benchmarks has led to principled evaluations and significant progress in associated fields, e.g., [3], [4], [5].

To that end, we developed two collections of defect scenarios—ManyBugs and IntroClass—consisting of 1,183 defects in 15 C programs. These benchmarks are diverse and allow for comparative evaluation of different types of automatic repair algorithms and experimental questions. Each program includes multiple defects, test suites, and human-written fixes. We illustrated the use of the benchmarks with three repair techniques, GenProg, TrpAutoRepair, and AE to provide baseline data that can be used in future studies. One of our goals in creating these benchmarks was to enable a broad set of use cases for automatic program repair researchers. We hope that the benchmarks will be useful, even for methods that don't require all of the included components. For example, a technique might not explicitly require a set of initially passing test cases, or an evaluation might not compare to the human-provided patches. We devoted a significant amount of effort to enhance usability and experimental reproducibility, with the goal of increasing the benchmarks' longterm contribution to the research community.

The scenarios, data (including baseline output and categorization information) and virtual machine images are available at http://repairbenchmarks.cs.umass.edu/. The web site provides detailed README files explaining the structure of the package, scenarios, results, and virtual machines information (including AMI numbers for the EC2 images). Errata and feedback on these resources should be sent to the authors.

## Acknowledgments

The authors gratefully acknowledge the partial support of AFOSR (FA9550-07-1-0532, FA9550-10-1-0277), US Defense Advanced Research Projects Agency (DARPA) (P-1070-113237), US Department of Energy (DOE) (DE-AC02-05CH11231), US National Science Foundation (NSF) (CCF-0729097, CCF-0905236, CCF-1446683, CNS-0905222), and the Santa Fe Institute. In addition, Martin Rinard provided insightful discussions regarding repair quality and identified and corrected several concerns in the defect scenarios.

## References

[1] "How science goes wrong," *Economist*, vol. 409, no. 8858, Oct. 2013, http://www.economist.com/news/leaders/21588069-scientific-research-has-changed-world-now-it-needs-change-itself-how-science-goes-wrong

[2] President's council of advisors on science and technology (PCAST) public meeting agenda [Online]. Available: http://www.whitehouse.gov/sites/default/files/microsites/ostp/PCAST/pcast_public_agenda_jan_2014.pdf, 2014.

[3] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *Proc. 17th Int. Conf. Parallel Archit. Compilation Techn.*, 2008, pp. 72–81.

[4] S. M. Blackburn, R. Garner, C. Hoffman, A. M. Khan, K. S. McKinley, R. Bentzur, A. Diwan, D. Feinberg, D. Frampton, S. Z. Guyer, M. Hirzel, A. Hosking, M. Jump, H. Lee, J. E. B. Moss, A. Phansalkar, D. Stefanović, T. VanDrunen, D. von Dincklage, and B. Wiedermann, "The DaCapo benchmarks: Java benchmarking development and analysis," in *Proc. 21st Annu. ACM SIGPLAN Conf. Object Oriented Programm. Syst. Languages Appl.*, 2006, pp. 169–190.

[5] "SPEC open systems group policies and procedures document," Aug. 2013.

[6] S. E. Sim, S. Easterbrook, and R. C. Holt, "Using benchmarking to advance research: A challenge to software engineering," in *Proc. 25th Int. Conf. Softw. Eng.*, 2003, pp. 74–83.

[7] T. Britton, L. Jeng, G. Carver, P. Cheak, and T. Katzenellenbogen, "Reversible debugging software," Judge Bus. School, Univ. Cambridge, Cambridge, U.K., Tech. Rep., 2013.

[8] A. Carzaniga, A. Gorla, A. Mattavelli, N. Perino, and M. Pezzè, "Automatic recovery from runtime failures," in *Proc. Int. Conf. Softw. Eng.*, 2013, pp. 782–791.

[9] Z. Coker and M. Hafiz, "Program transformations to fix C integers," in *Proc. Int. Conf. Softw. Eng.*, 2013, pp. 792–801.

[10] V. Debroy and W. E. Wong, "Using mutation to automatically suggest fixes for faulty programs," in *Proc. Int. Conf. Softw. Testing, Verification, Validation*, 2010, pp. 65–74.

[11] G. Jin, L. Song, W. Zhang, S. Lu, and B. Liblit, "Automated atomicity-violation fixing," in *Proc. 32nd ACM SIGPLAN Conf. Program. Language Des. Implementation*, 2011, pp. 389–400.

[12] D. Kim, J. Nam, J. Song, and S. Kim, "Automatic patch generation learned from human-written patches," in *Proc. Int. Conf. Softw. Eng.*, 2013, pp. 802–811.

[13] C. Le Goues, T. Nguyen, S. Forrest, and W. Weimer, "GenProg: A generic method for automated software repair," *IEEE Trans. Softw. Eng.*, vol. 38, no. 1, pp. 54–72, Jan./Feb. 2012.

[14] H. D. T. Nguyen, D. Qi, A. Roychoudhury, and S. Chandra, "SemFix: Program repair via semantic analysis," in *Proc. Int. Conf. Softw. Eng.*, 2013, pp. 772–781.

[15] J. H. Perkins, S. Kim, S. Larsen, S. Amarasinghe, J. Bachrach, M. Carbin, C. Pacheco, F. Sherwood, S. Sidiroglou, G. Sullivan, W.-F. Wong, Y. Zibin, M. D. Ernst, and M. Rinard, "Automatically patching errors in deployed software," in *Proc. ACM Symp. Operating Syst. Principles*, Big Sky, MT, USA, Oct. 12–14, 2009, pp. 87–102.

[16] Y. Wei, Y. Pei, C. A. Furia, L. S. Silva, S. Buchholz, B. Meyer, and A. Zeller, "Automated fixing of programs with contracts," in *Proc. Int. Symp. Softw. Testing Anal.*, 2010, pp. 61–72.

[17] W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest, "Automatically finding patches using genetic programming," in *Proc. Int. Conf. Softw. Eng.*, 2009, pp. 364–367.

[18] M. Järvisalo and A. V. Gelder, Eds., *Proc. 16th Int. Conf. Theory Appl. Satisfiability Testing*, Jul. 2013.

[19] U. Egly and C. Sinz, Eds., *Proc. 17th Int. Conf. Theory Appl. Satisfiability Testing*, Jul. 2014.

[20] C. Le Goues, S. Forrest, and W. Weimer, "Current challenges in automatic software repair," *Softw. Quality J.*, vol. 21, no. 3, pp. 421–443, 2013.

[21] W. Weimer, Z. P. Fry, and S. Forrest, "Leveraging program equivalence for adaptive program repair: Models and first results," in *Proc. IEEE/ACM 28th Int. Conf. Automated Softw. Eng.*, Nov. 2013, pp. 356–366.

[22] Y. Qi, X. Mao, and Y. Lei, "Efficient automated program repair through fault-recorded testing prioritization," in *Proc. Int. Conf. Softw. Maintenance*, Eindhoven, The Netherlands, Sep. 2013, pp. 180–189.

[23] C. Le Goues, M. Dewey-Vogt, S. Forrest, and W. Weimer, "A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8 each," in *Proc. Int. Conf. Softw. Eng.*, 2012, pp. 3–13.

[24] Y. Brun, E. Barr, M. Xiao, C. Le Goues, and P. Devanbu. (2013). Evolution vs. intelligent design in program patching. UC Davis: College Eng., Tech. Rep. [Online]. Available: https://escholarship.org/uc/item/3z8926ks

[25] E. K. Smith, E. Barr, C. Le Goues, and Y. Brun, "Is the cure worse than the disease? overfitting in automated program repair," in *Proc. Joint Meeting Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng.*, Bergamo, Italy, Sept. 2015, pp. 2–4.

[26] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in *Proc. Int. Conf. Softw. Eng.*, 2006, pp. 361–370.

[27] B. Liblit, M. Naik, A. X. Zheng, A. Aiken, and M. I. Jordan, "Scalable statistical bug isolation," in *Proc. ACM SIGPLAN Conf. Program. Language Des. Implementation*, 2005, pp. 15–26.

[28] (2014, Feb.) [Online]. Available: http://www.mozilla.org/security/bug-bounty.html

[29] (2014, Feb.) [Online]. Available: http://blog.chromium.org/2010/01/encouraging-more-chromium-security.html

[30] (2014, Feb.) [Online]. Available: http://msdn.microsoft.com/en-us/library/dn425036.aspx

[31] C. World. (2014, Feb.) [Online]. Available: http://www.computerworld.com/s/article/9179538/Google_calls_raises_Mozilla_s_bug _bounty_for_Chrome_flaws

[32] (2014, Feb.) [Online]. Available: http://msdn.microsoft.com/en-us/library/dn425036.aspx

[33] (2014, Feb.) [Online]. Available: http://www.daemonology.net/blog/2011-08-26-1265-dollars-of-tarsnap-bugs.html

[34] V. Dallmeier, A. Zeller, and B. Meyer, "Generating fixes from object behavior anomalies," in *Proc. IEEE/ACM Int. Conf. Automated Softw. Eng.*, 2009, pp. 550–554.

[35] A. Carzaniga, A. Gorla, N. Perino, and M. Pezzè, "Automatic workarounds for web applications," in *Proc. Int. Symp. Found. Softw. Eng.*, 2010, pp. 237–246.

[36] M. Monperrus, "A critical review of 'Automatic patch generation learned from human-written patches': Essay on the problem statement and the evaluation of automatic software repair," in *Proc. Int. Conf. Softw. Eng.*, 2014, pp. 234–242.

[37] D. Gopinath, S. Khurshid, D. Saha, and S. Chandra, "Data-guided repair of selection statements," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 243–253.

[38] S. Kaleeswaran, V. Tulsian, A. Kanade, and A. Orso, "Minthint: Automated synthesis of repair hints," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 266–276.

[39] M. Harman, "The current state and future of search based software engineering," in *Proc. Int. Conf. Softw. Eng.*, 2007, pp. 342–357.

[40] Z. P. Fry, B. Landau, and W. Weimer, "A human study of patch maintainability," in *Proc. Int. Symp. Softw. Testing. Anal.*, 2012, pp. 177–187.

[41] W. Weimer, "Advances in automated program repair and a call to arms," in *Proc. 5th Int. Symp. Search Based Softw. Eng.*, St. Petersburg, Russia, Aug. 2013, pp. 1–3.

[42] V. Dallmeier and T. Zimmermann, "Extraction of bug localization benchmarks from history," in *Proc. IEEE/ACM 22nd Int. Conf. Automated Softw. Eng.*, 2007, pp. 433–436.

[43] R. Just, D. Jalali, L. Inozemtseva, M. D. Ernst, R. Holmes, and G. Fraser, "Are mutants a valid substitute for real faults in software testing?" in *Proc. 22nd ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2014, pp. 654–665.

[44] H. Do, S. Elbaum, and G. Rothermel, "Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact," *Empirical Softw. Eng.*, vol. 10, no. 4, pp. 405–435, Oct. 2005.

[45] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand, "Experiments of the effectiveness of dataflow-and control flow-based test adequacy criteria," in *Proc. Int. Conf. Softw. Eng.*, 1994, pp. 191–200.

[46] National Institute of Standards and Technology (NIST), "Samate—Software software assurance metrics and tool evaluation," 2012.

[47] T. Durieux, M. Martinez, M. Monperrus, R. Sommerard, and J. Xuan. (2015). Automatic repair of real bugs: An experience report on the Defects4J dataset. *CoRR, abs/1505.07002* [Online]. Available: http://arxiv.org/abs/1505.07002

[48] J. S. Bradbury, I. Segall, E. Farchi, K. Jalbert, and D. Kelk, "Using combinatorial benchmark construction to improve the assessment of concurrency bug detection tools," in *Proc. Workshop Parallel Distrib. Syst.: Testing, Anal. Debugging*, 2012, pp. 25–35.

[49] S. Zhang, D. Jalalinasab, J. Wuttke, K. Muşlu, W. Lam, M. D. Ernst, and D. Notkin, "Empirically revisiting the test independence assumption," in *Proc. Int. Symp. Softw. Testing. Anal.*, San Jose, CA, USA, Jul. 2014, pp. 385–396.

[50] C. Le Goues, S. Forrest, and W. Weimer, "Representations and operators for improving evolutionary software repair," in *Proc. Genetic Evol. Comput. Conf.*, 2012, pp. 959–966.

[51] A. Arcuri and G. Fraser, "On parameter tuning in search based software engineering," in *Proc. Int. Symp. Search Based Softw. Eng.*, 2011, pp. 33–47.

[52] Y. Qi, X. Mao, Y. Lei, and C. Wang, "Using automated program repair for evaluating the effectiveness of fault localization techniques," in *Proc. Int. Symp. Softw. Testing. Anal.*, 2013, pp. 191–201.

[53] Programming language popularity [Online]. Available: http://langpop.com, 2014.

[54] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. T. Devanbu, "Fair and balanced? Bias in bug-fix datasets," in *Proc. ACM SIGSOFT Symp. Found. Softw. Eng.*, 2009, pp. 121–130.

[55] E. Fast, C. Le Goues, S. Forrest, and W. Weimer, "Designing better fitness functions for automated program repair," in *Proc. Genetic Evol. Comput. Conf.*, 2010, pp. 965–972.

[56] Black Duck Software. ( Inc., http://www.ohloh.net/, Feb. 2014.

[57] Wikipedia, "Python (programming language)," http://en.wikipedia.org/wiki/Python_(programming_language), Feb. 2014.

[58] C. Cadar, D. Dunbar, and D. Engler, "KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs," in *Proc. 8th USENIX Conf. Operating Syst. Des. Implementation*, San Diego, CA, USA, 2008, pp. 209–224.

[59] J. S. Bradbury and K. Jalbert, "Automatic repair of concurrency bugs," in *Proc. Int. Symp. Search Based Softw. Eng.—Fast Abstracts*, Sep. 2010, pp. 1–2.

[60] A. Arcuri and X. Yao, "A novel co-evolutionary approach to automatic software bug fixing," in *Proc. Congr. Evol. Comput.*, 2008, pp. 162–168.

[61] Y. Pei, C. A. Furia, M. Nordio, Y. Wei, B. Meyer, and A. Zeller, "Automated fixing of programs with contracts," *IEEE Trans. Softw. Eng.*, vol. 40, no. 5, pp. 427–449, May 2014.

[62] P. Liu and C. Zhang, "Axis: Automatically fixing atomicity violations through solving control constraints," in *Proc. Int. Conf. Softw. Eng.*, 2012, pp. 299–309.

[63] D. Jeffrey, M. Feng, N. Gupta, and R. Gupta, "BugFix: A learning-based tool to assist developers in fixing bugs," in *Proc. Int. Conf. Program Comprehension*, 2009, pp. 70–79.

[64] J. L. Wilkerson, D. R. Tauritz, and J. M. Bridges, "Multi-objective coevolutionary automated software correction," in *Proc. Genetic Evol. Comput. Conf.*, 2012, pp. 1229–1236.

[65] B. Demsky, M. D. Ernst, P. J. Guo, S. McCamant, J. H. Perkins, and M. C. Rinard, "Inference and enforcement of data structure consistency specifications," in *Proc. Int. Symp. Softw. Testing. Anal.*, 2006, pp. 233–244.

[66] M. Orlov and M. Sipper, "Flight of the FINCH through the Java wilderness," *Trans. Evol. Comput.*, vol. 15, no. 2, pp. 166–192, 2011.

[67] D. Gopinath, M. Z. Malik, and S. Khurshid, "Specification-based program repair using SAT," in *Proc. 17th Int. Conf. Tools Algorithms Construction Anal. Syst.*, 2011, pp. 173–188.

[68] M. Carbin, S. Misailovic, M. Kling, and M. C. Rinard, "Detecting and escaping infinite loops with Jolt," in *Proc. Eur. Conf. Object Oriented Program.*, 2011, pp. 609–633.

[69] B. Elkarablieh and S. Khurshid, "Juzi: A tool for repairing complex data structures," in *Proc. Int. Conf. Softw. Eng.*, 2008, pp. 855–858.

[70] S. Sidiroglou and A. D. Keromytis, "Countering network worms through automatic patch generation," *IEEE Security Privacy*, vol. 3, no. 6, pp. 41–49, Nov. 2005.

[71] S. Jha, S. Gulwani, S. A. Seshia, and A. Tiwari, "Oracle-guided component-based program synthesis," in *Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng.*, 2010, vol. 1, pp. 215–224.

[72] D. Perelman, S. Gulwani, D. Grossman, and P. Provost, "Test-driven synthesis," *SIGPLAN Not.*, vol. 49, no. 6, pp. 408–418, Jun. 2014.

[73] S. Lu, Z. Li, F. Qin, L. Tan, P. Zhou, and Y. Zhou, "Bugbench: Benchmarks for evaluating bug detection tools," in *Proc. Workshop Eval. Softw. Defect Detection Tools*, 2005, pp. 16–20.

[74] E. Schulte, Z. Fry, E. Fast, W. Weimer, and S. Forrest, "Software mutational robustness," *Genetic Program. Evolvable Mach.*, vol. 15, no. 3, pp. 281–312, 2014.

[75] Z. P. Fry and W. Weimer, "A human study of fault localization accuracy," in *Proc. Int. Conf. Softw. Maintenance*, 2010, pp. 1–10.

[76] J. C. Knight and P. Ammann, "An experimental evaluation of simple methods for seeding program errors," in *Proc. 8th Int. Conf. Softw. Eng.*, 1985, pp. 337–342.

[77] Y. Jia and M. Harman, "An analysis and survey of the development of mutation testing," *IEEE Trans. Softw. Eng.*, vol. 37, no. 5, pp. 649–678, Sep./Oct. 2011.

[78] C. Parnin and A. Orso, "Are automated debugging techniques actually helping programmers?" in *Proc. Int. Symp. Softw. Testing. Anal.*, Toronto, ON, Canada, 2011, pp. 199–209.

[79] H. He and N. Gupta, "Automated debugging using path-based weakest preconditions," in *Proc. 7th Int. Conf. Fundam. Approaches Softw. Eng.*, 2004, pp. 267–280.

[80] A. Arcuri, "On the automation of fixing software bugs," in *Proc. Doctoral Symp.—30th Int. Conf. Softw. Eng.*, 2008, pp. 1003–1006.

**Claire Le Goues** received the BA degree in computer science from Harvard University and the MS and PhD degrees from the University of Virginia. She is an assistant professor in the School of Computer Science, Carnegie Mellon University, where she is primarily affiliated with the Institute for Software Research. She is interested in how to construct high-quality systems in the face of continuous software evolution, with a particular interest in automatic error repair. More information is available at: http://www.cs.cmu.edu/ clegoues.

**Neal Holtschulte** received the BA degree in mathematics from Williams College. He is currently working toward the PhD degree at the University of New Mexico, where he studies automated program repair and biologically inspired computation with Professor Melanie Moses. He is also interested in genetic programming, program structure, and computer science education.

**Edward K. Smith** received the BS degree from the University of Maryland in 2013. He is currently working toward the PhD degree in the College of Information and Computer Science, University of Massachusetts, Amherst. His research interests include human factors, programming languages, and software engineering. More information is available on his homepage: https://people.cs.umass.edu/tedks/.

**Yuriy Brun** received the the MEng degree from the Massachusetts Institute of Technology in 2003, and the PhD degree from the University of Southern California in 2008. He is an assistant professor in the College of Information and Computer Science, University of Massachusetts, Amherst. He completed his postdoctoral work in 2012 at the University of Washington as a CI fellow. His research focuses on software engineering, distributed systems, and self-adaptation. He received the US National Science Foundation (NSF) Career Award in 2015, a Microsoft Research Software Engineering Innovation Foundation Award in 2014, and an IEEE TCSC Young Achiever in Scalable Computing Award in 2013. He is a member of the IEEE, the ACM, and the ACM SIGSOFT. More information is available on his homepage: http://people.cs.umass.edu/brun/

**Premkumar Devanbu** received the BTech degree from IIT Madras, in Chennai, India, and the PhD degreee from Rutgers University. After spending nearly 20 years as both a developer and a researcher at Bell Labs and its various offshoots, he left Industry to join the CS faculty at UC Davis in late 1997, where he is currently a professor of computer science.

**Stephanie Forrest** received the BA degree from St. John's College and the MS and PhD degrees from the University of Michigan. She is currently Regents distinguished professor of computer science at the University of New Mexico and a member of the External Faculty of the Santa Fe Institute. Her research studies complex adaptive systems, including immunology, evolutionary computation, biological modeling, and computer security. She is a fellow of the IEEE.

**Westley Weimer** received the BA degree in computer science and mathematics from Cornell University and the MS and PhD degrees from the University of California, Berkeley. He is currently an associate professor at the University of Virginia. His main research interests include static and dynamic analyses to improve software quality and fix defects.