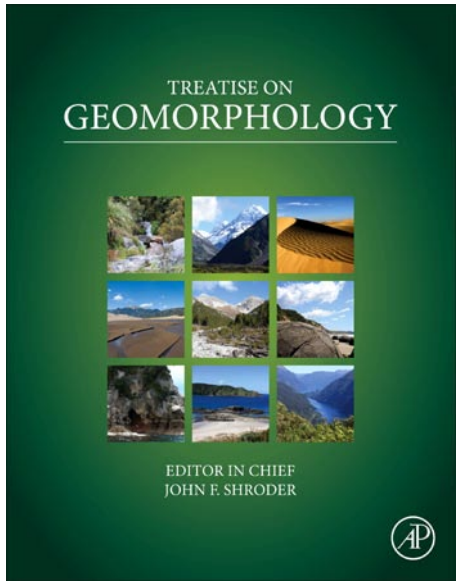
Provided for non-commercial research and educational use only.
Not for reproduction, distribution or commercial use.

This chapter was originally published in the *Treatise on Geomorphology*, the copy attached is provided by Elsevier for the author's benefit and for the benefit of the author's institution, for non-commercial research and educational use. This includes without limitation use in instruction at your institution, distribution to specific colleagues, and providing a copy to your institution's administrator.



All other uses, reproduction and distribution, including without limitation commercial reprints, selling or licensing copies or access, or posting on open internet sites, your personal or institution's website or repository, are prohibited. For exceptions, permission may be sought for such use through Elsevier's permissions site at:

http://www.elsevier.com/locate/permissionusematerial

Slingerland R., and Syvitski J.P.M. A Community Approach to Modeling Earth- and Seascapes. In: John F. Shroder (Editor-in-chief), Baas, A.C.W. (Volume Editor). Treatise on Geomorphology, Vol 2, Quantitative Modeling of Geomorphology, San Diego: Academic Press; 2013. p. 44-49.

© 2013 Elsevier Inc. All rights reserved.

Author's personal copy

2.4 A Community Approach to Modeling Earth- and Seascapes

R Slingerland, The Pennsylvania State University, University Park, PA, USA
JPM Syvitski, University of Colorado—Boulder, Boulder, CO, USA

© 2013 Elsevier Inc. All rights reserved.

Glossary

Babel An open-source, language-interoperability tool (and compiler) that automatically generates the 'glue code' for inter-component communication.
Bocca A development environment tool to create, edit, and manage Common Component Architecture (CCA) components and ports associated with a particular project.
Common Component Architecture (CCA) A component-architecture standard adopted by the US Department of Energy, its national labs, and many academic computational centers to allow software components to be combined.
Community modeling The collective efforts of individuals to code, debug, test, document, run, and apply a suite of modeling components coupled in a framework or community modeling system.
CSDMS An integrated community of experts developing and disseminating integrated software modules that predict the movement of fluids (wind, water, and ice), sediment, and solutes in landscapes, seascapes, and their sedimentary basins.
CSDMS Modeling Tool A user-friendly graphical user interface (GUI) that exploits Ccaffeine, a simple set of scripting commands that instantiate, connect, and disconnect CCA-compliant components.
Framework A set of agreed-upon protocols that allow the software components to function together.
Modeling components Modular code, commonly with a standardized interface to allow different modules to communicate with other components written in a different programming language.
Open-source code Software that is freely available and modifiable. Its attributes are flexibility, tailorability, modularity, and open-endedness in contrast to commercial software.

Abstract

Developing a unified, predictive science of surface processes requires a quantitative understanding of critical surface-dynamics processes. An efficient approach to acquire this understanding is community modeling, defined here as the collective efforts of individuals to code, debug, test, document, run, and apply a suite of modeling components coupled in a framework or community modeling system. The modeling components each consist of modular code, commonly with a standardized interface to allow different modules to communicate with other components written in a different programming language. The framework is a set of agreed-upon protocols that allow the components to function together. Because of the framework, users can assemble components coded and vetted by specialists into complex models tuned to their specific objectives. The advantages of community modeling are efficient use of community resources and more effective integration of scientists and software specialists.

2.4.1 Background

Starting in the 1970s, geoscientists began to translate conceptual models of complex, interacting geomorphic systems into computer codes to address problems that were not solvable analytically. The pioneering works of Strelkoff (1970) solving the open-channel-flow equations, Ahnert (1976) modeling the evolution of a landscape and channel network, and Harbaugh and Bonham-Carter (1970) modeling sedimentary systems showed us what was possible. With the advent of personal computers in the 1980s the trend accelerated, and journals, such as *Computers and Geosciences*, founded in 1976, were publishing over 100 computer codes a year. By the 1990s, the value of quantitative, model-driven science in Earth-surface studies was so apparent that funding panels

Slingerland, R., Syvitski, J.P.M., 2013. A community approach to modeling earth- and seascapes. In: Shroder, J. (Editor in Chief), Baas, A.C.W. (Ed.), Treatise on Geomorphology. Academic Press, San Diego, CA, vol. 2, Quantitative Modeling of Geomorphology, pp. 44–49.

Author's personal copy

began to expect it in proposals. A new generation of young scientists more comfortable with algorithmic computation and, in some cases, better trained in mathematics led to a flowering in geomorphology. Noteworthy examples are the insights gained in tectonic geomorphology from coupled tectonic and landscape evolution models and in the evolution of coastal depositional systems from a new generation of morphodynamic models.

During this exciting development phase, geomorphology codes were typically small, involving single developers. There were also few repositories that would publish or make public your code, an exception being the journal *Computers and Geosciences*, where there is an expectation but no requirement to share your code. Some scientists distributed their code widely, but most code remained outside of the peer-review process. Some examples of published geomorphology codes include the water-balance and transport model, HydroTrend (Syvitski et al., 1998; Kettner and Syvitski, 2008); the morphodynamics and stratigraphic models, SedFlux (Syvitski and Hutton, 2001; Hutton and Syvitski, 2008) and SedSim (Tetzlaff and Harbaugh, 1989; Martinez and Harbaugh, 1993); and the morphodynamics codes of Slingerland et al. (1994), Parker (2007), and Pelletier (2008).

2.4.2 Concept of a Community Modeling System

By 2002, it was apparent to many in the Earth-surface-science community that developing a unified, predictive science of surface processes was beset by two large and growing problems – the community was fragmented and the quantitative understanding of critical surface-dynamics processes was uneven. To address these issues, the National Science Foundation (USA) sponsored a workshop in 2002 directed toward developing a 'Community Sediment Model'. It was envisioned as a series of integrated, quantitative predictive models of basin and landscape evolution, encompassing both the subaerial and submarine realms. Subsequent workshops, guided by parallel developments in geophysics and climate science, refined the concept and, in 2007, led to the establishment and funding of CSDMS, the Community Surface Dynamics Modeling System. Other supportive community efforts included the Chesapeake Community Modeling Program, the Coastal Sediment Transport Modeling System (Warner et al., 2008), and ROMS (the Regional Ocean Modeling System). Community modeling as defined here involves the collective efforts of individuals to code, debug, test, document, run, and apply models and modeling frameworks (Voinov et al., 2010).

In its most basic form, a community modeling system is a suite of modeling components coupled in a framework (Voinov et al., 2008). The modeling components each consist of modular code, commonly with a standardized interface to allow different modules to communicate, that performs specific tasks, such as reading digital elevation data from a file or computing grain-settling velocities according to a specific formula. Components typically can communicate with other components written in a different programming language and, thus, are different from ordinary subroutines, software modules, or classes in an object-oriented language. Individuals of relevant expertise freely give their time to code, debug, test, and document the various components and then donate them to the system. This by itself leads to an improvement in community efficiency. However, the real power of community modeling lies in the framework, a set of agreed-upon protocols that allow the components to function together. Because of the framework, users can assemble components coded and vetted by specialists into complex models tuned to their specific objectives. The advantages of community modeling include: first, efficient use of community resources by cutting redundancy among researchers and institutions. Second, it promotes more effective integration of scientists and software specialists working on a particular Earth-surface system. Finally, it allows users, characteristically with valuable data sets, to participate in model definition and interaction with the data.

2.4.3 Open-Source and Readily Available Code

Community modeling relies on open-source code to address the practical need of contributing developers to examine and modify the code. Open-source code provides complete information transfer. Transparency is important because code is the ultimate statement of the scientific hypotheses embodied in a numerical model and their implementation. A scientific article may provide the theoretical equations, but the solution to these equations can take numerous forms, and each solution has its own pyramid of assumptions and limitations. Open-source code allows for full peer review and replication of results – the foundation of modern science (Syvitski and Grunsky, 2010). Further, code availability should not depend on a gatekeeper, who subjectively determines who gets to see the code; this also runs contrary to the transparency needed in science.

Community modeling therefore relies on, "software licensing and distribution designed to encourage use and improvement of software written by volunteers by ensuring that anyone can copy the source code and modify it freely" (Jesiek, 2003). Open-source software is not necessarily freeware, but the source code must be freely available and modifiable. Its attributes are flexibility, tailorability, modularity, and open-endedness, in contrast to commercial software.

2.4.4 Community Modeling and the CSDMS Approach

The largest and most inclusive communal modeling effort in hydrology, geomorphology, sedimentology, and stratigraphy, with overlap in related fields of environmental engineering, oceanography, and tectonics is the Community Surface Dynamics Modeling System or CSDMS (pronounced 'systems'). CSDMS is an integrated community of experts developing and disseminating integrated software modules that predict the movement of fluids (wind, water, and ice), sediment, and solutes in landscapes, seascapes, and their sedimentary basins. The organization operates under a cooperative agreement with the National Science Foundation (NSF) with additional financial support from industry and other government agencies.

Author's personal copy

The CSDMS Model Repository comprises a searchable inventory of models, some of which have been made into components that users can link through the CSDMS model-coupling framework into a stand-alone model. The CSDMS model repository in January 2010 offers the community over 3 million lines of code. Users can then run their model on the CSDMS high-performance computer and verify and validate their results using high-quality data sets from the CSDMS Data Repository. CSDMS Working Groups self-organized from the scientific community identify important knowledge gaps and encourage code development in those areas.

CSDMS is a complete modeling environment involving a model repository of numerous research-grade codes with augmented services and tools, such as from the Earth System Modeling Framework (Collins et al., 2005). CSDMS employs the Common Component Architecture (Kumfert et al., 2006) and Open Modeling Interface (OpenMI) standards (Gregersen et al., 2007) to provide model coupling, language interoperability, use of unstructured, structured, and object-oriented code, and structured and unstructured grids.

To aid the CSDMS community effort, a series of protocols were established to provide the needed technical and coding recommendations to model developers. Software contributions to the CSDMS Model Repository should:

1. hold an open-source 'GPL v2' or a 'GPL v2 compatible' license;
2. be widely available to the community of scientists through an international model or code repository;
3. undergo a level of peer review;
4. either be written in an open-source language or have a pathway for use in an open-source environment;
5. be written or refactored to allow for componentization by having an interface, with specific exchange items documented;
6. be accompanied with a formally defined metadata file, along with test files; and
7. be clean and well documented.

These protocols provide extensibility to software and allow for state-of-the-art tools to convert stand-alone models into flexible, 'plug-and-play' components that can be assembled into larger applications (Syvitski et al., 2011). The protocols also allow a migration pathway toward high-performance computing (HPC).

In a world of multiple computer languages and hardware architectures, how is all of this possible? CSDMS software engineers use the tools of the Common Component Architecture (CCA) to convert member-contributed code into linkable components (Hutton et al., 2010) (**Figure 1**). CCA is a component architecture standard adopted by the U.S. Department of Energy, its national labs, and many academic computational centers to allow software components to be combined (Kumfert et al., 2006). Three framework tools following the CCA standard have been adopted by CSDMS: Babel, Bocca, and Ccaffeine.

Babel is an open-source, language-interoperability tool (and compiler) that automatically generates the 'glue code' for inter-component communication (Dahlgren et al., 2007). It currently supports C, C++, Fortran (77, 90, 95, and 2003), Java, and Python. For two components written in different programming languages to exchange data, Babel only needs to know about their interfaces. These interface descriptions may be written in either XML (eXtensible Markup Language) or SIDL (Scientific Interface Definition Language) and include the names and data types of all arguments and the return values for each member function. CSDMS uses OpenMI as its model interface standard – a standardized set of rules and supporting infrastructure for how a component must be written or refactored in order for it to more easily exchange

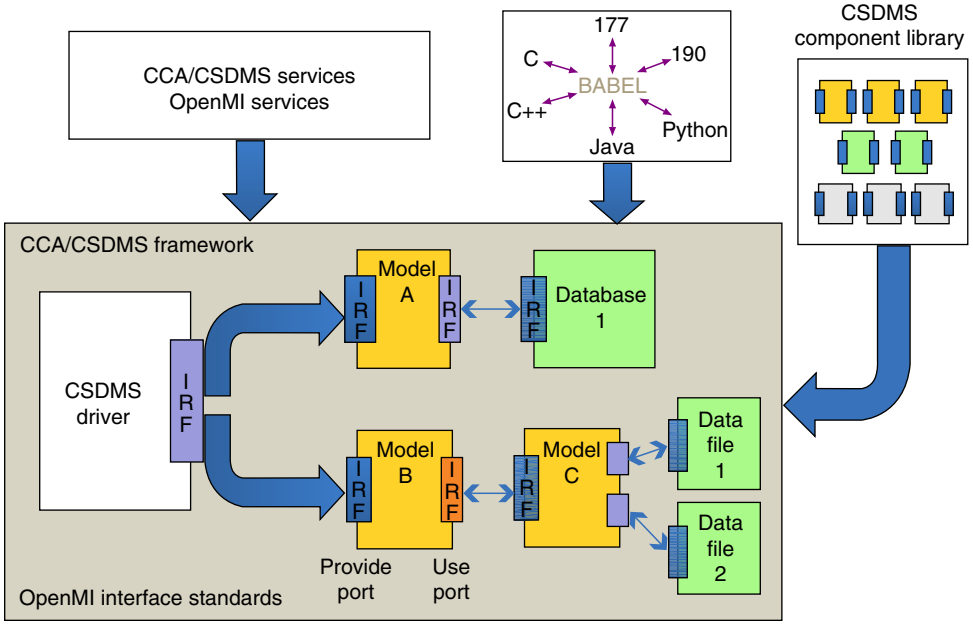


Figure 1 Schematic CSDMS Modeling Architecture consists of a library of component models, a CCA/CSDMS Framework, OpenMI Interface Standards, a suite of CCA/CSDMS and OpenMI Services, and the language neutral tool/compiler Babel.

Author's personal copy

data with other components that adhere to the same standard (Moore and Tindall, 2005; Gregersen et al., 2007). Additional OpenMI functions handle other differences among components, such as differing units, time steps, and dimensionality. CSMDS also employs services from Earth System Modeling Framework (ESMF) for mapping between unstructured and structured grids (e.g., triangular to raster) within an HPC environment.

Bocca helps create, edit, and manage CCA components and ports associated with a particular project (Elwasif et al., 2007). It is a development environment tool that allows rapid component prototyping. Once CCA-compliant components and ports are prepared using Bocca, CSDMS members can then assemble models into applications, with the CSDMS Modeling Tool (CMT).

The components can be assembled into a functional surface process model using the CMT (**Figure 2**). It is a user-friendly graphical user interface (GUI) that exploits Ccaffeine, a simple set of scripting commands that instantiate, connect, and disconnect CCA-compliant components. Ccaffeine can be used at an interactive command prompt or with a 'Ccaffeine script', but, for new users, the easiest approach is the CMT. This tool allows users to select components from a palette and drag them into an arena. Components in the arena can be connected to one another by clicking on buttons that represent their ports. A component with a 'config' button allows its parameters to be changed in a tabbed dialog. Once components are connected, clicking on a 'run' button on the 'driver' component starts the application. The CMT offers significant extensions and improvements to the basic Ccaffeine GUI, including offering a powerful, open-source (US Department of Energy (DOE)) visualization package called VisIt that is specifically designed for HPC use with multiple processors.

As an example, consider a scientist who wants to investigate the role of sediment cohesion on the planform of deltas. The scientist starts by compiling a list of the relevant processes to be included. He/she then downloads and installs the CMT onto a local computer. From the CSDMS libraries, the scientist chooses components that simulate the processes of interest and links them using the CMT. If, for example, a bedload-transport module needs to know the settling velocities of grains, she would link an out-port on the bedload-transport module to an in-port on a settling-velocity module, which would ingest the grain diameters and compute settling velocities. A similar connection back to the bedload-transport module would feed it the settling velocities. Other modules would define the initial and boundary conditions of interest. The person can then either run the new application on the CSDMS supercomputer or download the executable model and run it on their personal computer or server. All this is possible because the CMT supports: (1) Linux, OSX, and Windows platforms; (2) language interoperability; (3) legacy (non-protocol) code and structured code (procedural and object-oriented); (4) both structured and unstructured grids; and (5) a large offering of open-source tools.

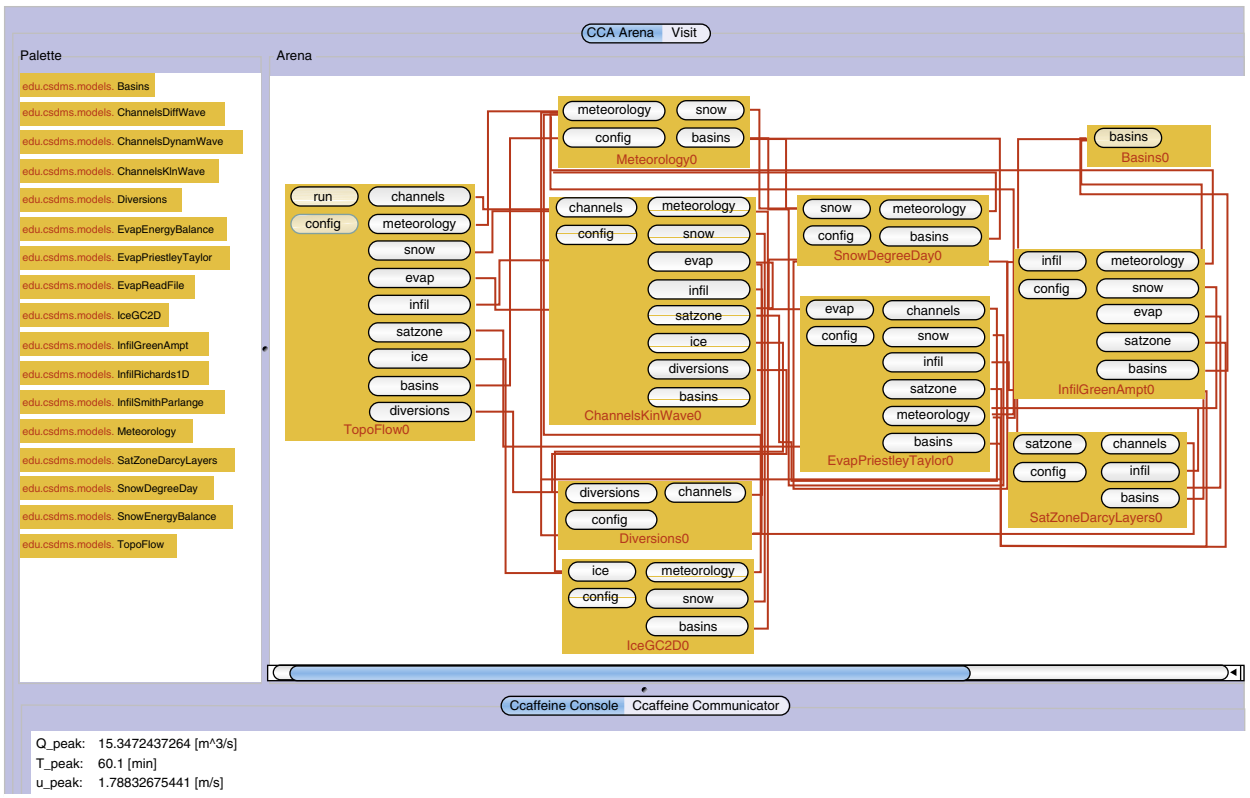


Figure 2 A 'wiring diagram' for a CSDMS project. The CCA framework called 'Ccaffeine' provides a 'visual programming' GUI for linking components to create working applications.

Author's personal copy

2.4.5 Challenges

Voinov et al. (2008) outlined several technical challenges faced by community-modeling efforts in Earth-surface dynamics, including a lack of standards for data and model interfaces and a lack of software to facilitate community collaborations. Probably the two most serious challenges are poorly known fundamental algorithms describing Earth-surface processes and significant social and institutional barriers to community model development. Well-documented, peer-reviewed code should be seen as worthy of merit with effective venues for peer review, publication, and citation.

2.4.6 Summary

Community modeling efforts, such as CSDMS, provide a competitive yet cooperative environment that can produce more reliable and more flexible simulation models than individuals working alone. Freely available open-source code eliminates the endless rewriting of the same initial algorithms, allowing more time spent on new advances. The CSDMS protocols create honesty in what modelers claim, and the CSDMS architecture allows for faster verification and comparison of different approaches on new data sets. Communication is greatly increased among users and coders, and, therefore, a more integrated community is built. If a new and improved model component is developed, then this new component is provided with faster penetration into the community and likely will replace older components. New model couplings will allow hypothesis testing, sensitivity experiments on key parameters, and the identification of new avenues of research.

References

Ahnert, F., 1976. Brief description of a comprehensive three-dimensional process-response model of landform development. Zeitschrift für Geomorphologie 25, 29–49.

Collins, N., Theurich, G., DeLuca, C., et al., 2005. Design and implementation of components in the Earth System Modeling Framework. International Journal of High Performance Computing Applications 19(3), 341–350.

Elwasif, W., Norris, B., Allan, B., Armstrong, R., 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.

Gregersen, J.B., Gijsbers, P.J.A., Westen, S.J.P., 2007. OpenMI: open modeling interface. Journal of Hydroinformatics 9(3), 175–191.

Harbaugh, J.W., Bonham-Carter, G., 1970. Computer Simulation in Geology. John Wiley and Sons, New York.

Hutton, E.W.H., Syvitski, J.P.M., 2008. SedFlux2.0: an advanced process-response model that generates three-dimensional stratigraphy. Computers and Geosciences 34, 1319–1337.

Hutton, E.W.H., Syvitski, J.P.M., Peckham, S.D., 2010. Producing CSDMS-compliant morphodynamic code to share with the RCEM community. In: Vionnet, C., García, G.M.E., Latrubesse, E.M., Perillo, G.M.E. (Eds.), River, Coastal and Estuarine Morphodynamics RCEM 2009. CRC Press, Taylor and Francis, London, pp. 959–962.

Jesiek, B., 2003. Democratizing software: open source, the hacker ethic, and beyond. First Monday 8(10). http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1082/1002

Kettner, A.J., Syvitski, J.P.M., 2008. HydroTrend v3.0: a climate-driven hydrological transport model that simulates discharge and sediment load leaving a river system. Computers and Geosciences 34, 1170–1183.

Kumfert, G., Bernholdt, D.E., Epperly, T.G.W., Kohl, J.A., McInnes, L.C., Parker, S., Ray, J., 2006. How the common component architecture advances computational science. Journal of Physics: Conference Series 46(1), 479.

Martinez, P.A., Harbaugh, J.W., 1993. Simulating Nearshore Environments. Computer Methods in Geosciences. Pergamon Press, New York, vol. 12, 265 pp.

Parker, G., 2007. 1D Sediment Transport Morphodynamics with applications to Rivers and Turbidity Currents, E-book, http://vtchl.uiuc.edu/people/parkerg/morphodynamics_e-book.htm

Pelletier, J.D., 2008. Quantitative Modeling of Earth Surface Processes. Cambridge University Press, Cambridge, 304 pp.

Slingerland, R.L., Furlong, K., Harbaugh, J., 1994. Simulating Clastic Sedimentary Basins/Physical Fundamentals and Computing Procedures. Prentice Hall, Englewood Cliffs, NJ, USA, 219 pp.

Strelkoff, T., 1970. Numerical solution of Saint-Venant equations. Journal of the Hydraulics Division 96(1), 223–252.

Syvitski, J.P., Hutton, E.H., 2001. 2D SEDFLUX 1.0C: an advanced process-response numerical model for the fill of marine sedimentary basins. Computers and Geoscience 27(6), 731–754.

Syvitski, J.P.M., DeLuca, C., David, O., Peckham, S., Hutton, E.W.H., Gooding, J., 2011. Cyber-infrastructure and community environmental modeling. In: Fernando, H.J.S. (Ed.), Handbook in Environmental Fluid Dynamics. Taylor and Francis Group, Boca Raton, FL.

Syvitski, J.P.M., Grunsky, E., 2010. Recommended protocols for model software developers. Computers and Geosciences (in review).

Syvitski, J.P.M., Morehead, M., Nicholson, M., 1998. HYDROTREND: a climate-driven hydrologic-transport model for predicting discharge and sediment to lakes or oceans. Computers and Geoscience 24(1), 51–68.

Tetzlaff, D.M., Harbaugh, J.W., 1989. Simulating Clastic Sedimentation. Van Nostrand Reinhold, New York, 202 pp.

Voinov, A., DeLuca, C., Hood, R., Peckham, S., Sherwood, C., Syvitski, J.P.M., 2010. A community approach to Earth systems modeling. EOS Transactions of the AGU 91(13), 117–124.

Voinov, A., Hood, R.R., Daues, J.D., Assaf, H., Stewart, R., 2008. Building a Community Modelling and Information Sharing Culture. Environmental Modelling, Software and Decision Support: State of the Art and New Perspectives. A.J. Jakeman, A.A. Voinov, A.E. Rizzoli and S.H. Chen. Elsevier, Amsterdam.

Warner, J.C., Sherwood, C.R., Signell, R.P., Harris, C., Arango, H.G., 2008. Development of a three-dimensional, regional, coupled wave, current, and sediment-transport model. Computers and Geosciences 34, 1284–1306.

Relevant Websites

http://ches.communitymodeling.org
CCMP is dedicated to advancing the cause of accessible, open-source environmental models of the Chesapeake Bay in support of research and management efforts.

http://www.cstms.org
The Coastal Sediment Transport Modeling System (CSTMS) is an open-source model that couples hydrodynamics (circulation and waves), sediment transport, and morphodynamics.

http://www.myroms.org
The Regional Ocean Modeling System (ROMS) framework.

Author's personal copy

Biographical Sketch

Professor Rudy L. Slingerland received his graduate education in geology (MS 1974, PhD 1977) at Pennsylvania State University. He has served as a professor at Penn State for over 25 years. Between 1997 and 2003, he was head of the Department of Geosciences and presently he is the interim associate dean for research, College of Earth and Mineral Sciences. He has mentored 29 MSc and PhD students and received the 2005 Wilson Award for Excellence in Teaching. His research interest is in sedimentary processes and deterministic modeling over a wide variety of environments and timescales. Current projects investigate: (1) clinoforms genesis in the Gulf of Papua, (2) the conditions that give rise to river channel bifurcations, (3) composition of sediment delivered to offshore basins, (4) geometry and internal characteristics of deltas, (5) the role of horizontal motions in orogenic landscapes in the Himalayas, and (6) feedback loops between evolving land-use practices and sediment erosion off the landscape in the Appalachian mountains. Rudy has been closely involved with the Community Surface Dynamics Modeling System (CSDMS) effort from the first hour; he has been part of the organizing committee for the workshops that laid out this initiative and was one of the lead authors on the CSDMS position papers.

Professor James P.M. Syvitski received a PhD in both oceanography and geological sciences (1978) at the University of British Columbia, where he developed a quantitative understanding of particle dynamics across the land–sea boundary. He has had a variety of appointments within Canadian universities (1978–95: University of Calgary, Dalhousie University, University of Laval, Memorial University, and INRS-oceanologie) and was employed as a senior research scientist within the Geological Survey of Canada at the Bedford Institute of Oceanography (1981–95). James served as director of INSTAAR – an Earth and Environmental Systems Institute from 1995 to 2007 and presently holds faculty appointments in geological sciences, applied mathematics, atmosphere and ocean sciences, hydrological sciences, and geophysics. James has over 500 publications, including authorship or co-authorship of 65 peer-reviewed books and has served in various editorial positions for many international journals. Professor Syvitski has taken leadership roles in large international projects (e.g., Sedimentology of Arctic Fiords Experiment (SAFE), Arctic Delta Failure Experiment (ADFEX), SEDFLUX, COLDSEIS, STRATAFORM, EuroSTRATAFORM, and Community Surface Dynamics Modeling System (CSDMS)) and served as an advisor for National Science Foundation (NSF), Office of Naval Research (ONR), Arctic Research Consortium of the U.S. (ARCUS), Land–Ocean Interactions in the Coastal Zone (LOICZ), International Geosphere–Biosphere Programme (IGBP), International Union of Geological Sciences (IUGS), International Union for Quarternary Research (INQUA), SCOR, and Global Water System Project (GWSP), and various energy, mining, and environmental companies. Professor Syvitski works in the forefront of computational geosciences: sediment transport, land–ocean interactions and Earth-surface dynamics and has won numerous awards for his efforts. In 2007, James became the executive director of CSDMS. In 2011, James was appointed by ICSU to be the new chair of the IGBP.