International Environmental Modelling and Software Society (iEMSs)
2010 International Congress on Environmental Modelling and Software
Modelling for Environment's Sake, Fifth Biennial Meeting, Ottawa, Canada
David A. Swayne, Wanhong Yang, A. A. Voinov, A. Rizzoli, T. Filatova (Eds.)
http://www.iemss.org/iemss2010/index.php?n=Main.Proceedings

Linking Data, Models and Tools: An Overview

H.R.A. (Bert) Jagers [a]

[a]*Deltares, PO Box 177, 2600 MH, Delft, The Netherlands (bert.jagers@deltares.nl)*

Abstract: Complex questions trigger researchers around the world to link data, numerical models and tools of different origins together for integrated modeling of the environment and related socio-economic fields. Different researchers have, however, chosen different ways to link such resources. As a result a seemingly wide range of interfaces and frameworks have been defined: some have used low level interfaces others more abstract and object oriented ones; some systems may require little or no code changes whereas others promote to fundamental rewriting of your code. So, if we all try to link data, models and tools, why are we then using such different environments? The basic idea is that the various approaches address supposedly conflicting demands like generality, flexibility, ease of use, accuracy and performance. What are the benefits of the various approaches? This paper addresses these questions by looking at a.o. common component architecture (CCA), earth system modeling framework (ESMF), FRAMES, object modeling system (OMS) and OpenMI. Are they really conflicting or do they to a large degree complement each other?

Keywords: integrated modeling, frameworks, interfaces, interoperability

1 INTRODUCTION

Environmental research integrates a large number of disciplines including atmospheric sciences, hydrology, geomorphology, geology, chemistry and ecology. A wide variety of models across these and other disciplines needs to be coupled to solve the challenges of today and tomorrow. Traditional (sub)mono-disciplinary numerical models have generally not been developed with interoperability in mind (especially not beyond the scope of the initial developers). The interaction of researchers, data, models and tools across (and to some degree even within) disciplines brought with it a range of 'new' challenges in the fields of information technology and semantics, such as: what data to exchange, how to exchange it and what does it actually mean? Many research communities around the world have tackled these questions in (relative) isolation such that different habits, approaches, conventions, interfaces and standards have emerged. As international co-operation increases (especially triggered by a limited number of researchers and common research models and tools) we reach a tipping point [Gladwell, 2000] and the need for interoperability among those communities becomes obvious. So, how different are those approaches?

In the following chapter I'll describe a non-exhaustive list of component coupling technologies that seem to be competing and mutually exclusive since they are all aimed at improving the reusability through componentization, at reducing software development cost and at increasing the effectiveness of integrated research. However, it turns out not to be an either/or decision. To avoid confusion, I'll use the following definitions which have been loosely based on the corresponding wikipedia descriptions.

Architecture is a general description of the structure of all generic parts of a framework.

Component is a software package or a module that encapsulates a set of related functions. Science components generally represent a coherent subset of the physical processes for the whole (or part of the) simulation domain.

Environment is a collection of central software services (infrastructure) used to initialize, start and finalize the components of a simulation. The environment may or may not play an important role in the communication between individual components.

Framework is a reusable implementation of a software architecture. It includes one or more of the following parts: a run-time environment, support libraries, components, their interfaces and conventions.

Interface is a formal, abstract definition of the functions/methods to be exposed/used by a component such that it can interact with the run-time environment and other components.

Implementation is a realization of an architecture or abstract component.

Coupling refers to (sequential or parallel) data transfer between components at run-time. This can happen in memory or via intermediate data files/repositories.

2 THE PLAYERS

This chapter gives a brief alphabetical overview of model coupling technologies developed by various groups. More information can be found in/on the referenced papers and websites.

2.1 CCA – Common Component Architecture

The CCA Forum[1] was founded in 1998 to define a standard for a *scientific, high-performance* component architecture that includes HPC features not available in other generic component architectures such as CORBA, COM, .NET and JavaBeans. The CCA specifications were designed to (1) maintain the performance of components, (2) be non-exclusive with respect to inter-component communication mechanisms, (3) allow for parallellization across components, and (4) allow for configuration of components both prior and during execution.

In CCA the Scientific Interface Definition Language (SIDL) is used to describe the language specific component interface and to define 'uses' and 'provides' ports for the input and output arguments of the routines; these ports may represent scalars, arrays or functions. A CCA-compliant framework should provide (1) SIDL support to generate actual component interface wrappers, (2) services concerning communication, security, thread creation and management, memory management and error handling, (3) a configuration API to instantiate and couple components, and (4) a repository API to access component repositories. The Babel tools are the *de facto* standard for generating the wrapper 'glue' code to make routines written in Fortran, C, C++, Java or Python interoperable on a plug-and-play basis as described by Kumfert [2003].

The main CCA-compliant framework Ccaffeine, which has been developed for parallel computing, can be configured via a command line tool and via a graphical user interface. CCA has been shown to be interoperable with ESMF and MCT, and the CSDMS developers[2] have successfully combined the Ccaffeine framework with the OpenMI 1.4 Java implementation. The CCA developments [Bernholdt et al., 2006] are currently led by TASCS[3]; this virtual organization is funded through the SciDAC (Scientific Discovery through Advanced Computing) program of the US Department of Energy.

2.2 CHyMP – Community Hydrology Modeling Platform

CHyMP[4] is an initiative by the Consortium of Universities for the Advancement of Hydrological Science (CUAHSI) to develop, provide and support advanced simulation models to the academic community within a community-based 'development-user-feedback' framework. This development is still in its infancy, but its long term goal is to stimulate the development and uptake of a number of large community model components in the field of hydrology (contrary to the wide

[1] CCA Forum website: `http://www.cca-forum.org`
[2] Community Surface Dynamics Modeling System website: `http://csdms.colorado.edu`
[3] Center for Technology for Advanced Scientific Component Software website: `http://tascs-scidac.org`
[4] CHyMP website: `http://www.cuahsi.org/chymp.html`

variety of smaller, incompatible and unsupported models that exist today). This CHyMP initiative is closely related to the hydrology focus group of CSDMS.

This initiative should, however, not be confused with the similarly named development of the Community Hydrology Prediction System (CHPS) by the US National Weather Service (NWS) that uses Delft-FEWS[5] for a nationwide early warning system. Such an operational system requires a robust approach and uses mostly one-way data streams with file-based data exchange.

2.3 ESMF – Earth System Modeling Framework

ESMF[6] is a high-performance framework aimed at improving the software interoperability and reuse in climate, numerical prediction, data assimilation, and other Earth science applications. ESMF provides both a coupling superstructure and a utility infrastructure; it supports both MPI and openMP for parallelization. The component code sits between these two layers, making calls to the infrastructure libraries beneath it and being scheduled and synchronized by the superstructure above it. The utility layer includes type definitions, time, clock and alarm functions, parallel data communication and regridding routines, message logging tools, etcetera.

ESMF distinguishes between gridded components (physics and dynamics) and coupler components (interpolation and mapping). All components must define initialize, run and finalize methods: the components' input and output arguments are bundled together in inputState and outputState data structures. Components may optionally use an internal state. These states can store (bundles of) Arrays, (bundles of) Fields, and other States. An Array is a distributed, multi-dimensional array that can carry information such as its type, kind, rank, and associated halo widths. To adopt ESMF, components may wrap their existing Fortran/C arrays into ESMF Array structures. A Field represents a physical scalar or vector field: it contains a data Array along with grid information and metadata. Gridded and coupler components may be nested in gridded components that can be coupled to other components at a higher hierarchical level. Generally, the drive module and all components are linked together into one executable.

ESMF is supported on UNIX, Linux and Windows HPC platforms. ESMF developments include (a) integration with workflow management and visualization services to create modeling environments, (b) automatic generation of couplers, executables, and metadata, (c) web services, and (d) support for a wider variety of grids and numerical methods. Saint and Murphy [2010] use web services for a one-way link from ESMF to OpenMI. The ESMF project is sponsored by the US Department of Defense (DoD), NASA, NSF and NOAA; as of November 2009, it is part of NOAA Environmental Software Infrastructure and Interoperability (NESII) group.

2.4 FRAMES – Framework for Risk Analysis of Multi-Media Environmental Systems

FRAMES[7] is an operational modeling environment in use by the US Environmental Protection Agency (EPA) within which collections of models and modeling tools (e.g., data retrieval and analysis) can communicate with each other. FRAMES applies predefined connection schemes and dictionaries to guarantee correct coupling of the components by end users. 3MRA[8] is an actual set of 17 modules placed within FRAMES that collectively simulate the release, fate & transport, exposure, and risk (human and ecological) associated with wastestream contaminants deposited in various land-based waste management units (e.g., landfills, waste piles). Because of the many processes and parameters involved, model results are based on ten thousands individual simulations[9]. To keep the total simulation time within limits, the components of 3MRA are based on highly simplified formulations for each domain which is quite the opposite of the climate

[5]Delft-FEWS website: `http://public.deltares.nl/display/FEWSDOC`
[6]ESMF website: `http://www.earthsystemmodeling.org`
[7]FRAMES 1 and 2 information: `http://mepas.pnl.gov/FRAMESV1` and `/FRAMESV2`
[8]Multi-media, Multi-pathway, Multi-receptor Risk Analysis website:
`http://www.epa.gov/athens/research/projects/3mra`
[9]Supercomputer for Model Uncertainty and Sensitivity Evaluation website:
`http://www.epa.gov/athens/research/modeling/supermuse/supermuse.html`

model components coupled by frameworks like ESMF. However, the demand for more complete representations is increasing.

For FRAMES v3 the current one-way file-based communication method will be (optionally) replaced by a faster in-memory two-way communication method. This is needed for more complex model components (resulting in more data exchange) as well as for more complex component interaction. This new coupling method will be based on OpenMI.

2.5 HLA – High Level Architecture

The High Level Architecture (HLA) was developed by the Defense Modeling and Simulation Office (DMSO) of the US DoD as a general purpose architecture for distributed real-time training/simulation environments. Typically, this concerns tightly coupled networks in which data exchanges are frequent, but usually small. The HLA baseline definition was completed on 1996; in 2000 it was accepted as general IEEE 1516 standard[10]. It defines the general architecture (federation rules), the interfaces of components (federates) and the environment (RTI: run-time infrastructure) and an object model template (OMT) that provides a common method for recording information and that defines a.o. a federation object model (FOM: data exchanges during a simulation) and a simulation object model (SOM: description of component/federate and its possible exchanges) that can be queried at run-time. Data exchange occurs via the RTI.

HLA is *not* an implementation, it just provides an architectural sketch. The existing HLA RTI implementations (by a.o. Raytheon, MÄK, Pitch, and `http://sourceforge.net/projects/ohla/`) are not 100% compatible because the IEEE standard contains some errors and doesn't fully prescribe the interface implementation. A dynamic link compatible (DLC) API has been defined (SISO-STD-004.1-2004) to make the implementations more consistent. All compatibility issues should have been resolved by the new HLA Evolved IEEE 1516-2010 standard.

2.6 Kepler

Kepler[11] offers a general purpose (nested) workflow framework that supports continuous time, discrete event, and dynamic or parallel data flow concepts for application across a broad range of scientific and engineering disciplines. It's a Java-based application that allows the user to visually assemble workflows by means of directed sequence graphs of components ('actors'). Kepler comes with a default library of some 350 actors for a wide range of tasks including numerical integration, image processing, accessing webservices, reading and writing standard file formats, plotting, and running external command line applications. Development of the open source Kepler software is led by a team from the universities of Davis, Santa Barbara, and San Diego. Taverna[12] is a similar Java-based workflow system developed in the context of the myGrid project led by Carole Goble of the University of Manchester, now part of OMII-UK.

2.7 MCT – Model Coupling Toolkit

MCT[13] is an MPI-based library of Fortran90 modules which can be used to create parallel integrated grid-based models (both structured and unstructured). Version 2 of the toolkit was developed for and used in building the cpl6 coupler [Craig et al., 2005] of Community Climate System Model CCSM3 as described by Larson et al. [2005] and Jacob et al. [2005]. It uses an AttributeVector type to store the local data to be exchanged in a 2D parameter-location array; it uses a GlobalSegmentMap type to describe the global partitioning of a numerical grid across multiple processes. Based on these data types, MCT supports efficient parallel MxN intercomponent data transfer and MxM intracomponent data redistribution, intergrid interpolation using matrix-vector

[10] IEEE website: `http://ieeexplore.ieee.org/xpl/standards.jsp?psf_t=1516`
[11] Kepler website: `https://kepler-project.org`
[12] Taverna website: `http://www.taverna.org.uk`
[13] MCT website: `http://www.mcs.anl.gov/mct`

multiplication, spatial integration and time averaging. MCT can be used in single or multiple executable systems and allows sequential or concurrent execution. Much of what was learned and implemented in MCT and cpl6 has been included in ESMF which will be used for CCSM4.

2.8 OASIS and PALM

Earth system modelers organized in ENES[14] initiated the Partnership for Research Infrastructures in earth System Modelling (PRISM) project to develop a common software infrastructure. Building on their earlier work, CERFACS[15] led the development of the open source OASIS3 and OASIS4 frameworks[16] [Valcke and Morel, 2006]. The OASIS frameworks consist of a driver, transformer and a MPI-based PRISM System Model Interface Library (PSMILe). The individual OASIS3 component executables include initialization, variable definition, get and put, and finalization calls to the statically linked PSMILe. The driver component initializes and connects the components at run time based on configuration files[17]. It also calls the data transformer to repartition and/or regrid the data; data that doesn't require either of these actions will be passed directly from the providing to the using component. Although OASIS3 components may be multi-threaded, the driver and transformer are single threaded (they are parallel in OASIS4). OASIS3 (transformer) requires grid coordinate data to be specified using netCDF files and it supports scalar 2D grid data only. The OASIS4 PSMILe interface includes support for vector quantities and grid definition (1D, 2D and 3D); hence it no longer requires data files for grid information.

Concurrent with these developments, CERFACS developed for the MERCATOR project the closely related proprietary PALM[18] framework for oceanographic data assimilation applications [Valcke and Morel, 2006]. Contrary to the OASIS couplers, PALM supports the dynamic addition and removal of components during the execution by means of MPI2 features. However, it currently lacks the parallel interpolation features of OASIS4.

2.9 OMS – Object Modeling System

The OMS[19] is a domain-specific, reusable framework with a set of interdependent Java classes developed by the US Department of Agriculture (USDA) in collaboration with other agencies and organizations involved with agro-environmental modeling. OMS provides an integrated programming, simulation and analysis environment. Individual components are plain Java classes with an execution method and optional initialization and finalize methods. Component methods, input and output variables as well as are identified and described by means of Java annotations such as @In, @Out, @Unit, and @Execute. Both time and spatial loops are moved out of the components such that most input and output arguments are scalars; individual components are relatively simple. OMS and specific model developers together have migrated (or are in the process of migrating) various watershed models, such as the Soil and Water Assessment Tool (SWAT), J2000 [Krause, 2002], and Precipitation Runoff Modeling System (PRMS) into OMS modules.

2.10 OpenMI – Open Modeling Interface

Version 1.4 of the OpenMI standard[20] was developed within the EU-funded HarmonIT project by researchers from a.o. Delft Hydraulics (now part of Deltares), DHI and Wallingford Software (now part of MWH Soft). This interface standard enables end users to couple components created by different developers without recompilation. It requires simulation engines to be imple-

[14] European Network for Earth System Modelling website: `http://www.enes.org`
[15] Centre Européen de Recherche et Formation Avancées en Calcul Scientifique: `http://www.cerfacs.fr`
[16] OASIS stands for: Ocean Atmosphere Sea Ice Soil
[17] Each component has a PMIOD (potential model input and output description) and SMIOC (specific model input and output configuration). The SMIOC files combined with the SCC (specific coupling config) of the simulation determine the run time exchanges.
[18] Projet d'Assimilation par Logiciel Multi méthode: `http://www.cerfacs.fr/globc/PALM_WEB`
[19] OMS development website: `http://honeycomb.javaforge.com/project/oms`
[20] OpenMI website: `http://www.openmi.org`

mented as 'linkable components' that provide methods to (1) initialize the component, (2) query the component for (providing and accepting) 'exchange items', (3) define links, (4) get values from the component, and (5) finalize the component. An exchange item is a quantity defined on an 'element set' which is a location specification (set of either labels or coordinates). Providing components are responsible for the interpolation of data to the element set of the requesting component. This first version of the standard is strictly based on a pull-based approach as it doesn't include a method for setting values. After initializing all components and their links, a simulation starts by asking the 'final' component in the workflow for data. Subsequently, that component will start the necessary computation and, as needed, it will request values from the linked components without intervention of a central framework. Deadlocks in cyclic workflows are prevented by requiring that a component must return 'best guess' values *without calling other components* if it's already waiting for data due to a previous GetValues() call. A first reference implementation of the standard was created using C#[21]. A Java implementation was later provided by Alterra; they combined OpenMI with formal ontologies in the SEAMLESS[22] Integrated Framework. Although both reference implementations use only a single execution thread, this not strictly required by the OpenMI standard. OpenMI has been shown to be compatible with remote and multithreaded engines, and webservices.

In 2007 the OpenMI Association was founded to take formal ownership of the standard and the associated reference implementation(s). The upcoming version 2.0 of the standard (see Gijsbers et al. [2010] and Donchyts et al. [2010] for details) adds a SetValues() method and separates time progress from the GetValues() method; these changes will make OpenMI easier to use when coupling with (geospatial) databases and data assimilation. In this context it is worthwhile to mention the Dutch OpenDA[23] initiative to develop a flexible open source calibration and data assimilation framework based on a component interface that is consistent with OpenMI.

2.11 TIME – The Invisible Modelling Environment

TIME[24] supports developers in creating, testing and delivering environmental simulation models; it has been developed by CSIRO [Rahman et al., 2005]. It is a .NET framework that includes object libraries for standardized data IO, GIS operations, data visualisation, uncertainty assessment and non-linear optimisation. A GUI specific for the model is automatically generated based on metadata tags to component variables. E2/WaterCAST [Cook et al., 2009] extends the TIME framework with functionality to link sub-catchment hydrological and constituent models along streams down to the tidal limits in estuaries.

3 COMPARING DEVELOPMENTS AND CONCLUSIONS

FRAMES and CHyMP differ from the rest by their focus on the end-user and collaborative side. The same holds for other (local and web-based) modeling environments like CSDMS, DeltaShell [Donchyts and Jagers, 2010], OpenWEB[25] and iemHUB[26]. Instead of developing another coupling technology, these initiatives tend to adopt (and adapt) such technologies; hence, I'll ignore them in this comparison. Furthermore, it should be noted that CCA, HLA and OpenMI in essence only define architectures and interfaces, whereas the other initiatives have the actual implementation as objective. The standard body for HLA doesn't even provide a reference implementation, whereas the CCA and OpenMI developers are at least creating one.

The interfaces defined by CCA, ESMF, OASIS, OMS, OpenMI and TIME are similar in the sense that they all use initialize, run, finalize, get and set concepts. However, a comparison by Lloyd et al. [2009] showed that the amount of code needed varies significantly: OMS 3.0 requires the

[21] Runs on Linux via the open source mono evironment: `http://www.mono-project.com`.
[22] System for Environmental and Agricultural Modelling; Linking European Science and Society website: `http://www.seamlessassociation.org`
[23] OpenDA association website: `http://www.openda.org`
[24] TIME website: `http://www.toolkit.net.au/Tools/TIME`
[25] `https://openweb.uk.net` uses the OpenMI-based Pipistrelle C# environment by HR Wallingford
[26] iemHUB website: `http://iemhub.org`

topic	CCA	ESMF	HLA	Kepler	MCT	OASIS	OMS	OpenMI	TIME
defines framework	✓	✓	✓	✓		✓	✓		✓
defines interfaces	✓	✓	✓		✓	✓	✓	✓	
provides (reference) implementation	r✓	✓		✓	✓	✓	✓	r✓	✓
defines object model		✓			✓	✓		✓	
code invasiveness [Lloyd et al., 2009]	−−	+	?	?	?	?	++	−	?
plug & play (and graphical coupling)	✓		(✓)	✓				✓	✓
support for HPC environment	✓	✓	✓		✓	✓			
C/FORTRAN support	✓	✓	(✓)	W	✓	✓	W	W	W
Java support	✓		(✓)	✓			✓	✓	
.NET support								✓	✓

Table 1. Comparison of coupling technologies

least due to its plain Java approach and the strategic use of annotations, but CCA generates a lot of language interoperability code for each component. CCA, HLA and Kepler don't impose any particular (spatial) object model. ESMF, MCT and OASIS use numerical grids as spatial data representations, whereas OpenMI takes an approach consistent with OGC[27] conventions. OMS removes in general the spatial loop from the model components, such that the component variables are scalars and no complex objects are needed. The same holds for TIME, but it supports raster and network data types. OMS and TIME object types may be extended using plain Java or C#.

CCA, HLA, Kepler, OpenMI and TIME allow the end user to couple pre-compiled components from different developers at run-time and (except for HLA) provide a graphical environment for that. ESMF, MCT and OASIS focus on the high-performance user community with primary programming languages Fortran and C. Kepler, OMS, OpenMI and TIME support the Fortran and C users only indirectly through wrappers. Only OpenMI and TIME provide support for the .NET platform. Which languages HLA supports depends on the implementation (generally Java or C). CCA is the only architecture that really addresses the issue of language interoperability through the use of SIDL and Babel.

There is a trend to guide users in coupling components. Ontologies and other metadata conventions are important to guarantee the validity of links and for the automatic discovery of possible links. Frameworks with an operational focus, like FRAMES, already apply such conventions to some degree. In climate research, US Earth System Curator[28] and European METAFOR[29] projects work together with ESMF and OASIS (component) developers to adopt Climate & Forecast conventions[30].

Although the technologies described here were all developed with a common vision of a componentized architecture, they have addressed different parts of the integrated modeling challenge (generality, flexibility, ease of use, accuracy and/or performance). One might be able to use the similarities in their interfaces to implement a generic wrapper generator (see e.g. the Bespoke Framework Generator by Armstrong et al. [2009]). This would improve the portability of components among compatible frameworks.

REFERENCES

Armstrong, C. W., R. W. Ford, and G. D. Riley. Coupling integrated earth system model components with BFG2. *Concurrency and Computation: Practice and Experience*, 21:767–791, 2009.

[27] Open Geospatial Consortium website: `http://www.opengeospatial.org`
[28] Earth System Curator website: `http://www.earthsystemcurator.org`
[29] Common Metadata for Climate Modelling Digital Repositories: `http://metaforclimate.eu`
[30] CF metadata conventions website: `http://cf-pcmdi.llnl.gov`

Bernholdt, D. E., B. A. Allan, R. Armstrong, F. Bertrand, K. Chiu, T. L. Dahlgren, K. Damevski, W. R. Elwasif, T. G. W. Epperly, M. Govindaraju, D. S. Katz, J. A. Kohl, M. Krishnan, G. Kumfert, J. W. Larson, S. Lefantzi, M. J. Lewis, A. D. Malony, L. C. McInnes, J. Nieplocha, B. Norris, S. G. Parker, J. Ray, S. Shende, T. L. Windus, and S. Zhou. A component architecture for high-performance scientific computing. *International Journal of High Performance Computing Applications*, 20:163–202, 2006.

Cook, F. J., P. W. Jordan, D. K. Waters, and J. M. Rahman. WaterCAST – whole of catchment hydrology model an overview. In Anderssen, R. S., R. D. Braddock, and L. T. H. Newham, editors, *18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation*, pages 3492–3499. Modelling and Simulation Society of Australia and New Zealand, July 2009.

Craig, A., R. Jacob, B. Kauffman, T. Bettge, J. Larson, E. Ong, C. Ding, and Y. He. Cpl6: The new extensible, high-performance parallel coupler for the community climate system model. *International Journal for High Performance Computing Applications*, 19(3):309–327, 2005.

Donchyts, G., S. Hummel, S. Vaneçek, J. Groos, A. Harper, R. Knapen, J. Gregersen, P. Schade, A. Antonello, and P. Gijsbers. OpenMI 2.0 - What's new? In *Proceedings iEMSs 2010*, 2010.

Donchyts, G. and H. R. A. Jagers. DeltaShell – an open modelling environmental. In *Proceedings iEMSs 2010*, 2010.

Gijsbers, P., S. Hummel, S. Vaneçek, J. Groos, A. Harper, R. Knapen, J. Gregersen, P. Schade, A. Antonello, and G. Donchyts. From OpenMI 1.4 to 2.0. In *Proceedings iEMSs 2010*, 2010.

Gladwell, M. *The tipping point: how little things can make a big difference*. Little, Brown and Company, New York, 2000.

Jacob, R., L. Larson, and E. Ong. MxN communication and parallel interpolation in CCSM3 using the model coupling tookit. *International Journal for High Performance Computing Applications*, 19(3):293–307, 2005.

Krause, P. Quantifying the impact of land use changes on the water balance of large catchments using the J2000 model. *Physics and Chemistry of the Earth*, 27(9-10):663–673, 2002.

Kumfert, G. Understanding the CCA standard through Decaf. Technical report, Lawrence Livermore National Laboratory, 2003.

Larson, J., R. Jacob, and E. Ong. The model coupling toolkit: A new Fortran90 toolkit for building multi-physics parallel coupled models. *International Journal for High Performance Computing Applications*, 19(3):277–292, 2005.

Lloyd, W., O. David, J. C. Ascough II, K. W. Rojas, J. R. Carlson, G. H. Leavesley, P. Krause, T. R. Green, and L. R. Ahuja. An exploratory investigation on the invasiveness of environmental modeling frameworks. In Anderssen, R. S., R. D. Braddock, and L. T. H. Newham, editors, *18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation*, pages 909–915. Modelling and Simulation Society of Australia and New Zealand and International Association for Mathematics and Computers in Simulation, July 2009.

Rahman, J. M., S. P. Perraud, H. Hotham, N. Murray, B. Leighton, A. Freebairn, G. Davis, and R. Bridgart. Evolution of TIME. In Zerger, A. and R. Argent, editors, *MODSIM 2005 International Congress on Modelling and Simulation*, pages 697–703. Modelling and Simulation Society of Australia and New Zealand, December 2005.

Saint, K. and S. Murphy. End-to-end workflows for coupled climate and hydrological modeling. In *Proceedings iEMSs 2010*, 2010.

Valcke, S. and T. Morel. OASIS and PALM, the CERFACS couplers. Technical Report TR/CMGC/06/38, Centre Européen de Recherche et Formation Avancées en Calcul Scientifique, 2006.