Software Carpentry Bibliography

September 1, 2015

References

[1] Karen S. Ackroyd, Steve H. Kinder, Geoff R. Mant, Mike C. Miller, Christine A. Ramsdale, and Paul C. Stephenson. Scientific Software Development at a Research Facility. *IEEE Software*, pages 44–51, July-August 2008.

[2] Victor R. Basili, Danelia Cruzes, Jeffrey C. Carver, Lorin M. Hochstein, Jeffrey K. Hollingsworth, Marvin V. Zelkowitz, and Forrest Shull. Understanding the High-Performance-Computing Community: A Software Engineer's Perspective. *IEEE Software*, 25(4):29–36, July 2008.

[3] Susan M. Baxter, Steven W. Day, Jacquelyn S. Fetrow, and Stephanie J. Reisinger. Scientific Software Development Is Not an Oxymoron. *PLoS Computational Biology*, 2(9):e87, 2006.

[4] Martin Blom. Is scrum and xp suitable for cse development? *Procedia Computer Science*, 1(1):1511 – 1517, 2010.

[5] Ronald F. Boisvert and Ping Tak Peter Tang, editors. *The Architecture of Scientific Software*. Springer, 2001.

[6] Alan Calder, Jonathan Dursi, Bruce Fryxell, Tomek Plewa, Greg Weirs, Todd Dupont, Harry Robey, Jave Kane, Bruce Remington, Frank Timmes, Guy Dimonte, John Hayes, Mike Zingale, Paul Drake, Paul Ricker, Jim Stone, and Kevin Olson. Validating Astrophysical Simulation Codes. *Computing in Science & Engineering*, 6(5):10–20, 2004.

[7] J. C. Carver. Development of a Mesh Generation Code with a Graphical Front-End: A Case Study. *Journal of Organizational and End-User Computing*, page 16, 2011.

[8] Jeffrey Carver, Lorin Hochstein, Richard Kendall, Taiga Nakamura, Marvin Zelkowitz, Victor Basili, and Douglass Post. Observations about Software Development for High End Computing. *CTWatch Quarterly*, 2(4A):33–38, November 2006.

[9] Jeffrey C. Carver. Post-Workshop Report for the Third International Workshop on Software Engineering for High Performance Computing Applications. *ACM Software Engineering Notes*, 11(4):38–43, 2007.

[10] Jeffrey C. Carver. First international workshop on software engineering for computational science and engineering (secse08). In *Proceedings of the 30th International Conference on Software Engineering (ICSE08)*, pages 1071–1072. ACM, 2008.

[11] Jeffrey C. Carver. Second international workshop on software engineering for computational science and engineering (secse09). In *Proceedings of the 31st International Conference on Software Engineering (ICSE09)*, pages 484–485. IEEE, 2009.

[12] Jeffrey C. Carver, Richard P. Kendall, Susan E. Squires, and Douglass E. Post. Software Development Environments for Scientific and Engineering Software: A Series of Case Studies. In *Proceedings of the 29th International Conference on Software Engineering (ICSE07)*, 2007.

[13] Jeffrey Clark Carver. Report from the Second International Workshop on Software Engineering for Computational Science and Engineering. *Computing in Science & Engineering*, 11(6):14–19, 2009.

[14] T. L. Clune and K. Kuo. Test Driven Development: Lessons from a Simple Scientific Model. *AGU Fall Meeting Abstracts*, page A1100, December 2010.

[15] Carlton A. Crabtree, A. Gunes Koru, Carolyn Seaman, and Hakan Erdogmus. An Empirical Characterization of Scientific Software Development Projects According to the Boehm and Turner Model: A Progress Report. In *Second International Workshop on Software Engineering for Computational Science and Engineering (SECSE09)*, pages 22–27, 2009.

[16] Bronis R. de Supinski, Jeffrey K. Hollingworth, Shirley Moore, and Patrick H. Worley. Results of the PERI survey of SciDAC applications. *Journal of Physics: Conference Series*, 2007.

[17] Glenn Downing, Paul F. Dubois, and Teresa Cottom. Data Sharing in Scientific Simulations. *Computing in Science & Engineering*, 6(3):87–96, May-June 2004.

[18] P. F. Dubois. Maintaining Correctness in Scientific Programs. *Computing in Science & Engineering*, 7(3):80–85, May-June 2005.

[19] P. F. Dubois, T. Epperly, and G. Kumfert. Why Johnny Can't Build (Portable Scientific Software). *Computing in Science & Engineering*, 5(5):83–88, 2003.

[20] Paul F. Dubois. Designing Scientific Components. *Computing in Science & Engineering*, 4(5):84–90, September 2002.

[21] Steve M. Easterbrook and Timothy C. Johns. Engineering the Software for Understanding Climate Change. *Computing in Science & Engineering*, 11(6):65–74, November-December 2009.

[22] Steven L. Eddins. Automated Software Testing for Matlab. *Computing in Science & Engineering*, 11(6):48–55, 2009.

[23] S. Faulk, J. Gustafson, P. M. Johnson, A. Porter, W. Tichy, and Lawrence Votta. Measuring HPC Productivity. *International Journal of High Performance Computing Applications*, 18(4), 2004.

[24] S. Faulk, J. Gustafson, P. M. Johnson, W. Tichy, Lawrence Votta, and A. Porter. Toward Accurate HPC Productivity Measurement. In *Proceedings of the 26th International Conference on Software Engineering (ICSE04)*, 2004.

[25] Stuart Faulk, Eugene Loh, Michael L. Van De Vanter, Susan Squires, and Lawrence G. Votta. Scientific Computing's Productivity Gridlock: How Software Engineering Can Help. *Computing in Science & Engineering*, 11(6):30–39, 2009.

[26] Matthew Gentzkow and Jesse M. Shapiro. Code and data for the social sciences: A practitioner's guide, January 2014.

[27] Yolanda Gil, Pedro A. González-Calero, and Ewa Deelman. On the Black Art of Designing Computational Workflows. In *2nd Workshop on Workflows in Support of Large-Scale Science*, pages 53–62. ACM, 2007.

[28] Robert Gray and Diane Kelly. Investigating Test Selection Techniques for Scientific Software Using Hook's Mutation Sensitivity Testing. In *Third International Workshop on Software Engineering for Computational Science and Engineering (SECSE10)*, May 2010.

[29] C. Greenough and D. J. Worth. Computational Science and Engineering Department Software Development Best Practice. Technical Report RAL-TR-2008-022, SFTC Rutherford Appleton Laboratory, 2008.

[30] J. Gustafson. Purpose-Based Benchmarks. *International Journal of High Performance Computing Applications*, 18(4):475–487, 2004.

[31] Jo Erskine Hannay, Hans Petter Langtangen, Carolyn MacLeod, Dietmar Pfahl, Janice Singer, and Greg Wilson. How Do Scientists Develop and Use Scientific Software? In *Second International Workshop on Software Engineering for Computational Science and Engineering (SECSE09)*, 2009.

[32] L. Hatton. The T Experiments: Errors in Scientific Software. *Computational Science & Engineering*, 4(2):27–38, 1997.

[33] L. Hatton and A. Roberts. How Accurate is Scientific Software? *IEEE Transactions on Software Engineering*, 20(10):785–797, 1994.

[34] Michael A. Heroux. Improving CSE Software Through Reproducibility Requirements. In *Fourth International Workshop on Software Engineering for Computational Science and Engineering (SECSE11)*, pages 28–31, 2011.

[35] Michael A. Heroux and James M. Willenbring. Barely-Sufficient Software Engineering: 10 Practices to Improve Your CSE Software. In *Second International Workshop on Software Engineering for Computational Science and Engineering (SECSE09)*, 2009.

[36] L. Hochstein and V. R. Basili. The ASC-Alliance Projects: A Case Study of Large-Scale Parallel Scientific Code Development. *IEEE Computer*, 41(3):50–58, March 2008.

[37] L. Hochstein, J. Carver, F. Shull, S. Asgari, V. R. Basili, J. Hollingsworth, and M. Zelkowitz. Parallel Programmer Productivity: A Case Study of Novice HPC Programmers. In *Proceedings of Supercomputing 2005 (SC05)*, 2005.

[38] Lorin M. Hochstein, Forrest Shull, and Lynn B. Reid. The Role of MPI in Development Time: a Case Study. In *Proceedings of Supercomputing 2008 (SC08)*, pages 1–10, 2008.

[39] Daniel Hook and Diane Kelly. Mutation Sensitivity Testing. *Computing in Science & Engineering*, 11(6):40–47, 2009.

[40] Daniel Hook and Diane Kelly. Testing for Trustworthiness in Scientific Software. In *Second International Workshop on Software Engineering for Computational Science and Engineering (SECSE09)*, May 2009.

[41] James Howison and James D. Herbsleb. Scientific Software Production. In *Proceedings of the Computer Support for Cooperative Work 2011*, 2011.

[42] Jeffrey N. Johnson and Paul F. Dubois. Issue Tracking. *Computing in Science & Engineering*, 5(6):71–77, November 2003.

[43] Philip M. Johnson and Michael G. Paulding. Understanding HPC Development through Automated Process and Product Measurement with Hackystat. In *Second Workshop on Productivity and Performance in High-End Computing*, 2005.

[44] M. Jones and C. Scaffidi. Obstacles and opportunities with using visual and domain-specific languages in scientific programming. In *Visual Languages and Human-Centric Computing 2011 (VLHCC11)*, pages 9–16, September 2011.

[45] Lucas N. Joppa, Greg McInerny, Richard Harper, Lara Salido, Kenji Takeda, Kenton O'Hara, David Gavaghan, and Stephen Emmott. Troubling Trends in Scientific Software Use. *Science*, 340(6134):814–815, 2013.

[46] David Kane. Introducing Agile Development into Bioinformatics: An Experience Report. In *Proceedings of the Agile Development Conference 2005*, 2005.

[47] David Kane, Moses Hohman, Ethan Cerami, Michael McCormick, Karl Kuhlmman, and Jeff Byrd. Agile Methods in Biomedical Software Development: a Multi-Site Experience Report. *BMC Bioinformatics*, 7(1):273, 2006.

[48] Diane Kelly. A Study of Design Characteristics in Evolving Software Using Stability as a Criterion. *IEEE Transactions on Software Engineering*, 32(5):315–329, May 2006.

[49] Diane Kelly. An Analysis of Process Characteristics for Developing Scientific Software. *Journal of Organizational and End-User Computing*, 23(4):63–78, December 2011.

[50] Diane Kelly, Nancy Cote, and Terry Shepard. Software Engineers and Nuclear Engineers: Teaming up to do Testing. In *Proceedings of the Canadian Nuclear Society Conference*, June 2007.

[51] Diane Kelly, Robert Gray, and Yizhen Shao. Examining Random and Designed Tests to Detect Code Mistakes in Scientific Software. *Journal of Computational Science*, 2(1):47–56, March 2011.

[52] Diane Kelly and John Harauz. Software Development Processes and Analysis Software: A Mismatch and a Novel Framework. In *Proceedings of the Canadian Nuclear Society Conference*, 2011.

[53] Diane Kelly, Daniel Hook, and Rebecca Sanders. Five Recommended Practices for Computational Scientists Who Write Software. *Computing in Science & Engineering*, 11(5):48–53, 2009.

[54] Diane Kelly, Daniel Hook, and Rebecca Sanders. A Framework for Testing Computational Software. In J. Leng and W. Sharrock, editors, *Handbook of Research on Computational Science and Engineering: Theory and Practice*, pages 177–196. IGI Global, 2011.

[55] Diane Kelly and Rebecca Sanders. Mismatch of Strategies: Scientific Researchers and Commercial Software Suppliers, July 2007.

[56] Diane Kelly and Rebecca Sanders. Assessing the Quality of Scientific Software. In *First International Workshop on Software Engineering for Computational Science and Engineering (SECSE08)*, May 2008.

[57] Diane Kelly and Terry Shepard. A Little Knowledge about Software. *IEEE Software*, pages 46–48, March-April 2004.

[58] Diane Kelly and Terry Shepard. Eight Maxims for Software Code Inspections. *Journal of Software Testing, Verification, and Reliability*, 14(4):243–256, December 2004.

[59] Diane Kelly and Terry Shepard. Task-Directed Inspection. *Journal of Systems and Software*, 73(2):361–368, October 2004.

[60] Diane Kelly, Spencer Smith, and Nicholas Meng. Software Engineering for Scientists. *Computing in Science & Engineering*, 13(5):7–11, 2011.

[61] Diane Kelly, Stefan Thorsteinson, and Daniel Hook. Scientific Software Testing: Analysis in Four Dimensions. *IEEE Software*, pages 84–90, May-June 2011.

[62] Diane F. Kelly. A Software Chasm: Software Engineering and Scientific Computing. *IEEE Software*, 24:120, 118–119, 2007.

[63] R. Kendall, J. C. Carver, D. Fisher, D. Henderson, A. Mark, D. Post, C. E. Rhoades Jr, and S. Squires. Development of a Weather Forecasting Code: A Case Study. *IEEE Software*, 25(4):59–65, 2008.

[64] R. P. Kendall, J. Carver, A. Mark, D. Post, S. Squires, and D. Shaffer. Case Study of the Hawk Code Project. Technical Report LA-UR-05-9011, Los Alamos National Laboratory, 2005.

[65] R. P. Kendall, D. Post, S. Squires, and J. Carver. Case Study of the Eagle Code Project. Technical Report LA-UR-06-1092, Los Alamos National Laboratory, 2006.

[66] Richard Kendall, Andrew Mark, Douglass Post, Susan Squires, and Christine Halverson. Case Study of the Condor Code Project. Technical report, Los Alamos National Laboratory, 2005.

[67] Richard P. Kendall, Andrew Mark, Susan E. Squires, and Christine A. Halverson. Condor: Case Study of a Large-Scale, Physics-Based Code Development Project. *Computing in Science & Engineering*, 12(3):22–27, 2010.

[68] Richard P. Kendall, Douglass E. Post, and Andrew Mark. Case Study of the Nene Code Project. *Computing in Science & Engineering*, 12(3):28–33, 2010.

[69] Sarah Killcoyne and John Boyle. Managing Chaos: Lessons Learned Developing Software in the Life Sciences. *Computing in Science & Engineering*, 11(6):20–29, 2009.

[70] D. J. Kuck. Productivity in High Performance Computing. *International Journal of High Performance Computing Applications*, 18(4), 2004.

[71] G. Kumfert and T. Epperly. Software in the DOE: The Hidden Overhead of "The Build". Technical Report UCRL-ID-147343, Lawrence Livermore National Lab., February 2002.

[72] Hans Petter Langtangen, A. M. Bruaset, and Ewald Quak. *Advances in Software Tools for Scientific Computing*. Springer, 1999.

[73] Y. Li. Reengineering a scientific software and lessons learned. In *Fourth International Workshop on Software Engineering for Computational Science and Engineering (SECSE11)*, pages 41–45, 2011.

[74] Patrick Martin, Anatol Kark, and Darlene Stewart, editors. *2nd Workshop on Software Engineering for Science*, November 2009.

[75] David Matthews, Greg Wilson, and Steve Easterbrook. Configuration Management for Large-Scale Scientific Computing at the UK Met Office. *Computing in Science & Engineering*, November-December 2008.

[76] Nicholas Jie Meng, Diane Kelly, and Thomas R. Dean. Towards the Profiling of Scientific Software for Accuracy. In *Proceedings of the IBM CASCON 2011*, November 2011.

[77] Zeeya Merali. Error: Why Scientific Programming Does Not Compute. *Nature*, 467:775–777, 2010.

[78] C. Morris. Some lessons learned reviewing scientific code. In *Proceedings of the 30th International Conference on Software Engineering (ICSE08)*, 2008.

[79] C. Morris and J. Segal. Some Challenges Facing Scientific Software Developers: the Case of Molecular Biology. In *5th International IEEE Conference on E-Science*, pages 216–222, 2009.

[80] R. Mugridge. Test Driven Development and the Scientific Method. In *Proceedings of the Agile Development Conference 2003*, pages 47–52, 2003.

[81] Luke Nguyen-Hoan, Shayne Flint, and Ramesh Sankaranarayana. A Survey of Scientific Software Development. In *Proceedings of the ACM-IEEE Internation Symposium on Empirical Software Engineering and Measurement 2010*, 2010.

[82] Dianne P. O'Leary. Computational Software: Writing Your Legacy. *Computing in Science & Engineering*, 8(1):78–83, 2006.

[83] C. M. Pancake and C. Cook. What Users Need in Parallel Tool Support: Survey Results and Analysis. In *Proceedings of the Scalable High-Performance Computing Conference*, 1994.

[84] Victor Pankratius, Ali Jannesari, and Walter F. Tichy. Parallelizing Bzip2: A Case Study in Multicore Software Engineering. *IEEE Software*, 26(6):70–77, 2009.

[85] Joe Pitt-Francis, Miguel O. Bernabeu, Jonathan Cooper, Alan Garny, Lee Momtahan, James Osborne, Pras Pathmanathan, Blanca Rodriguez, Jonathan P. Whiteley, and David J. Gavaghan. Chaste: Using Agile Programming Techniques to Develop Computational Biology Software. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1878):3111–3136, September 2008.

[86] D. E. Post and R. P. Kendall. Large-Scale Computational Scientific and Engineering Code Development and Production Workflows. In *Proceedings of the 12th Workshop on Use of High Performance Computing in Meteorology*, October 2006.

[87] D. E. Post and Richard P. Kendall. Software Project Management and Quality Engineering Practices for Complex, Coupled Multi-Physics, Massively Parallel Computational Simulations: Lessons Learned from ASCI. Technical Report LA-UR-03-1274, Los Alamos National Laboratory, 2003.

[88] D. E. Post, Richard P. Kendall, and Robert F. Lucas. The Opportunities, Challenges and Risks of High Performance Computing in Computational Science and Engineering. *Advances in Computers*, pages 240–297, 2006.

[89] Douglass E. Post and Lawrence G. Votta. Computational Science Demands a New Paradigm. *Physics Today*, 58(1):35–41, January 2005.

[90] Yann Pouillon, Jean-Michel Beuken, Thierry Deutsch, Marc Torrent, and Xavier Gonze. Organizing Software Growth and Distributed Development: The Case of Abinit. *Computing in Science & Engineering*, 13(1):62–69, 2011.

[91] Prakash Prabhu, Thomas B. Jablin, Arun Raman, Yun Zhang, Jialu Huang, Hanjun Kim, Nick P. Johnson, Feng Liu, Soumyadeep Ghosh, Stephen Beard, Taewook Oh, Matthew Zoufaly, David Walker, and David I. August. A Survey of the Practice of Computational Science. In *Proceedings of the 24th ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis*, 2011.

[92] James Quirk. Computational Science: Same Old Silence, Same Old Mistakes, Something More Is Needed.... In Tomasz Plewa, Timur Linde, and V. Gregory Weirs, editors, *Adaptive Mesh Refinement: Theory and Applications*, volume 41 of *Lecture Notes in Computational Science and Engineering*, pages 3–28. Springer, 2005.

[93] Karthik Ram. Git can facilitate greater reproducibility and increased transparency in science. *Source Code for Biology and Medicine*, 8(1):7, 2013.

[94] Patrick J. Roache. Building PDE Codes to be Verifiable and Validatable. *Computing in Science & Engineering*, 6(5):30–38, 2004.

[95] A. Rodman and M. Brorsson. Programming Effort vs. Performance with a Hybrid Programming Model for Distributed Memory Parallel Architectures. In P. Amestoy, P. Berger, M. Daydé, I. Duff, V. Frayssé, L. Giraud, and D. Ruiz, editors, *Euro-Par'99: 5th International Euro-Par Conference*, volume 1685, pages 888–898. Springer, September 1999.

[96] R. Sanders. The Development and Use of Scientific Software. Master's thesis, Queen's University, 2008.

[97] R. Sanders and D. Kelly. Dealing with Risk in Scientific Software Development. *IEEE Software*, 25(4):21–28, July-August 2008.

[98] Rebecca Sanders and Diane Kelly. The Challenge of Testing Scientific Software. In *Proceedings of the Conference for the Association for Software Testing*, pages 30–36, July 2008.

[99] J. Segal. Two Principles of End-User Software Engineering Research. In *1st Workshop on End User Software Engineering*, May 2005.

[100] J. Segal. Some Problems of Professional End User Developers. In *IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 111–118, 2007.

[101] J. Segal. Models of Scientific Software Development. In *First International Workshop on Software Engineering for Computational Science and Engineering (SECSE08)*, 2008.

[102] J. Segal. Scientists and Software Engineers: A Tale of Two Cultures. In *Proceedings of the Psychology of Programming Interest Group*, 2008.

[103] J. Segal. Software Development Cultures and Cooperation Problems: a Field Study of the Early Stages Of Development of Software for a Scientific Community. *Computer Supported Cooperative Work*, 18(5/6):581–606, 2009.

[104] J. Segal. Some Challenges Facing Software Engineers Developing Software for Scientists. In *Second International Workshop on Software Engineering for Computational Science and Engineering (SECSE09)*, pages 9–14, 2009.

[105] J. Segal and S. Clarke. Software Engineers Don't Know Everything About End-User Programming. *IEEE Software*, September-October 2009.

[106] J. Segal and C. Morris. Developing Scientific Software. *IEEE Software*, 25(4):18–20, 2008.

[107] J. Segal and C. Morris. Developing Software For A Scientific Community: Some Challenges And Solutions. In J. Leng and W. Sharrock, editors, *Handbook of Research on Computational Science and Engineering: Theory and Practice*, pages 177–196. IGI Global, 2011.

[108] J. Segal and C. Morris. Scientific End-User Developers and Barriers to Use/Customer Engagement. *Journal of Organizational and End User Computing*, 23(4):51–63, 2011.

[109] Judith Segal. When Software Engineers Met Research Scientists: A Case Study. *Empirical Software Engineering*, 10(4):517–536, 2005.

[110] Judith Segal, Diane Kelly, and Jeffrey Carver. Guest Editorial Preface: Special Issue on Scientific End User Computing. *Journal of Organizational and End User Computing*, 23(4), December 2011.

[111] David E. Skinner, Jon Stearley, John Hules, and Jon Bashor. Report of the 3rd DOE Workshop on HPC Best Practices: Software Lifecycles. Technical report, US Department of Energy, September 2009.

[112] M. T. Sletholt, J. Hannay, D. Pfahl, H. C. Benestad, and H. P. Langtangen. A literature review of agile practices and their effects in scientific software development. In *Fourth International Workshop on Software Engineering for Computational Science and Engineering (SECSE11)*, pages 1–9, 2011.

[113] M. T. Sletholt, J. E. Hannay, D. Pfahl, and H. P. Langtangen. What do we know about scientific software development's agile practices? *Computing in Science & Engineering*, 14(2):24–37, March-April 2012.

[114] W. Spencer Smith, Lei Lai, and Ridha Khedri. Requirements Analysis for Engineering Computation: A Systematic Approach for Improving Software Reliability. *Reliable Computing*, 13:83–107, 2007.

[115] Susan Squires, Michael L. Van De Vanter, and Lawrence G. Votta. Software Productivity Research in High Performance Computing. *CTWatch Quarterly*, 2006.

[116] Susan Squires, Michael L. Van De Vanter, and Lawrence G. Votta. Yes, there is an "expertise gap" in hpc applications development. In *Third Workshop on Productivity and Performance in High-End Computing (PPHEC06)*, 2006.

[117] T. Sterling. Productivity Metrics and Models for High Performance Computing. *International Journal of High Performance Computing Applications*, 18(4), 2004.

[118] Victoria Stodden, Peixuan Guo, and Zhaokun Ma. Toward Reproducible Computational Research: An Empirical Analysis of Data and Code Policy Adoption by Journals. *PLoS ONE*, 8(6):e67111, 06 2013.

[119] Jin Tang. Developing Scientific Computing Software: Current Processes and Future Directions. Master's thesis, McMaster University, 2008.

[120] Erik Trainer, Chalalai Chaihirunkarn, and James Herbsleb. The Big Effects of Short-term Efforts: A Catalyst for Community Engagement in Scientific Software. September 2013.

[121] Erik H Trainer, Chalalai Chaihirunkarn, and James D Herbsleb. The big effaects of short-term efforts: Mentorship and code integration in open source scientific software. *Journal of Open Research Software*, 2(1):e18, 2014.

[122] Erik H Trainer and James D Herbsleb. Beyond code: Prioritizing issues, sharing knowledge, and establishing identity at hackathons for science. In *CSCW Workshop on Sharing, Re-use, and Circulation of Resources in Scientific Cooperative Work*, 2014.

[123] Michele Vallisneri and Stanislav Babak. Python and XML for Agile Scientific Computing. *Computing in Science & Engineering*, 10(1):80–87, January 2008.

[124] Gregory R. Watson and Nathan A. DeBardeleben. Developing Scientific Applications Using Eclipse. *Computing in Science & Engineering*, 8(4):50–61, 2006.

[125] Gregory R. Watson and Craig E. Rasmussen. A Strategy for Addressing the Needs of Advanced Scientific Computing Using Eclipse as a Parallel Tools Platform. Technical report, Los Alamos National Laboratory, December 2005.

[126] James M. Willenbring, Michael A. Heroux, and Robert T. Heaphy. The Trilinos Software Lifecycle Model. In *Proceedings of the 3rd International Workshop on Software Engineering for High Performance Computing Applications*, 2007.

[127] Greg Wilson. Software Carpentry: Getting Scientists to Write Better Code by Making Them More Productive. *Computing in Science & Engineering*, November-December 2006.

[128] Greg Wilson. Where's the Real Bottleneck in Scientific Computing? *American Scientist*, January-February 2006.

[129] Greg Wilson. Those Who Will Not Learn From History... *Computing in Science & Engineering*, May-June 2008.

[130] Greg Wilson. How Do Scientists Really Use Computers? *American Scientist*, 97(5):8–10, September-October 2009.

[131] Greg Wilson. Software carpentry: Lessons learned. *F1000 Research*, 3(62), 2014.

[132] Greg Wilson, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H.D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, and Paul Wilson. Best practices for scientific computing. *PLoS Biology*, 12(1):e1001745, January 2014.

[133] Greg Wilson and Andrew Lumsdaine. Software Engineering and Computational Science. *Computing in Science & Engineering*, 11(6):12–13, 2009.

[134] Gregory V. Wilson. What Should Computer Scientists Teach to Physical Scientists and Engineers? *IEEE Computational Science & Engineering*, Summer-Fall 1996.

[135] Nicole Wolter, Michael O. McCracken, Allan Snavely, Lorin Hochstein, Taiga Nakamura, and Victor Basili. What's Working in HPC: Investigating HPC User Behavior and Productivity. *CTWatch Quarterly*, 2(4A):9–17, November 2006.

[136] William A. Wood and William L. Kleb. Exploring XP for Scientific Research. *IEEE Software*, 20(3):30–36, 2003.