Global Quantity without Assignment

#### 5.6.1.5 Moiety

### 5.6.2 Transient Values

#### 5.6.2.1 Metabolite Reaction (Independent)

#### 5.6.2.2 Metabolite Reaction (Dependent)

#### 5.6.2.3 Metabolite Assignment (Concentration)

This will be implemented later

#### 5.6.2.4 Metabolite ODE (Concentration)

This will be implemented later

#### 5.6.2.5 Moiety

#### 5.6.2.6 Reaction

#### 5.6.2.7 Model Value Assignment

#### 5.6.2.8 Model Value ODE

#### 5.6.2.9 Compartment Assignment

This will be implemented later

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Self | none | none |
| Particle Number | Self (state controlled) | none |
| Concentration | Particle Number, parent compartment Volume | refreshConcentration |
| Particle Rate | All particle fluxes of reactions the metabolites participates in | refreshRate |
| Concentration Rate | Particle Rate, Concentration, parent compartment Volume and Rate | refreshConcentrationRate |
| Transition Time | All particle fluxes of reactions the metabolites participates in | refreshTransitionTime |

Table 5.7: Metabolite Reaction (Independent)

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Self | none | none |
| Particle Number | none | none |
| Concentration | Particle Number, parent compartment Volume | refreshConcentration |
| Particle Rate | All particle fluxes of reactions the metabolites participates in | refreshRate |
| Concentration Rate | Particle Rate, Concentration, parent compartment Volume and Rate | refreshConcentrationRate |
| Transition Time | All particle fluxes of reactions the metabolites participates in | refreshTransitionTime |

Table 5.8: Metabolite Reaction (Dependent)

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Self | none | none |
| Particle Number | Concentration, parent compartment Volume | refreshNumber |
| Concentration | all objects in expression | calculate |
| Particle Rate (NA) | none | none |
| Concentration Rate (NA) | none | none |
| Transition Time (NA) | none | none |

Table 5.9: Metabolite Assignment (Concentration)

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Self | none | none |
| Particle Number | Self (state controlled) | none |
| Concentration | Particle Number, parent compartment Volume | refreshConcentration |
| Particle Rate | all objects in expression, parent compartment Volume | calculate |
| Concentration Rate | Particle Rate, Concentration, parent compartment Volume and Rate | refreshConcentrationRate |
| Transition Time | Particle Number, Particle Rate | refreshTransitionTime |

Table 5.10: Metabolite ODE (Concentration)

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Self | Particle number of participating independent metabolites | none |
| Dependent Number | Self | none |

Table 5.11: Moiety

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Self | all variables of the kinetic function, associated compartment Volume, kinetic function, all metabolites involved | calculate |
| ParticleFlux | all variables of the kinetic function, associated compartment Volume | calculate |
| Flux | all variables of the kinetic function, associated compartment Volume | calculate |

Table 5.12: Reaction

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Self | none | none |
| Value | all objects in expression | calculate |
| Rate (not available) | none | none |

Table 5.13: Model Value Assignment

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Self | none | none |
| Value | Self (state controlled) | none |
| Rate | all objects in expression | calculate |

Table 5.14: Model Value ODE

#### 5.6.2.10 Compartment ODE

This will be implemented later

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Self | none | none |
| Volume | all objects in expression | calculate |
| Rate (not available) | none | none |

Table 5.15: Compartment Assignment

| Attribute | Dependencies | Refresh Method |
|---|---|---|
| Self | none | none |
| Volume | Self (state controlled) | none |
| Rate | all objects in expression | calculate |

Table 5.16: Compartment ODE

# Chapter 6

# Installation Structure

This section defines the installation structure for COPASI on different platforms. Each platform will adhere to the platform specific requirements.

## 6.1 Unix

The installation location needs to be available to COPASI at runtime and therefore the environment variable `COPASIDIR` pointing to this location must be set by the user.

```
$COPASIDIR
 +- bin
 |  +- CopasiSE
 |  +- CopasiUI
 +- share
 |  +- copasi
 |     +- doc
 |     |  +- html
 |     |     +- figures
 |     |     |  +- DefaultPlotAdded.jpg
 |     |     |  +- ModelSettingsDialog.jpg
 |     |     |  +- ObjectBrowserSelection.jpg
 |     |     |  +- ObjectBrowserTree.jpg
 |     |     |  +- PlotDefinition.jpg
 |     |     |  +- PlotWindow.jpg
 |     |     |  +- ReactionDialog.jpg
 |     |     |  +- ReactionOverview.jpg
 |     |     |  +- ReactionOverviewEmpty.jpg
 |     |     |  +- ReportDefinitionDialog.jpg
 |     |     |  +- TimeCourseDialog.jpg
 |     |     |  +- ...
 |     |     +- TutWiz-Step1.html
 |     |     +- TutWiz-Step2.html
 |     |     +- TutWiz-Step3.html
 |     |     +- TutWiz-Step4.html
 |     |     +- TutWiz-Step5.html
 |     |     +- TutWiz-Step6.html
 |     |     +- ...
 |     +- examples
 |     |  +- CircadianClock.cps
 |     |  +- Metabolism-2000Poo.xml
 |     |  +- YeastGlycolysis.gps
 |     |  +- brusselator.cps
 |     |  +- ...
```

```
|     +- icons
|        +- copasi icon small.png
+- README
+- ChangeLog
```

## 6.2 MacOS X

The installation location must be available to COPASI at runtime. However it is possible to determine the location through MacOS X.

```
$COPASIDIR
 +- CopasiSE.app
 |  +- Contents
 |  |  +- MacOS
 |  |     +- CopasiSE
 +- CopasiUI.app
 |  +- Contents
 |  |  +- MacOS
 |  |  |  +- CopasiUI
 |  |  +- Resources
 |  |  |  +- doc
 |  |  |     +- html
 |  |  |        +- figures
 |  |  |        |  +- DefaultPlotAdded.jpg
 |  |  |        |  +- ModelSettingsDialog.jpg
 |  |  |        |  +- ObjectBrowserSelection.jpg
 |  |  |        |  +- ObjectBrowserTree.jpg
 |  |  |        |  +- PlotDefinition.jpg
 |  |  |        |  +- PlotWindow.jpg
 |  |  |        |  +- ReactionDialog.jpg
 |  |  |        |  +- ReactionOverview.jpg
 |  |  |        |  +- ReactionOverviewEmpty.jpg
 |  |  |        |  +- ReportDefinitionDialog.jpg
 |  |  |        |  +- TimeCourseDialog.jpg
 |  |  |        |  +- ...
 |  |  |        +- TutWiz-Step1.html
 |  |  |        +- TutWiz-Step2.html
 |  |  |        +- TutWiz-Step3.html
 |  |  |        +- TutWiz-Step4.html
 |  |  |        +- TutWiz-Step5.html
 |  |  |        +- TutWiz-Step6.html
 |  |  |        +- ...
 |  |  +- Info.plist
 +- examples
 |  +- CircadianClock.cps
 |  +- Metabolism-2000Poo.xml
 |  +- YeastGlycolysis.gps
 |  +- brusselator.cps
 |  +- ...
 +- COPASI-README.rtf
```

## 6.3 Windows

The installation location must be available to COPASI at runtime. However it is possible to determine the location through Windows specific means.

```
$COPASIDIR
 +- bin
 |  +- CopasiSE
 |  +- CopasiUI
 +- share
 |  +- copasi
 |     +- doc
 |     |  +- html
 |     |     +- figures
 |     |     |  +- DefaultPlotAdded.jpg
 |     |     |  +- ModelSettingsDialog.jpg
 |     |     |  +- ObjectBrowserSelection.jpg
 |     |     |  +- ObjectBrowserTree.jpg
 |     |     |  +- PlotDefinition.jpg
 |     |     |  +- PlotWindow.jpg
 |     |     |  +- ReactionDialog.jpg
 |     |     |  +- ReactionOverview.jpg
 |     |     |  +- ReactionOverviewEmpty.jpg
 |     |     |  +- ReportDefinitionDialog.jpg
 |     |     |  +- TimeCourseDialog.jpg
 |     |     |  +- ...
 |     |     +- TutWiz-Step1.html
 |     |     +- TutWiz-Step2.html
 |     |     +- TutWiz-Step3.html
 |     |     +- TutWiz-Step4.html
 |     |     +- TutWiz-Step5.html
 |     |     +- TutWiz-Step6.html
 |     |     +- ...
 |     +- examples
 |     |  +- CircadianClock.cps
 |     |  +- Metabolism-2000Poo.xml
 |     |  +- YeastGlycolysis.gps
 |     |  +- brusselator.cps
 |     |  +- ...
 |     +- icons
 |        +- Copasi.ico
 |        +- CopasiDoc.ico
 +- LICENSE.txt
 +- README.txt
 +- ChangeLog
```

## 6.4 Handling Installation Differences

The handling of differences in the installation structure must be dealt with in one place within the COPASI code. The place for this is the class `COptions`. In this class the method:

```
template<class CType> static void getValue(const std::string &name, CType &value)
```

provides access to common options within COPASI. The following values will deal with installation dependent settings: `CopasiDir`, `TempDir`, `ExampleDir`, and `WizardDir`. The following code shows how to retrieve the location of the examples files for COPASI:

```
std::string ExampleDir;
COptions::getValue(``ExampleDir'', ExampleDir);
```

To assure that the values are correctly set any main program must call:

```
COptions::init(argc, argv);
```