2014

Wrap Your Model!

Hauke Sonnenberg

Michael Rustler

Victor Philippon

Nicolas Caradot

Andreas Matzinger

11th International Conference on Hydroinformatics
HIC 2014, New York City, USA

WRAP YOUR MODEL!

HAUKE SONNENBERG (1*), MICHAEL RUSTLER (1), VICTOR PHILIPPON (1),
NICOLAS CARADOT (1), ANDREAS MATZINGER (1)
(1): Kompetenzzentrum Wasser Berlin, Cicerostr. 24, Berlin, 10709, Germany
**corresponding author, E-mail: hauke.sonnenberg@kompetenz-wasser.de*

In order to promote the usage of simple model engines this paper proposes to apply the concept of model wrapping that makes model software applications accessible from a programming environment. The paper shows exemplarily for one model software (WTAQ) how it has been wrapped. The concept has been successfully applied to two other very different kinds of model software (EPANET, GompitZ). The authors promote the wrapping of environmental models to make their use and exchange easier and more flexible.

INTRODUCTION - WHY MODEL WRAPPING?

Environmental models are provided in the form of software solutions at very different levels of complexity. In its simplest form a model software consists only of an executable file, the so called model engine, containing the "computational core" that reads the model configuration, and input data (such as initial or boundary conditions) from input files, does all the calculations and generates output files containing the model results.

More complex software solutions include a graphical user interface (GUI), which supports the import and export of data in different file formats as well as additional features such as the visualisation and analysis of model results. Regarding usability, the complex solutions are more user-friendly than the simple model engines. However, they have three main drawbacks compared to their simple counterparts. Firstly, the more complex software solutions often lack the possibility to automate multiple model runs which are required for sensitivity analysis or parameter estimation. Also, complex model systems are often difficult to couple. Secondly, data visualisation and analysis is mostly limited to the features that are offered by the software. Thirdly, many complex models are relatively expensive commercial software solutions, whereas the model engines are often available in the form of free software.

So, there are good reasons for using a simple model engine. These are also mostly fast in execution, since they do not contain much more than the computational core of the model. As most of the complex systems use exactly the same model engines internally that are provided as stand-alone programs, the user can trust that model results will not differ whether produced by the simple or by the complex model software. However, using a simple model engine is more complicated and less user-friendly than using complex software. The user needs to know how to prepare input files in the proper format, how to invoke the model engine in the correct way and how to read and interpret the result files. As these steps require special technical knowledge scientists may be discouraged in using models in this simple form.

As there is a great potential in using model engines instead of complex software systems this paper aims at simplifying the use of different existing model engines by using a technique called "model wrapping". The idea is to "wrap" the different steps such as reading and writing input files, calling the model and reading output files into functions of a well-established programming language. By doing so all the model-specific requirements are performed within the "wrapper" functions and the user just needs to call these functions without caring about the technical and formal details that happen internally.

Wrapped models overcome the drawbacks of complex software solutions and allow users:

- to automate the creation of input data from data bases, in scenario analysis or for model calibration,
- to couple different models and
- to perform taylor-made analysis and visualisation of output data.

In this paper wrapping of environmental model engines in the R environment (R Core Team [7]) is proposed. This general approach is explained under Materials and Methods, including model requirements for wrapping. The approach was tested for three very different environmental models. One exemplary application for the groundwater model WTAQ (Barlow and Moench [1]) is described in detail in the second section of the paper.

MATERIALS AND METHODS

General approach

Very different kinds of model software (differing in subject, solving algorithms, file formats, programming language, etc.) can be brought together in one environment if the following requirements are met: (i) model engines can be run from the command line, (ii) model engines read input files and write output files and (iii) a description of the input and output file format is available.

The wrapper functions act as bridges between the user and the model software. Towards the user-side they provide a well-defined interface that is easy to use in the environment that the wrapper is designed for. Towards the model-side the wrapper functions take care of all the model-specific requirements. They "know" how the model engine expects the input file to be formatted, how to invoke the model engine on the command line and how to read and interpret the output files generated by the model engine. The idea is to comply with all model-specific requirements automatically, so that the user can concentrate on the important tasks such as input data preparation and output data analysis. The advantage of a wrapped model increases with the number of wrapped models available for the same environment since e.g. model coupling becomes feasible and data analysis or plotting functions can be reused for different types of model outputs.

Following the terminology of Martin [5], the wrapping takes place at different "levels of abstraction" (Fig. 1):

- At the first and lowest level access to the input files, to the model engine and to the output files is given. Accordingly, this level provides functions for (i) writing input files (function writeInputFile in Fig. 1), (ii) running the model engine with a given input file (function runModelEngine) and (iii) reading output files (function readOutputFile).
- At the second level of abstraction a function is provided that performs a full sequence of these three steps (i) to (iii) (function runModel in Fig.1). It takes a model configuration as input argument (config), passes it on to writeInputFile, calls

runModelEngine and uses readOutputFile, which returns the model results in an appropriate data type (result) available in the programming environment.

- Finally the wrapper may comprise a third level of abstraction at which functions are provided (or written by the user) that performs a task for which multiple model runs are required. This includes functions performing a sensitivity analysis or a model calibration.

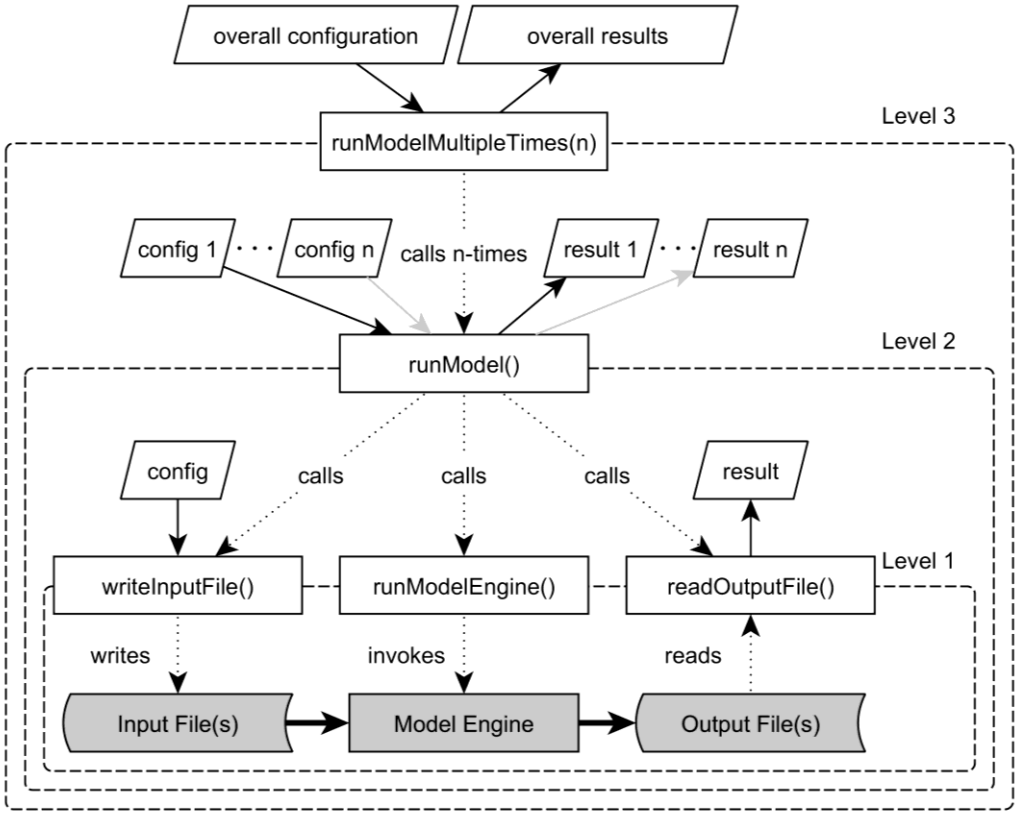

Figure 1. General data flow scheme of a simple model engine reading data from one or more input files and generating one or more result files (boxes shaded in grey). Wrapper functions at different levels of abstraction are indicated (see text).

R and RStudio as model wrapping environment

The popular programming language R (R Core Team [7]) and the Integrated Development Environment (IDE) RStudio [9] make an ideal environment for wrapping simple model software and for integrating wrapped models into one consistent environment.

R: A programming language and environment for data analysis and graphics

R is a programming language and environment for statistical computing and graphics. The software is free and available for different operating systems. The R environment includes data handling, data analysis and graphical facilities. The simple and effective programming language offers good input and output facilities such as reading and writing of text files and is able to call external programs from the command line shell of the operating system. R is highly extensible. Functionality is added in the form of user-defined functions which can be organised in so-called packages. Packages can be easily exchanged and installed into any R environment. They can

contain exemplary data and code demonstrating the usage of the functions in the package. This makes it easy to get started with the functionality that the package provides. Since all functions in an R package must be documented in a uniform way it is easy for the user to explore the purpose and data requirements of each function.

All these characteristics make R an appropriate environment for the "wrapping" of model engines. It allows reading, modifying and writing files with fewer commands than would be needed in other programming languages and it is able to invoke the executable model engines via the command line shell. For the documentation of the packages developed within this study the package "inlinedocs" (Inlinedocs development team [3]), available at the Comprehensive R Archive Network (CRAN, http://cran.r-project.org/), has been used. It is able to generate documentation files from specially formatted user-comments in the R source code files. The documentation of the wrapped model engines can be integrated into the documentation of the R-package so that the user is given background information on the model.

RStudio: An Integrated Development Environment (IDE) for R

RStudio [9] is an Integrated Development Environment (IDE) built just for R. It is free and open source, and works on common operating systems. It provides a source code editor supporting syntax highlighting and code completion (cf. Fig. 2) and allows executing R code directly from the source editor. RStudio integrates R help and documentation and contains a workspace browser and data viewer.

Within this study RStudio has been used as the environment for developing and testing the model wrapping R packages. It is also the environment recommended for using the developed packages since its code-completion and inline-documentation features (cf. Fig. 2) support the concept that lies behind the model wrapping idea: The user does not have to know the technical details of a model engine but is supplied with an easy-to-use, well documented interface.

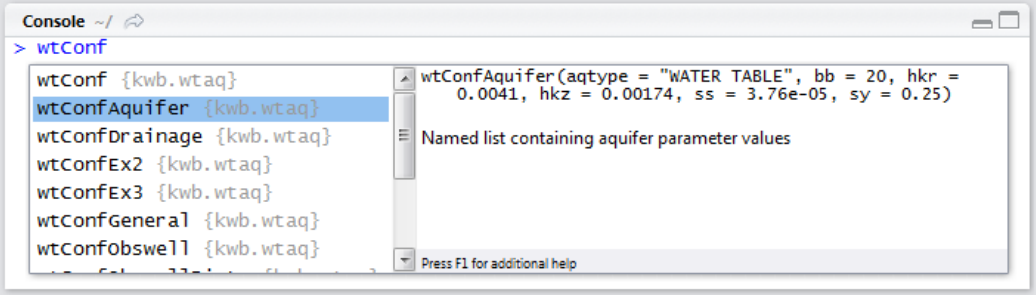

Figure 2. Code completion (left: list of function names matching the user input) and inline-documentation (right: list of arguments that the selected function [here: wtConfAquifer] accepts, together with a short description, both taken from R's help system) in RStudio.

EXAMPLE: WRAPPING OF THE WELL DRAWDOWN MODEL ENGINE WTAQ

The model wrapping approach described above has been applied successfully to three very different models: a groundwater drawdown model (WTAQ, see Barlow and Moench [1]), a water distribution network model (EPANET, see Rossman [8]) and a sewer deterioration model (GompitZ, see Le Gat [4]). In the following the application to WTAQ is shown exemplarily; i.e. how the model engine was wrapped into an R package that now allows running the model from within the R environment. The two other models have been wrapped in a similar manner.

The well drawdown model software WTAQ

WTAQ is an analytical well drawdown model able to predict the temporal evolution of the drawdown in one pumping well and a distinct number of observation wells for both confined and unconfined aquifers. The model engine is an open-source software application and provided for free by the United States Geological Survey (USGS, see Barlow and Moench [1]). The software does not provide a GUI but needs to be executed from the command line.

All input and output files are text files being formatted in a non-standard but well documented way. The input file contains information on aquifer and well characteristics as well as additional settings, e.g. related to the mathematical solver algorithm. The WTAQ model engine generates two types of text output files both of which contain the model results. One file (model.out) reads like a report. It contains a banner stating the owner and version of the software, and sections for the aquifer hydraulic properties, the program solution variables, and, finally, the characteristics and calculated drawdowns for the pumped well and for the observation wells, respectively. In the case that errors occurred during the model run (e.g. by an implausible set of input parameters) error messages are also written to this output file. The second output file (called plot file, model.plot) is intended to be used for plotting purposes. It contains the pure results in the form of one or more tables containing the time series of drawdowns for the different observation wells. The plot file appears in two different formats, depending on whether logarithmic or equidistant time steps are defined in the model parameters.

Wrapping of the well drawdown model software WTAQ

As described in general in the Materials and Methods section, the R package created to access the well drawdown model WTAQ provides functions at three different levels of abstraction.

Level 1-functions

WTAQ input files are text files in which each line contains a single or a group of model parameters (cf. Fig. 3, left). The wrapper provides a function that is able to read an existing input file and returns an appropriate R data structure (config in Fig. 3, right) being organised in sublists each of which contains model parameters of a certain group, e.g. general parameters in sublist "general", aquifer parameters in sublist "aquifer", etc. (Fig. 3, right). If parameters are changed in this data structure another wrapper function is able to convert them back into the file format required by WTAQ.

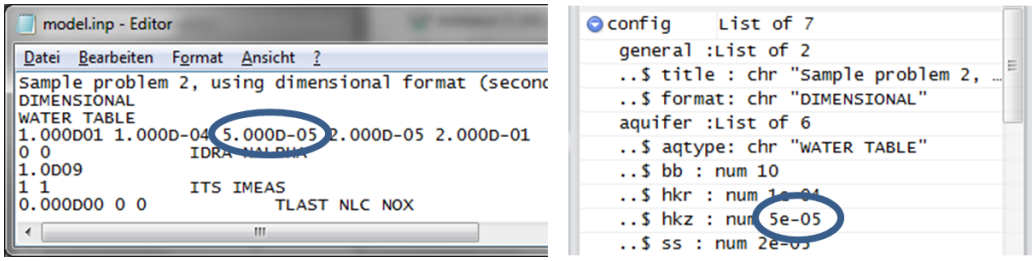

Figure 3. Cutouts of exemplary WTAQ input file (left) and the corresponding representation in an R list structure "config" (right, as shown in RStudio's data browser). The corresponding value "5.0 e-05" of model parameter "hkz" (vertical hydraulic conductivity of the aquifer) is indicated in both cases.

In order to support the user in generating a model configuration "from scratch", i.e. without having an existing input file, a set of functions has been implemented each of which accepts

model parameter values for a certain part of the configuration (e.g. related to the aquifer [wtConfAquifer], to the drainage [wtConfDrainage] or to general settings [wtConfGeneral], cf. Fig. 2, left).

If WTAQ is run in the usual way, the user is prompted to enter the paths of input and output files on the console. The corresponding wrapper function uses a trick in order to avoid this user interaction: it prepares a file ("arguments.txt") containing the paths of input and output files (model.inp, model.out, model.plot). The model engine is then invoked with an appropriate command line call instructing the model engine to read all required file paths from that file.

Due to the fact that the plot file appears in two different formats (see above), there are two different wrapper functions each of which is able to read one of the formats. Both return the contained data in the form of an R data frame.

Level 2-function

The level 2 function performing a complete model run (sample code in Fig. 4) accepts a WTAQ configuration in the form of an R list object (config in Fig. 4, cf. also Fig. 3, right) and returns the model results as an R data frame (line 8 in Fig. 4). The function considers that the plot file format differs depending on the model configuration. Consequently, it checks the configuration if logarithmic or linear time steps were calculated and calls the appropriate level 1 function (either readOutputFile_log or readOutpufFile_lin in Fig. 4) for reading the plot file. The function also performs an error check by parsing the output file for an error message and provides the user with an appropriate message (not shown in Fig. 4).

```
1  model.inp <- getPathToInputFile()
2  writeInputFile(config, model.inp)
3  model.out <- runModelEngine(model.inp)
4  if (logarithmicTimesteps(config))
5    result <- readOutputFile_log(model.out)
6  else
7    result <- readOutputFile_lin(model.out)
8  return (result)
```

Figure 4. Simplified code example of how the level 2 function (cf. runModel in Fig. 1) is implemented. The concept of temporal coupling by creating a "bucket brigade" (Martin [5]) was applied. The lines, in which a function of level 1 is called, are indicated with a bold dot to the left.

Level 3-function

Finally a wrapper function representing a third level of abstraction was defined. At this level multiple model runs are controlled. This level aims at considering well interference by calculating the superposition of drawdowns in a well field, in which at least two wells are pumping at the same time. Since WTAQ supports only one pumping well but many observation wells, the model engine is invoked as many times as there are wells pumping in parallel. In each run, WTAQ is requested to calculate the drawdowns caused by exactly one of the pumping wells. The results from the runs are collected to calculate the superposition of drawdowns. The final wrapper function takes a description of the well field (coordinates and geometry of wells), as well as the information on the pumping rates of the wells as input argument (referred to as "overall configuration" in Fig. 1) and returns the time series of superposed drawdowns for each well of the well field (referred to as "overall results" in Fig. 1).

The package contains the functions of the three levels described above. Furthermore plotting functions that visualise the drawdowns at the different wells and over time and that are

fed with the model results provided by a level 1 or level 2 functions have been implemented. The package also contains example data that make it easy to run a test simulation without the need to configure a model from scratch.

RESULTS AND DISCUSSION

In addition to WTAQ, wrappers have also been developed for the model engines EPANET (see Rossman [8]) and GompitZ (see Le Gat [4]).

The WTAQ-wrapper was tested in the framework of a sensitivity analysis of the WTAQ model. Therefore, a base configuration being read from an existing input file was modified and run within a loop. This test identified the most important model parameters of WTAQ. The functions also provide the means to auto-calibrate a WTAQ model.

Using the functions of the EPANET-wrapper a calibration of an existing EPANET model identifying the best assumption for an unknown pipe diameter was performed. The functions provided in the package were also used to automatically create and run different model scenarios in which theoretical pump curves were exchanged with real pump curves. Using the GompitZ-package a Monte Carlo analysis investigating the influence of different subsets of sewer inspection data for model calibration on the prediction of sewer pipe conditions was performed.

With the wrapper functions provided in the packages it is possible to use the models without having to know the formal structure of input or output files and without having to cope with the technical details of the communication with the model engines. For example, the WTAQ wrapper automatically adapts cross-dependent input parameters correctly in case one is changed by the user. This assures the formal correctness of the input file and minimises the effort for the user, who normally has to consider all cross-dependencies for each input file modification manually by consulting the model documentation. Consequently the focus can be shifted on retrieving and preparing the data needed by the models. The coding effort is highly reduced and less error-prone compared to “manual” input and output file handling and communication with the model engine.

Thanks to choosing R as environment the effective methods that R provides for data preparation and analysis can be used. Even if it may take a high amount of time to develop the interfaces and to write the wrapper functions it is worth the time, particularly if model engines are planned to be used more than once, in different contexts and by different users.

Having wrapped some models it was realised that the tasks to be performed are very similar each time. The general procedures were reused, e.g. how to extract sections of input/output files, how to invoke a program from the command line or how to call system functions, such as for creating temporary directories.

CONCLUSIONS AND OUTLOOK

The principle of model wrapping has been introduced and shown for one example. It can be applied to many other model engines meeting the requirements stated in this paper.

Wrapping simple model engines provides the advantages of:

- making models accessible from a programming environment,
- facilitating multiple model runs (as e.g. required for calibration, sensitivity analysis),
- enable model coupling,
- extending model software by features that the programming environment provides.

Using R as environment for model wrapping provides the advantages of:

- using a convenient environment that is widely used in science and easy to start with,

- using the strengths of R in data preparation, visualisation and statistical analysis,
- using R's documentation standard to document the usage of wrapped models,
- facilitating and promoting the exchange of models between users.

Not only models but also data sources can be "wrapped", e.g. databases that are designed according to a standardised schema, such as the CUAHSI Community Observations Data Model (ODM, see Tarboton D. G. *et al.* [10]). An R interface to an ODM-compliant database is currently being developed in the framework of another KWB project. The more "wrappers" are available the easier and more flexible the work on (multiple) environmental models becomes. Interested people are invited to use and revise the packages described in this paper and are encouraged to wrap other models in similar packages.

ACKNOWLEDGEMENTS

The presented work was carried out in the frame of the KWB research projects OPTIWELLS-2 (Philippon *et al.* [6]) and SEMA (Caradot *et al.* [2]) both of which are funded by Veolia Water.

The authors thank the developers of model engines who provide their software for free. Special thanks go to the maintainers of R and RStudio who permanently develop their software further and who offer their products for free. Also thanks to all the people who share their R packages.

REFERENCES

[1] P. M. Barlow and A. F. Moench, *"WTAQ Version 2 - A computer program for analysis of aquifer tests in confined and water-table aquifers with alternative representations of drainage from the unsaturated zone"*, U.S. Geological Survey Techniques and Methods 3-B9, (2011), 41 p.

[2] N. Caradot, H. Sonnenberg, I. Kropp, T. Schmidt, A. Ringe, S. Denhez, M. Timm, A. Hartmann and P. Rouault, *"What is the reliability of sewer deterioration models?"*, *IWA World Water Congress*, Lisbon, Portugal, (2014), accepted.

[3] Inlinedocs development team, *"inlinedocs: Convert inline comments to documentation"*, R package version 1.9.2, http://CRAN.R-project.org/package=inlinedocs, (2013).

[4] Y. Le Gat, *"Modelling the deterioration process of drainage pipelines"*, Urban Water Journal, Vol. 5, No. 2, (2008), pp 97-106.

[5] R. C. Martin, *"Clean Code: A Handbook of Agile Software Craftsmanship"*, Prentice Hall PTR, (2008).

[6] V. Philippon, A. Sainz-Garcia, H. Sonnenberg, M. Alary, K. Böhm, G. Grützmacher and M. Rustler, *"A tool for minimizing the energy demand of drinking water well fields"*, *IWA Water, Energy and Climate Conference* Mexico City, Mexico, (2014), accepted.

[7] R Core Team, *"R: A Language and Environment for Statistical Computing"*, R Foundation for Statistical Computing, Vienna, Austria, http://www.R-project.org/, (2014).

[8] L. A. Rossman, *"EPANET 2 - User Manual"*, U.S. Environmental Protection Agency, National Risk Management Research Laboratory, Office Of Research And Development, Cincinnati, OH, (2000).

[9] RStudio, *"RStudio: Integrated development environment for R"*, http://www.rstudio.org/, (2014).

[10] Tarboton D. G., Horsburgh J. S. and Maidment D. R., *"CUAHSI Community Observations Data Model (ODM) Version 1.1 Design Specifications"*, Consortium of Universities for the Advancement of Hydrologic Science, Inc., http://wdc.cuahsi.org/wdc/Docs/ODM1.1DesignSpecifications.pdf, (2008).