Rapid Real-Time Interdisciplinary Ocean Forecasting Using Adaptive Sampling and Adaptive Modeling and Legacy Codes: Component Encapsulation Using XML

Constantinos Evangelinos[1], Robert Chang[1], Pierre F.J. Lermusiaux[2], and Nicholas M. Patrikalakis[1]

[1] Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A.
[2] Harvard University, Cambridge, MA 02138, U.S.A.

Abstract. We present the high level architecture of a real-time interdisciplinary ocean forecasting system that employs adaptive elements in both modeling and sampling. We also discuss an important issue that arises in creating an integrated, web-accessible framework for such a system out of existing stand-alone components: transparent support for handling legacy binaries. Such binaries, that are most common in scientific applications, expect a standard input stream, maybe some command line options, a set of input files and generate a set of output files as well as standard output and error streams. Legacy applications of this form are encapsulated using XML. We present a method that uses XML documents to describe the parameters for executing a binary.

1 Introduction

Effective ocean forecasting is essential for efficient human operations 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 of the last decade have given rise to complete Ocean Prediction systems [1] that are used in operational settings. Recent developments in the availability of high-performance computing and networking infrastructure now make it possible to construct distributed computing systems that address computationally intensive problems in interdisciplinary oceanographic research, coupling physical and biological oceanography with ocean acoustics [2].

Poseidon [3] is such a distributed computing based project, that brings together advanced modeling, observation tools, and field and parameter estimation methods for oceanographic research. The project has three main goals:

1. To enable efficient interdisciplinary ocean forecasting, by coupling physical and biological oceanography with ocean acoustics in an operational distributed computing framework.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 375–384, 2003.

2. To introduce adaptive modeling and adaptive sampling of the ocean in the forecasting system, thereby creating a dynamic data-driven forecast.
3. To allow 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 to allow the ocean scientist/forecaster to concentrate on the task at hand as opposed to the micro-management of the underlying forecasting mechanisms.

The Poseidon project employs the Harvard Ocean Prediction System (HOPS) [4] as its underlying advanced interdisciplinary forecast system. HOPS is a portable and generic system for interdisciplinary nowcasting and forecasting through simulations of the ocean. It provides a framework for obtaining, processing, and assimilating data in a dynamic forecast model capable of generating forecasts with 3D fields and error estimates. HOPS has been successfully applied to several diverse coastal and shelf regions [1], and analyses have indicated that accurate real-time operational forecast capabilities were achieved. HOPS' advanced DA scheme (Error Subspace Statistical Estimation - ESSE [5]) that is quasi-optimal and at the same time provides an estimate of the dominant uncertainty modes in the forecast is central to the project's stated goal of adaptive modeling and sampling.

The architecture of Poseidon is being designed based on HOPS, but keeping in mind the future HOPS developments and also very importantly allowing the replacement of certain elements of HOPS by other components, e.g. employing different physical oceanographic models for adaptive physical modeling. Moreover, the ESSE methodology, that is computing and data intensive, is also an important driving force behind the architectural design decisions.

One of the first practical problems we faced in the process of designing Poseidon had to deal with the fact that HOPS (as well as other ocean modeling systems, eg. for physical oceanography ROMS [6] or for ocean acoustics OASES [7]) are, like most scientific applications, legacy[1] programs. They consist of binaries that expect a standard input (*stdin*) stream, maybe some command line options, 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) hardcoded in scripts. While such an 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 remote use over the Web.

After examining various ways of dealing with this issue in the context of our distributed computing architecture and keeping in mind that the Poseidon system should allow for future handling of non-HOPS components, we opted to keep working with Fortran binaries and to encapsulate their functionality and

[1] The term "legacy" should not be misconstrued to imply aged code in this context: these are all codes with an active development community and recent enhancements. For various reasons they are still being written in Fortran 77.

requirements using the eXtensible Markup Language (XML) [8]. Thereby we are creating a computer-readable manual for the binaries, allowing us to generate a GUI, check for parameter correctness and drive execution in a transparent manner.

In what follows, Section 2 briefly describes ESSE-based forecast workflows, and adaptive sampling and modeling in the Poseidon system. Section 3 discusses the restrictions legacy software impose on our system design and some proposed solutions. Section 4 describes the XML schema [9] designed for constraining XML descriptions of encapsulated binaries. Section 5 presents some results from our initial XML-based implementation. Section 6 concludes the paper.

2 Forecast Workflow and Adaptivity

The initial implementation of Poseidon is based on HOPS. Poseidon distributes its forecasting and data assimilation components, and will include new HOPS developments in adaptive modeling and adaptive sampling. The HOPS forecasting software is distributed by separating/parallelizing its sequential modules. For example, physical and biological time-stepping modules are run simultaneously on different CPUs with data-exchanges between modules. In this section, the focus is on the data assimilation and adaptivity components of HOPS.

Data assimilation (DA) provides estimates of natural fields which are better than what 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. These computations are very expensive. The ESSE schemes provide an optimal reduction of the problem: only the dominant errors are minimized. For example, if a variance criterion is used to combine data and dynamics, the "dominant errors" are then defined by the dominant ordered eigendecomposition of a normalized form of the error covariance matrix. Even with such optimal error reductions, ESSE still involves massive throughput requirements. However, by design, it provides tremendous opportunities for scalable parallelism within the Poseidon system.

The ESSE workflow is as follows: First, the interdisciplinary error subspace is initialized based on an error decomposition on multiple scales (Fig. 1, far left oval). These dominant initial errors are then evolved by an ensemble of perturbed nonlinear and stochastically-forced dynamical model integrations (Fig. 1, center left oval). As the size of the ensemble is increased, convergence criteria are evaluated. Usually, converged ensemble sizes are $O(100 - 1000)$. This provides a significant opportunity for throughput parallelism. Individual model integrations can also be parallel simulations (depending on the problem size and interdisciplinary nature of the problem), further increasing the scope for parallelism, while also stressing available computing and data resources.

Once error estimates have converged, adaptive sampling forecasts are issued (Fig. 1, bottom left oval). For example, future sampling patterns of autonomous underwater vehicles are computed so as to maximize the reduction of forecast errors. As new data are made available, data-forecast misfits are computed and

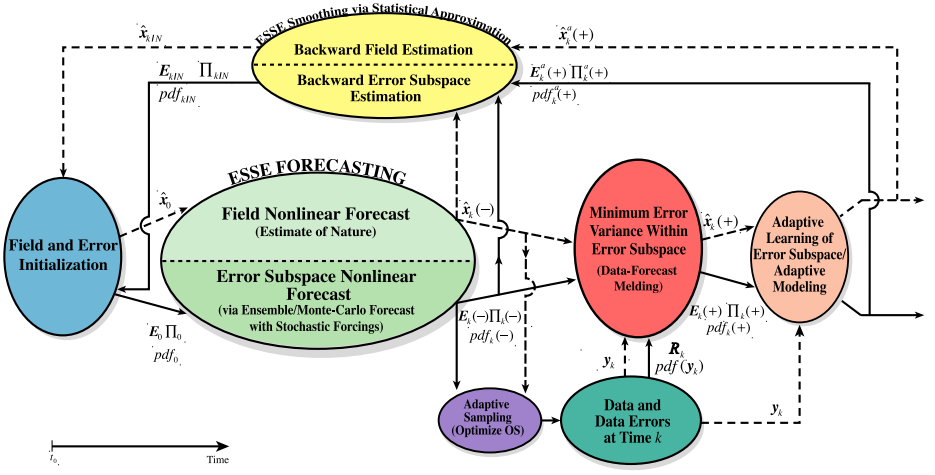

Fig. 1. The ESSE schematic workflow, adapted from [10]

used to correct the predicted fields by minimum error variance estimation in the interdisciplinary error subspace (Fig. 1, center right oval). Outputs are the filtering field and error estimates. A posteriori data misfits are then calculated and used for adaptation of the dominant errors and adaptation of model structures and parameters (Fig. 1, right oval). Ultimately, the smoothing via ESSE is carried out (Fig. 1, top oval) to correct, based on future data, the past coupled fields and uncertainties [10].

Automated objective adaptive modeling allows the optimal use of approximate models for rapidly evolving ocean dynamics. Presently, a model quantity is said to be adaptive if its formulation, classically assumed constant, is made variable as a function of data values. Both structural as well as parametric adaptation are possible. Physical adaptive modeling includes regime transition (e.g., well-mixed to stratified) and evolving turbulent mixing parameterizations. Biogeochemical adaptive modeling includes variations of biological assemblages with time and space (e.g., variable zooplankton dynamics, summer to fall phytoplankton populations, etc) and evolving biogeochemical rates and ratios. This is especially important because biogeochemical modeling is in its infancy and model uncertainties are very large. The adaptive component also greatly facilitates quantitative comparisons of competing biogeochemical models, thus ultimately leading to better scientific understanding.

From a computational point of view, three basic cases are considered in the Poseidon design: i) a single interdisciplinary model is run and adapted, ii) a single physical model is coupled to a set of competing biological models whose parameters are adapted, iii) competing interdisciplinary models are run and their parameters are adapted. In all cases, the basis of the adaptation is the value of the misfits between model estimates and data. When misfits are large, either models are adapted or, in the later two cases, eliminated when too inadequate.

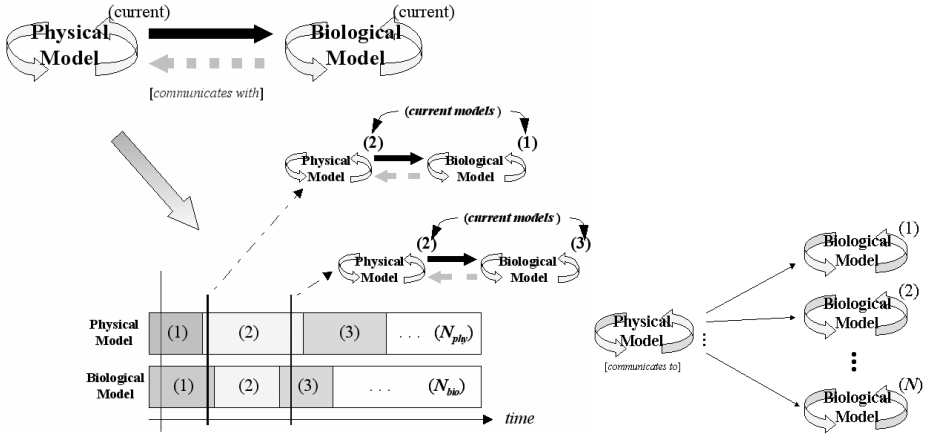

Fig. 2. Approaches for adaptive biophysical modeling

Within ESSE, misfits are large when they become significant compared to their expected values, the ESSE uncertainties.

In the first case (Fig. 2, left), specific model input files are continuously checked by the model software as it is running. These input files are updated when data-forecast misfits are significant. Model parameters and/or model structures are then modified accordingly. For example, the carbon-to-chlorophyll ratio is modified or the number of state variables is increased (adding mesozooplankton). The later is a structural adaptation which is efficiently implemented using a set of C function pointers, with little change to the legacy software. Adaptive updates on the physics or biology sides do not have to be concurrent. In the second case (Fig. 2, right), it is the forecasts of competing biogeochemical models which are compared based on their respective data-forecast misfits. The parameter values are updated and the best biological functional forms can then be selected. The third case is as the second except that several physical as well as several biological models are run and compared. In both cases ii) and iii), only the parameters of the competing model structures are adapted (i.e. fitted) to the data. The best model structures correspond to the fitted model which has the least significant misfits.

3 Legacy Codes

The software components used in Poseidon are legacy Fortran 77 codes (be it HOPS or ocean acoustics codes). As such they do not fit well within a modern distributed computing model: For example, there is no support for dynamic memory allocation, remote procedure calls and component-based programming. At worst, they require manual intervention to work with each other (conversion, concatenation, file editing). At best, they have been designed to interoperate in a prescribed workflow. The individual components of such a workflow 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.

There are two major options for dealing with this issue of legacy: Migration and Encapsulation (wrapping). The cleanest approach is to migrate the code and rewrite all applications in a modern language, employing object-oriented and component technologies. Use of platform-agnostic programs (by transferring the codes to Java for example) and Mobile Agents [11] for the distributed calculations would be a radical form of such modernization. Such an option is obviously costly. Moreover, it is error-prone as it is impractical to convert existing procedural programs to object-oriented components [12].

The more traditional approach is to encapsulate legacy codes as modern software components using wrappers. For example, CORBA [13] wrappers (for clear standardization reasons) 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 [14]. However this powerful encapsulation approach involves breaking the Fortran code into separately callable components that a main program can access.

While CORBA was not designed with high performance computations in mind, there is recent activity in the area of software components for scientific computation, centered mostly around the Common Component Architecture Forum (CCA) [15]. We chose not to follow this approach as it would still entail significant changes to our software components; moreover it would have to force their developers to migrate to a new paradigm to allow us to keep up with enhancements and bug fixes to their codes. While this is possible (but not necessarily practical) for in-house codes, it is not reasonable for third party codes and impossible with programs that are provided in binary format.

Another approach compatible with stand-alone binaries involves wrapping at the binary level, preparing the input files and arguments to the binary, executing the program and capturing and interpreting the output. This can again be accomplished using CORBA or Mobile Agents (losing much of their advantages in the process). It can also be done via a master application generating job scripts for a Grid computing [16] environment.

This way 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 runtime parameters. The advent of XML [8] 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.

Some related work has been done but none focused on wrapping legacy software components at the binary level to allow users to modify their runtime parameter values. Sneed [17] worked on techniques for encapsulating legacy COBOL programs with an XML interface. This requires some modifications of legacy software within an architecture so as to adapt the components for

reading and writing XML interfaces. Walker et al [18,19] illustrate the software architecture of a problem solving environment (PSE) used for the construction of scientific applications from software components. Users visually construct domain-specific applications using a Java/CORBA-based PSE for scientific simulations and computations. Each legacy component is encapsulated as a CORBA object, with its interface and constraints defined in XML.

The main difference between our approach and approaches such as the above is that we aim to encapsulate software components without having to adapt them for XML. We are in fact generating software metadata (in XML format) that describe completely how to control the legacy code's runtime behavior.

4 XML Schema Design

The XML interface to the binaries should be self-contained and should not require modifications to the binaries. By providing a detailed XML description for a binary, this binary is treated as a black-box. An application should then be able to parse in the XML description, and from its contents, determine the specifics on how to properly execute the binary with the appropriate input parameters.

Several key concerns have to be addressed and supported in the XML schema [20]. One issue is that the resulting XML documents should provide as much information to the user as possible, so that well informed decisions can be taken while making changes to the runtime parameters. There should be a set of default parameter values so that manual entry of all values for each execution is avoided, especially since there can be hundreds of them.The XML descriptions conforming to our schema design should also be capable of specifying input and output files for the binary execution. Since runtime parameters should be checked for legality after any user changes, the schema must support datatypes and constraints on parameter values. This is to facilitate the execution of the binary and ensure that all input parameters are acceptable. Each parameter value can be validated against its constraints and datatype before proceeding.

Our schema design supports the description of the binary's input and output files, as well as the runtime input parameters that are read in from *stdin*. It also handles command line arguments and other runtime parameter sources. It consists of two layers: the top layer handles the basic information about the binary, child layers include the runtime inputs. 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. A more detailed description of the schema with example XML descriptions of software components will be presented in a forthcoming paper.

5 Initial Results

We initially tested our implementation with the Primitive Equation (PE) Model binary in the HOPS system. The PE model is at the heart of the forecasting system: it is the solver for the ocean temperature, salinity and velocity fields.

The PE Model binary has runtime parameters read in from *stdin*. A sample of the runtime parameter data is provided in the *stdin* file *pemodel.in*. We wrote a schema-conforming XML description based on the values and types of these parameters.

After validating the XML description using the schemas, our application was able to process the parameters for display by the GUI, as shown in Fig. 3. The system presents the contents of the *stdin* file in an organized manner that is easily understood. Instead of editing the input file directly, the user can update parameter values in the GUI and have changes checked for validity (type, range) before the system generates the new input file and execution script automatically.

A subset of the PE Model *stdin* file generated by our system is included in Fig. 4. The entire PE Model input file consists of 39 *cards*, which are analogous to the *sets* defined in our schema. Each *card* contains the group of variables relating to a specific aspect of the PE Model binary. For instance, *Card 10* deals with the tidal mixing variables used during execution. The PE Model binary ignores every other line that contains the textual description of the parameter value(s) for each *card*. The last line beginning with "99" signifies the end of the input file for the binary.

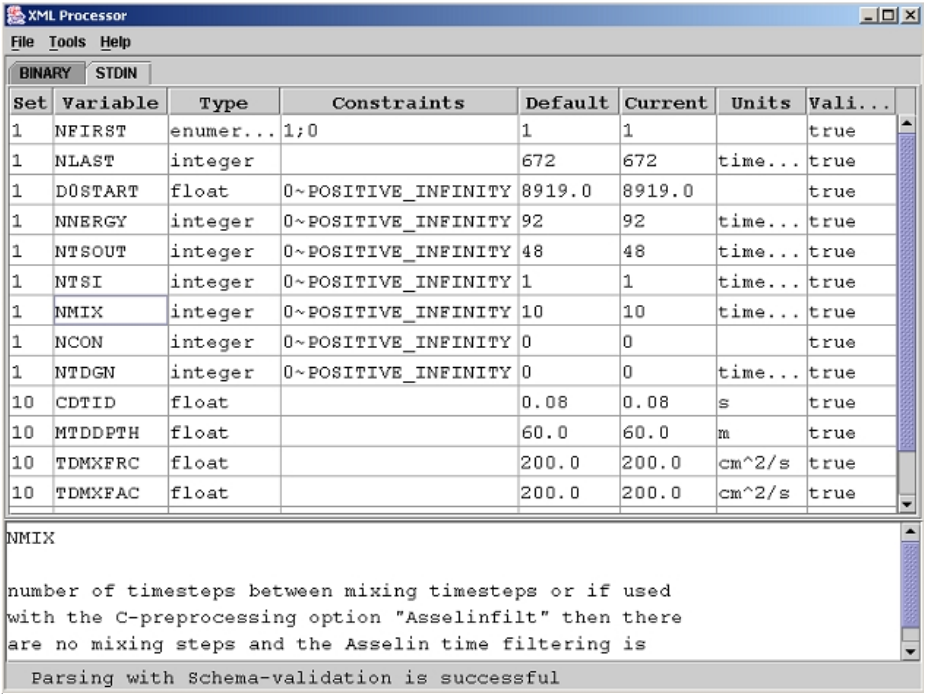

Fig. 3. Screen capture of GUI displaying runtime parameters

```
*** CARD 1:  Various Initialization Parameters
1  672  8919.00  92  48  1  10  0  0
*** CARD 10:  Tidal Mixing
0.08  60.0  200.0  200.0  0.1
*** CARD 39:  Input File Defining Tidal Regions
dev/null
99   END of input data
```

Fig. 4. Subset of *stdin* file generated automatically by our system

6 Conclusion and Future Research

Portions of the high level architecture for Poseidon, a distributed computing system for Real-Time Interdisciplinary Ocean Forecasting employing adaptive modeling and sampling were presented. This includes ESSE-based workflows and adaptive modeling and sampling schemes used to enhance forecasting capabilities. Poseidon efficiently manages legacy codes by using an XML-based method for the encapsulation of legacy binaries. The schema-validated XML descriptions provide a machine (and human) readable standard for handling legacy codes. By wrapping the binaries using XML, their input and output files and *stdin*-provided runtime parameters are described. A prototype system was implemented and shown to generate a graphical interface displaying user-prescribed parameters. The GUI allows for user customization of parameters and validates user changes. It produces input files and a script file for the execution of the binaries.

Future research includes extending our approach to encompass build-time information. To allow XML-descriptions of binaries from other systems, the list of supported datatypes needs to be expanded. The full system will be integrated within a Grid infrastructure using the Globus toolkit [21].

Acknowledgements. The authors would like to acknowledge the assistance of Professors J.J. McCarthy, A.R. Robinson and H. Schmidt, Drs. P.J. Haley, S. Lalis and R. Tian and Mr. S.K. Geiger. This work was funded in part from NSF/ITR (under grant EIA-0121263) and from DoC (NOAA via MIT Sea Grant) (under grant NA86RG0074).

References

1. Robinson, A.: Forecasting and simulating coastal ocean processes and variabilities with the Harvard Ocean Prediction System. In Mooers, C., ed.: Coastal Ocean Prediction. AGU Coastal and Estuarine Studies Series. American Geophysical Union (1999) 77–100
2. Patrikalakis, N.M., Abrams, S.L., Bellingham, J.G., Cho, W., Mihanetzis, K.P., Robinson, A.R., Schmidt, H., Wariyapola, P.C.H.: The digital ocean. In: Proceedings of Computer Graphics International, GCI '2000, Geneva, Switzerland, IEEE Computer Society Press (2000) 45–53 Los Alamitos, CA: IEEE, 2000.

3. Patrikalakis, N.: (Poseidon: A Distributed Information System for Ocean Processes) http://czms.mit.edu/poseidon/.
4. Robinson, A.: (Harvard Ocean Prediction System (HOPS)) http://oceans.deas.harvard.edu/HOPS/HOPS.html.
5. Lermusiaux, P., Robinson, A.: Data assimilation via Error Subspace Statistical Estimation. Part I: Theory and schemes. Month. Weather Rev. **127** (1999) 1385–1407
6. Haidvogel, D., Arango, H., Hedstrom, K., Malanotte-Rizzoli, A.B.P., Shchepetkin, A.: Model evaluation experiments in the North Atlantic basin: Simulations in nonlinear terrain-following coordinates. Dyn. Atmos. Oceans **32** (2000) 239–281
7. Schmidt, H., Tango, G.: Efficient global matrix approach to the computation of synthetic seismograms. Geophys. J. R. Astr. Soc. **84** (1986)
8. : (eXtensible Markup Language) http://www.w3.org/XML.
9. Roy, J., Ramanujan, A.: XML schema language: Taking XML to the next level. IT Professional **3** (2001) 37–40
10. Lermusiaux, P., Robinson, A., Haley, P., Leslie, W.: Advanced interdisciplinary data assimilation: Filtering and smoothing via Error Subspace Statistical Estimation. In: The OCEANS 2002 MTS/IEEE, Holland Publications (2002) 795–802
11. Houstis, C., Lalis, S., Christophides, V., Plexousakis, D., Vavalis, E., Pitikakis, M., Kritikos, K., Smardas, A., Gikas, C.: A service infrastructure for e-Science: the case of the ARION system. In: 14th Intern. Conf. on Advanced Information Systems Engineering (CAiSE 2002). Number 2512 in Lecture Notes in Computer Science, Toronto, Canada, Springer (2002) 175–187 E-Services and the Semantic Web workshop (WES2002).
12. Terekhov, A., Verhoef, C.: The realities of language conversions. IEEE Software (2000) 111–124
13. : (Common Object Request Broker Architecture) http://www.corba.org.
14. : (Java Native Interface) http://java.sun.com/j2se/1.4.1/docs/guide/jni.
15. : (Common Component Architecture Forum) http://www.cca-forum.org.
16. Foster, I., Kesselman, C., eds.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (1999)
17. Sneed, H.M.: Wrapping legacy COBOL programs behind an XML-interface. In: Proceedings of the Eighth Working Conference on Reverse Engineering. (2001) 189–197
18. Walker, D.W., Li, M., Rana, O.F.: An XML-based component model for wrapping legacy codes as Java/CORBA components. In: Proc. of the Fourth Intern. Conf. on High Performance Computing in the Asia-Pacific Region, Beijing, China, IEEE Computer Society Press (2000) 507–512
19. Walker, D.W., Li, M., Rana, O.F., Shields, M.S., Huang, Y.: The software architecture of a distributed problem-solving environment. Concurrency: Practice and Experience **12** (2000) 1455–1480
20. : (XML Schema Specification) http://www.w3.org/XML/Schema.
21. : (The Globus Project) http://www.globus.org.