Filip CLAEYS

A Generic Software Framework for Modelling and Virtual Experimentation with Complex Environmental Systems

Thesis submitted in fulfillment of the requirements
for the degree of Doctor (PhD) in Applied Biological Sciences:
Environmental Technology

G
Summary

Computer-based modelling and virtual experimentation (*i.e.*, any procedure in which the evaluation of a model is required, such as simulation and optimization) has proven to be a powerful mechanism for solving problems in many areas, including environmental science. Since the late 1960's, a *plethora* of software systems have been developed that allow for modelling, simulation, and to a lesser extent, also other types of virtual experimentation. However, given the persistent desire to model more complex systems, and the trend towards more complex computational procedures based on model evaluation, it may be required to re-evaluate and improve existing software frameworks, or even to suggest new frameworks. Moreover, recent developments in information technology have caused a growing trend towards flexible deployment and integration of software components, *e.g.*, in a web-based, distributed or embedded context. In order to be able to handle the need for malleable deployment and integration of modelling and virtual experimentation systems with other software components, re-evaluation of existing frameworks and/or development of new frameworks may also be required.

One particular domain of environmental science that has in recent years attracted the interest of researchers in the field of software frameworks, is water quality management (which, amongst other, includes the study of water quality in treatment plants, sewer networks and river systems). This domain can be considered mature since the inception of standard unit process models such as the Activated Sludge Model (ASM) series. Moreover, the complexity of the most advanced integrated models that are currently studied in this domain is such that mainstream software systems simply cannot handle them anymore.

The work that is presented in this dissertation concerns the design and implementation of an advanced framework for modelling and virtual experimentation with complex environmental systems, which attempts to accomplish the following goals: deliver support for complex modelling and complex virtual experimentation, offer a wide variety of deployment and integration options, and ensure compliance with modern architectural standards. As most systems that are studied in environmental science are continuous, focus is on complex continuous system modelling.

The framework that was developed is generic in nature, although the design and implementation process has been strongly guided by demands coming from the field of water quality modelling. Development was done over a longer period of time, *i.e.*, mainly from 1995 until 1998 and from 2003 until 2007. The WEST modelling and virtual experimentation product that is currently commercially available (and is mainly used for modelling and simulation of wastewater treatment plants) is an emanation of the work that was done during the first period. The work done during the second period, has resulted into a

software framework named “Tornado” and should eventually find its way to WEST-4, which is a major rewrite of the former WEST-3 product.

Tornado implements a hierarchical virtual experimentation framework in which new virtual experiments can be created by re-using already existing experiments. The composition graph that represents the virtual experiments that have so far been developed consists of 15 different types of experiments. Using these experiments, a broad range of problems can be solved, ranging from simple simulation studies to large-scale risk analyses using comprehensive Monte Carlo simulation, such as recently performed in the scope of the CD4WC project dealing with support for the EU Water Framework Directive. All experiment types are highly configurable through an extensive set of properties, which can be dynamically queried and modified. New experiment types can be added by computer scientists or environmental scientists, provided that the design and implementation principles of Tornado are well-understood.

Most virtual experiment types in Tornado are guided by numerical solver algorithms. For the incorporation of these solvers into the kernel, a generalized framework for abstraction and dynamic loading of solver plug-ins was devised. This framework allows for solvers to be classified according to their purpose (integration, optimization, *etc.*) and for new solvers to be developed on the basis of a library of base classes. At run-time, solvers can be dynamically loaded or removed from the system for reducing memory consumption.

The flexibility and orthogonal design of Tornado is well illustrated by its ability to perform solver setting scenario analysis, which is a method that allows for applying variable sweeping techniques to solver settings (such as integrator tolerances), instead of to model parameters. Using this technique, overall execution times of large-scale compound virtual experiments can be shortened by applying improved solver settings that are discovered through *a priori* exploration of the solver setting space.

In the scope of Tornado, two declarative, hierarchical, object-oriented, equation-based modelling languages play an important role: MSL and Modelica. MSL has been available as of the first Tornado version and has been the cornerstone of model development in the scope of many water quality modelling studies. Since MSL does not support the concepts of acausal and multi-domain modelling, the Modelica language was introduced in a later version of the framework. Albeit very powerful, Modelica is also a complex language. Consequently, a hybrid approach is adopted in which the standard OpenModelica model compiler front-end is used in combination with a custom model compiler back-end for Tornado. This model compiler back-end allows for the generation of executable models that can be used by the Tornado virtual experimentation kernel, as well as for the generation of S-functions that can be adopted in the scope of MATLAB/Simulink.

Since executable models need to be fast to generate, the Tornado executable model format was designed in such a way that meta-information is represented in XML and computational information is described in a compiled general-purpose programming language, *i.e.*, C. In this way, executable models can quickly be produced, using the user’s C compiler of choice.

In order to ensure the run-time performance of executable models, three techniques were implemented that reduce the model complexity and hence provide a considerable performance improvement of 10 to 30%. These techniques are known as constant folding, equiv substitution, and lifting of equations. The first computes the results of constant sub-expressions at compile-time, the second removes aliases from a set of equations, and the third allows for detecting initial and output equations and for moving them to appropriate equation sections.

As a result of the manipulations that are performed on models during the executable code generation process, the output of this process is often not recognizable anymore to a user. Consequently, stability is an important issue: in case a problem does occur, it must be reported in a way that is understandable to a user. Two techniques were implemented that allow for improved stability and error reporting in executable models: code instrumentation and the generation of bounds checking code. The first replaces potentially dangerous constructs (*e.g.*, division-by-zero) by “safe” versions in which arguments are first checked on validity before being applied. The second generates code that performs range checking at run-time (*e.g.*, to detect negative values). By generating this code only for variables that need it, a performance improvement of up to 50% could be realized with respect to kernel-based range checking. In general, a situation is created in which run-time performance of safe, optimized Tornado models is comparable to the performance of unsafe, non-optimized models. Hence, the time spent on performing safety checks is compensated by the time gained through optimization.

In order to allow for the integration of Tornado in other software, several application programming interfaces (API's) were developed. Some of these are comprehensive and allow for any Tornado-related operation to be performed, while others are restricted and only allow for the most frequently occurring operations. Comprehensive API's were developed for C++ and .NET. Restricted API's were developed for C, the Java Native Interface, MATLAB MEX, OpenMI and CLIPS. A special case is the CMSLU interface that allows for the Tornado kernel to be called to run any type of experiment from within a Tornado model.

Several stand-alone applications were developed on the basis of the Tornado kernel. One of these applications is a suite of command-line programs that is developed along with the Tornado kernel itself and is useful for testing, automation and advanced use. Next to the command-line suite, a number of graphical applications were developed: WEST++, WEST-3 and EAST are based on the initial version of Tornado, while MORE and WEST-4 are based on the most recent version. These graphical applications illustrate that for application development, various technologies can be applied to one and the same Tornado kernel.

With respect to remote and web-based use, a number of technologies were discussed that are potentially applicable to Tornado. Some of these technologies have already been used in the scope of applications (sockets, DCOM and SOAP), while others have merely been adopted in prototype projects (.NET Remoting, ASP.NET). Still some others have not yet been applied, but could in case a need for this would arise (CORBA, OPC, Java RMI).

Finally, coarse-grained gridification was applied to Tornado at the level of sub-experiments (*i.e.*, a set of simulation jobs is carried out concurrently on a pool of computational nodes). As a result, Tornado can be deployed in a distributed fashion on the basis of the Typhoon cluster software (which was developed especially to serve as a light-weight solution for the execution of jobs generated by Tornado), as well as on the basis of the gLite grid middleware. Through the application of these distributed technologies, the extreme computational workload (14,400 simulations each requiring 30' of computation time) that was a result of the CD4WC project could be processed in a timely fashion (approximately 10 days).

Tornado was developed from scratch using object-oriented design and implementation principles. C++ was used as a programming language and design patterns such as the singleton, factory, facade and proxy patterns were adopted. Platform-independence of the kernel (but not necessarily its external interfaces) was ensured and thread-safety for high-level entities was provided. For these high-level entities, XML-based persistent formats were devised and described on the basis of XML Schema Definitions. Furthermore, a mechanism that provides for run-time querying of entity properties was provided, which alleviates the need for modification of entity interfaces in case of changes to the set of properties supported by an entity.

Overall, it can be stated that through the design and development of the Tornado framework, the solution of water quality management problems that were hitherto hindered by performance limitations or limitations to the degree to which complexity could be handled, has now become possible. Moreover, thanks to the design of the framework, it is expected that Tornado will be able to adapt to the expected continued increase in complexity for a considerable period of time. In order to render the Tornado framework more complete, a need for future work and the exploration of additional research areas exists. In some cases, this additional research has already been initiated.