

# Water Science Software Institute: Agile and Open Source Scientific Software Development

**Stan Ahalt, Larry Band, Laura Christopherson, Ray Idaszak, and Chris Lenhardt** | University of North Carolina at Chapel Hill

**Barbara Minsker** | University of Illinois at Urbana-Champaign

**Margaret Palmer** | University of Maryland, College Park

**Mary Shelley** | University of Maryland, College Park

**Michael Tiemann** | Red Hat

**Ann Zimmerman** | Ann Zimmerman Consulting

An Open Community Engagement Process (OCEP) applies open source mechanics and software engineering to water science research. To operationalize OCEP, the authors conceptualize a Water Science Software Institute whose mission is to support and accelerate water science by transforming both the software and research cultures of the water science community.

Research has historically underappreciated software; yet, software is fundamental to research. Both scientists and software engineers experience challenges when using and developing software: the wide range of software types (for example, simulation models, algorithms, analytical tools, digital libraries, and middleware); creation of software development standards; software application complexity; software reuse and sustainability; needs of science software app stores, portals, and interfaces; and costs of development, deployment, and maintenance of new software.

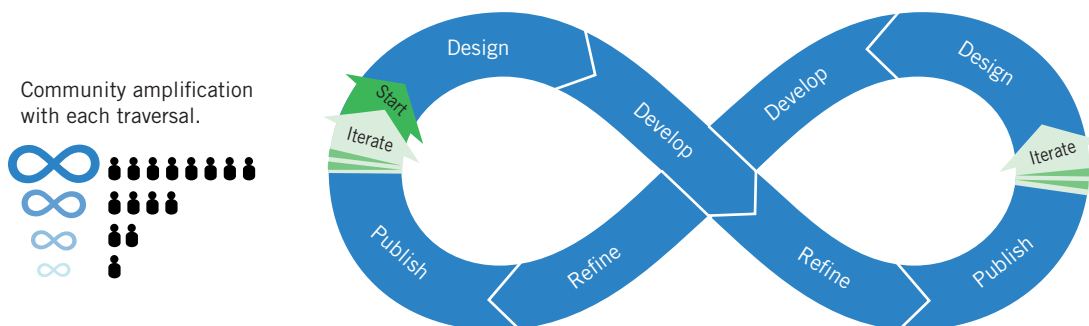
In addition to these issues, the academic research community experiences challenges related to resource allocation, funding, and academic cultural norms. For example, researchers spend enormous amounts of time developing niche, ad hoc software solutions to answer their individual research questions. This software is not applicable to other research questions, datasets, or researcher needs; so it is not sustainable or interoperable with other software developed by the scientific community. Furthermore, the academic incentive system discourages resource sharing because of its focus on individual advancement through publications and citations. Thus, sharing software code

is uncommon. This leads to unnecessary duplication of effort and little opportunity for collaborative code checking and improvement.

In many cases, academic researchers often rely on graduate students and postdoctoral fellows who lack formal training in software engineering or understanding the value thereof. So software is developed without employing proven software engineering practices (such as test-driven development and continuous integration) that help make software more error-free and sustainable. Without proper software code documentation, the software generated by junior-level researchers tends to be difficult to maintain beyond the completion of their fellowships.<sup>1</sup> Furthermore, rigorous testing of software and the implementation of quality assurance practices are rare among the academic research community, even though problems with software code have led to major errors in academic research and even the retraction of published work.<sup>1</sup>

## The Water Science Software Institute

With funding from the National Science Foundation's (NSF) Scientific Software Innovation Institutes program, our team is conceptualizing a Water Science Software Institute (WSSI), which we intend to



**Figure 1.** The Water Science Software Institute's Open Community Engagement Model. The model draws heavily on the combined principles and practices of open source, Agile software development, and open community engagement.

evolve into a full-fledged institute by accomplishing its goals of

- enabling and accelerating new water science,
- increasing the productivity and research capability of the water science research community,
- transforming the research and software culture of the water science community through activities such as educational outreach and software engineer and scientist collaborative software development, and
- establishing an evaluative framework for assessing impact.

WSSI currently partners with nine leading universities and industries, and intends to engage with other large NSF projects, institutes, and programs; local, state, and federal agencies; international entities; additional industry leaders; and nongovernmental organizations.

The WSSI approach hypothesizes that three key components are necessary to accelerate and significantly advance progress within domain science:

- identify and address critical software barriers that inhibit new science;
- diffuse Agile and open source principles, shown to work in software development, into the research process itself; and
- elevate the status, culture, and knowledge of software among domain scientists.

The WSSI tests these hypotheses and addresses these needs via the Open Community Engagement Process (OCEP). The OCEP engages domain and stakeholder communities through an iterative process of science problem and software barrier identification; development and application of scientific software

within the framework of current best practices (for example, Agile software methodology, test-driven development, and so on); and dissemination of scientific results, software, and software development best practices as they apply in the domain community. We describe our process in detail in this article and regard it as the central theme to operationalize the WSSI.

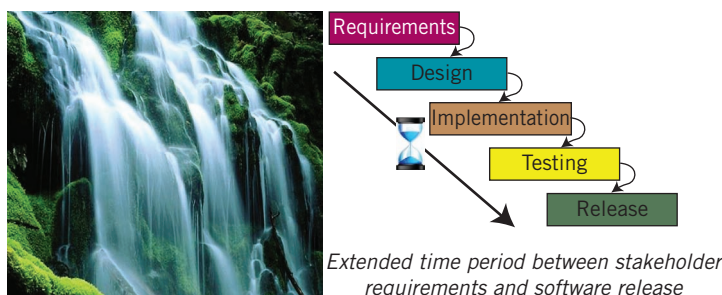
### The WSSI's OCEP Model

Figure 1 depicts the WSSI's OCEP model. It draws heavily on Agile development principles and open source mechanics as well as our previous work.<sup>2</sup> Agile software development assumes that requirements will evolve throughout the life of a project. It accommodates the need for change through an iterative, incremental approach with continual collaboration between stakeholders. Stakeholders include academic researchers; industry leaders; nongovernmental organizations; governmental policymakers; international entities; and federal, state, and local agency officials. At least one software engineer participates. Stakeholders might come together on issues that are scaled over years. Other times, they come together on issues that require more rapid attention.

Open source development espouses values of universal access to and use of software by a community of individuals who are empowered to review, request, and develop improvements to the code. Contributions to the software in the form of code review, suggested improvements, implemented improvements, code extensions, and so on—if valued—result in credibility and reputation gains within the open source community.<sup>3</sup>

OCEP is comprised of four main stages that are repeated when new research questions arise or new scientific communities are engaged (see Figure 1).

To test key parts of this framework, we organize a sprint designed to traverse steps central to the process.



**Figure 2.** The waterfall model of software development. Software is produced through a sequence of cascading phases, with stakeholder involvement only at a project's beginning and end.

The term sprint is derived from a variation of Agile software-development methodology called Scrum.<sup>4</sup> A software sprint begins when the overarching questions identified by the stakeholder and domain community are broken down into smaller, tractable problems that we can address via software development. WSSI staff leading the sprint then make necessary preparations to bring domain scientists and software engineers together to work intensively on a solution.

### Open Source Software Development

The principles and methodology of open source software development can enable and accelerate new, transformative water science, as well as modify academic research culture.<sup>5</sup> R, an open source statistical software (see [www.r-project.org](http://www.r-project.org)), is an example. R is based on user-contributed packages, such as data mining and statistical analysis packages. Users may tailor an existing R package to meet their needs or develop a new R package. These packages are freely published and made available to the community of R users. A given package can serve a group as small as two or as large as thousands of users. Open source methods of modular incremental contributions to software promote increasing participation in development and use of software by a broad class of users, including those with little research funding. Public sharing of R packages provides an open social forum for community commentary on the software, including different contexts for use, enhancements, and bug reports. Peer review of contributed R packages keeps the quality of R packages high and bug-free. In an open source community, bugs are quickly discovered and corrected due to the inherent value of bug-fixing for the individual user and/or the rest of the user community.

As a particular manifestation of a community cyberinfrastructure that promotes R-like open source methodology, we call attention to the HydroShare project.<sup>6</sup> HydroShare is an NSF-funded collaborative

environment for sharing hydrologic data and models. HydroShare expands upon the data-sharing capability of the Hydrologic Information System of the Consortium of Universities for the Advancement of Hydrologic Sciences by broadening the classes of data accommodated, expanding system abilities to include model and model component sharing, and taking advantage of emerging social media functionality. In implementation, HydroShare mimics each of the R-like attributes we described and provides a tangible community portal for use by the interdisciplinary WSSI community.

HydroShare—as separately funded from the WSSI with its own objectives—will not be the only community cyberinfrastructure with these attributes; rather, we are using it as an initial exemplar and necessary vehicle and WSSI component to demonstrate a concrete mechanism by which the WSSI will accelerate science. The key aspect of other R-like and HydroShare-like community cyberinfrastructures is that each has attributes and capabilities that are designed, implemented, and managed to be interoperable.

### Agile Software Development

Agile software development can enable and accelerate new, transformative water science, as well as transform the academic research culture.<sup>7</sup> Agile is an iterative and incremental approach to software development, with continual involvement of stakeholders. Agile software methodologies are replacing the older waterfall model of software development. In the waterfall model, software is produced through a sequence of cascading phases, with stakeholder involvement only at a project's beginning and end (see Figure 2). The waterfall model generally cascades as follows: exhaustive requirements gathering, design, implementation, testing, and release.

For complex software, the gathering of requirements can take months or even years. This creates significant lag between when stakeholders specify the software and when they can first use the software, which may result in software that no longer meets stakeholder expectations. Furthermore, if software developers misinterpret the requirements, the misinterpretation is often not discovered until months or years later, after the code's release.

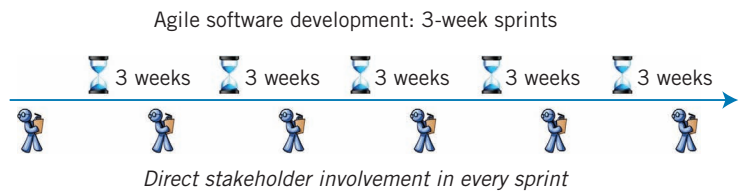
The Agile approach is responsive to situations where requirements are too complex or evolve too rapidly to permit exhaustive requirements gathering before coding begins. Additionally, continual stakeholder involvement ensures that any misinterpretations on the part of the software developers are realigned quickly, with minimal project impact.

While there are different Agile methodologies, a representative Agile approach begins with the identification of representative use cases—written descriptions describing interactions between the user and the system for the purpose of achieving a specific work goal. An Agile sprint lasts typically two to four weeks, with each week constituting an Agile iteration (see Figure 3). An Agile iteration involves the implementation of a specific system feature that can be traced back to one or more use cases. A feature is published to the system only if the team's stakeholder approves it. Otherwise, the software code is rejected in favor of further development until stakeholder approval is received. Stakeholders have the opportunity to comment on the resulting software code and direct course corrections, bug fixes, and so on. This process ensures continual stakeholder alignment and the opportunity for continual innovation. After each iteration, there is an opportunity to review the evolving system, modify the course, and incorporate new use cases in future development efforts.

These Agile methods are integral to the WSSI's OCEP model. They allow for research hypotheses to be incompletely fleshed out at the onset of a research project. Agile permits incremental and iterative research that values and seeks stakeholder input, thereby ensuring continual alignment. Agile encourages incremental results in publishable units (for example, an R-like package) at the end of each research iteration, and it enables more frequent opportunities for innovation and reuse in new use cases.

To suit the research needs of WSSI projects, Agile methodology must be tailored to research timelines. Agile iterations are typically two to four weeks long and team members typically meet briefly on a daily basis. Team meetings provide a quick review of each developer's accomplishments since the previous team meeting, the current day's development plan, and any encountered obstacles. Iterations must be lengthened to accommodate research timelines, and daily team meetings are impractical. Team meetings between domain scientists and software developers must instead occur periodically (perhaps every few weeks or months) and may be irregularly scheduled. Developers, however, may meet more frequently; although it is important that at least one domain scientist remains closely involved in developer meetings to ensure that the intent of the broader stakeholder community is maintained throughout longer iterations.

The Agile approach adopted in the WSSI's OCEP lets software development and academic



**Figure 3.** Schematic of an Agile sprint. An Agile sprint lasts typically two to four weeks, with each week constituting an Agile iteration, involving implementation of a specific system feature.

research progress in a synchronized manner. A community-wide challenge is how to inculcate professional software development practices into the culture of academic researchers who have historically not adhered to or have not been trained in such practices.<sup>8</sup> In the WSSI's OCEP model, the concurrent application of Agile methodologies to both research and software development enables the diffusion of more sustainable software practices into the water science research community.

### Amplification Effect and Change in Culture

The application of Agile and open source methodology to academic research and software development has the additional benefit of having an amplification effect through which a finite number of WSSI staff can broadly change the software and research culture across the water science community, while also enabling and accelerating new, transformative water science. The concept of an amplification effect was first realized by author Michael Tiemann. It is our belief that WSSI (or any software institute) will not be successful in changing the software and research culture of the entire water science community solely through the completion of individual projects. The water science community is too large and the WSSI staff too limited to make an appreciable impact, regardless of how long this approach is attempted. Rather, what is necessary is an amplifying step, depicted on the left side of Figure 1. When a research group works with the WSSI, the Agile and open source structure allows other research groups to rapidly pick up that group's work. Additional research groups would then pick up the derivative work, and so on. The idea, as stated by Tiemann, is that "lots of little ideas, if permitted to freely combine, can themselves be understood to be a really great idea."<sup>9</sup> In this manner, small innovations rapidly develop into large ones. Further, this approach should also encourage emphasizing the value of each innovation as representative of a community rather than emphasizing the value of a single innovator.

### Example Traversal of OCEP

The OCEP model begins with the design stage, where stakeholders begin addressing a pressing social, environmental, or public health need related to water science. An environmental emergency requiring rapid progress on solutions that cannot be addressed with current cybertools may spark an OCEP project. A National Research Council report detailing major water science research questions may also encourage pursuit of OCEP. Other examples include a request for proposals calling for the identification and resolution of a larger question, need, and/or barrier, and/or a community-driven activity such as a workshop implemented with a goal of defining research questions.

These needs begin to be addressed by discussing related research questions and hypotheses. Driving questions incentivize participation in OCEP: the more driving questions in play, the more incentives for community members to participate. The approach applies Agile principles as research questions—and hypotheses are iteratively, openly, and incrementally processed by the community.

After the team identifies research questions and defines working hypotheses, a working group of 10 to 15 people identify any potential barriers to hypothesis testing (for example, inadequate tools, personnel, resources, and/or gaps in expertise). This positions the working group to begin to identify potential solutions and design experiments, analytical approaches, and models for effective hypothesis testing. From these discussions, the team develops requirements (such as use cases), which will then undergo an independent review.

An organization such as the National Socio-Environmental Synthesis Center, a WSSI partner, can facilitate the activities described in the design stage. During design stage activities, participants engage in intense, hands-on sessions that result in distillation or integration of data, ideas, theories, and/or methods from a variety of sources. Creation of a shared conceptual framework and initial code architecture are performed. This pooling of knowledge and ideas helps to address water science issues that might not emerge using traditional methods.

During the development stage, stakeholders collaboratively refactor existing code or develop new code to overcome barriers and satisfy requirements drafted in the design stage. The team performs a formal code review. A dedicated documentation professional ensures high-quality documentation. One way WSSI has approached the development stage is through a one-week hackathon, where a group of software developers

collaborate on developing working code that is of utility to the community as a whole.<sup>10</sup>

A milestone is reached in the development stage when new water science and cyberinfrastructure capabilities are realized, recorded, and incrementally improved as a result of new working code production. This code can contribute to hypothesis refinement and new theory development.

In the refine stage, stakeholders reexamine driving questions, hypotheses, and analytical results in the context of the emerging answer or solution. Potential new science, if possible, is identified. Mirroring Agile, further code modifications can be made to support the emergence of new questions and hypotheses or support revised questions and hypotheses. Agile and open source principles are applied as stakeholders continually assess and revise the working code and introduce new use cases.

In the final stage of this OCEP iteration, participants publish data, software, research findings, and lessons learned from developing the software. This will amplify and extend OCEP's impact to a larger water science community. In keeping with open source ideals, participants can exploit social media for ratings and commentary on the product's usefulness in specific contexts.

The iterate arrow shown on the right side of Figure 1 indicates repetition of all four stages that includes community distribution, testing, replication, and refinement of published software, data, and methods through open sharing of products. It signifies a new iteration of OCEP where scientists may revise existing questions and hypotheses, or develop new questions and hypotheses for the purpose of producing new or refined results. The community may modify published products for different applications or contexts. For example, the community members may discover code for southern climate modeling, modify it for application in a northern climate, and publish it for use by the larger community. In another example, multiple experimental watersheds within a network (for example, the Long-Term Ecological Research Network or Critical Zone Observatory sites) might implement and test the new cyberinfrastructure at their sites. The community may continue to use and develop the original work and any derivatives, resulting in amplification and extension to an ever-increasing number of individuals.

With each iteration of OCEP, incentives for participation continue to arise. Opportunities for innovation continue to emerge, and the number of users developing original research increases. This body of increasing knowledge is disseminated to the



community through workshops, conference talks, and publications.

Implementation of WSSI's OCEP might prove challenging because of the many differences between the academic research community and the traditional software engineering community. For instance, the academic research community tends to be motivated by research productivity, while the software engineering community, primarily situated in the commercial sector, is motivated by profit margins. Additionally, we might encounter resistance from academic researchers who are not used to sharing code or testing it via Agile and more formal software engineering processes. We anticipate that these and other challenges will be overcome as academic researchers recognize the value of OCEP through the use of OCEP-developed software and OCEP-supported scientific findings. Through continual evaluation of our efforts, we plan to identify the strengths and weaknesses of OCEP so that improvements can be made to better support scientific inquiry and development.

### Initial Results

An initial OCEP traversal occurred from February to June 2013, representing the OCEP design (February–March), development (April), and refine (May–June) steps. The development step culminated in a hackathon. The development involved a hydroecological modeling framework—Regional Hydro-Ecologic Simulation System (Rhessys)—originally developed in 1993.<sup>11</sup> Rhessys processes digital terrain data, remote sensing data, and digitized soil maps to automate the simulation of hydrological and ecological flux and storage processes. It is written in C and available on GitHub (<https://github.com>).

Rhessys was the perfect case study for an initial evaluation of the OCEP development step. A committed user-base already existed for Rhessys, and it commands strong interest from WSSI members. The WSSI conceptualization proposal submitted to the NSF suggested that WSSI activities might begin by furthering the study of green infrastructure impacts on stormwater transport and its effects on an urban ecosystem. Rhessys was designed for this type of study, but could benefit from improvements so it could more fully address this topic and become more usable by a larger community. The water scientists interested in enhancing Rhessys came with well-defined research questions and extensive experience with the framework. Additionally, detailed field data in urban ecosystems existed and could be used to test new and improved functionality.

The hackathon team included 10 domain scientists, five software engineers, and six other WSSI members, including an evaluator and a computer scientist. Nineteen of these participants were co-located in Chapel Hill, North Carolina, and two participated virtually.

We evaluated the hackathon by observation and developed a survey based on the literature and on outcomes that hackathon organizers aimed to achieve. We administered the survey to the 16 software engineers, domain scientists, and computer scientists who were the primary participants. Fifteen of the 16 surveys were returned for a response rate of 94 percent.

Evaluation focused on three main questions:

1. Did the hackathon prove effective for developing scientific software? Were Rhessys capabilities improved?
2. Were better software engineering practices understood by the scientists involved?
3. Were better software engineering practices used by the scientists involved?

The hackathon exceeded expectations. In answer to the first question, results showed that new and improved Rhessys functionality was developed, including a Web-GIS for visualizing flow tables and integration with the National Center for Supercomputing Applications' Cyberintegrator—a workflow application used for Web service development and provisioning the automation of analytical processes. Significant progress was made on routines for manual rerouting of stormwater flow, and the ability to add and visualize trees at the parcel level.

The create flowpaths (CF9) functionality, which accounts for nontopographic drainage of rooftops, was successfully refactored. Its optimization has led to an exponential improvement in calculation speed. Prior to the first hackathon, calculations took  $O(n^2)$  but now take  $O(n \times \log n)$  time to complete. Researchers can now obtain results in minutes rather than hours.

In answer to the second question about scientist understanding of software engineering practices, results showed that the software engineers identified software testing (for example, unit and integration testing) and software requirements (such as eliciting, analyzing, specifying, and prioritizing functional and nonfunctional requirements) as two key concepts that scientists should know to produce higher-quality software. Most of the software engineers reported that hackathons are an effective way for scientists to increase their knowledge of standard

practice in software engineering. They thought that the scientists had improved their knowledge because of the hackathon. This coincided with the self-reports of the domain scientists who believed their knowledge had increased because of the hackathon.

Many of the participating domain scientists learned new computational techniques (for example, using the Google Street View API and basic Python scripting) and confirmed their understanding of test-driven development, continuous integration, and their importance. However, it is unclear how in-depth their knowledge of these best practices were for a couple of reasons. First, mini-teams employing these practices possessed pre-existing knowledge of them and mini-teams not employing these practices worked on activities that did not require them. Second, although scientists were exposed to the idea of test-driven development during the hackathon, the team did not write test cases prior to the hackathon as suggested by the OCEP design step. Instead, the team wrote test cases during the hackathon more as a form of documentation rather than as guides to development. Thus, test cases were not as thorough as would be required if they were intended to drive development rather than document development. In the next OCEP traversal, we will emphasize the importance of test-driven development earlier in the OCEP cycle and encourage completion of test cases in advance.

In answer to the third question about scientist use of software engineering best practices, results showed that collaborating with software engineers changed how the scientists typically work. The team established a new code repository and a continuous integration system for building, testing, and tracking new code. Participants established bug reports and code metrics that, in combination with the continuous integration testing system, enabled code coverage analysis. These systems and practices had not been implemented prior to this. They allow both software engineers and domain scientists to coordinate their programming activities as code is hardened and features are added to Rhessys, which will lead to a more sustainable, robust product in the future.

The evaluation results also point to factors that influence the planning and organization of hackathons. These include participant motivation, knowledge, and expertise; support and encouragement from leadership; and planning advance activities, such as creation of a shared conceptual framework, to establish mutual understanding prior to coding. This team was motivated to succeed and represented a breadth and depth of knowledge and expertise in both the science and the software

engineering. We believe these factors contributed to the sprint's success.

The leadership team was supportive of the hackathon as a risk-taking endeavor. The knowledge that this would be the first attempt at a hackathon for the WSSI team unified and inspired participants to view the first hackathon as an experimental, learning endeavor. Unanticipated obstacles and mistakes were expected to arise, which would require the group to work together to resolve, and were so viewed as learning opportunities. The spirit of adventure that underpinned this hackathon contributed to the feelings of satisfaction by participants.

As a part of WSSI, we have developed a model for scientific software development that we believe will enable a finite number of scientists and engineers to broadly change the software and research culture across the water science community, resulting in advancement and acceleration of new, transformative water science. Our model is based in large part on Agile and open source software development principles. We further believe that our model can be applied more broadly to NSF's Scientific Software Innovation Institutes program to further software development and research in other scientific disciplines. ■

### Acknowledgments

This work is funded by NSF award 1216817 Conceptualization of a WSSI and awards 1148453 and 1148090 HydroShare: An Interactive Software Infrastructure for Sustaining Collaborative Community Innovation in the Hydrologic Sciences. Any opinions, findings, conclusions, or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the NSF.

### References

1. J. Hsu and Innovation News Daily, "Secret Computer Code Threatens Science," *Scientific Am.*, 13 Apr. 2012; [www.scientificamerican.com/article/secret-computer-code-threatens-science](http://www.scientificamerican.com/article/secret-computer-code-threatens-science).
2. S. Ahalt et al., "Water Science Software Institute: An Open Source Engagement Process," *Proc. IEEE Int'l Workshop Software Eng. for Computational Science and Eng.*, 2013, pp. 40–47.
3. E.S. Raymond, *The Cathedral and the Bazaar*, O'Reilly, 1999.
4. "Scrum Sprint," Oct. 2009; <http://scrummethodology.com/scrum-sprint>.
5. G. Wilson et al., "Best Practices for Scientific Computing," Cornell Univ. Library, 2012; <http://arxiv.org/abs/1210.0530>.

6. D. Tarboton et al., "HydroShare: An Online, Collaborative Environment for the Sharing of Hydrologic Data and Models," *3rd Biennial Colloquium on Hydrologic Science and Engineering*, 2012; [www.cuahsi.org/pagefiles/slides-from-presentation-to-cuahsi-colloquium-july-2012.pptx](http://www.cuahsi.org/pagefiles/slides-from-presentation-to-cuahsi-colloquium-july-2012.pptx).
7. Agile Alliance, "What is Agile Software Development?" 2012; [www.Agilealliance.org/the-alliance/what-is-Agile](http://www.Agilealliance.org/the-alliance/what-is-Agile).
8. Z. Merali, "...ERROR...Why Scientific Programming Does Not Compute," *Nature*, vol. 467, 2010, pp. 775–777.
9. M. Tiemann, "Amplifying Creativity and Business Performance with Open Source," 16 Feb. 2010; [www.opensource.com/business/10/2/amplifying-creativity-and-business-performance-open-source](http://www.opensource.com/business/10/2/amplifying-creativity-and-business-performance-open-source).
10. GMOD EvoHack Organizing Committee, "GMOD Evo Hackathon Open Call," 2010; [www.gmod.org/wiki/GMOD\\_Evo\\_Hackathon\\_Open\\_Call](http://www.gmod.org/wiki/GMOD_Evo_Hackathon_Open_Call).
11. L. Band et al., "Forest Ecosystem Processes at the Watershed Scale: Incorporating Hillslope Hydrology," *Agricultural and Forest Meteorology*, vol. 63, 1993, pp. 93–126.

---

**Stan Ahalt** is director of the Renaissance Computing Institute (RENCI), professor in the computer science department, and principal investigator of the NSF Conceptualization of a WSSI award at the University of North Carolina at Chapel Hill. His research interests include signal processing, high-performance scientific and industrial computing, pattern recognition applied to national security problems, and high-productivity domain specific languages. Ahalt has a PhD in electrical and computer engineering from Clemson University. He is a member of Microsoft's Technical Computing Advisory Committee, the Council on Competitiveness High-Performance Computing Advisory Committee, a National Lambda Rail board member, and chairs the National Consortium for Data Science steering committee. Contact him at [ahalt@cs.unc.edu](mailto:ahalt@cs.unc.edu).

---

**Larry Band** is the Voit Gilmore Distinguished Professor of Geography, and director of the Institute for the Environment, and co-principal-investigator on the NSF Conceptualization of a WSSI award at the University of North Carolina at Chapel Hill. His research interests include ecohydrology of watersheds, with an emphasis on numerical modeling and geographic information analysis of coupled cycling of water, carbon, and nutrients; coevolution of forest ecological and hydrological systems; and human/environment interactions. Band has a PhD in geology from the University of California, Los Angeles. He was named the 2014 Birdsall-Dreiss Lecturer. Contact him at [lband@email.unc.edu](mailto:lband@email.unc.edu).

---

**Laura Christopherson** works on the WSSI project for RENCI. Her research explores the ways in which technology influences communication and information behavior, and how people respond creatively and adapt to new technology. Christopherson holds a doctorate in information science from the School of Information and Library Science at the University of North Carolina at Chapel Hill. Contact her at [laura@renci.org](mailto:laura@renci.org).

---

**Ray Idaszak** is the director of collaborative development environments at RENCI at the University of North Carolina at Chapel Hill, and co-principal-investigator on the NSF HydroShare award. His research interests include diffusing advanced software engineering, open source, and visualization techniques in large sponsored research projects, including the NSF WSSI and HydroShare project awards. Idaszak has a BS in computer science engineering from the University of Illinois Urbana-Champaign. Contact him at [rayi@renci.org](mailto:rayi@renci.org).

---

**Chris Lenhardt** is the domain scientist for environmental data sciences and systems at RENCI at the University of North Carolina at Chapel Hill. His research interests include helping to create knowledge-management frameworks for science data and information, and studying the implications of emerging technologies. He is active in the Federation for Earth Science Information Partners. Lenhardt has an MS in international relations from The London School of Economics and Political Science. Contact him at [clenhardt@renci.org](mailto:clenhardt@renci.org).

---

**Barbara Minsker** is a professor and the Arthur and Virginia Nauman Faculty Scholar in the Department of Civil and Environmental Engineering at the University of Illinois at Urbana-Champaign, and co-principal-investigator of the NSF Conceptualization of a WSSI award. Her research interests include using information technology to improve understanding and management of complex environmental systems, with a focus on water and sustainability. Minsker has a PhD in civil and environmental engineering from Cornell University. She has received the NSF CAREER Award, Presidential Early Career Award for Scientists and Engineers, American Society for Civil Engineers' Walter L. Huber Civil Engineering Research Prize, and Xerox Award for Faculty Research. Contact her at [minsker@illinois.edu](mailto:minsker@illinois.edu).

---

**Margaret Palmer** is a professor at the University of Maryland, College Park, director of the National Socio-Environmental Synthesis Center, and co-principal-investigator on the NSF Conceptualization of a WSSI award. Her research interests include watershed restoration ecology. Palmer has a PhD in coastal oceanography from the



University of South Carolina. She is an American Association for the Advancement of Science fellow, Aldo Leopold Leadership fellow, and Ecological Society of America elected fellow. Contact her at [mpalmer@umd.edu](mailto:mpalmer@umd.edu).

**Mary Shelley** is assistant director of computational synthesis at the National Socio-Environmental Synthesis Center. Her research interests include research computing, collaboration between interdisciplinary science teams and computing specialists, computational capacity among scientists, and computing support to advance research goals. Shelley has an MA in geography from the University of Maryland, College Park. Contact her at [mshelley@sesync.org](mailto:mshelley@sesync.org).

**Michael Tiemann** is vice president of open source affairs at Red Hat. In his current role, he integrates and informs technology and open source strategies for Red Hat and its partners and customers in the public and private sectors. He is the original author of the GNU C++ compiler, former Chief Technology Officer for Red Hat, and co-founder of Cygnus Solutions—the first open source business model. Tiemann has a BS in computer science

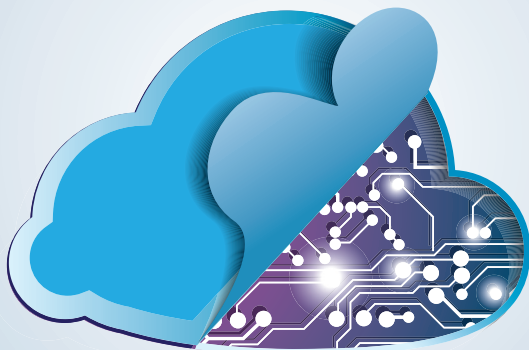
engineering from the University of Pennsylvania. He is a board member of the Open Source Initiative, GNOME Foundation, and Open Source for America. Contact him at [tiemann@redhat.com](mailto:tiemann@redhat.com).


**Ann Zimmerman** is an independent consultant specializing in program planning and evaluation for science and technology projects and organizations at Ann Zimmerman Consulting. Her research interests include collaboration, interdisciplinary research, and the influence of new technologies on the practice and organization of science and the production of new knowledge. Zimmerman has a PhD in information from the University of Michigan. Contact her at [annzconsulting@gmail.com](mailto:annzconsulting@gmail.com).



*Selected articles and columns from IEEE Computer Society publications are also available for free at <http://ComputingNow.computer.org>.*

New in 2014



IEEE  computer society

# IEEE CLOUD COMPUTING

IEEE Computer Society's newest magazine tackles the emerging technology of cloud computing.

Subscribe today!

[computer.org/  
cloudcomputing](http://computer.org/cloudcomputing)

IEEE  
COMMUNICATIONS  
SOCIETY