OpenDA black box wrapper cookbook

Nils van Velzen

October 16, 2013

1 What not to expect from this document

This document is not a thorough reference manual on configuring and using the black box wrapper of OpenDA. The purpose of this document is to give you some overview information to get you started making your own black box wrapper for your model. There are many example implementations available and at some point we hope that there will be an user guide available for the black box wrapper explaining all the details on configuration and implementation. This kind of detailed information is not to be found in this document. This document should be used in combination with the existing examples and the OpenDA course[1] where you already can find a lot of information on the black box wrapper.

2 The design of the black box wrapper

Using models in black box form in OpenDA means that data assimilation and model calibration techniques are applied to a model without changing the existing model code. OpenDA and the black box wrapper have no knowledge of the model internals (black box) and only uses the input and output files of the model. The black box implementation of OpenDA can also be used in a slightly different way as well like we do for C# models and EFDC, but this is beyond the scope of this document.

The black box wrapper is an implementation of a stochastic model in OpenDA. The difference with a normal model is that the black box wrapper

[1] The OpenDA course is available as a separate download from the OpenDA website.

does not contain any model equations. It is a layer that helps you to link and configure a model to OpenDA using the black box approach. The black box wrapper is illustrated in Figure **??**, the elements of this figure will be explained in some more detail in the following sections.

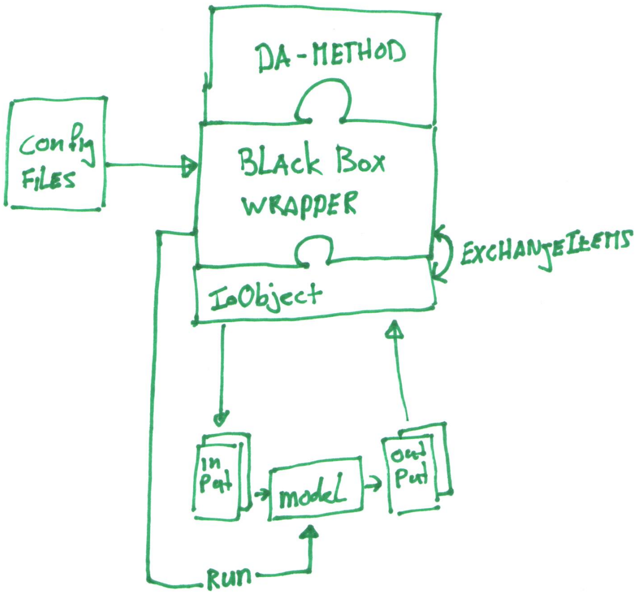

Figure 1: Overview of a black box configuration in OpenDA. The model interacts with the generic black box wrapper of OpenDA implementing an OpenDA stochastic model. The black box wrapper builds its state, parameters and predictions using exchangeItems. The exchangeItems contain values that are read from model output files or need to be written to model input files. The creation of the exchangeItems and reading from/writing to files is implemented by one or more ioObjects. The black box wrapper will run the model executable when all input files are prepared.

3 ExchangeItems

Values from the input and output files are represented in the OpenDA black box wrapper in the form of exchangeItems. An exchangeItem is a container of values with optionally some meta information. In the most simple form an exchangeItem contains a single value or array and has en ID. ExchangeItems can have three different roles:

- input: the values have been read from one of the output files of the model. These values are used by the algorithm but (updated) values will not be written back to the file.
- output: the values will be computed by the algorithm and will be written to the input file of the model.
- inout: the values are both read and written to a file of the model that is both input as output (typically restart files).

There is one special kind of exchangeItem that is now worth mentioning and that is the timeSeriesExchangeItem. This kind of exchangeItem contains a sequence of values that are all associated to a timestamp (meta information). This kind of exchangeItem is typically used for storing model results that need to be matched to observations.

4 Model wrappers

The model uses files for input and output and the black box wrapper uses exchangeItems. Therefore we need to write some code that creates all the exchangeItems and handles the reading and writing of the values in the exchangeItems from and to the model input and output files.

For each file format we need to handle we have to implement a java class that implements the IoObjectInterface interface. This interface only contains three public methods:

- initialize: creates all exchangeItems. The values of the input and inout exchangeItems are read from file.
- getExchangeItems: Returns an array containing all the exchangeItems of this file.

- finish: Write the values of output and inout exchangeItems to file.

OpenDA already contains quite a number of implementations of exchangeItems. In almost all cases you can use one of the existing implementations as container of the values for your model. Useful existing implementations include:

- org.openda.exchange.DoubleExchangeItem: ExchangeItem for a single double
- org.openda.exchange.DoublesExchangeItem: ExchangeItem for an array of doubles
- org.openda.exchange.timeseries.TimeSeries: Timeseries of double values each having a time associated.

Your implementation only needs to create those exchangeItems that you think you will need. In an first implementation you do not need to bother about all possible data in the files. Additional exchangeItems can always be added later when needed.

5 Recipe

- Identify the input and output files that contain values you need and determine how these values are stored
- Write an IoObjectInterface class for each file format you need such that you have exchangeItems for all the data you are going to use in OpenDA. Try to use existing implementations of exchangeItems when possible. For inspiration look at available example implementations. Note: Use a "simple" wrapper like reactive_advection_pollution_wrapper from the OpenDA course or model_nemo as inspiration and not one of the very complex wrappers with a lot of support and functionality.
- set up a black box configuration environment (model input and configuration files). Best way is to copy an existing example and replace model input files and change the black box configuration files. The basic ideas of the black box configuration are explained in the OpenDA course.
- Done