Available online at www.sciencedirect.com

SCIENCE DIRECT®

Ocean Modelling 13 (2006) 197–220

Ocean Modelling

www.elsevier.com/locate/ocemod

Web-enabled configuration and control of legacy codes: An application to ocean modeling

C. Evangelinos [a,*], P.F.J. Lermusiaux [b], S.K. Geiger [a], R.C. Chang [a], N.M. Patrikalakis [a]

[a] *Department of Mechanical Engineering, MIT, Cambridge, MA 02139, USA*
[b] *Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02139, USA*

Received 16 August 2005; received in revised form 30 September 2005; accepted 4 October 2005
Available online 17 November 2005

Abstract

For modern interdisciplinary ocean prediction and assimilation systems, a significant part of the complexity facing users is the very large number of possible setups and parameters, both at build-time and at run-time, especially for the core physical, biological and acoustical ocean predictive models. The configuration of these modeling systems for both local as well as remote execution can be a daunting and error-prone task in the absence of a graphical user interface (GUI) and of software that automatically controls the adequacy and compatibility of options and parameters. We propose to encapsulate the configurability and requirements of ocean prediction codes using an eXtensible Markup Language (XML) based description, thereby creating new computer-readable manuals for the executable binaries. These manuals allow us to generate a GUI, check for correctness of compilation and input parameters, and finally drive execution of the prediction system components, all in an automated and transparent manner. This web-enabled configuration and automated control software has been developed (it is currently in "beta" form) and exemplified for components of the interdisciplinary Harvard ocean prediction system (HOPS) and for the uncertainty prediction components of the error subspace statistical estimation (ESSE) system. Importantly, the approach is general and applies to other existing ocean modeling applications and to other "legacy" codes.
© 2005 Elsevier Ltd. All rights reserved.

PACS: 92.10.−c; 92.20.−h; 92.60.Wc; 95.75.−z

Keywords: XML; Encapsulation; Metadata; GUI; Numerical ocean models; HOPS; ROMS; MITgcm; Primitive equation; Data assimilation; Ocean observing and prediction systems

[*] Corresponding author. Address: Room 54-1518, Department of Earth, Atmospheric and Planetary Sciences, MIT, 77 Massachusetts Avenue, Cambridge, MA 02139, USA.
E-mail address: ce107@mit.edu (C. Evangelinos).

1463-5003/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ocemod.2005.10.002

1. Introduction

Effective ocean modeling and forecasting is essential for scientific investigations and other human operations and activities in the ocean. Application areas include among others fisheries management, pollution control and maritime and naval operations. The advances in physical oceanography numerical models and data assimilation (DA) schemes have given rise to complete ocean prediction systems that are used in operational settings. Computationally intensive modeling research in interdisciplinary ocean science is emerging, on multiple scales and for multiple applications, including the coupling of physical and biological oceanography with ocean acoustics (Patrikalakis et al., 2000; Robinson and Lermusiaux, 2004). The learning curve to achieve confident and robust utilization of disciplinary modeling systems is often steep; the additional complexities of interdisciplinary computations renders the learning curve even steeper. Recent developments in high-performance computing, networking infrastructure and configuration and control software now make it possible to construct distributed computing systems that address such computationally intensive problems. The present contribution aims to help utilize some of these developments, focusing on modern web-enabled user-software interactions and automated oversight of complex interdisciplinary ocean prediction and data assimilation systems, from the setup-time to the run-time of a specific ocean application.

There is now a body of literature on computational ocean modeling (e.g. Haidvogel and Beckmann, 1999; Kantha and Clayson, 2000). Such research occurs on multiple scales, from unstructured mesh for coastal ocean applications (Pietrzak et al., 2005) to specific approaches for the world ocean (Semtner, 1997), global climate modeling (Griffies, 2004) or even the age of properties in the ocean (Deleersnijder et al., 2001). Computational schemes have been developed within several ocean prediction models implemented for many regions and scales (e.g. Lynch and Davies, 1995; Mooers, 1999). Recent examples include, for the: US eastern coastal oceans (Signell et al., 2000; Lynch et al., 2001; Robinson and the LOOPS Group, 1999), northwestern Atlantic (Chassignet and Malanotte-Rizzoli, 2000), Atlantic Ocean (Halliwell et al., 2001; Stammer and Chassignet, 2000), Pacific Ocean and US western coastal oceans (De Szoeke et al., 2000), Mediterranean Sea (Pinardi and Woods, 2002; Onken et al., 2004), European NorthSeas (Berntsen and Svendsen, 1999; Burchard and Bolding, 2000), and other basins and the global ocean (Dutay et al., 2002; Gent et al., 1998). Some models are also utilized by multiple users for various purposes, e.g., MOM (Pacanowski and Griffies, 2000), NLOM/DART (Wallcraft, 1991), ROMS (Haidvogel et al., 2000; Arango et al., 2000), POM (Ezer et al., 2000), POP (Smith et al., 1992), MITgcm (Marshall et al., 1997a,b) and TOMS (Arango, 2001). Some of these systems have initiated interdisciplinary modeling and forecasting research.

On the other hand, efficient, web-enabled and user-friendly numerical systems for environmental and earth science research and applications are just starting to be developed. Recent progress includes solver interfaces for parallel oceanic and atmospheric models (Frickenhaus et al., 2005), Java frameworks for parallel ecosystem simulations (Wenderholm, 2005) and home desktop hosted GUI-driven programs for climate, meteorological and air pollution modeling (EDGCM, 2005; Hurley et al., 2005). A lot of effort (e.g. DAGman, 2005; Deelman et al., 2004; Hategan et al., 2004; Buyya and Venugopal, 2004) within the wider context of grid computing, has been concentrated on the workflow aspects of the composition of more complicated interdisciplinary applications; adapting monolithic legacy workflows to execution on the Grid is another aspect of the same problem. A few projects have looked into efficient computational and web-based integration of diverse environmental software and applications (e.g. Argent, 2004). Specific Internet-based execution environments are also being developed (Fatoohi et al., 2005). Such advances will be useful for state-of-the-art prediction systems for complex natural processes.

The present effort is part of a larger NSF-sponsored distributed computing project (Patrikalakis, 2005), that brings together advanced modeling, observation tools, and estimation methods for oceanographic research. The focus is on one of the objectives of this larger initiative: To contribute to the seamless access, analysis, and visualization of experimental and simulated forecast data, through a science-friendly web interface that hides the complexity of the underlying distributed heterogeneous software and hardware resources. The aim is thus to allow the ocean scientist/forecaster to concentrate on the task at hand as opposed to the micro-management of the underlying modeling mechanisms and computational details.

As a starting point we employ the Harvard ocean prediction system (HOPS) (Robinson et al., 2005; Robinson, 1999; Lozano et al., 1996; Robinson et al., 2002) as an advanced ocean forecast system. HOPS is a portable and generic system for interdisciplinary nowcasting and forecasting through numerical simulations of the ocean. It provides a framework for obtaining, processing, and assimilating data in a dynamic model capable of generating forecasts with 3D fields and error estimates. HOPS has been successfully applied to several diverse coastal and shelf regions, and analysis has indicated that real-time operational forecast capabilities were achieved. HOPS is also coupled to an advanced estimation system (error subspace statistical estimation—ESSE (Lermusiaux and Robinson, 1999; Lermusiaux et al., 2002)) that allows quasi-optimal DA and provides real-time estimates of the dominant uncertainty modes in the forecast. With such error estimates, ESSE allows quantitative adaptive sampling and adaptive modeling (Lermusiaux et al., 2004). ESSE is also part of our encapsulation efforts.

One of the first practical problems is that HOPS (as well as other ocean modeling systems, e.g. for ocean physics, ROMS (Haidvogel et al., 2000), or acoustics, OASES (Schmidt and Tango, 1986)) are, like the vast majority of scientific applications, what computer scientists tend to term as "legacy code". The term "legacy" should not be misconstrued to imply outdated code in our context: these are all models with an active development community and recent enhancements. For various reasons, several codes are still being written at least partially in Fortran 77. Even when Fortran 90/95 is used, most user interactions occur via the command line.

These command-line driven applications consist of native binaries (as opposed to fat binaries or bytecode for Java/CIL etc.) that expect a standard input (stdin) stream, maybe some command-line options and a set of input files, and generate a set of output files as well as standard output (stdout) and error (stderr) streams. In such a setup, any workflows are either executed interactively (a very common approach) or (after all potential problems are handled) hard-coded in scripts. While this command-line driven approach, which dates from the days when graphical user interfaces (GUIs) were not available, is efficient for a skilled user, it is cumbersome and error-prone and entails a steep learning curve. Moreover it is not suited for efficient web-driven remote usage and does not fit the novel distributed computing model of software components interacting through remote procedure calls.

We examined various ways of dealing with this issue of "legacy", with the more expensive ranging from costly rewriting of the code (Terekhov and Verhoef, 2000) to reformating the code in discrete, well-defined components (e.g. CCA, 2004). Given our specific requirements however and keeping in mind that our system was intended from the outset to be able to handle non-HOPS models in the future, we opted instead to keep working with Fortran binaries developed by the ocean modelers and to encapsulate their functionality and requirements using the eXtensible Markup Language (XML, 2005). In this manner we create a computer-readable manual for the codes, allowing us to generate a GUI, check for parameter correctness and drive execution in a transparent manner. This methodology is general by design and it applies to many other "legacy" codes, allowing for the automated generation of corresponding validating GUIs.

In what follows, Section 2 briefly describes HOPS/ESSE-based forecast workflows and their characteristics, and our motivations for this work. Section 3 discusses requirements that legacy software, such as HOPS/ESSE, MITgcm or ROMS, impose on our system design and outlines some solutions. Section 4 describes the new XML schema (XML schema, 2005) based language (Legacy Computing Markup Language, LCML) utilized for constraining descriptions of encapsulated binaries. Section 5 presents results to illustrate our implementation, Section 6 discusses some related work and Section 7 concludes.

2. Motivation: ocean prediction and data assimilation workflows

As a motivating example of the complexity of an ocean prediction workflows, we first concentrate in this section on the setup and initialization of HOPS for ocean simulations and predictions. For each application to a specific ocean region, the most frequent tasks are, successively: the definition of modeling domains (grid generation, topography, coastlines); the preparation of the synoptic and historical data sets (data file management, adding salt to temperature profiles, etc.); the griding of these data for initialization and possible assimilation (objective analyses, feature models, etc.); the computation of the non-observed or partially observed variables for initialization of the full numerical ocean state (e.g. the use of geostrophic equilibrium for computing a first-guess at the internal velocities from density); the preparation of ocean-atmosphere fluxes saved as forcing fields; the setup of the files for the ocean simulation or prediction; running the numerical predictive model (e.g. run the primitive equation (PE) model); and finally, the visualization of the output datasets.

For all these tasks, a series of codes need to be compiled and set up. Parameter files need to be defined and directories and links created for each simulation. Finally, for each ocean application or scientific study, a multitude of setups, initializations and simulations are carried out. This includes for example multiple runs for model tuning and/or for ensemble predictions. With the recent progress in computer science and web-based technologies, the details related to the management of all of these tasks and their results should now be carried out by automated web-enabled software.

The specific software modules that are most frequently involved in the above setup and initialization steps of HOPS are illustrated in Fig. 1. First, repeated utilization of the **GRIDS** program generate an appropriate 2D horizontal grid discretization of the solution domain. The land mask points for the grid (if the solution domain covers both the ocean and land) are then generated by repeated applications of **PE_mask**. The resulting 3D grid's bottom topography is then conditioned by **Cond_Topo** through filtering, which aims to eliminate possible numerical instabilities. If the initialization or data assimilation is based on objective analyses (OA) of ocean observations, either the full-matrix (global) objective analysis **OAG** or a local approximation **OA** software module is utilized to grid the synoptic and historical data. In doing so, data files are prepared, for example remote or in situ field measurements are concatenated into different modular ocean data sets (MODS, i.e. ASCII format files, usually processed by multiple data management codes and possibly by the **AddSalt** program). The initial conditions and external forcing are then prepared by **PE_initial** and **PE_forcing** respectively, leading to the execution of the **PE_model** forecast binary.

During a numerical simulation, ocean data can be assimilated, in part to control the loss of predictability due to the non-linear growth of errors. Data assimilation (DA, Robinson et al., 1998) leads to estimates of natural fields that are better than can be obtained by using only observations or a dynamical model. Data and models are usually combined in accord with their respective uncertainties, by quantitative minimization of a cost function. Model and data uncertainties need to be adequate (Killworth et al., 2003). DA computations can be very expensive. The simplest and robust DA scheme utilized in HOPS is an optimal interpolation (OI) scheme, which combines the OA-ed data with the model forecast by blending, in agreement with data uncertainties. The other scheme in use with HOPS, ESSE (Lermusiaux and Robinson, 1999; Lermusiaux et al., 2002), aims to provide an optimal reduction of the DA problem: only the dominant errors are minimized. For example, if a variance criterion is used to combine data and dynamics, the "dominant errors" are defined by the dominant ordered eigen-decomposition of a normalized form of the error covariance matrix. Even with such reductions, ESSE still involves massive throughput computations. However, by design, it provides tremendous opportunities for scalable parallelism.

The present ESSE DA workflow, builds on top of the standard HOPS workflow as follows. First, the interdisciplinary error subspace is initialized (Fig. 2, far left oval) based on a dominant error decomposition on multiple scales (Lermusiaux et al., 2000; Lermusiaux, 2002) (using binary **mapcor**) and by perturbing (binary **pert**) the central initial state using these dominant error modes and a simple random number model for the truncated errors. The dominant initial errors are then evolved by an ensemble of perturbed non-linear and stochastically forced dynamical model integrations (**PE_model**: Fig. 2, center left oval). As the size of the

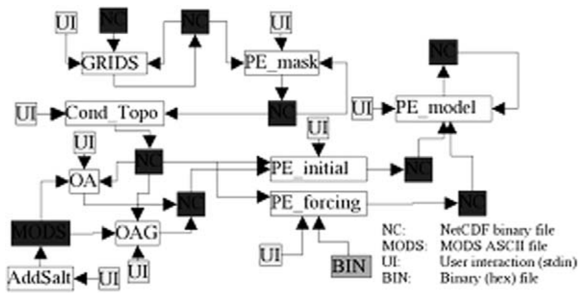


Fig. 1. The HOPS pipelined setup, initialization and simulation workflow. Except for the infrequent usage of forcing files in "native" binary format, all binary (blue) files are in the NetCDF portable format. User interactions (stdin) are yellow. Other external ASCII and NetCDF input files (e.g. ocean data) are omitted for clarity. (For interpretation of the references in color in this figure legend, the reader is referred to the web version of this article.)

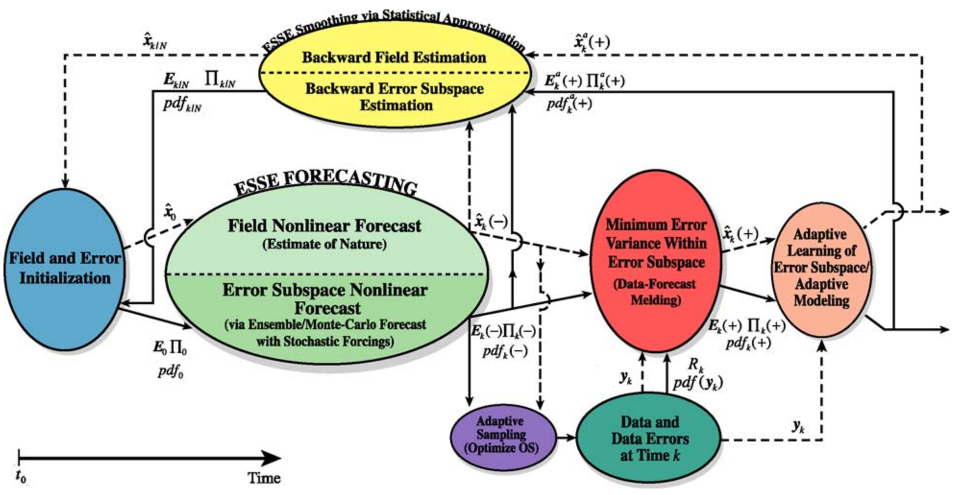

ESSE Smoothing via Statistical Approximation
Backward Field Estimation
Backward Error Subspace Estimation
ESSE FORECASTING
Field Nonlinear Forecast (Estimate of Nature)
Error Subspace Nonlinear Forecast (via Ensemble/Monte-Carlo Forecast with Stochastic Forcings)
Field and Error Initialization
Minimum Error Variance Within Error Subspace (Data-Forecast Melding)
Adaptive Learning of Error Subspace/ Adaptive Modeling
Adaptive Sampling (Optimize OS)
Data and Data Errors at Time k
Time
t_0
$\hat{x}_{kIN}$
E_{kIN} Π_{kIN} pdf_{kIN}
$\hat{x}^a_k(+)$
$E^a_k(+)$ $\Pi^a_k(+)$ $pdf^a_k(+)$
$\hat{x}_0$
E_0 Π_0 pdf_0
$\hat{x}_k(-)$
$E_k(-)\Pi_k(-)$ $pdf_k(-)$
$\hat{x}_k(+)$
$E_k(+)$ $\Pi_k(+)$ $pdf_k(+)$
y_k
R_k $pdf(y_k)$

Fig. 2. The ESSE schematic workflow.

ensemble is increased, a repetitive sequence of three binaries calculates the: ensemble spread or uncertainty matrices (**differ**), singular value decompositions (SVD) of these matrices (**svddif**), and convergence criteria (**chcksvd**) usually comparing the error subspaces provided by the SVDs. Converged ensemble sizes are often O(100–1000). This provides a significant opportunity for throughput parallelism. Individual model integrations (**PE_model** runs) can also be parallel simulations (depending on the problem size and interdisciplinary nature of the problem).

Once error estimates have converged, adaptive sampling forecasts are issued (Fig. 2, bottom left oval). For example, future sampling patterns of autonomous underwater vehicles are computed using modified integer programming (Yilmaz, 2005) so as to maximize the reduction of forecast errors. As new data are available, data-forecast misfits are computed and used to correct the predicted fields by minimum error variance estimation (**meld**) in the error subspace (Fig. 2, center right oval). Outputs are filtered fields and error estimates. A posteriori data misfits are then calculated and used for adaptation of the dominant errors (Fig. 2, right oval). Ultimately, the smoothing via ESSE is carried out using a set of software (**ESSE_smooth**: Fig. 2, top oval) to correct, based on future data, the past fields and uncertainties (Lermusiaux et al., 2002).

The computationally challenging management of the ESSE/HOPS workflows is compounded by the fact that just as in the case of standalone HOPS, each one of the ESSE binaries also requires build- and run-time configuration (user interaction as in Fig. 1). The specifics will be described in Section 3. Given the rather heavy-weight and complex nature of the individual workflow binaries and the fixed nature of the ESSE workflow, Grid-enabled (Foster et al., 2003) workflow execution provides a natural fit. Beyond workflow management which, in the case of the Grid, has had a lot of research activity, issues with configuring (build- and run-time) executables also need to be tackled, preferably in a web-enabled manner.

Importantly, our configuration software needs to be modular and flexible enough so as to easily include new developments to the above ocean prediction and data assimilation system workflow. In the particular case of HOPS-ESSE, this includes adaptive modeling and adaptive sampling software, and plans to further distribute certain forecasting and data assimilation components by separating/parallelizing sequential modules. For example, physical and biological time-stepping modules can be run simultaneously on different grids by different CPUs with data exchanges between modules.

3. Legacy codes

The ocean modeling and data assimilation software of HOPS are legacy codes mostly in Fortran 77 and matlab. As such they do not immediately fit well within a modern distributed computing model: for example, there is no support for dynamic memory allocation and no straightforward provision for remote procedure calls. At worst, some ocean modeling codes require manual intervention (conversion, concatenation, file editing) to work with each other using file I/O exchange. At best, as in the case of HOPS, they have been designed to interoperate in prescribed workflows. The individual components of such workflows are the legacy binaries and the connections between them are files and/or streaming input/output. This type of setup is common among scientific codes that have evolved over many years.

3.1. The modernization conundrum

Seen from an information technology viewpoint (for an overview see Argent (2004)), there are two major options for dealing with the issue of legacy: Migration and Encapsulation (wrapping). The cleanest approach for adapting to a modern distributed computing infrastructure appears to be migrating the code and rewrite all applications in a language platform employing object-oriented and component technologies such as C++/CORBA (Serrano et al., 2002). Moving to the use of platform–agnostic platforms (by transferring the codes to Java, Seymour and Dongarra, 2003, for example) and Mobile Agents (Houstis et al., 2002) for the distributed calculations would be a radical form of such modernization. Such an option is very flexible if implemented in the context of a complete framework for future code development (e.g. Wenderholm, 2005), but it is obviously extremely costly in terms of programming effort and accounting of all evolutionary work done so far. Moreover, it is error-prone as it is impractical to convert existing procedural programs to object-oriented components (Terekhov and Verhoef, 2000). Finally, and very importantly for the case of scientific applications that are performance sensitive, it comes at a heavy price in performance, especially for Java-based solutions.

The invasive but more traditional approach is to encapsulate legacy codes as modern software components using wrappers. For example, CORBA (OMG, 2005), or better still, Common Component Architecture Forum (CCA, 2004), can be used. The code can also be wrapped in C/C++ and then called from Java (in a more complicated Mobile Agent setting for distributed computing) using JNI (JNI, 2004; Bubak et al., 2001). However this powerful encapsulation approach involves breaking the Fortran code into separately callable components (Sang et al., 2002) that a main program can access, which once again is a very expensive exercise without commensurate returns. One can also choose to encapsulate the whole of a program instead (Walker et al., 2000a,b; Fatoohi et al., 2005; Wohlstadter et al., 2001).

A non-invasive approach is to keep employing the legacy binaries in predefined (but configurable) workflows with all data exchange between binaries continuing to take place through file I/O. The binaries configuration however is shifted from hand-edited files and scripts to automatically generated GUIs. This way of working with legacy codes reduces to devising an extensible encapsulation of the software components (as binaries) that treats them as black boxes with a set of inputs/outputs and a set of valid types and ranges of compile-time and run-time parameters. The advent of XML provides a standards-based way to accomplish this. XML describes data through the use of custom tags thus eliminating the need to conform to a specific programming structure and offering the possibility to integrate legacy software with new technology.

Due to both some common "ancestral" links and adoption of dominant programming languages in use at the time of development, a lot of ocean codes share common build- and run-time coding options: C-preprocessor directives are mainly used to incorporate many different and possibly alternative code paths/modules in the same program, selectable at compile time with the resulting speed and memory savings. Namelists have been chosen by many models as a simple yet powerful way to provide run-time parameter and option handling without having to write a lot of user interaction code. Due to a heritage of development on Unix platforms, the build system is Makefile based (either directly or through some approach that creates Makefiles). NetCDF (2005) is often the preferred I/O approach for model data input and output while ASCII files (including the case of redirected standard input) are usually used to provide for run-time parameters and options. In the following subsections we present (in alphabetical order) three different ocean models in more detail in light of their build- and run-time configuration requirements.

3.2. HOPS

HOPS was built based on the Geophysical Fluid Dynamics Laboratory primitive equation model, (e.g. Bryan and Cox, 1967). As summarized in Sections 1 and 2, it has evolved in the last 20 years into a relatively complex system, including software modules for physical, biological and acoustical modeling, initialization, boundary conditions, data assimilation, adaptive sampling and skill evaluation. In this subsection, we review only computational properties of the ocean physics PE model software module. Other modules have similar build- and run-time characteristics.

The PE model employs stretched terrain-following or flat coordinates in the vertical and a choice between a cartesian, aligned or rotated spherical polar grid in the horizontal. It can be used in a rigid-lid, surface pressure or free-surface configuration, with various options for tidal forcing. The PE software consist of a suite Fortran 77 codes with no dynamic memory allocation and mostly NetCDF-based I/O. Therefore the PE binary needs to be compiled either with large enough arrays to run with smaller problem sizes or be recompiled every time it is used in different configurations or applied to another application. The PE of HOPS uses mostly GNU make (Stallman and McGrath, 2005) and platform specific Makefiles (that differ in very few locations). The vast majority of code configuration is done at build-time through the use of multiple alternative code sections selectable via more than 100 different C-preprocessor directives declared in the Makefile (see Fig. 3). Some of these directives are not orthogonal and therefore have dependencies (requirements or conflicts with each other).

While preprocessor directives handle the specialization of the code paths, adaptation of the model to a particular region of the ocean is done via modifying an include file (“param.h”) as seen in Fig. 4.

What is shown in Figs. 3 and 4 are but a small subset of the total space of compile-time configurability of the PE model of HOPS. As can easily be surmised, the task of configuring a robust model adequate for a specific application is challenging and the learning curve for a new user can be steep.

This level of complexity continues at the level of run-time configuration, which is provided by a minimum of one file (piped through standard input—see Fig. 5). Run time parameters in that file are read from

```
#  -Dbottom          Second-order in time bottom stress calculation.
#  -Dcoast           Coastal boundary conditions
#  -Doias            Intermittent OI assimilation option
#  -Dforcing         space/time variable forcing.(Do not use with both
#                    "analytical" and "dblprec" also active)
#  -Dresetjulian     Start surface forcing clock from supplied value.
#  -Dnkfix           Revise protections & input for Niiler-Kraus mixed-layer.
#  -Dpttrcsrc        specification of point tracer sources
#  -Drivsrc          River sources.  Use with PTTRCSRC
#  -Dfrozentrc       Initially freeze tracers for specified number of timesteps
#  -Dvel_conv        Require convergence in velocity when solving PBAR
#  -Ddblsigma        Double "sigma" transformation in the vertical
#  -Dpressbias       Remove bias from pressure gradient
#  -Dshapmean        Remove "mean" tracer field before shapiro filtering
#  -Dbotfrc          Bottom friction.
#  -Dcstfrc          Coastal friction.
#  -Dnest2larger     Two-way nesting to larger grid.
#  -Dnest2smaller    Two-way nesting to smaller grid.
#  -Dusrdiagnostic   Report user defined diagnostics.
#  -Dnesttime        Times various elements of the run.
#  -Dposmxtid        Sets (tidal) vertical mixing in pre-defined areas and
#                    over given depth range.
#  -Drmdocinc        Remove documentation in all include files.
#  -Dsundate         SUN's intrisic date/time function
#  -Dsunflush        regularly flush output buffers in SUN systems.
#
#*********************************************************************

CPPFLAGS = -Dbottom -Dsecondmean -Dshapnocoastflux -Dpindex \
           -Dcoast -Doias -Dforcing -Dresetjulian -Dnkfix \
           -Dpttrcsrc -Drivsrc -Dfrozentrc -Dvel_conv -Ddblsigma \
           -Dpressbias -Dshapmean -Dbotfrc -Dcstfrc \
           -Dnest2larger -Dnest2smaller -Dusrdiagnostic -Dnesttime \
           -Dposmxtid -Drmdocinc -Dsundate -Dsunflush
```

Fig. 3. Heavily edited-out example subset of the PE model of HOPS built-time Makefile.

```
#ifndef rmdocinc
c  IMT     Number of tracer points in the x-direction.
c  JMT     Number of tracer points in the y-direction.
c  IJMX    Either IMT or JMT, whichever is greater.
c  KM      Number of vertical levels.
c  LBC     Number of arrays of slab incidental data (usually, LBC=2).
c  LSEG    Maximum number of sets of Start and End indices per row or
c          per column (number of coastal segments plus one).
c  MISLE   Maximum number of islands in the model basin.
c  MPROF   Maximum number of points in mean TS profile.
c  NT      Number of tracer type variables (generally, NT=2 for T & S).
c  MCLEN   Maximum number of points in a coastal segment.
c  MCSEG   Maximum number of coastal segments.
c  TDBOXMX maximum number of tidally active boxes.
c  XMDAT   maximum number of points in arrays for passing to external
c          models.  A loose upper bound is given by max(nx*ny*nz)
#endif
      integer ijmx,imt,imtjmt,imtkm,imtm1,imtm2,imtp1,imu,imum1,imum2,
     *        jmt,jmtm1,jmtm2,jmtp1,jscan,km,kmm1,kmp1,kmp2,lbc,lseg,
     *        misle,mprof,ndices,nkflds,nslab,nswich,nt,ntmin2,nwds
#ifdef coast
      integer mcseg,mclen
#endif
#if defined posmxtid & ( !defined mixtide | !defined ext_tide )
      integer tdboxmx
#endif
#if defined nest2larger | defined nest2smaller
      integer xmdat
#endif
      parameter(imt=131,jmt=144,ijmx=144,km=16,lseg=5,misle=4,lbc=2)
      parameter(nt=2)
      parameter(mprof=1000)
#ifdef coast
      parameter(mcseg=7,mclen=2000)
#endif
#if defined posmxtid & ( !defined mixtide | !defined ext_tide )
      parameter (tdboxmx=100)
#endif
#if defined nest2larger | defined nest2smaller
      parameter(xmdat=4071*km)
#endif
```

Fig. 4. Heavily edited-out example subset of the PE model of HOPS compile-time parameters file.

alternating lines (the intervening lines serving as titles/comments for each parameter provided). Each line (“card” in the PE model terminology) has a different number of parameters (of differing data types) and usually addresses a particular subset of the run-time parameters. Some of these parameters may constrain subsequent parameter entries (e.g. the number of entries to follow—see card 15 in Fig. 5). Others are strings that provide the filenames for reading in grid, IC and forcing fields as well as outputting run-time diagnostics, fields and checkpoints or other run-time parameter files. A glossary to help the user can be appended to the end of the file—after “card 99”—the code ignores it.

The very modular nature of these options (and their range constraints or interdependencies) can be very useful for the seasoned user of the PE model but can also exacerbate difficulties faced by a new user. When the whole HOPS system is utilized, including the other software modules for acoustical and biological modeling, skill evaluation and ESSE assimilation, error predictions and adaptive sampling (not described here), the number of possibilities and options are further increased.

3.3. MITgcm

MITgcm (MIT general circulation model) (Marshall et al., 1997a,b) is a is a *z*/*p*-coordinate finite volume numerical model designed for study of the atmosphere, ocean, and climate. It allows for a non-hydrostatic

```
1    NFIRST  NLAST  DOSTART  NNERGY  NTSOUT  NTSI  NMIX   NCON  NTDGN
       1     2304  12071.667  384     384     1     10     0      0
2    DTTS    DTUV     DTSF  (seconds)
      225     225      225
3    MIXVEL  MIXTRC  MIXZTD  (mixing scheme: momentum, tracers and vorticity)
       1       1       1
4    NORD NTIM NFRQ (momentum, tracers, vorticity, and transport)
     4 1 1   4 1 1   2 2 1   4 1 0
5    AM      AH
     1.E9    2.E7
6    AIDIF  FKPM  VVCLIM   WVMIX  FRICMX  FKPH   VDCLIM   WDMIX
      1.0    0.5   50.5    15.0    50.0   0.01    50.01    0.75
7    MLDOPT  MLDVAL   MLDMIN   MLDMAX   EKFAC   MCOEF   NCOEF  WSDFAC
       1     8.0E+2   1.0E+2   7.0E+3   0.085  0.1754  -5.182  0.0004
8    MXSCAN  SOR      CRIT      ACOR
      9000   1.0     1.0E-6     0.0
9    CDBOT
     2.5E-3
10    CDTID      MTDDPTH   TDMXFRC   TDMXFAC   SADV
     2.592E-4    36.0      0.0625    200.0    0.2
11   DVBRLX   TVBRLX   DTBRLX   TTBRLX  DCSFRC  TCSFRC  DBTFRC  TBTFRC
      4.0    -7200.0    3.0     -300.0   1.0   -10800.0  1.0    21600.0
12   (1)  (2)  (3)  (4)  (5)  (6)  (7)  (8)  (9)  (10)   IOPT(I)
      3    3    3    3    0    9    0    2    2    9
13 (PSI) (Vt) (Vi) (Vb) (Vg)  (W@V) (W@T) (w@V) (w@T)  (KE) (VOR)   IOUT(01-11)
    1     1     1   0    0     1     1     1     1      0     1
14 (T) (S) (RHO) (Buoy) (MLD) (Vtide) (Stide) (Ttide) (TrcBal) (Err) IOUT(12-21)
1   1    0     0      0      0       0       0       0       0
15   NLEV      LEV(nlev) in ascending numerical order
      16      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
16   TITLRUN (a80):  Application title.
GOM:  June 2000 ini;  June 6-7, 2001 Ass;  June 2000 FNMOC  (Run 120)
17   OUTNAME (a80): output PE fields NetCDF file name.
/home/projects/ASCOT/Itr/PE/GOM/pe_gom_PB_out.nc
18   NRGNAME (a80): output PE energy and diagnostics NetCDF file name.
/home/projects/ASCOT/Itr/PE/GOM/pe_gom_PB_nrg.nc
19   TRKNAME (a80): output Langrangian trajectories NetCDF file name, if any.
/dev/null
....<edited out> ...
39   TIDEBOX (a80): input ASCII file defining tidal regions, if applicable.
/home/projects/ASCOT/Itr/PE/GOM/tidebox_GOM.dat
99   END of input data
```

Fig. 5. Heavily edited-out example subset of the PE model of HOPS run-time parameters file.

formulation and employs cartesian, spherical polar or general orthogonal curvilinear grids in the horizontal and shaved cells in the vertical to represent the topography. MITgcm is a modular Fortran 77 code with no dynamic memory allocation and emerging full support for NetCDF-based I/O (for parallel I/O the native MDS package is recommended). Thus MITgcm binaries need to be recompiled every time depending on the model setup resolution, etc. The MITgcm is very portable; at the deeper level it employs Makefiles—GNU or system—but that level of complexity is hidden from the user through the use of a meta-make tool called ‘‘genmake2’’. In the spirit of GNU configure (Taylor, 1998), this tool automatically detects the presence and required functionality of support software and proceeds to construct a platform specific Makefile. Like ‘‘configure’’, ‘‘genmake2’’ takes further command-line arguments that allow the customization of its behavior.

At the most basic level, code configuration is carried out both at build-time and at run-time. The former is handled through the heavy use of alternative code sections selectable via C-preprocessor directives defined in the Makefile. The latter is done by switching activated code paths on through run-time switches. Thus to employ a given functionality, it is not enough to compile it in but one also needs to enable it at run-time. This approach cuts down on the necessity for recompilation. This type of software design allows for great computational modularity in MITgcm: multiple separately maintained packages with clearly defined entry and exit points in the main code path can be activated depending on the modeling needs.

The user is encouraged to work at the "genmake2" level instead of making changes to the Makefile directly. To activate a package one needs to provide it as an argument to genmake2 "-enable 'pkgname1 pkgname2 ...' "; similarly to disable a package, the "-disable" flag is provided. Modifications to the standard files in the main model or in enabled packages can be all put in subdirectories which are specified to "genmake2" using the "-mods" flag. Sanity checks for package dependencies are automatically handled by genmake2. Instead of specifying the list of packages on the command line, one can write them directly in a file called "packages.conf" (one package name per line) in a directory specified using the "mods" flag. The directory containing modifications is expected to always include a file called "SIZE.h" (see Fig. 6) where the definitions of the main array sizes used in the code are stated.

Similarly to the PE model of HOPS, other include files usually modified are "CPP_OPTIONS", where basic options about the code's use are set and package-specific options are selected and defined, e.g. "GMREDI_OPTIONS.h"—see Fig. 7, for a Gent/McWiliams/Redi SGS Eddy parameterization.

After invoking "genmake2" and "make" for several targets specified in sequence, the MITgcm binary is built. Run-time configuration of MITgcm is done entirely via namelists. MITgcm expects to find files "data" (see Fig. 8) and "data.pkg" (see Fig. 9). Depending on which packages are being used, additional files, e.g. "data.diagnostics", "data.gmredi", etc., also need to be specified. Variables can be in any order, lines can be commented out and variables not appearing are set to their default values.

As for the PE model of HOPS, the modular nature of the build-time and run-time options of the MITgcm is essential but can also exarcebate difficulties faced by new users. A web-enabled system for the user-friendly configuration and control of such legacy software is needed.

3.4. ROMS

ROMS (Haidvogel et al., 2000; Arango et al., 2000) in its latest release (2.2) of May 2005 is a free-surface hydrostatic PE ocean model; it employs stretched terrain-following coordinates in the vertical and an orthogonal curvilinear grid in the horizontal. ROMS is written in Fortran 90 (with dynamic memory allocation, a flexible choice with some possible performance implications, e.g. Ashworth et al., 2001) and NetCDF-based I/O. The more modern features of Fortran 90 allow ROMS to be more flexible in its use and do not require recompilation for slight adjustments, for example, the model resolution. On the other hand, most of the build-

```
C     Voodoo numbers controlling data layout.
C     sNx - No. X points in sub-grid.
C     sNy - No. Y points in sub-grid.
C     OLx - Overlap extent in X.
C     OLy - Overlat extent in Y.
C     nSx - No. sub-grids in X.
C     nSy - No. sub-grids in Y.
C     nPx - No. of processes to use in X.
C     nPy - No. of processes to use in Y.
C     Nx  - No. points in X for the total domain.
C     Ny  - No. points in Y for the total domain.
C     Nr  - No. points in Z for full process domain.
      INTEGER sNx, sNy, OLx, OLy, nSx, nSy, nPx, nPy, Nx, Ny, Nr
      PARAMETER (sNx =  30, sNy =  32, OLx =   3, OLy =   3, nSx =   1,
     &           nSy =   1, nPx =  12, nPy =   5, Nx  = sNx*nSx*nPx,
     &           Ny  = sNy*nSy*nPy, Nr  =  23)
C     MAX_OLX  - Set to the maximum overlap region size of any array
C     MAX_OLY    that will be exchanged. Controls the sizing of exch
C                routine buufers.
      INTEGER MAX_OLX
      INTEGER MAX_OLY
      PARAMETER ( MAX_OLX = OLx,
     &            MAX_OLY = OLy )
      integer     nobcs
      parameter ( nobcs = 4 )
```

Fig. 6. Edited-out example subset of MITgcm build-time array configuration file.

```
#ifndef GMREDI_OPTIONS_H
#define GMREDI_OPTIONS_H
#include "PACKAGES_CONFIG.h"
#include "CPP_OPTIONS.h"
#ifdef ALLOW_GMREDI
C Designed to simplify the Ajoint code:
C  exclude the clipping/tapering part of the code that is not used
#define GM_EXCLUDE_CLIPPING
#define GM_EXCLUDE_AC02_TAP
#undef  GM_EXCLUDE_TAPERING
C This allows to use Visbeck et al formulation to compute K_GM+Redi
#undef  GM_VISBECK_VARIABLE_K
C This allows the leading diagonal (top two rows) to be non-unity
C (a feature required when tapering adiabatically).
#define  GM_NON_UNITY_DIAGONAL
C Allows to use different values of K_GM and K_Redi ; also to
C be used with the advective form (Bolus velocity) of GM
#undef  GM_EXTRA_DIAGONAL
C Allows to use the advective form (Bolus velocity) of GM
C  instead of the Skew-Flux form (=default)
#undef  GM_BOLUS_ADVEC
C Following option avoids specific recomputation in adjoint
C routines of gmredi_x/y/rtransport
C It's not needed, only for tests, and very memory-consuming
#undef  GM_AUTODIFF_EXCESSIVE_STORE
#endif /* ALLOW_GMREDI */
#endif /* GMREDI_OPTIONS_H */
```

Fig. 7. Edited-out example of MITgcm build-time configuration files (for the GM/Redi package).

```
 &PARM01
 tRef= 24.0 , 23.0 , 22.0 , 21.0 , 20.0 ,
       19.0 , 18.0 , 17.0 , 16.0 , 15.0 ,
       14.0 , 13.0 , 12.0 , 11.0 , 10.0 ,
        9.0 ,  8.0 ,  7.0 ,  6.0,   5.0 ,
        4.0 ,  3.0 ,  2.0 ,
 sRef= 34.65, 34.75, 34.82, 34.87, 34.90,
       34.90, 34.86, 34.78, 34.69, 34.60,
       34.58, 34.62, 34.68, 34.72, 34.73,
       34.74, 34.73, 34.73, 34.72, 34.72,
       34.71, 34.70, 34.69,
 no_slip_sides=.false.,
 no_slip_bottom=.TRUE.,
 viscAz=1.E-3,
....<edited out> ...
 eosType='POLY3',
 readBinaryPrec=32,
....<edited out> ...
 &
 &PARM05
 bathyFile='bathy_fl.bin',
 hydrogThetaFile = 'wghc_ptmp_r2',
 hydrogSaltFile  = 'wghc_salt_r2',
 &
```

Fig. 8. Edited-out example subset of MITgcm run-time main configuration file.

time configuration of ROMS is still done using C-preprocessor directives to select alternative code blocks, as for the PE model of HOPS. These directives are specified in an include file ("cppdefs.h") that is automatically inserted in all relevant source code files. Compilation is currently controlled using GNU make, with system-specific Makefiles being automatically included depending on the platform and user-specified command-line options for the make command.

```
 &PACKAGES
 useKPP       = .TRUE.,
 useGMRedi    = .TRUE.,
 useGrdchk    = .FALSE.,
 useECCO      = .TRUE.,
# useDiagnostics = .TRUE.,
 &
```

Fig. 9. Edited-out example subset of MITgcm run-time package configuration file.

The compile-time configuration choices (see Fig. 10) cover both the setting of individual option values to true or false, or the combined settings of values according to predefined model scenarios (by turning the scenario flag on). There are more than 350 such options with complexity further increased with any interdependencies.

At the level of run-time parameters, ROMS relies on a file (piped through standard input, see Fig. 11, or given as the first command-line argument) that specifies both run-time parameter values and filenames including those of other files containing further run-time parameterization. The syntax of the ROMS parameter files is more flexible than that of the PE model of HOPS. Essentially input parameters can be entered in any order, with the parameter name followed by "=" or "==" and the parameter value. Comments in the Fortran 90

```
**-------------------------------------------------------------------------------
**  Choose a pre-defined model application.  If building a new application,
**  choose a unique CPP flag for it and select the appropriate configuration.
**-------------------------------------------------------------------------------
*/
#undef  ADRIA02         /* for Adriatic Sea Application */
#undef  BASIN           /* for Big Bad Basin Example */
#undef  BASIN_BLOB      /* for Basin Blob Example */
....<edited out, move to one of many sets of specific options> ...
# if defined MY_APPL

**-------------------------------------------------------------------------------
**  Detailed description of all available CPP options.
**-------------------------------------------------------------------------------
**
**  Select model dynamics for MOMENTUM equations:
**  (The default advection is third-order upstream bias)
*/
#undef  UV_ADV          /* turn ON or OFF advection terms */
#undef  UV_COR          /* turn ON or OFF Coriolis term */
#undef  UV_C2ADVECTION  /* turn ON or OFF 2nd-order centered advection */
#undef  UV_C4ADVECTION  /* turn ON or OFF 4th-order centered advection */
#undef  UV_SADVECTION   /* turn ON or OFF splines vertical advection */
#undef  UV_VIS2         /* turn ON or OFF Laplacian horizontal mixing */
#undef  UV_VIS4         /* turn ON or OFF biharmonic horizontal mixing */
#undef  UV_LOGDRAG      /* turn ON or OFF logarithmic bottom friction */
#undef  UV_LDRAG        /* turn ON or OFF linear bottom friction */
#undef  UV_QDRAG        /* turn ON or OFF quadratic bottom friction */
#undef  UV_PSOURCE      /* turn ON or OFF point Sources/Sinks */
....<edited out, move to a particular predefined model instance> ...
# elif defined WINDBASIN

**  Options for Wind-Driven Constant Coriolis Basin.
*/
#undef UV_ADV
#define UV_COR
#define UV_QDRAG
#define SOLVE3D
#define SPLINES
....<edited out>....
```

Fig. 10. Heavily edited example subset of ROMS build-time configuration file.

```
      TITLE = ROMS/TOMS 2.2 - Adriatic Sea Application, Adria02
    VARNAME = External/varinfo.dat
     NtileI == 1                              ! I-direction partition
     NtileJ == 8                              ! J-direction partition
     NTIMES =  387360
         DT == 60.0d0
    NDTFAST == 20
....<edited out>....
! Harmonic/biharmonic horizontal diffusion of tracer: [1:NAT+NPT,Ngrids].
       TNU2 == 5.0d0  5.0d0                   ! m2/s
       TNU4 == 2*0.0d0                        ! m4/s
....<edited out>....
! Logical switches (TRUE/FALSE) to activate writing of fields into
! HISTORY output file.
Hout(idUvel) == T                        ! 3D U-velocity
Hout(idVvel) == T                        ! 3D V-velocity
....<edited out>....
Hout(idTvar) == T T                      ! temperature and salinity
....<edited out>....
!                                1 1 1 1 1 1 1
!               1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
Hout(idBott) == T T F T T T T F T T F F F F T F
....<edited out>....
    NFFILES == 7                         ! number of forcing files
    FRCNAME == lami_wind_frc.nc \
               lami_bulk_frc.nc \
               lami_cloud_frc.nc \
               lami_swrad_frc.nc \
               tides_frc_sep20.nc \
               adria48_rivers_sed.nc \
               swan_lami_erick_lowres_frc.nc
....<edited out>....
! Input ASCII parameter filenames.
    APARNAM =  External/assimilation.in
    SPOSNAM =  External/stations_adria02.in
    FPOSNAM =  External/floats_adria02.in
    BPARNAM =  External/bioFasham.in
    SPARNAM =  External/sediment_adria02.in
    USRNAME =  External/MyFile.dat
```

Fig. 11. Heavily edited-out example subset of ROMS run-time parameters file.

mode follow the "!" symbol. Fortran syntax multiplicative assignments ("*") and unix shell continuation lines ("′") are also allowed.

As in the case of the PE model of HOPS, some of these parameters may constrain subsequent parameter entries (e.g. see "NFFILES" in Fig. 11). Others are shortcuts for "TRUE" and "FALSE" switches. Finally, specific strings provide the filenames for reading in grid, IC and forcing fields as well as output run-time diagnostics, fields and checkpoints or other run-time parameter files.

4. Legacy Computing Markup Language (LCML) schema design

Based on our requirements laid out in Section 3, an XML-based encapsulation of the configuration options and parameters of the binaries should be self-contained; it should not require any modifications to the binaries. By providing a detailed description in XML for a binary, we may treat it as a black box. A controlling application should then be able to parse in the XML description, and from its contents, determine the specifics on how to properly build and execute the binary with the appropriate input parameters.

We opted to employ XML schemata (XML schema, 2005) to constrain the vocabulary and syntax of our XML descriptions. We call the resulting XML-schema-based language the "Legacy Computing Markup Language (LCML)". Several key concerns have to be addressed and supported by LCML. A first issue is that the resulting LCML documents should provide as much useful information to the user as possible, so that well

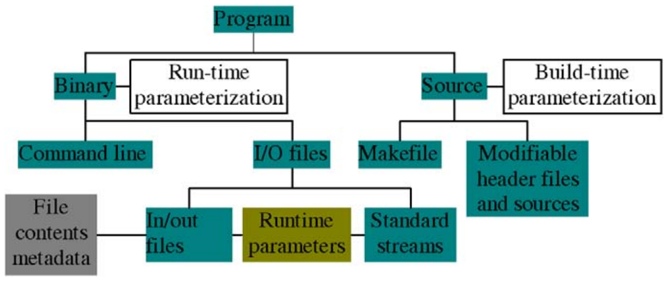


Fig. 12. Hierarchy of schemata.

informed decisions can be taken while making changes to parameters. There should be a set of default parameter values so that manual entry of all values for each compilation or execution is avoided, especially since there can be hundreds of them. The LCML descriptions should also be capable of specifying input and output files for the binary execution. Since run-time parameters should be checked after user changes, the schema must support data types and constraints or requirements on parameter values and combinations thereof. This is to facilitate the building or execution of the binary and ensure that all compile or input parameters are acceptable. Each parameter value can then be validated against its constraints and data type before proceeding. Support must also exist for more complicated data types such as variable length arrays (representing compound lines) consisting of multiple constituent data types. This was one of the more difficult problems faced in the design of LCML (Geiger, 2004).

Our hierarchical schema design, shown in Fig. 12, supports the description of the legacy program such the ocean numerical models of HOPS, MITgcm and ROMS (Section 3), from using the source to build a binary, to running a binary. In the absence of source code only the left side of the tree is available, while given source code we may have multiple binaries built with different compilation options and parameters. LCML descriptions at the source code level involve Makefile parameterizations, in terms of architecture, compiler options, include and library paths, preprocessor defines, alternative sources etc. as well as parameters (initial values as well as constants) provided in include files and sections of source files. LCML descriptions at the binary level include the names and locations of input (and output) files, as well as the run-time input parameters that are read in from stdin. Also described are command-line arguments and other run-time parameter sources (specific files with predetermined names or names provided at the command line). Most elements in the schema have parameters for name and description. These parameters are very useful for generating a GUI that provides sufficient information for the users. The initial work is described in Chang (2003) while a more detailed description of the schemata with example LCML descriptions of software components are presented in Geiger (2004).

5. Initial results

Based on the ideas outlined above, we developed a generic GUI-generator in Java (called LEGEND for LEGacy Encapsulation for Network Distribution) that can parse the LCML description of a code and present the user with a validating GUI specifically tailored to the build-time and run-time parameterization of the code. Our tool is currently a Java applet but its classes (specifically, the LCML API) could be re-used in a server-side framework employing EJB (EJB, 2005)/JSP (JSP, 2005) and portlets. The tool allows the user to customize the Makefile and source/include files in order to build a binary with the required capabilities and attributes. Through the tool, the user can then build the application, either locally or remotely, through a job script submission to a queuing system. Given a binary, the tool allows the user to customize the run-time behavior of the binary, specify input and output files and finally execute the binary, again either locally or remotely via a queuing system. In what follows, we illustrate this new tool by its application to the PE dynamical model of the HOPS system (Sections 2 and 3.2). We also briefly show an example for the ESSE system, focusing only on the ensemble uncertainty predictions, specifically on the binaries and output/input directories that usually remain constant within such predictions.

5.1. Build-time GUIs

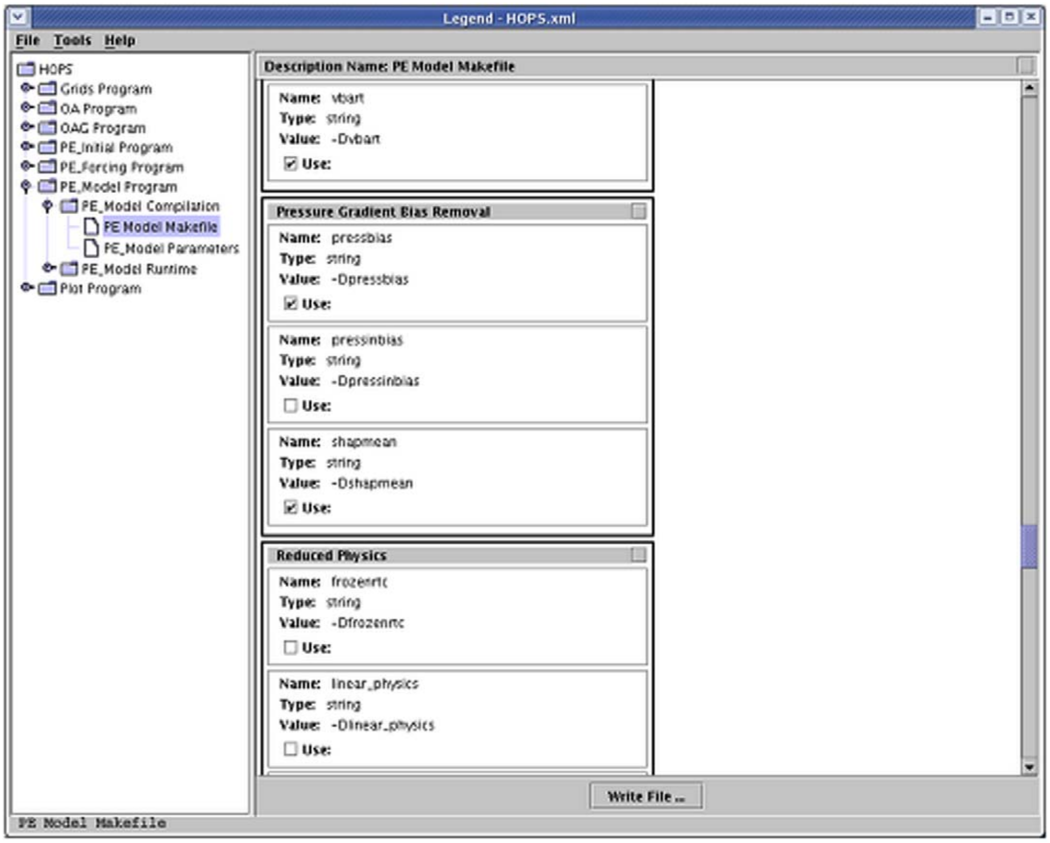
Before compiling the PE Model code, the user has to make several Makefile choices and select a PE model configuration among multiple C preprocessing options. These options are read in from the Makefile. The user can also select several compile-time constants and other parameters. We wrote an LCML description of these options and parameters. The GUIs generated for this build-time parameterization are shown in Fig. 13 for the Makefile and Fig. 14 for the include file containing compile-time constants for the PE model.

Information about preprocessor macros and values for the constants (parameters and dimensions) used at compile time are available and utilized by the GUI (Fig. 14). User interaction is either via value substitution or toggling of an ''on–off'' flag. User selections are validated according to the data types and constraints described in the LCML descriptions, taking into account interdependencies between parameters and options.

5.2. Input run-time parameter GUIs

The PE model binary has run-time parameters read in from stdin. We wrote an LCML description based on the values and types of these parameters. After validating the description using the schemata (Section 4), LEGEND processes the description and produce a GUI for the run-time parameters of the PE model, as shown in Fig. 15.

The system presents the contents of the stdin stream in an organized manner that is easily understood. Parameters are shown grouped together in the set structure imposed by the code (but can also be presented indexed by name—for more easily locating a particular one—or in table format for compactness). Their



Description Name: PE Model Makefile

Name: vbart
Type: string
Value: -Dvbart
☑ Use:

Pressure Gradient Bias Removal

Name: pressbias
Type: string
Value: -Dpressbias
☑ Use:

Name: pressinbias
Type: string
Value: -Dpressinbias
☐ Use:

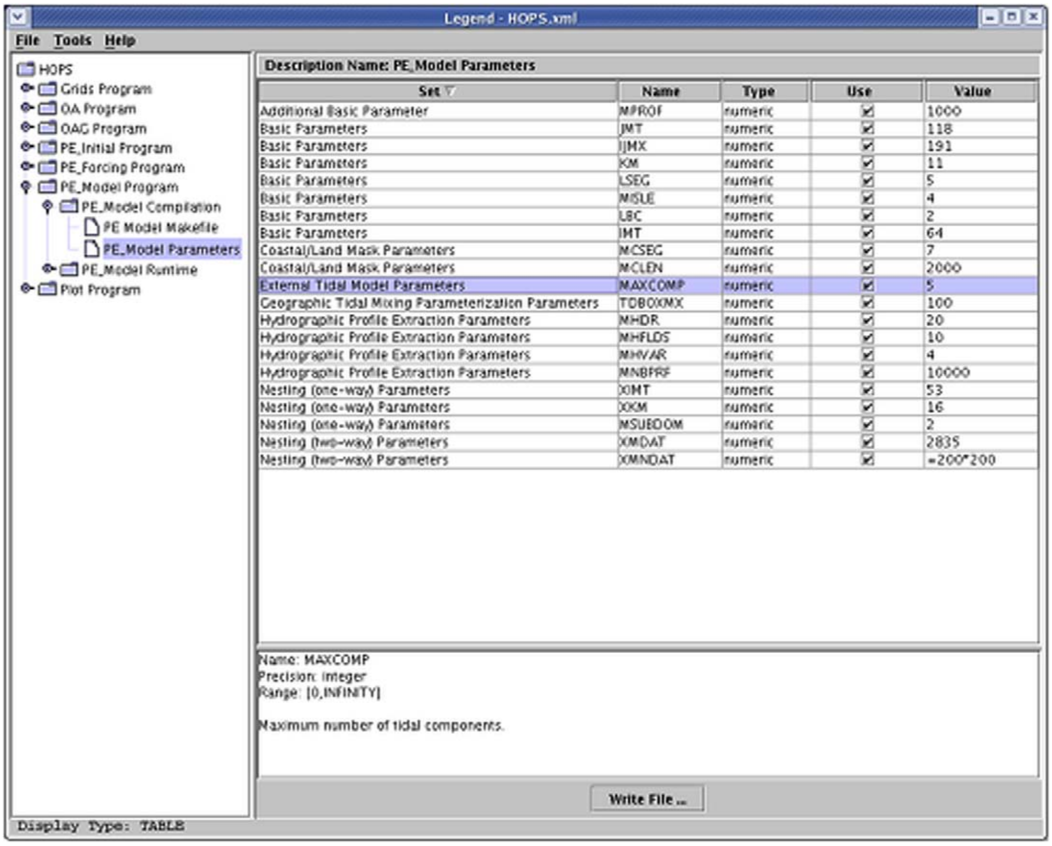
Name: shapmean
Type: string
Value: -Dshapmean
☑ Use:

Reduced Physics

Name: frozenrtc
Type: string
Value: -Dfrozenrtc
☐ Use:

Name: linear_physics
Type: string
Value: -Dlinear_physics
☐ Use:

Fig. 13. Screen capture of GUI displaying Makefile options.



Description Name: PE_Model Parameters

Set	Name	Type	Use	Value
Additional Basic Parameter	MPROF	numeric	✓	1000
Basic Parameters	JMT	numeric	✓	118
Basic Parameters	IJMX	numeric	✓	191
Basic Parameters	KM	numeric	✓	11
Basic Parameters	LSEG	numeric	✓	5
Basic Parameters	MISLE	numeric	✓	4
Basic Parameters	LBC	numeric	✓	2
Basic Parameters	IMT	numeric	✓	64
Coastal/Land Mask Parameters	MCSEG	numeric	✓	7
Coastal/Land Mask Parameters	MCLEN	numeric	✓	2000
External Tidal Model Parameters	MAXCOMP	numeric	✓	5
Geographic Tidal Mixing Parameterization Parameters	TDBOXMX	numeric	✓	100
Hydrographic Profile Extraction Parameters	MHDR	numeric	✓	20
Hydrographic Profile Extraction Parameters	MHFLDS	numeric	✓	10
Hydrographic Profile Extraction Parameters	MHVAR	numeric	✓	4
Hydrographic Profile Extraction Parameters	MNBFRF	numeric	✓	10000
Nesting (one-way) Parameters	XIMT	numeric	✓	53
Nesting (one-way) Parameters	XKM	numeric	✓	16
Nesting (one-way) Parameters	MSUBDOM	numeric	✓	2
Nesting (two-way) Parameters	XMDAT	numeric	✓	2835
Nesting (two-way) Parameters	XMNDAT	numeric	✓	=200*200

```
Name: MAXCOMP
Precision: integer
Range: [0,INFINITY]

Maximum number of tidal components.
```

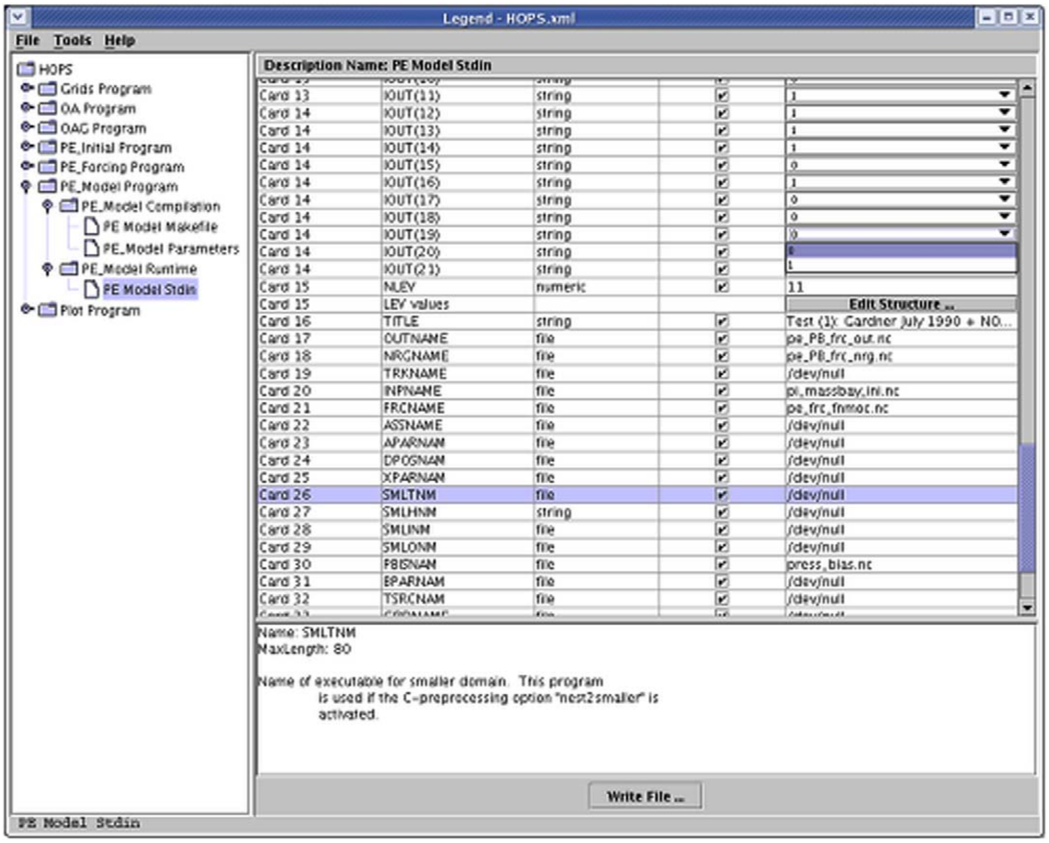
Fig. 14. Screen capture of GUI displaying compile-time parameters.

current value and data type is shown and additional information regarding their meaning and use is available in the bottom frame. Beyond the default values provided for all parameters (as required by the schema), importing and exporting of values allows a user to re-use the parameters of previous simulations, thereby lessening one's workload. This way, instead of editing the input file, the user updates parameter values in the GUI directly. Changes in the GUI are then checked for validity (type, range and conflicts with other parameters—see Fig. 16) before the system generates the new stdin input file and execution script automatically. An example of such automatically created stdin files was illustrated in Fig. 5.

Importantly, with our build-time and run-time GUIs, the user does not see any of the intricate details of the input files (e.g. Figs. 5, 8 and 11). The user updates all parameters and submits all requests in a modern and friendly web-based local environment. The required files are then created in the background. Should developers or advanced users modify the ocean model and its input characteristics, corresponding incremental changes to the LCML description would still allow LEGEND to automatically generate the appropriate GUI.

5.3. ESSE GUI for ensemble uncertainty prediction

To illustrate the application of LEGEND to a modern data assimilation system, we focus on the uncertainty prediction module of the ESSE system, specifically on the corresponding workflows. Such predictions currently involve a C-shell which manages the perturbation of the initial conditions and ensemble of stochastic PE model runs, including the successive computations of the singular value decomposition of the ensemble spread until a criterion estimating convergence is satisfied. The C-shell workflow contains different types of variables, including limit values, constant variables and evolving variables. In Fig. 17, we illustrate a GUI for the constant variables involved in such workflows, including the: (i) binaries, from the codes which carry



Description Name: PE Model Stdin

Card 13	IOUT(11)	string	✓	1
Card 14	IOUT(12)	string	✓	1
Card 14	IOUT(13)	string	✓	1
Card 14	IOUT(14)	string	✓	1
Card 14	IOUT(15)	string	✓	0
Card 14	IOUT(16)	string	✓	1
Card 14	IOUT(17)	string	✓	0
Card 14	IOUT(18)	string	✓	0
Card 14	IOUT(19)	string	✓	0
Card 14	IOUT(20)	string	✓	0
Card 14	IOUT(21)	string	✓	1
Card 15	NLEV	numeric	✓	11
Card 15	LEV values			Edit Structure ...
Card 16	TITLE	string	✓	Test (1): Gardner July 1990 + NO...
Card 17	OUTNAME	file	✓	pe_PB_frc_out.nc
Card 18	NRGNAME	file	✓	pe_PB_frc_nrg.nc
Card 19	TRKNAME	file	✓	/dev/null
Card 20	INPNAME	file	✓	pi_massbay_ini.nc
Card 21	FRCNAME	file	✓	pe_frc_fnmoc.nc
Card 22	ASSNAME	file	✓	/dev/null
Card 23	APARNAM	file	✓	/dev/null
Card 24	DPOSNAM	file	✓	/dev/null
Card 25	XPARNAM	file	✓	/dev/null
Card 26	SMLTNM	file	✓	/dev/null
Card 27	SMLHNM	string	✓	/dev/null
Card 28	SMLINM	file	✓	/dev/null
Card 29	SMLONM	file	✓	/dev/null
Card 30	PBISNAN	file	✓	press_bias.nc
Card 31	BPARNAM	file	✓	/dev/null
Card 32	TSRCNAM	file	✓	/dev/null

```
Name: SMLTNM
MaxLength: 80

Name of executable for smaller domain.  This program
        is used if the C-preprocessing option "nest2smaller" is
        activated.
```

Fig. 15. Screen capture of GUI displaying run-time parameters.

out checks for physically admissible perturbations to the stochastic PE model code, (ii) output/input directories that usually remain constant, and (iii) management variables.

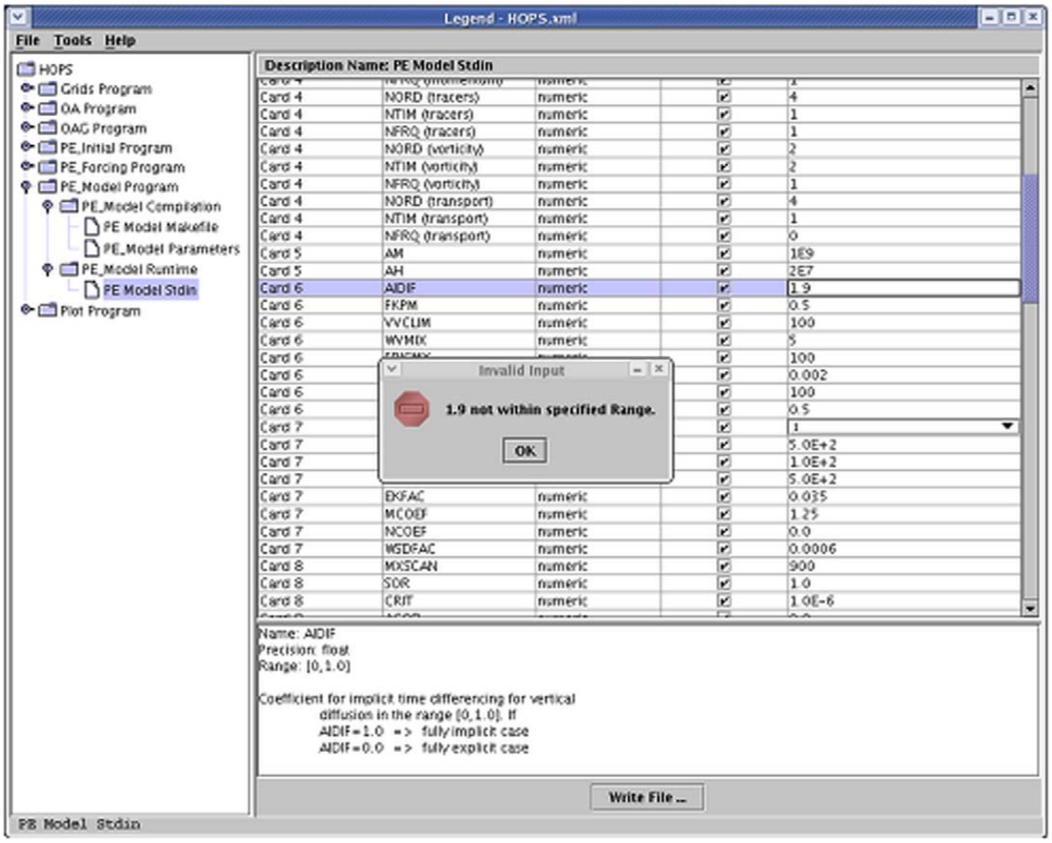
5.4. LCML issues

The LCML descriptions of the binaries are meant to be written once and only revised/enlarged when configuring the build- or run-time behavior of the code changes. Code enhancements that do not affect the user-interaction thus do not warrant any changes in the LCML descriptions. Developers need only to carry out incremental changes to the LCML code descriptions. The latter, as always in the case of XML, are rather verbose files (see Fig. 18 for an example) but are not very difficult to write with an XML or specialized LCML editor (Geiger, 2004).

6. Related work

Among the earliest work done to encapsulate legacy binaries for use over the web was Javamatic (Phanouriou and Abrams, 1997). Their system required the user to write Java classes to describe the application or attempt to graphically compose derived/extended Javamatic classes. The application description was not separate from the GUI look and feel.

The bioinformatics community has had a great need for simplified access to command-line driven genomics tools and several efforts have attempted to meet that need: W2H (Senger et al., 1998) and AppLab (Senger, 1999) paved the way, followed by PISE (Letondal, 2001) and GenePattern (Reich and the GenePattern team, 2005). All of the latter tools were XML-based but followed significantly different implementation



Description Name: PE Model Stdin

Card 4	NORD (tracers)	numeric	✓	4
Card 4	NTIM (tracers)	numeric	✓	1
Card 4	NFRQ (tracers)	numeric	✓	1
Card 4	NORD (vorticity)	numeric	✓	2
Card 4	NTIM (vorticity)	numeric	✓	2
Card 4	NFRQ (vorticity)	numeric	✓	1
Card 4	NORD (transport)	numeric	✓	4
Card 4	NTIM (transport)	numeric	✓	1
Card 4	NFRQ (transport)	numeric	✓	0
Card 5	AM	numeric	✓	1E9
Card 5	AH	numeric	✓	2E7
Card 6	AIDIF	numeric	✓	1.9
Card 6	FKPM	numeric	✓	0.5
Card 6	VVCLIM	numeric	✓	100
Card 6	WVMIX	numeric	✓	5
Card 6			✓	100
Card 6			✓	0.002
Card 6			✓	100
Card 6			✓	0.5
Card 7			✓	1
Card 7			✓	5.0E+2
Card 7			✓	1.0E+2
Card 7			✓	5.0E+2
Card 7	EKFAC	numeric	✓	0.035
Card 7	MCOEF	numeric	✓	1.25
Card 7	NCOEF	numeric	✓	0.0
Card 7	WSDFAC	numeric	✓	0.0006
Card 8	MXSCAN	numeric	✓	900
Card 8	SOR	numeric	✓	1.0
Card 8	CRIT	numeric	✓	1.0E-6

Invalid Input

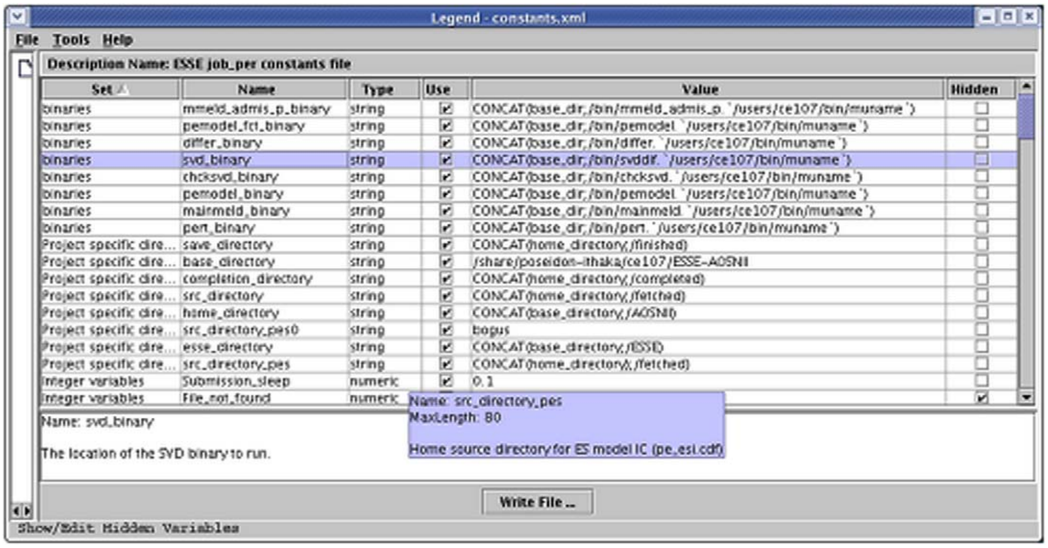
1.9 not within specified Range.

OK

Name: AIDIF
Precision: float
Range: [0,1.0]

Coefficient for implicit time differencing for vertical
diffusion in the range [0,1.0]. If
AIDIF=1.0 => fully implicit case
AIDIF=0.0 => fully explicit case

Fig. 16. Screen capture of GUI validating user input.



Description Name: ESSE job_per constants file

Set	Name	Type	Use	Value	Hidden
binaries	mmeld_admis_p_binary	string	✓	CONCAT(base_dir,/bin/mmeld_admis_p.`/users/ce107/bin/muname`)	☐
binaries	pemodel_fct_binary	string	✓	CONCAT(base_dir,/bin/pemodel.`/users/ce107/bin/muname`)	☐
binaries	differ_binary	string	✓	CONCAT(base_dir,/bin/differ.`/users/ce107/bin/muname`)	☐
binaries	svd_binary	string	✓	CONCAT(base_dir,/bin/svddif.`/users/ce107/bin/muname`)	☐
binaries	chcksvd_binary	string	✓	CONCAT(base_dir,/bin/chcksvd.`/users/ce107/bin/muname`)	☐
binaries	pemodel_binary	string	✓	CONCAT(base_dir,/bin/pemodel.`/users/ce107/bin/muname`)	☐
binaries	mainmeld_binary	string	✓	CONCAT(base_dir,/bin/mainmeld.`/users/ce107/bin/muname`)	☐
binaries	pert_binary	string	✓	CONCAT(base_dir,/bin/pert.`/users/ce107/bin/muname`)	☐
Project specific dire...	save_directory	string	✓	CONCAT(home_directory,/finished)	☐
Project specific dire...	base_directory	string	✓	/share/poseidon-ithaka/ce107/ESSE-AOSNII	☐
Project specific dire...	completion_directory	string	✓	CONCAT(home_directory,/completed)	☐
Project specific dire...	src_directory	string	✓	CONCAT(home_directory,/fetched)	☐
Project specific dire...	home_directory	string	✓	CONCAT(base_directory,/AOSNII)	☐
Project specific dire...	src_directory_pes0	string	✓	bogus	☐
Project specific dire...	esse_directory	string	✓	CONCAT(base_directory,/ESSE)	☐
Project specific dire...	src_directory_pes	string	✓	CONCAT(home_directory,/fetched)	☐
Integer variables	Submission_sleep	numeric	✓	0.1	☐
Integer variables	File_not_found	numeric			✓

Name: src_directory_pes
MaxLength: 80

Home source directory for ES model IC (pe_esl.cdf)

Name: svd_binary

The location of the SVD binary to run.

Fig. 17. Screen capture of a GUI for variables that usually remain constant within the uncertainty prediction module of the ESSE system.

philosophies: CORBA is used for AppLab, Perl for PISE and Java/JNI for GenePattern. The emphasis is on command-line arguments and, quite significantly, workflows. There is no support for configuring the building

```
<set>
  <setName>Card 6</setName>
  <setInfo>Card 6</setInfo>
  <var>
    <name>CLASS1</name>
    <info>Field ID classification to use in primary NetCDF file:
            [1] PE classification.
            [2] QG classification.</info>
    <type>string</type>
    <value>1</value>
    <use>true</use>
    <hidden>false</hidden>
    <enumeration>1;2</enumeration>
  </var>
</set>
  <setName>Card 7</setName>
  <setInfo>Card 7</setInfo>
  <var>
    <name>NFIELDS</name>
    <info>Number of fields to plot.</info>
    <type>numeric</type>
    <value>1</value>
    <use>true</use>
    <hidden>false</hidden>
    <precision>integer</precision>
    <range>[0,1000]</range>
  </var>
</set>
```

Fig. 18. Heavily edited-out example LCML description.

of an application. Of the systems mentioned, PISE has the most common features with LCML/LEGEND in that it supports the concepts of dependencies and validity ranges. The web services/grid computing community has placed a lot of emphasis on distributed workflow execution. Grid workflow solutions like Keppler (Altintas et al., 2005), Karajan (Hategan et al., 2004), GRB (Aloisio and Cafaro, 2002) and Gridbus (Buyya and Venugopal, 2004) provide some limited access to command-line configuration for the individual workflow components. Once again the underlying description language is XML but the GUI deals with the workflow aspect.

More detailed configuration of Grid-enabled applications has been provided by the emergence of more involved GUI-generation tools such as GUIGen (Reinerfeld et al., 2002) and Gridspeed (Suzumura et al., 2004). Handling of run-time configuration of input files is done via parameter substitution in template files specified in some scripting or programming language (GUIGen) or in a Java-based template engine (Apache velocity—gridspeed).

In the engineering community some similar abstraction efforts have been made: e.g. XGui for the ICE project (Clarke and Namburu, 2002) and MAUI (2002) with the latter being far more generic. In both cases the GUI presentation is part of the application description, requiring more work at the level of writing the application description.

Finally, in the area of environmental applications, efforts have arisen in application specific platforms such as WRFIS/GUI (McCaslin et al., 2004). More general approaches in the context of a grid computing infrastructure can be seen in DMEFS (Haupt et al., 2002) and ECMWF's (PrepIFS, 2005). Both are XML-based and provide for configuring the run-time and (in the case of the latter) build-time options of command-line driven binaries. Moreover they form very complete environments for the remote control of applications, from configuration through execution to postprocessing.

LCML/LEGEND is largely orthogonal to any of the workflow-oriented work mentioned above. In fact it was purposefully designed to allow coupling to more than one workflow solution. The strength of our work lies in that LCML provides for a generic approach to the abstraction of an application's configuration, without bothering the application developer with specifying GUI presentation details, but providing for validation of the input values through constraints and dependencies. Our system handles not only run-time options and parameters but also build-time ones. Our LCML customizations go beyond command-line arguments and the simple value-pair substitutions in template files and allow for varying length multi-type arrays and other complex objects.

7. Conclusion and future research

We developed a novel XML-schema-based language (LCML) and implemented new software (LEGEND) for the web-enabled configuration and control of "legacy" codes and applied them to components of an interdisciplinary ocean modeling and data assimilation system. Schema-validated LCML descriptions are implemented to provide a machine (and human) readable standard for managing the tasks involved in carrying out research and operations with such complex computational command-line-driven codes. All options and parameterizations of the binaries are described using LCML, and LEGEND automatically generates validating GUIs to handle the interactions with the user.

By wrapping the codes using LCML, their build-time and compile-time parameterizations, as well as their input and output files, stdin and command-line-provided run-time parameters are described. A prototype system was implemented as a Java applet and graphical interfaces displaying user-configurable parameters were generated, based on the LCML description. The GUI allows for user customization of parameters and validates user changes. Include files, source files and Makefiles are modified and input and script files for the compilation and execution of binaries, respectively, are produced.

LCML and LEGEND are available under an MIT open source license from http://www.sf.net/lcml and http://www.sf.net/legend-lcml. We expect LCML to slowly evolve to deal with cases where it may be discovered that it cannot adequately describe the particulars of a specific code. LEGEND has the potential of being enhanced to have direct Grid connectivity or its code may be re-used in a new more powerful LCML-parsing GUI generator.

Current research includes completing production-quality LCML descriptions for the whole HOPS and ESSE systems, and for ROMS (Haidvogel et al., 2000) and MITgcm (Marshall et al., 1997a,b). Our approach can also be extended to cover more codes from the ocean forecasting/simulation and data assimilation community. In the future, as our new LCML and LEGEND are applied to other models, our schema and software may need to be further enhanced to handle unusual structures of input files, etc. While these structures are only limited by the imagination of ocean model developers, we expect that over time any necessary changes to LCML will be ever more uncommon and incremental.

Other usability extensions involve style sheets allowing for a more flexible presentation of the GUI and supports for units. For example, this could allow a user to provide parameter values in the units she/he is more accustomed to. The software would then automatically transform units and validate input values against the allowed values in the units the program uses.

We have designed our system in expectation of being able to deal with codes that employ other build-time systems, for example cmake (2004) or others. As any such systems are themselves either legacy-binary or configuration file driven, we expect that the same approach will be easily used to cover them.

Going beyond the current capabilities of our software tools which drive the building and execution of ocean model binaries through submission of jobs to queuing systems, we have started to integrate the full system within a Grid infrastructure using the Globus toolkit (Globus, 2005). So instead of LEGEND working with local files, it would create files and execute scripts on Globus-enabled remote platforms. That would enable the remote use of LEGEND from essentially any Internet connected machine. A server-centric approach to the GUI generation that uses JSP/portlets instead of an applet is being investigated in relation to integration within a Grid Portal. Another direction of interest is integrating our applet in the ARION system (Houstis et al., 2002), allowing for a generic applet to drive remote binaries. Further work could include the introduction of the concept of virtual data files, a capability that is already partially in place in our tools with support for pre and poststaging of I/O files. Just as XML metadata about building and running a legacy binary are central to our system, metadata about data file contents and locations (of replicas, etc.) are going to be important for a future system that would support a fully virtual data file that is described by information about its contents (or instructions about how it can be generated dynamically) instead of a name and a location.

Currently, our system workflows are predefined and our software tools serve to prepare the individual components of these workflows prior to their utilization. In the long term, one can envision other scenarios involving legacy binaries where workflows are flexible and composed interactively by the user or dynamically and automatically by an expert system. A full, machine-readable description of the parameters in, and other attributes of, input and output files used to connect workflow components is a prerequisite for a fully flexible system allowing the validated composition of such workflows. This would extend our approach by further integrating data file metadata and requiring descriptions for output files as well. Going beyond validation, mismatched component connections can sometimes be handled by mediating agents, the requirements for which can be generated from the available XML binary and data file descriptions. Given these requirements, a (relatively simple) mediating agent could even be automatically composed from a set of predefined components.

Acknowledgments

The authors would like to acknowledge the assistance of Dr. P.J. Haley and Professors J.J. McCarthy, A.R. Robinson and H. Schmidt. Discussions with W. Leslie, S. Lalis and R. Tian were also appreciated. We thank Prof. Killworth and two anonymous reviewers for a rapid and useful review. This work was funded under the support of Dr. F. Darema from NSF/ITR (under Grant EIA-0121263) and in part from DoC (NOAA via MIT Sea Grant, under Grant NA86RG0074) and from ONR for PFJL (under Grant N00014-05-1-0335, N00014-05-1-0370, N00014-01-1-0771 and N00014-97-1-0239).

References

Aloisio, G., Cafaro, M., 2002. Web-based access to the Grid using the Grid Resource Broker portal. Concurr. Comput.: Pract. Exper. 14, 1145–1160.

Altintas, I., Birnbaum, A., Baldridge, K., Sudholt, W., Miller, M., Amoreira, C., Potier, Y., Ludaescher, B., 2005. A framework for the design and reuse of grid workflows. In: International Workshop on Scientific Applications on grid computing (SAG'04)LNCS, vol. 3458. Springer.

Arango, H.G., 2001. A community terrain-following ocean modelling system. Available from: <http://marine.rutgers.edu/cool/coolresults/ams2001/index.html/TOMS.ppt>.

Arango, H.G. et al., 2000. ROMS. The regional ocean modeling system. Available from: <http://marine.rutgers.edu/po/models/roms/index.php>.

Argent, R.M., 2004. An overview of model integration for environmental applications—components, frameworks and semantics. Environ. Modell. Software 19 (3), 219–234.

Ashworth, M., Emerson, D.R., Sandham, N.D., Yao, Y., Li, Q., 2001. Parallel DNS using a compressible turbulent channel flow benchmark. In: ECCOMAS Computational Fluid Dynamics Conference 2001, Swansea, Wales, UK, 4–7 September.

Berntsen, J., Svendsen, E., 1999. Using the SKAGEX dataset for evaluation of ocean model skills. J. Mar. Syst. 18 (4), 313–331, for model information: http://www.mi.uib.no/BOM/.

Bryan, K., Cox, M.D., 1967. A numerical investigation of the oceanic general circulation. Tellus 19 (1), 54–80.

Bubak, M., Kurzyniec, D., Luszczek, P., 2001. Convenient use of legacy software in Java with Janet package. Future Gener. Comput. Syst. 17, 987–997.

Burchard, H., Bolding, K., 2000. GOTM—a public-domain-model for the water column. Available from: <http://www.gotm.net>.

Buyya, R., Venugopal, S., 2004. The gridbus toolkit for service oriented grid and utility computing: an overview and status report. In: 1st IEEE International Workshop on Grid Economics and Business Models (GECON 2004, April 23, 2004, Seoul, Korea). IEEE Press, New Jersey, USA, ISBN 0-7803-8525-X, pp. 19–36.

CCA Forum, 2004. Common Component Architecture. Available from: <http://www.cca-forum.org/>.

Chang, R.C., 2003. The Encapsulation of Legacy Binaries using an XML-Based Approach with Applications in Ocean Forecasting. M.Eng. in Electrical Engineering and Computer Science thesis, Massachusetts Institute of Technology.

Chassignet, E.P., Malanotte-Rizzoli, P., 2000. Ocean circulation model evaluation experiments for the North Atlantic basin, Elsevier Science, special DAMEE issue, Dyn. Atmos. Oceans 32, 155–432, for model information: http://oceanmodeling.rsmas.miami.edu/micom/.

Clarke, J.A., Namburu, R.R., 2002. A distributed computing environment for interdisciplinary applications. Concurr. Comput.: Pract. Exper. 14, 1161–1174.

Cross-Platform Make, 2004. Available from: <http://www.cmake.org/>.

DAGman developers in the Condor Group, 2005. The directed acyclic graph manager (DAGMan). Available from: <http://www.cs.wisc.edu/condor/dagman>.

De Szoeke, R.A., Springer, S.R., Oxilia, D.M., 2000. Orthobaric density: a thermodynamic variable for ocean circulation studies. J. Phys. Oceanogr. 30 (11), 2830–2852, for model information: http://posum.oce.orst.edu/.

Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M.-H., Vahi, K., Livny, M., 2004. Pegasus: mapping scientific workflows onto the gridLNCS, vol. 3165, pp. 11–20.

Deleersnijder, E., Campin, J.-M., Delhez, E.J.M., 2001. The concept of age in marine modelling: I. Theory and preliminary model results. J. Mar. Syst. 28 (3–4), 229–267.

Dutay, J.-C., Bullister, J.L., Doney, S.C., Orr, J.C., Najjar, R., Caldeira, K., Campin, J.-M., Drange, H., Follows, M., Gao, Y., 2002. Evaluation of ocean model ventilation with CFC-11: comparison of 13 global ocean models. Ocean Modell. 4 (2), 89–120, TY—JOUR, http://www.ipsl.jussieu.fr/OCMIP/phase2/poster/dutay.CFC.pdf.

EdGCM, 2005. Available from: <http://www.edgcm.org>.

Enterprise JavaBeans Technology, 2005. Available from: <http://java.sun.com/products/ejb/>.

Ezer, T. et al., 2000. The Princeton Ocean Model, Program in Atmospheric and Oceanic Sciences, P.O. Box CN710, Sayre Hall, Princeton University, Princeton, NJ 08544-0710. Available from: <http://www.aos.princeton.edu/WWWPUBLIC/htdocs.pom/>.

Fatoohi, R., Gokhale, N., Viswesan, S., 2005. iJob: an Internet-based job execution environment using asynchronous messaging. Inform. Software Technol. 47 (8), 565–574.

Foster, I., Kesselman, C. (Eds.), 2003. The Grid 2: Blueprint for a New Computing Infrastructure, second ed. Morgan Kaufmann.

Frickenhaus, S., Hiller, W., Best, M., 2005. FoSSI: the family of simplified solver interfaces for the rapid development of parallel numerical atmosphere and ocean models. Ocean Modell. 10 (1–2), 185–191.

Geiger, S.K., 2004. LEGacy Encapsulation for Network Distribution. S.M. in Ocean Engineering thesis, Massachusetts Institute of Technology.

Gent, P.R., Bryan, F.O., Danabasoglu, G., Doney, S.C., Holland, W.R., Large, W.G., McWilliams, J.C., 1998. The NCAR Climate System Model global ocean component. J. Climate 11 (6), 1287–1306, for model information: http://www.cgd.ucar.edu/csm/models/ocn-ncom/.

Globus Alliance, 2005. The Globus Toolkit. Available from: <http://www.globus.org/toolkit/>.

Griffies, S., 2004. Fundamentals of Ocean Climate Models. Princeton University Press, 496 pp.

Haidvogel, D.B., Beckmann, A., 1999. Numerical Ocean Circulation Modeling. World Scientific Pub. Co., ISBN 1860941141, 318 pp.

Haidvogel, D.B., Arango, H.G., Hedstrom, K., Beckmann, A., Malanotte-Rizzoli, P., Shchepetkin, A.F., 2000. Model evaluation experiments in the North Atlantic Basin: simulations in nonlinear terrain-following coordinates. Dyn. Atmos. Oceans 32 (3–4), 239–281, for model information: http://marine.rutgers.edu/po/models/.

Halliwell, G., Bleck, R., Chassignet, E.P., Smith, L., 2001. Evaluation of the HYbrid Coordinate Ocean Model (HYCOM) Using Atlantic Ocean Simulations. Available from: <http://hycom.rsmas.miami.edu/halliwell_01/slides_files/v3_document.htm>.

Hategan, M., von Laszewski, G., Amin, K., 2004. Karajan: a grid orchestration framework. Supercomputing 2004, Pittsburgh, 6–12 November (Refereed Poster). Available from: <http://www.sc-conference.org/sc2004>.

Haupt, T., Bangalore, P., Henley, G., 2002. Mississippi computational portal. Concurr. Comput.: Pract. Exper. 14, 1275–1287.

Houstis, C., Lalis, S., Christophides, V., Plexousakis, D., Vavalis, E., Pitikakis, M., Kritikos, K., Smardas, A., Gikas, C., 2002. A service infrastructure for e-Science: the case of the ARION system. In: Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE 2002), E-Services and the Semantic Web workshop (WES2002), Toronto, Canada, LNCS, vol. 2512, Springer, pp. 175–187.

Hurley, P.J., Physick, W.L., Luhar, A.K., 2005. TAPM: a practical approach to prognostic meteorological and air pollution modeling. Environ. Modell. Software 20 (6), 737–752.

Java Native Interface, 2004. Available from: <http://java.sun.com/j2se/1.5.0/docs/guide/jni/>.

JavaServer Technology, 2005. Available from: <http://java.sun.com/products/jsp/>.

Kantha, L.H., Clayson, C.A., 2000. Numerical Models of Oceans and Oceanic Processes. Academic Press, 940 pp.

Killworth, P.D., Li, J.-G., Smeed, D.A., 2003. On the efficiency of statistical assimilation techniques in the presence of model and data error. J. Geophys. Res. 108 (C4), 3113.

Lermusiaux, P.F.J., 2002. On the mapping of multivariate geophysical fields: sensitivity to size, scales and dynamics. J. Atmos. Oceanic Technol. 19, 1602–1637.

Lermusiaux, P.F.J., Robinson, A.R., 1999. Data assimilation via error subspace statistical estimation. Part I: Theory and schemes. Month. Weather Rev. 127, 1385–1407.

Lermusiaux, P.F.J., Anderson, D.G.M., Lozano, C.J., 2000. On the mapping of multivariate geophysical fields: error and variability subspace estimates. Q.J.R. Meteorol. Soc., 1387–1430.

Lermusiaux, P.F.J., Robinson, A.R., Haley Jr., P.J., Leslie, W.G., 2002. Advanced interdisciplinary data assimilation: filtering and smoothing via error subspace statistical estimation. In: Proceedings of the OCEANS 2002. MTS/IEEE, Holland Publications, pp. 795–802.

Lermusiaux, P.F.J., Evangelinos, C., Tian, R., Haley Jr., P.J., McCarthy, J.J., Patrikalakis, N.M., Robinson, A.R., Schmidt, H., 2004. Adaptive coupled physical and biogeochemical ocean predictions: a conceptual basis. In: Darema, F (Ed.), Computational Science—ICCS 2004, LNCS, vol. 3038, pp. 685–692.

Letondal, C., 2001. A web interface generator for molecular biology programs in Unix. Bioinformatics 17 (1), 73–82.

Lozano, C.J., Robinson, A.R., Arango, H.G., Gangopadhyay, A., Sloan, Q., Haley Jr., P.J., Anderson, L., Leslie, W.G., 1996. An interdisciplinary ocean prediction system: assimilation strategies and structured data models. In: Malanotte-Rizzoli, P. (Ed.), Modern Approaches to Data Assimilation in Ocean Modeling. Elsevier Science, pp. 413–452.

Lynch, D.R., Davies, A.M., 1995. Quantitative skill assessment for coastal ocean models. In: Lynch, D.R., Davies, A.M. (Eds.), Coastal and Estuarine Studies, vol. 47. American Geophysical Union, Washington, DC, pp. 153–174.

Lynch, D.R., Naimie, C.E, Ip, J.T., Lewis, C.V., Werner, F.E., Luettich, R., Blanton, B.O., Quinlan, J., McGillicuddy Jr., D.J., Ledwell, J.R., Churchill, J., Kosnyrev, V., Davis, C.S., Gallager, S.M., Ashjian, C.J., Lough, R.G., Manning, J., Flagg, C.N., Hannah, C.G., Groman, R.C., 2001. Real-time data assimilative modeling on Georges Bank. Oceanography 14 (1), 65–77. Available from: <http://www-nml.dartmouth.edu/Publications/external_publications/PUB-99-2/>.

Marshall, J., Hill, C.N., Perelman, L., Adcroft, A., 1997a. Hydrostatic, quasi-hydrostatic, and nonhydrostatic ocean modeling. J. Geophys. Res. 102 (C3), 5733–5752, for model information: http://mitgcm.org/.

Marshall, J., Adcroft, A., Hill, C.N., Perelman, L., Heisey, C., 1997b. A finite-volume, incompressible Navier Stokes model for studies of the ocean on parallel computers. J. Geophys. Res. 102 (C3), 5753–5766.

MAUI: rapidly developing a graphical user interface (GUI) for an application, 2002. Available from: <http://csmr.ca.sandia.gov/projects/maui/index.php>.

McCaslin, P.T., Smart, J.R., Shaw, B., Jamison, B.D., 2004. Graphical user interface to prepare the standard initialization for WRF. In: 20th Conference on Weather Analysis and Forecasting/16th Conference on Numerical Weather Prediction, 84th AMS Annual Meeting 11–15 January.

Mooers, C.N.K. (Ed.), 1999. Coastal Ocean Prediction. AGU Coastal and Estuarine Studies Series. American Geophysical Union, Washington, DC, 523 pp.

Network Common Data Format, 2005. Available from: <http://my.unidata.ucar.edu/content/software/netcdf/index.html>.

Open Management Group, 2005. Common Object Request Broker Architecture. Available from: <http://www.corba.org/>.

Onken, R., Robinson, A.R., Lermusiaux, P.F.J., Haley Jr., P.J., Anderson, L.A., 2004. Data-driven simulations of synoptic circulation and transports in the Tunisia–Sardinia–Sicily region. J. Geophys. Res. 108 (C9), 8123–8136.

Pacanowski, R.C., Griffies, S.M., 2000. Modular Ocean Model 3.0 Manual, GFDL. Available from: <http://www.gfdl.gov/smg/MOM/web/guide_parent/guide_parent.html>.

Patrikalakis, N.M., 2005. Poseidon: a distributed information system for ocean processes. Available from: <http://czms.mit.edu/poseidon/>.

Patrikalakis, N.M., Abrams, S.L., Bellingham, J.G., Cho, W., Mihanetzis, K.P., Robinson, A.R., Schmidt, H., Wariyapola, P.C.H., 2000. The digital ocean. In: Proceedings of Computer Graphics International, GCI '2000, Geneva, Switzerland, June. IEEE Computer Society Press, Los Alamitos, CA, pp. 45–53.

Phanouriou, C., Abrams, M., 1997. Transforming command-line driven systems to Web applications. Comput. Networks ISDN Syst. 29, 1497–1505.

Pietrzak, J., Deleersnijder, E., Schrter, J., 2005. The second international workshop on unstructured mesh numerical modelling of coastal, shelf and ocean flows, Delft, The Netherlands, September 23–September 25. Ocean Modell. 10 (1–2), 1–3.

Pinardi, N., Woods, J.D., 2002. Ocean Forecasting: Conceptual Basis and Applications. Springer-Verlag, Berlin.

PrepIFS, 2005. Available from: <http://www.ecmwf.int/services/prepifs>.

Reich, M., the GenePattern team, 2005. GenePattern. Available from: <http://www.broad.mit.edu/cancer/software/genepattern.

Reinerfeld, A., Stüben, H., Schintke, F., Din, G., 2002. GUIGen: a toolset for creating customized interfaces for grid user communities. Future Gener. Comput. Syst. 18, 1075–1084.

Robinson, A.R., 1999. Forecasting and simulating coastal ocean processes and variabilities with the Harvard ocean prediction system. In: Mooers, C.N.K. (Ed.), Coastal Ocean Prediction, AGU Coastal and Estuarine Studies Series. American Geophysical Union, pp. 77–100.

Robinson, A.R., Lermusiaux, P.F.J., 2004. Prediction systems with data assimilation for coupled ocean science and ocean acoustics. In: Tolstoy, A. et al. (Eds.), Proceedings of the Sixth International Conference on Theoretical and Computational Acoustics. World Scientific Publishing, pp. 325–342.

Robinson, A.R., the LOOPS Group, 1999. Realtime forecasting of the multidisciplinary coastal ocean with the littoral ocean observing and predicting system (LOOPS). In: Third Conference on Coastal Atmospheric and Oceanic Prediction and Processes, New Orleans, LA, 3–5 November, American Meteorological Society, pp. 30–35. Available from: <http://www.deas.harvard.edu/robinson/PAPERS/AMS_NO.html>, for model information: http://www.deas.harvard.edu/robinson/activities.html.

Robinson, A.R., Lermusiaux, P.F.J., Sloan III, N.Q., 1998. Data assimilation. In: Brink, K.H., Robinson, A.R. (Eds.), The Sea: The Global Coastal Ocean I: Processes and Methods, vol. 10. John Wiley and Sons, New York, NY, pp. 541–594.

Robinson, A.R., Lermusiaux, P.F.J., Haley Jr., P.J., Leslie, W.G., 2002. Predictive skill, predictive capability and predictability in ocean forecasting. In: Proceedings of The OCEANS 2002 MTS/IEEE Conference. Holland Publications, pp. 787–794.

Robinson, A.R., Haley Jr., P.J., Lermusiaux, P.F.J., Leslie, W.G., 2005. Harvard ocean prediction system (HOPS). Available from: <http://oceans.deas.harvard.edu/HOPS/HOPS.html>.

Sang, J., Follen, G., Kim, C., Lopez, I., 2002. Development of CORBA-based engineering applications from legacy Fortran programs. Inform. Software Technol. 44, 175–184.

Schmidt, H., Tango, G., 1986. Efficient global matrix approach to the computation of synthetic seismograms. Geophys. J.R. Astr. Soc., 84.

Semtner, A.J., 1997. Introduction to a numerical method for the study of the circulation of the world ocean. J. Comput. Phys. 135 (2), 149–153.

Senger, M., 1999. AppLab: CORBA-Java based application wrapper, CCP11 Newsletter, 8.

Senger, M., Flores, T., Glatting, K.-H., Hotz-Wagenblatt, A., Suhai, H., 1998. W2H: WWW interface the GCG sequence analysis package. Bioinformatics 14, 452–457.

Serrano, M.A., Carver, D.L., de Oca, C.M., 2002. Reengineering legacy systems for distributed environments. J. Syst. Software 64, 37–55.

Seymour, K., Dongarra, J., 2003. Automatic translation of Fortran to JVM bytecode. Concurr. Comput.: Pract. Exper. 15 (3–5), 207–222.

Signell, R.P., Jenter, H.L., Blumberg, A.F., 2000. Predicting the physical effects of relocating Boston's sewage outfall. J. Estuarine Coastal Shelf Sci. 50 (1), 59–72, for model information: http://crusty.er.usgs.gov/ecomsi.html.

Smith, R.D., Dukowicz, J.K., Malone, R.C., 1992. Parallel ocean general circulation modeling. Physica D 60, 38–61, for model information: http://www.lanl.gov/orgs/t/t3/globalclimate.shtml.

Stallman, R.M., McGrath, R., 2005. GNU Make: A Program for Directing Recompilation, Free Software Foundation. Available from: <http://directory.fsf.org/devel/build/make.html>.

Stammer, D., Chassignet, E.P., 2000. Ocean state estimation and prediction in support of oceanographic research. Oceanography 13, 51–56.

Suzumura, T., Nakada, H., Matsuoka, S., Casanova, H., 2004. GridSpeed: A Web-based Grid Portal Generation Server. In: Proceedings of the 7th International Conference on High Performance Computing and Grid in Asia Pacific Region (HPCAsia'04), Tokyo.

Taylor, I.L., 1998. The GNU configure and build system, Cygnus Solutions.

Terekhov, A., Verhoef, C., 2000. The realities of language conversions. IEEE Software 17 (6), 111–124.

Walker, D.W., Li, M., Rana, O.F., 2000a. An XML-based component model for wrapping legacy codes as Java/CORBA components. In: Proceedings of the Fourth International Conference on High Performance Computing in the Asia-Pacific Region, Beijing, China. IEEE Computer Society Press, pp. 507–512.

Walker, D.W., Li, M., Rana, O.F., Shields, M.S., Huang, Y., 2000b. The software architecture of a distributed problem-solving environment. Concurr. Comput.: Pract. Exper. 12, 1455–1480.

Wallcraft, A.J., 1991. The Navy Layered Ocean Model Users Guide, Naval Research Laboratory, Stennis Space Center, MS, 21, NOARL Report 35. Available from: <http://www7320.nrlssc.navy.mil/html/images/users_guide.ps.gz>, for model information: http://www7320.nrlssc.navy.mil/global_nlom/.

Wenderholm, E., 2005. Eclpss: a Java-based framework for parallel ecosystem simulation and modeling. Environ. Modell. Software 20 (9), 1081–1100.

Wohlstadter, E., Jackson, S., Devanbu, P., 2001. Generating Wrappers for Command Line Programs: The Cal-Aggie Wrap-O-Matic Project. In: 23rd International Conference on Software Engineering (ICSE'01), p. 243.

eXtensible Markup Language, 2005. Available from: <http://www.w3.org/XML/>.

XML schema specification, 2005. Available from: <http://www.w3.org/XML/Schema.

Yilmaz, N.K., 2005. Path Planning of Autonomous Underwater Vehicles for Adaptive Sampling. Mechanical Engineering Department, MIT, Ph.D. thesis in final preparation, September, Cambridge, MA.