Evaluation of Model Coupling Frameworks for Use by the Community Surface Dynamics Modeling System (CSDMS)

Scott D. Peckham
University of Colorado, Scott.Peckham@colorado.edu, Boulder, CO, US

ABSTRACT

The Community Surface Dynamics Modeling System (CSDMS) is a recently NSF-funded project that represents an effort to bring together a diverse community of surface dynamics modelers and model users. Its members are from both universities and industry and are divided into five working groups: terrestrial, coastal, marine, education and knowledge transfer and cyber/numerics. Key goals of the CSDMS project are to (1) promote open-source code sharing and re-use, (2) to develop a review process for code contributions, (3) promote recognition of contributors, (4) develop a "library" of low-level software tools and higher-level models that can be linked as easily as possible into new applications and (5) provide resources to simplify the efforts of surface dynamics modelers. The architectural framework of CSDMS is being designed to allow code contributions to be in any of several different programming languages (language independence), to support a migration towards parallel computation and to support multiple operating systems (platform independence). In addition, the architecture should permit structured, unstructured and adaptive grids. This paper provides a brief summary of component-based software engineering and several different coupling frameworks and architectures including ESMF, OpenMI, and CCA within the context of the CSDMS project goals.

INTRODUCTION

A variety of different "coupling frameworks" are currently in use or under development in support of large modeling projects in other communities. One of these, ESMF (Earth System Modeling Framework; Hill et al., 2004; Collins et al., 2005), is primarily centered on Fortran 90, structured grids and Unix-based platforms. ESMF has significant buy-in from the weather and climate modeling community in the U.S. and several federal agencies. Another framework called OASIS4 (part of PRISM) is under development by climate modelers in Europe. OpenMI (Open Modeling Interface; Gregersen et al., 2007; Moore and Tindall, 2005) has emerged from the hydrologic community in Europe and although more of a modeling interface standard for hydrologic applications than a coupling framework it is supported by a variety of software tools for migration, linking, performance monitoring and visualization of results. Several hydrologic software companies in Europe are contributing to the development of OpenMI, including Delft Hydraulics, DHI (Danish Hydraulics Institute) and Wallingford Software. OpenMI is primarily centered on the Microsoft Windows platform, including the .NET framework and a programming language called C# ("C sharp", similar to Java). It is not intended for high-performance computing. Much of the core functionality of OpenMI is also being implemented in Java. A third, DOE-funded "coupling framework" called CCA (Common Component Architecture; Bernholdt et al., 2006) achieves language interoperability using a tool called Babel (Dahlgren et al., 2007). Babel currently supports rapid communication between components written in C, C++, F77, F90, F95, F03, Java and Python. It supports parallel computation and many different operating systems but does not yet have native support for Windows. CCA has been used to wrap MCT (Model Coupling Toolkit) so that it can be used to couple CCA components (McInnes et al., 2006). Prototypes have also been completed showing how ESMF components can be wrapped in CCA in order to make them interoperable with other CCA components (Larson et al., 2004; Zhou et al., 2003,2004). CCA should also be able to utilize OpenMI components written in Java. For all of these reasons CCA is very attractive for the CSDMS project and has been selected as its base architecture.

BRINGING TOGETHER A DIVERSE COMMUNITY

The surface dynamics modeling community is diverse and has many different types of models written in many different programming languages. While Fortran appears to dominate in the realm of ocean and

climate models, many of the large modeling efforts outside of this domain are written in other popular languages such as C, C++, Java and Python. Many of these other models include graphics and GUIs. In addition to this diversity of languages, surface dynamics modelers also use a wide variety of operating systems, data formats, web services and supporting tools. This heterogeneity is an intrinsic aspect of this community and in some cases has been a barrier to cooperation between different groups. Differences between programming languages create difficulties when a program written in one language tries to call a subroutine (and pass data to or from it) that is written in another, possibly very different language. Rewriting or translating code from one language to another is a tedious, time-consuming and error-prone endeavor and moreover does not allow contributors to retain primary, ongoing responsibility for their code. In view of these facts, language interoperability tools such as Babel and CHASM that can automatically create the "glue code" for communication between any pair of supported languages are very attractive. These two tools, designed specifically to support high-performance scientific computing, are an integral part of the CCA effort. By contrast, the ESMF framework and the OpenMI toolkit are primarily focused on one or two languages and have a lesser need for these types of tools.

While language interoperability is an important concern, the fundamental challenge to a project like CSDMS is to develop the robust and intuitive interfaces that are required in order to enable plug-and-play couplings between models and components that were not designed to work together. This goal poses significant challenges from both the social and technical points of view. While two models may provide similar functionality and compute similar quantities, significant differences in their interfaces can make it difficult and time-consuming to swap out one for the other in a given application. As an interface standard for hydrologic models, OpenMI is focused on this issue. However, it does not support high-performance computing and is built around the capabilities of Microsoft's .NET framework. Moreover, its hydrologic scope may be less broad than that of CSDMS. As an architecture standard, CCA does not impose any type of interface. However, there are a growing number of CCA-compliant components such as solvers and meshing tools that have well-defined interfaces and new projects that hope to use these components should strive for interfaces that are compatible with them. Also, many of the tools in the CCA toolchain, such as Babel and Bocca have been designed to cleanly separate interface issues from implementation issues. Bocca (Elwasif et al., 2007) is a recent addition to the CCA toolchain that greatly simplifies the process of building applications from CCA components. Since it is not feasible for projects like CSDMS to rewrite and/or test large quantities of user-contributed code, the only workable solution to this interface problem is to impose interface standards through a wrapping process that is as straightforward as possible. It will be the responsibility of the CSDMS working groups to solicit, evaluate and prioritize code contributions from their respective subcommunities as well as to assist with the development of suitable interface standards.

COMPONENT BASED SOFTWARE ENGINEERING

Every programmer is familiar with the numerous advantages of breaking a large application into many smaller subroutines. Similarly, there are major advantages to building applications from subroutines that have been written and tested previously, often by experts. The use of shared libraries, numerical recipes and high-level languages such as Matlab, IDL, Mathematica and Python all demonstrate this concept and result in enormous productivity gains. Component technology is a natural extension of these ideas, but offers a number of additional advantages over subroutine programming. Components generally conform to object-oriented design principles and are instantiated and connected to one another within a "framework". It is the presence of a framework that represents the main point of departure from subroutine programming. Components are linked together within a framework to create applications and the framework provides an extensible set of services that each component can access directly. Services may, for example, address tasks related to communication, security, thread creation and management, memory management and error handling. Some frameworks (e.g. CCA) allow frequently-used components to be promoted to framework services.

The CCA component architecture standard was specifically designed to support the needs of high-performance, open-source, scientific computing. It supports multiple different frameworks that can be tailored to a specific need or type of computing such as parallel (e.g. Ccaffeine) or distributed (e.g. XCAT). While a variety of component architectures and frameworks have also been developed in the

commercial sector (e.g. CORBA, COM, Active X, .NET and JavaBeans), these do not support the performance, data types, languages and other specific needs of the scientific modeling community. CCA employs a minimalist or "loose" design philosophy that greatly simplifies the task of wrapping or incorporating existing software. This gives developers a great deal of flexibility but it does not provide any "automatic" way for the software to take advantage of multiple processors. ESMF offers a variety of specific structures for representing and sharing data (arrays, fields, bundles, location streams and meshes) that provide a somewhat direct path to parallel computation (e.g. through domain decomposition). For some models, however, it may be difficult or time-consuming to map their internal data representation to one of these standardized forms.

Components may encapsulate an entire model, or a smaller unit of functionality. A rich set of smaller components provides a greater variety of plug-and-play configurations. Components often take the form of a class with a set of method functions that are all related to a common theme. For example, a component might provide an approach to modeling a given physical process or phenomenon like infiltration or ocean waves and have method functions to return numerous quantities that can be computed using this approach. An alternate approach to modeling the same process or phenomenon can then be implemented as a component of the same "type", in the sense that it returns the same set of computed quantities and has the same interface. This allows components of the same type to be used interchangeably in any application that requires this type of functionality. Components of the same type may also be used to provide different numerical approaches to solving a given problem, such as equation solvers and mesh generators. Toolkits such as PETSc (Portable, Extensible Toolkit for Scientific Computation) and MCT (Model Coupling Toolkit; Larson et al., 2001) and research centers such as PERI (Performance Engineering Research Institute) and ITAPS (Interactive Technologies for Advanced Petascale Simulations Center) are working to provide these types of components and have close ties with the CCA effort.

Component technology enables "plug and play" functionality in a heterogeneous computing environment. Components can be written in different languages and can be replaced, added or deleted from an application at run-time via dynamic linking. They can also be moved to another remote location (a different address space) without recompiling other parts of an application. Components can have multiple different interfaces and promote code re-use and a clean separation of functionality.

SUMMARY

The functional requirements of the CSDMS project, including support for code contributions in several different languages, support for multiple operating systems and a migration pathway toward high-performance computing all point to CCA as the most appropriate available architecture for the CSDMS project. CCA provides a powerful foundation for high-performance, scientific computing and has already helped to advance computational science (Kumfert et al., 2006). Proof-of-concept studies have shown that CCA is interoperable with ESMF. Work is ongoing to learn more about the technical aspects of OpenMI. This interface standard or a variant of it may address many of the interface needs of the CSDMS project. The CSDMS project aims to be interoperable with these other frameworks and to provide a powerful and user-friendly environment for model development and coupling. In addition, the CSDMS project is working in close cooperation with the NSF-funded CUAHSI project (Consortium of Universities for the Advancement of Hydrologic Science) to ensure compatibility with the data services (e.g. HIS) and other products that are being developed as part of that effort.

REFERENCES

Bernholdt, D.E., B.A. Allan, R. Armstrong, F. Bertrand, K. Chiu, T.L. Dahlgren, K. Damevski, W.R. Elwasif, T.G.W. Epperly, M. Govindaraju, D.S. Katz, J.A. Kohl, M. Krishnan, G. Kumfert, J.W. Larson, S. Lefantzi, M.J. Lewis, A.D. Malony, L.C. McInnes, J. Nieplocha, B. Norris, S.G. Parker, J. Ray, S. Shende, T.L. Windus and S. Zhou (2006) A component architecture for high-performance scientific computing, *Intl. J. High Performance Computing Applications, ACTS Collection Special Issue*, 20(2), 163-202.

Collins, N., G. Theurich, C. DeLuca, M. Suarez, A. Trayanov, V. Balaji, P. Li, W. Yang, C. Hill, and A. da

Silva (2005) Design and implementation of components in the Earth System Modeling Framework, *Intl. J. High Performance Computing Applications*, 19(3), 341-350.
Dahlgren, T., T. Epperly, G. Kumfert and J. Leek (2007) *Babel User's Guide*, March 23, 2007 edition, Center for Applied Scientific Computing, U.S. Dept. of Energy and University of California Lawrence Livermore National Laboratory, 269 pp.
Elwasif, W., B. Norris, B. Allan and R. Armstrong (2007) Bocca: A development environment for HPC components, Proceedings of the 2007 symposium on component and framework technology in high-performance and scientific computing, Montreal, Canada, Association for Computing Machinery, New York, pp. 21-30. PDF online at: http://portal.acm.org/citation.cfm?id=1297390
Gregersen, J.B., Gijsbers, P.J.A., and Westen, S.J.P. (2007) OpenMI : Open Modeling Interface. *Journal of Hydroinformatics*, 9(3), 175-191.
Hill, C., C. DeLuca, V. Balaji, M. Suarez, and A. da Silva (2004) The architecture of the Earth System Modeling Framework, *Computing in Science and Engineering*, 6(1), 18-28.
Kumfert, G., D.E. Bernholdt, T. Epperly, J. Kohl, L.C. McInnes, S. Parker and J. Ray (2006) How the Common Component Architecture advances computational science, *Journal of Physics: Conference Series,* 46, 479-493.
Larson, J. W., Jacob, R. L., Foster, I. T., and Guo, J. (2001) The Model Coupling Toolkit, In: Alexandrov, V. N., Dongarra, J. J., Juliano, B. A., Renner, R. S., and Tan, C. J. K., editors, *Proceedings of the International Conference on Computational Science (ICCS) 2001*, volume 2073 of Lecture Notes in Computer Science, pp. 185–194, Springer-Verlag, Berlin.
Larson, J.W., B. Norris, E.T. Ong, D.E. Bernholdt, J.B. Drake, W.R. Elwasif, M.W. Ham, C.E. Rasmussen, G. Kumfert, D.S. Katz, S. Zhou, C. Deluca and N.S. Collins (2004) Components, the Common Component Architecture, and the climate/weather/ocean community, In: 84th American Meteorological Society Annual Meeting, Seattle, American Meteorological Society.
McInnes, L.C., B.A. Allan, R. Armstrong, S.J. Benson, D.E. Bernholdt, T.L. Dahlgren, L.F. Diachin, M. Krishnan, J.A. Kohl, J.W. Larson, S. Lefantzi, J. Nieplocha, B. Norris, S.G. Parker, J. Ray and S. Zhou (2006) Parallel PDE-based simulations using the Common Component Architecture, *In: Numerical Solution of Partial Differential Equations on Parallel Computers*, Eds. Bruaset, A.M and A. Tveito, Springer, Berlin, pp. 327-381.
Moore, R.V and Tindall,I. (2005) An Overview of the Open Modelling Interface and Environment (the OpenMI). *Environmental Science & Policy*, 8, 279-286.
Zhou, S., A. DaSilva, B. Womack and G. Higgins (2003) Prototyping of the ESMF using DOE's CCA, In NASA Earth Science Technology Conference 2003, College Park, MD. Online at: http://esto.gsfc.nasa.gov/conferences/estc2003/papers/A4P3(Zhou).pdf
Zhou, S., B. Womack and G. Higgins (2004) Grid-enabled earth system models. Online at: http://esto.gsfc.nasa.gov/conferences/estc2004/papers/a4p1.pdf

APPENDIX A: LINKS TO ONLINE RESOURCES

Babel
http://www.llnl.gov/CASC/components/babel.html

CCA (Common Component Architecture)
http://www.cca-forum.org

COM (Component Object Model, Microsoft, incl. COM+, DCOM & ActiveX Controls)
http://www.microsoft.com/com/default.mspx

CORBA (Object Management Group)
http://www.omg.org/gettingstarted
http://www.omg.org/gettingstarted/history_of_corba.htm

CUAHSI (Consortium of Universities for the Advancement of Hydrologic Science)
http://www.cuahsi.org
http://www.cuahsi.org/his.html (Hydrologic Information System, HIS)

ESMF (Earth System Modeling Framework)
http://www.esmf.ucar.edu

FMS (Flexible Modeling System, GFDL, NOAA)
http://www.gfdl.noaa.gov/~fms

ITAPS (Interoperative Technologies for Advanced Petascale Simulations Center)
http://www.itaps-scidac.org

JavaBeans (Sun Microsystems)
http://java.sun.com/products/javabeans
http://java.sun.com/products/ejb

MCT (Model Coupling Toolkit, DOE)
http://www-unix.mcs.anl.gov/mct

.NET (Microsoft Corporation)
http://www.microsoft.com/net

OpenMI (Open Modeling Interface)
http://www.openmi.org

PERI (Performance Engineering Research Institute, formerly TOPS, DOE)
http://www.peri-scidac.org

PETSc (Portable, Extensible Toolkit for Scientific Computation)
http://www.mcs.anl.gov/petsc

PRISM (Program for Integrated Earth System Modeling)
http://www.prism.enes.org

SciDAC Program (DOE)
http://www.scidac.gov

XCAT (Indiana University)
http://www.extreme.indiana.edu/xcat