# Tornado: A versatile and efficient modelling & virtual experimentation kernel for water quality systems

F. Claeys [a], D.J.W. De Pauw[a], L. Benedetti[a], I. Nopens[a] and P.A. Vanrolleghem[a b]

[a]BIOMATH, Department of Applied Mathematics, Biometrics and Process Control
Ghent University, Coupure Links 653, 9000 Gent, Belgium
e-mail: filip.claeys@biomath.ugent.be

[b]model*EAU*, Département de Génie Civil
Pavillon Pouliot, Université Laval, Québec, G1K 7P4, QC, Canada

**Abstract:** Recently, a new kernel for modelling and virtual experimentation (*i.e.* any evaluation of a model) in the domain of water quality management was developed. This kernel was named "Tornado" and will be included in the new generation of the WEST product family, as well as in several other products (*e.g.* DHI's MOUSE-TRAP) and projects. Most important issues during development were versatility and efficiency.
This paper focuses mainly on the rationale for the new development, and the major features of the resulting kernel. It is argued that classical approaches (such as the adoption of MATLAB/SIMULINK, custom FORTRAN codes and/or domain-specific simulators) all have specific disadvantages, hence the need for a kernel that offers a compromise between versatility and efficiency.
Tornado was developed in C++ using advanced language features, yielding a code base that offers fast execution, portability and increased readability. The software is composed of distinct environments for modelling and virtual experimentation. The modelling environment allows for the specification of complex ODE and DAE models in object-oriented, declarative languages such as MSL and Modelica. A model compiler translates these high-level models into efficient, flattened code. The experimentation environment allows for running atomic virtual experiments (such as simulations and steady-state analyses) as well as compound experiments (optimizations, scenario analyses, etc.) on the basis of flattened models.
Important Tornado features are the fact that new virtual experiment types and numerical solvers can easily be added and loaded dynamically. Further, Tornado is available for several platforms and can be deployed in multiple ways: it can be used as a numerical back-end for graphical, command-line and web-based applications, and can be integrated in cluster and grid computing infrastructures. Several types of API's are currently provided: C, C++, .NET and MEX. The persistency layer is XML-centric.

**Keywords:** Modelling; Virtual experimentation; Software kernels; Distributed execution

## 1 INTRODUCTION

The water quality domain is typically subdivided into river systems, sewer systems and wastewater treatment plants (WWTP). For each of these sub-domains, several mathematical models have evolved over time, some of which have received a formal or *de facto* standardization status. For WWTP for instance, the ASM (Activated Sludge Model) series has been standardized by the International Water Association since 1987 (Henze et al. [2000]). Also, for each sub-domain various software tools have been developed. Often these tools are targeted towards one single model or towards a set of related models, in only few cases are they more generic in nature.

Since a number of years, the desire for integrated modelling in the water quality domain is growing, *i.e.* one wants to be able to build comprehensive models describing entire water quality processes, across the historical sub-domain boundaries. Also, knowledge of systems is constantly growing and models are becoming increasingly detailed. As a result, a need for powerful software infrastructures exists.

Water quality models typically consist of large

sets of non-linear Ordinary Differential Equations (ODE) and/or Differential-Algebraic Equations (DAE). These equations are mostly well-behaved, although discontinuities occur regularly. The complexity of water quality models is therefore not in the nature of the equations, but in the sheer number. In WWTP, smaller models such as the well-known Benchmark Simulation Model (BSM) (Copp [2002]) consist of approximately 150 derived variables. Larger systems have up to 1,000 derived variables and over 10,000 (partly coupled) parameters.

## 2 RATIONALE FOR THE DEVELOPMENT OF TORNADO

The work on software tools in the area of water quality that is being conducted at BIOMATH (in collaboration with HEMMIS NV, Kortrijk, Belgium) is mainly driven by three goals: allow for complex modelling and virtual experimentation, allow for multiple ways of deployment (*i.e.* enable tools to be used from within stand-alone, web-based, distributed and embedded applications, through various types of graphical and textual interfaces), and ultimately allow for intelligent model-based decision support.

Historically, a number of approaches have been followed when developing software for water quality modelling:

As in many other domains, custom (FORTRAN or C) codes were popular in the past and still are with certain groups. Main advantage in this case is of course efficiency (*i.e.* simulation speed). Drawbacks on the other hand are numerous, and include low readability, maintainability, reusability and extensibility. Most often there is no clear separation between model and execution environment.

Another well-known approach is to use MATLAB M-files for modelling purposes. Readability, maintainability, reusability and extensibility are evidently much better here than in the case of custom models, but this is at the expense of efficiency. For applications in the water quality domain (other than prototyping) efficiency is insufficient, as is stated in the COST Simulation Benchmark report (Copp [2002]).

To a large extent, efficiency problems can be overcome through the use of SIMULINK causal block diagrams. However, in order to reach a level of performance that is convenient enough for most applications, new SIMULINK S-functions must be implemented using large chunks of procedural C or FORTRAN code (Copp [2002]). This again is detrimental to readability, maintainability, reusability and extensibility.

Finally, several specialized applications for water quality modelling and simulation have entered the market in the past. These applications can present good or acceptable scores in terms of performance, readability and maintainability. However, they are often closed environments and lack reusability and extensibility potential. For WWTP, the most well-known tools are BioWin, EFOR, GPS-X, SIMBA, STOAT and WEST (Copp [2002]).

In view of the disadvantages of the above approaches, an attempt was made to design and develop a software framework that offers a compromise between flexibility, versatility and efficiency.

## 3 DESIGN OF THE SOFTWARE

In general, the Tornado software system that was built can be described as a unified, flexible and portable kernel for modelling and virtual experimentation. The term “unified” refers to the fact that ideas from a number of former kernels developed at BIOMATH were re-used and merged into one new kernel. The new kernel is “flexible” since it can be easily extended in several ways, be applied to a wide variety of problems and be put to work under various circumstances. It is also “portable” since it is available for win32 and linux, and potentially also other platforms.

One of the major principles of the kernel is that it consists of strictly separated environments for modelling and virtual experimentation. Both model building and virtual experimentation are regarded as hierarchical in nature. Building models in a hierarchical fashion (either top-down or bottom-up) is evidently not a novel approach. However, looking at virtual experiments in a hierarchical way is less evident. In the case of modelling we consider the top-level model to be a coupled model, where each of the sub-models is either atomic, or a coupled model in its own right. In the case of virtual experimentation, we can distinguish between atomic experiments (which cannot be further decomposed) and compound experiments (which are made up of a number of atomic and/or compound sub-experiments).

Some other principles that have guided the design

and development of the kernel are the following:

- All development was to be done in one-and-only-one advanced, object-oriented language, to allow for a homogeneous code base and easy detail debugging. To this end, C++ was chosen since it still offers a good compromise between efficiency and advanced features such as high-level data constructs (maps, vectors, ...), abstract interfaces, smart pointers, exception handling, namespaces, ...
- As few as possible commercial third-party components were to be used. In fact, the only such component that has been retained after careful consideration is Elcel Technology's OpenTop (http://www.elcel.com), which is a general purpose Java-like library and is used in Tornado for threading and XML parsing.
- The design should be clean and based on modern software design patterns (Gamma et al. [2003]).
- All code is to be thread-safe at the level of virtual experiments.
- Wide-character strings should be supported at all levels to allow for internationalization.
- Platform-dependent code and general convenience routines should be hidden beneath an abstraction layer.
- Focus should be on clarity and performance.
- UML (Unified Modelling Language) diagrams should be made available for all relevant entities.
- A one-to-one mapping should exist between the structure laid out in the UML diagrams and the persistent representation of entities.

The design of the Tornado framework was based on the three-tier principle, *i.e.* a strict separation was made between application layer, persistency layer, and business logic layer (further subdivided into business process and business objects sub-layers). This approach is essential to allow for extensibility and multiple ways of deployment.

In principle, XML (Extensible Markup Language) is to be used for all information that is to be made persistent in Tornado, except in case this is impossible for human readability or efficiency. In particular, two types of information are not represented as XML: high-level models (readability) and executable model equations (efficiency).

## 4 FUNCTIONALITY

### 4.1 Modelling environment

The main elements of the Tornado modelling environment are **model compiler** and **model builder**. The model compiler converts models described in a high-level modelling language to flattened, executable model code. The model builder then compiles and links the executable model code into a binary object that can be dynamically loaded into the experimentation environment (Figure 1). At the

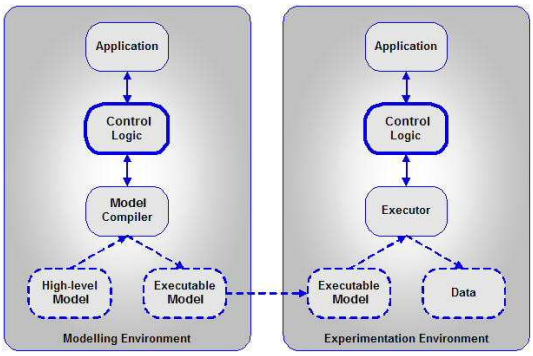


**Figure 1.** Tornado conceptual diagram

moment Tornado has full support for the MSL language (Vanhooren et al. [2003]) and partial support for Modelica (http://www.modelica.org). Both languages are equation-based, declarative (*i.e.* non-procedural), object-oriented, and allow for the construction of new models through inheritance and composition. Modelica also has the added advantage that it allows for non-causal modelling, *i.e.* it allows for equations to be automatically rewritten, so that different variables can be computed from them.

In general, the model compilation process consists of the following phases: flattening of inheritance and decomposition hierarchies, causality assignment, sorting of equations, detection of sets of algebraic equations, and optimization of the generated code (Kops et al. [1999]). During the latter phase, various types of optimizations can be performed, some of which may have a major effect on the size and speed of the generated code. Noteworthy optimizations are canonization of expressions, removal of aliases and splitting of the equation set into sets that have to be computed once at the beginning/end of the simulation process, and sets that should be computed at each (output) timepoint.

The executable model generated by the model compiler consists of two parts: the actual equations (C

code), and a description of the original (unflattened) model structure (XML). The latter includes meta-information on sub-models, parameters and variables.

The model builder internally relies on a plain C compiler. Several compilers are currently supported: Microsoft Visual C++ 6.0 (win32), Microsoft Visual C++ 7.1 (win32), Borland C++ 5.5 (win32), LCC (win32), INTEL C++ 9.0 (win32 & linux) and g++ (linux). Tests have shown that the Microsoft Visual C++ 7.1 and INTEL C++ 9.0 compilers usually yield the best run-time performance, as is illustrated in Table 1 on the basis of the *Galindo CL* WWTP model (Ayesa et al. [2006]).

**Table 1.** Performance of C compilers for *Galindo CL*

| Compiler | Run (s) | Compile (s) | Model size (KByte) |
|---|---|---|---|
| BC++ 5.5 | 752 | 1 | 352 |
| LCC | 689 | 2 | 291 |
| MSVC++ 6.0 | 561 | 32 | 232 |
| MSVC++ 7.1 | 503 | 42 | 228 |
| INTEL C++ 9.0 | 501 | 58 | 226 |

### 4.2 Experimentation environment

The Tornado experimentation environment consists of a main *singleton* object that acts as a *factory* for virtual experiments. The following virtual experiment types have so far been implemented:

- **ExpSimul**: Dynamic simulation of ODE or DAE systems.
- **ExpSS**: Discovery of the steady-state of a system using root finding algorithms.
- **ExpOptim**: Minimization of a weighted sum of objectives, computed on the basis of simulated trajectories.
- **ExpCI**: Generation of confidence information for an optimum found by ExpOptim.
- **ExpScen**: Evaluation of a set of scenarios through the generation of ExpSimul jobs with different parameters and/or initial conditions. The values for parameters and initial conditions are either spaced explicitly or sampled from a number of distributions.
- **ExpMC**: Evaluation of a reduced set of scenarios by applying Monte Carlo and Latin Hypercube Sampling (LHS) techniques.
- **ExpSens**: Local sensitivity analysis based on the computation of differences between a reference simulation and perturbed simulations.

Each of the above experiment types relies on one or more numerical solvers. These solvers are binary objects that can be loaded dynamically. A hierarchy of solver types has been created consisting of root solvers, integration solvers, optimization solvers, confidence information solvers, scenario analysis solvers, and Monte Carlo analysis solvers. It must be admitted that in some cases interpreting the engine that drives a certain experiment type as a "solver" is somewhat artificial. However, it was beneficial for establishing a clear design for the framework. Most solvers that are available for Tornado have been taken from well-known free numerical libraries such as LAPACK, DASSL, MINPACK, ODEPACK and RANLIB (http://www.netlib.org), and SUNDIALS (http://www.llnl.gov/CASC/sundials).

The fact that solvers are dynamically loadable binary objects makes it very easy for new solvers to be added to the system. Another factor that is beneficial in this respect is the fact that solvers, as well as other entity types within the Tornado framework, contain a set of so-called "properties". These are self-describing attribute-value pairs that can be queried at run-time. In this way, no specific interfaces have to be developed for each entity that is added or modified. Property structures contain the following information: name, description, value, type, default value and range of validity.

Each experiment type relies on certain input data and will eventually generate output data. In Tornado, a generic I/O framework has been set up that is based on **input providers** and **output acceptors**. Several types of input providers are supported, including input files, internal data buffers, input generator algorithms (such as pulse, sine, ramp, ...) and custom external providers. The output acceptors that are supported are output files, internal data buffers, and custom external acceptors such as plots. In many cases, several forms of interpolation can be used (zero-hold, linear, Lagrange). Multiple providers and acceptors can be used in one experiment.

In the design and implementation of experiment types, orthogonality has been favored to a large extent. For instance, ExpOptim, ExpScen and ExpMC support the same types of objectives, which are all computed from the simulated trajectory of a certain variable. These objective types include: Min (min-

imum value), Max (maximum value), Avg (time-weighted average value), StdDev (standard deviation), Int (integral), EndValue (value at end of trajectory), ValueOnTime (value at a specific point of the trajectory), DiffSum (sum of differences at different timepoints between reference trajectory and simulated trajectory), DiffMax (maximum difference between reference trajectory and simulated trajectory).

Table 2 illustrates the performance gain that has been obtained in Tornado with respect to some other tools. The comparison is done on the basis of the *BSM1 OL* model (Copp [2002]) that was implemented in a FORTRAN stand-alone program as well as in MATLAB/SIMULINK and WEST v3. Since not every tool offers the same solvers, entirely identical settings could not be chosen.

**Table 2.** Simulation times for *BSM1 OL*

| Software | Solver | Time (s) |
|---|---|---|
| FORTRAN | RK5, Step=5e-3 | 65 |
| MATLAB/SIMULINK | ODE45, Step=1e-4 | 72 |
| WEST v3 | RK4, Step=1e-4 | 70 |
| Tornado | RK4, Step=1e-4 | 35 |

### 4.3 Interfaces

Since one of the major goals of Tornado is to support multiple ways of deployment, it is of the utmost importance to have a wide variety of interfaces available. The following interfaces are therefore foreseen at this point:

- **C++**: As Tornado was entirely developed in this language, it comes with a C++ interface by default.
- **C**: To access Tornado from Rapid Application Development (RAD) languages such as Delphi or from within scripting languages such as Tcl, Perl or Python, the availability of a C interface is a necessary condition.
- **.NET**: A .NET interface allows for integration of Tornado in a plethora of applications and languages such as VisualBasic.NET or C#. It also allows for building web-based applications through the ASP.NET framework.
- **MEX**: MATLAB is a major factor in scientific computing. Interfaceability with MATLAB is therefore essential. Tornado has been wrapped by a MEX interface that allows for MATLAB to call the Tornado engine and to pass data back and forth.
- **OpenMI**: In the scope of the European Harmon-IT project, a specification for a mechanism that allows for the linkage of model engines has recently been established. The specification is based on .NET and describes a formal way of linking data containers of one model engine to another. Since this open modelling interface (OpenMI, cf. http://www.openmi.org) was specifically intended for use within the world of hydroinformatics, Tornado will also be made available as an OpenMI compliant component.

### 4.4 Distributed execution

Since most problems that are being tackled with Tornado are computationally complex, it is important to provide a means for distributed execution. From the onset, it was decided only to support coarse-grained gridification (*i.e.* gridification of compound experiments). Fine-grained gridification would entail splitting up atomic experiments (such as simulations) into constituents, which is believed to cause severe overhead.

The ultimate goal with regard to the use of distributed execution is transparency, *i.e.* it should in the end become possible to use several execution environments through exactly the same set of user manipulations. At the moment Tornado supports two such environments in a semi-automated manner: Typhoon (formerly known as WDVE, cf. Claeys et al. [2006]) and LCG-2 (http://public.eu-egee.org). In order to allow for transparent distributed execution, a generic XML-based job description format has been introduced. In Typhoon, this job description is directly interpreted by the job dispatcher. In LCG-2, generic job descriptions are automatically converted to the format required by LCG-2. Noteworthy is the fact that Tornado and Typhoon support the notion of a "set of jobs" whereas LCG-2 does not. This means that one XML file that describes a set of jobs will ultimately be translated to several individual job description files for LCG-2.

As an illustration of the performance gained by the introduction of distributed execution, one can consider the scenario analysis experiment for the *Marselisborg* WWTP (DK). This experiment consists of 1,084 simulations for which each simulation used to take 30 min with pre-Tornado kernels. In Tornado, individual simulation times could be reduced to 6 min and through gridification onto a 40 node LCG-2 infrastructure, the total execution time for the entire scenario analysis experiment could

be reduced to a mere 3.5 hours, as shown in Table 3. Another example, resulting from a LHS-based methodology based on Tornado, can be found in Benedetti et al. [2006].

**Table 3.** Execution times for *Marselisborg*

| Software | 1 run | 1,084 runs |
|---|---|---|
| pre-Tornado | 30 min | 23 days |
| Tornado | 6 min | 4.5 days |
| Tornado + LCG-2 | 6 min | 3.5 hours |

## 5 Applications

It is our belief that because of its flexibility and versatility, Tornado is suitable for inclusion in a wide range of applications. At the moment, four applications already make use of the Tornado kernel:

- **WEST** is a specialized WWTP tool that is commercialized by HEMMIS NV (Kortrijk, Belgium) (Vanhooren et al. [2003]). It has an attractive user-interface and will use Tornado through its .NET interface as of the next major product release.
- **EAST** is a research application for modelling and virtual experimentation that is being developed by BIOMATH. It is available for several platforms and uses Tornado through its native C++ interface.
- DHI's **MOUSE-TRAP** is a well-known commercial tool for modelling water quality in sewers (http://www.dhisoftware.com/mouse). As of its next version it will use Tornado for back-end computations through Tornado's C interface.
- Finally, the Tornado distribution itself contains a suite of **CUI tools**, convenient for testing the Tornado kernel as well as for batch-oriented processing. The CUI suite uses Tornado's native C++ interface.

Figure 2 gives an overview of the various applications and interfaces that have been constructed on top of Tornado.

## 6 Conclusion

A new kernel for modelling and virtual experimentation in the water quality domain was developed. It allows for high-level modelling and efficient execution of virtual experiments. Multiple ways of deployment and extension of the kernel are available.

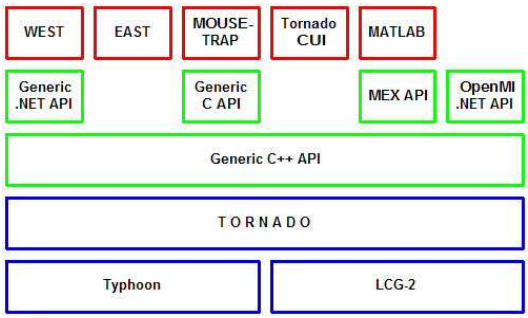


**Figure 2.** Tornado-based applications & interfaces

## References


Ayesa, E., A. De la Sota, P. Grau, J. Sagarna, A. Salterain, and J. Suescun. Supervisory control strategies for the new WWTP of Galindo-Bilbao: The long run from the conceptual design to the full-scale experimental validation. In *The 2nd IWA Conference on Instrumentation, Control and Automation*, Busan, Korea, 2006.

Benedetti, L., D. Bixio, F. Claeys, and P. Vanrolleghem. A model-based methodology for benefit/cost/risk analysis of wastewater systems. In *Proceedings of the iEMSs 2006*, Burlington, VT, 2006.

Claeys, F., M. Chtepen, L. Benedetti, B. Dhoedt, and P. Vanrolleghem. Distributed virtual experiments in water quality management. *Water Science and Technology*, 53(1):297–305, 2006.

Copp, J. *The COST simulation benchmark*. European Commission, 2002.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison-Wesley, 2003.

Henze, M., W. Gujer, T. Mino, and M. van Loosdrecht. *Activated Sludge Models ASM1, ASM2, ASM2d, and ASM3*. IWA Task Group on Mathematical Modelling for Design and Operation of Biological Wastewater Treatment, 2000.

Kops, S., H. Vangheluwe, F. Claeys, and P. Vanrolleghem. The process of model building and simulation of ill-defined systems: Application to wastewater treatment. *Mathematical and Computer Modelling of Dynamical Systems*, 5(4): 298–312, 1999.

Vanhooren, H., J. Meirlaen, Y. Amerlinck, F. Claeys, H. Vangheluwe, and P. Vanrolleghem. WEST: modelling biological wastewater treatment. *Journal of Hydroinformatics*, 5(1):27–50, 2003.