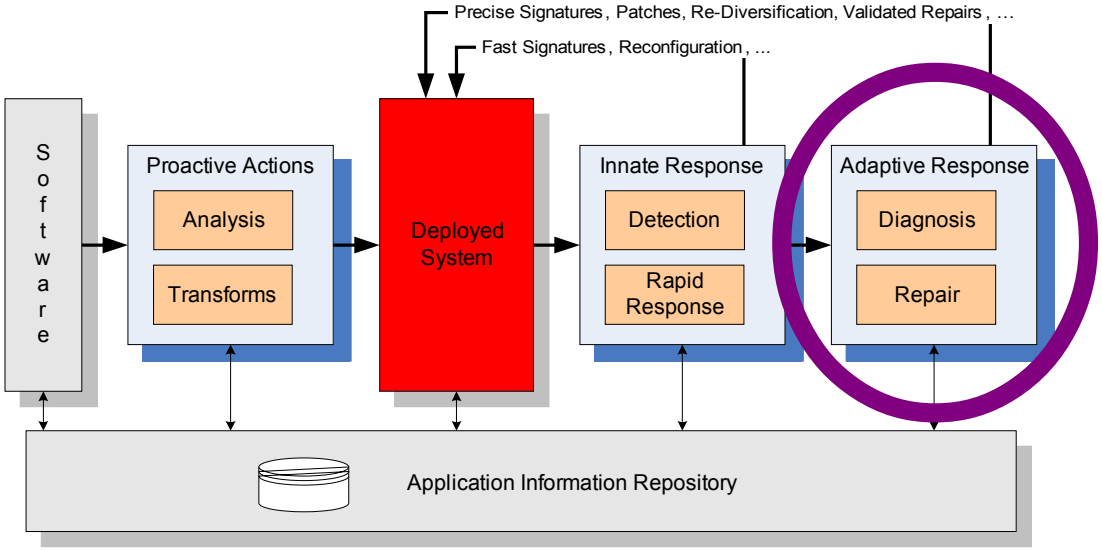
Automated Program Repair

Westley Weimer (UVA), Stephanie Forrest (UNM)

MURI Final Review Briefing
June 13, 2012

Helix Context

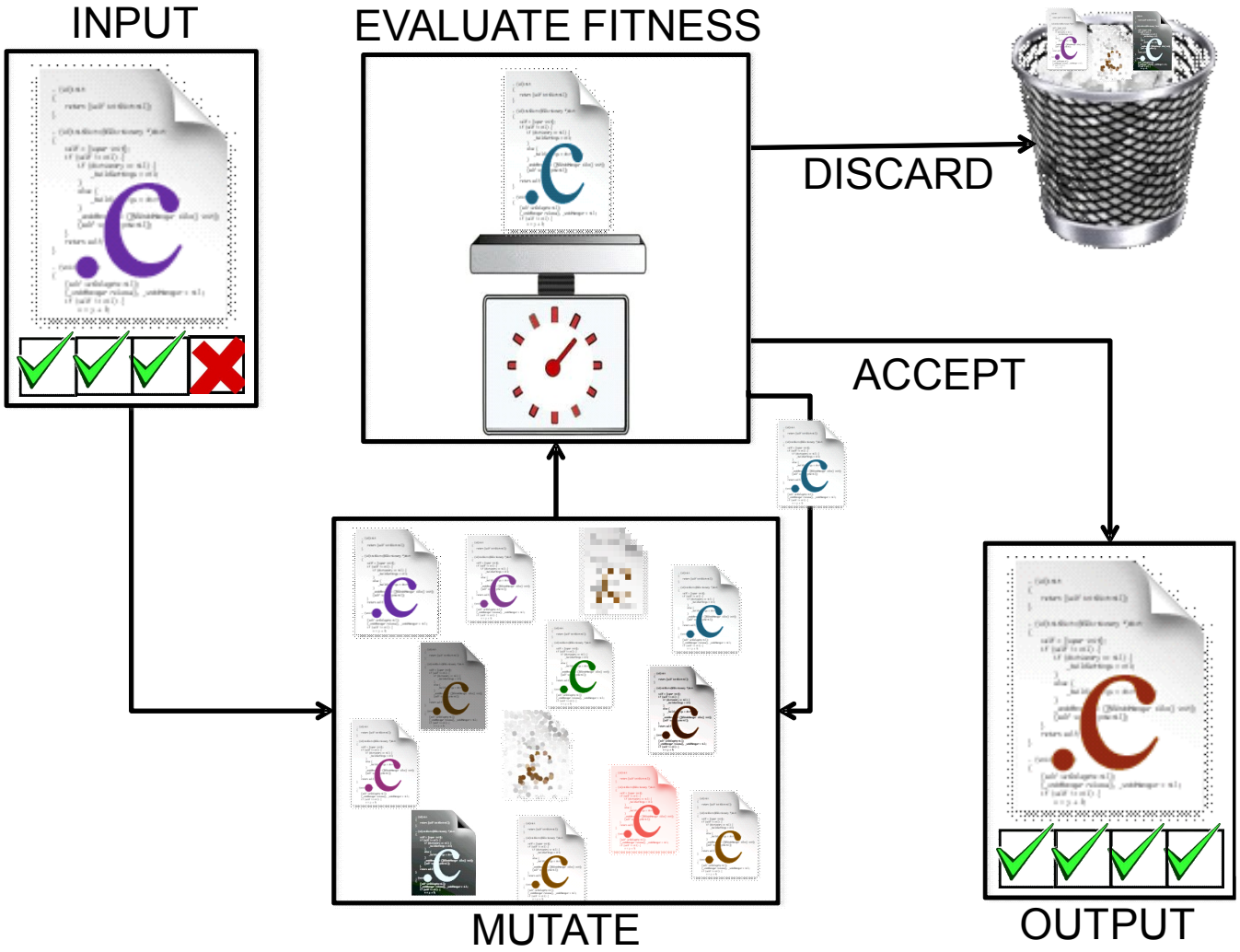
- Given an unannotated program, its test suite, and a bug …
- Produce a patch repairing that bug while retaining other functionality!



Idea Genesis

- Brainstormed idea at first MURI meeting
  - Forrest (UNM) on GA, Weimer (UVA) on PL
  - "MU" in MURI was absolutely necessary
- First results (gcd & nullhttpd) at second
- Also supported by: NSF, AFOSR, DARPA
  - "Since you and I would never have conceived this project without the MURI collaboration, I think they deserve credit for almost everything." – SF

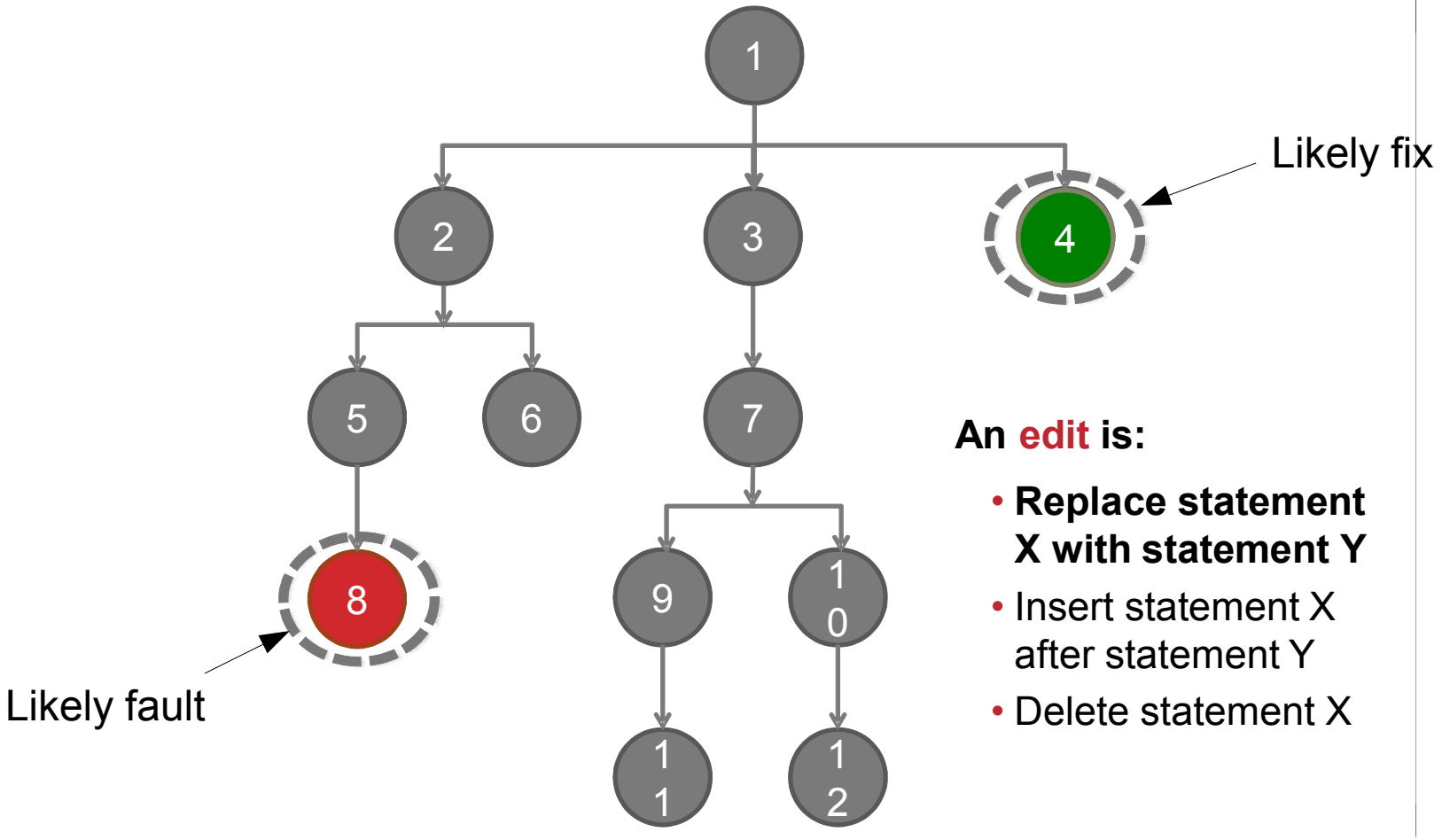
GenProg Architecture



Evolving A Candidate Repair

- Mimic humans: find a line and change it
- Operate on abstract syntax trees
- Find nodes implicated by fault localization
  - Likely to be the source of the bug
- Program contains seeds of its own repair
- Find nodes implicated by fix localization
  - Replace or augment bad code with good code

Editing A Program



An edit is:

- **Replace statement X with statement Y**
- Insert statement X after statement Y
- Delete statement X

Fitness and Test Cases

- We use standard test cases to determine

  (1) If a candidate patch is an acceptable final repair

  (2) Otherwise, whether to retain it into the next generation of the genetic algorithm (fitness)

- Only (1) needs the full test suite
- Internal calculations (2) can subsample!
- Test cases, candidate repairs, and genetic algorithm runs can be done independently
  - Orders of magnitude beyond our initial results ...

Confidence

- We want confidence that our technique generalizes to real-world bugs
- Over time, the research question shifts:
  - Are there bugs of many types that we can repair?
  - *now*: What fraction of important bugs can we repair, and at what cost and quality?
- Intuition:
  - Suppose we worked for a company and used our technique on the next 100 bugs reported

Evaluation Plan

- Decide on and finalize our algorithm
- Use version control histories
  - Look for bugs where the humans checked in fixes, checked in test cases, and rated at least 3/5 severity
  - Check all pairs of viable versions to find such bugs
- Use public cloud-computing resources
  - Place commodity prices on our technique
- Thirteen hours each to repair all bugs ...

Benchmarks

Program	LOC	Tests	Defects	Description
fbc	97,000	773	3	Language (legacy)
gmp	145,000	145	2	Precision math
gzip	491,000	12	5	Data compression
libtiff	77,000	78	24	Image processing
lighttpd	62,000	295	9	Web server
php	1,046,000	8,471	44	Language (web)
python	407,000	355	11	Language (general)
wireshark	2,814,000	63	7	Packet analyzer
total	**5,139,000**	**10,193**	**105**	

Orders of magnitude larger than previous/competing work.

Repaired 55/105 For $8 Each

Program	Defects	Cost per Non-Repair		Cost per Repair	
		Hours	US$	Hours	US$
fbc	1 / 3	8.52	5.56	6.52	4.08
gmp	1 / 2	9.93	6.61	1.60	0.44
gzip	1 / 5	5.11	3.04	1.41	0.30
libtiff	17 / 24	7.81	5.04	1.05	0.04
lighttpd	5 / 9	10.79	7.25	1.34	0.25
php	24 / 44	13.00	8.80	1.84	0.62
python	1 / 11	13.00	8.80	1.22	0.16
wireshark	1 / 7	13.00	8.80	1.23	0.17
total	55 / 105	11.22		1.60	

Total cost: $403 - or $7.32 per repair (incl. cost of failures)

Cost Comparisons

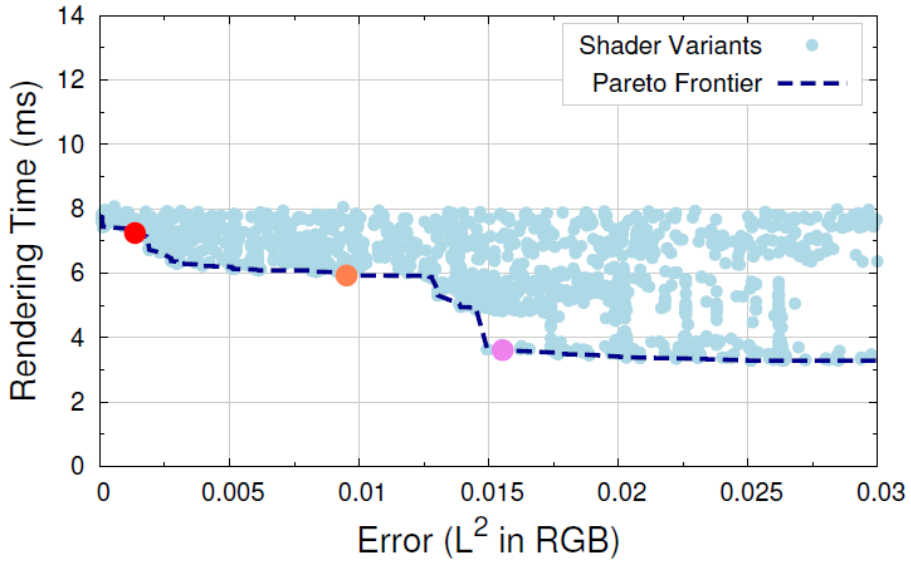
- JBoss issue tracking: median 5.0, mean 15.3 hours
- IBM: $25 per defect during coding, rising at build, Q&A, post-release, etc.
- Tarsnap.com: $17, 40 hours per non-trivial repair.
- Bug bounty programs in general:
  - At least $500 per security-critical bug.
  - One of our php bugs has an associated security CVE.

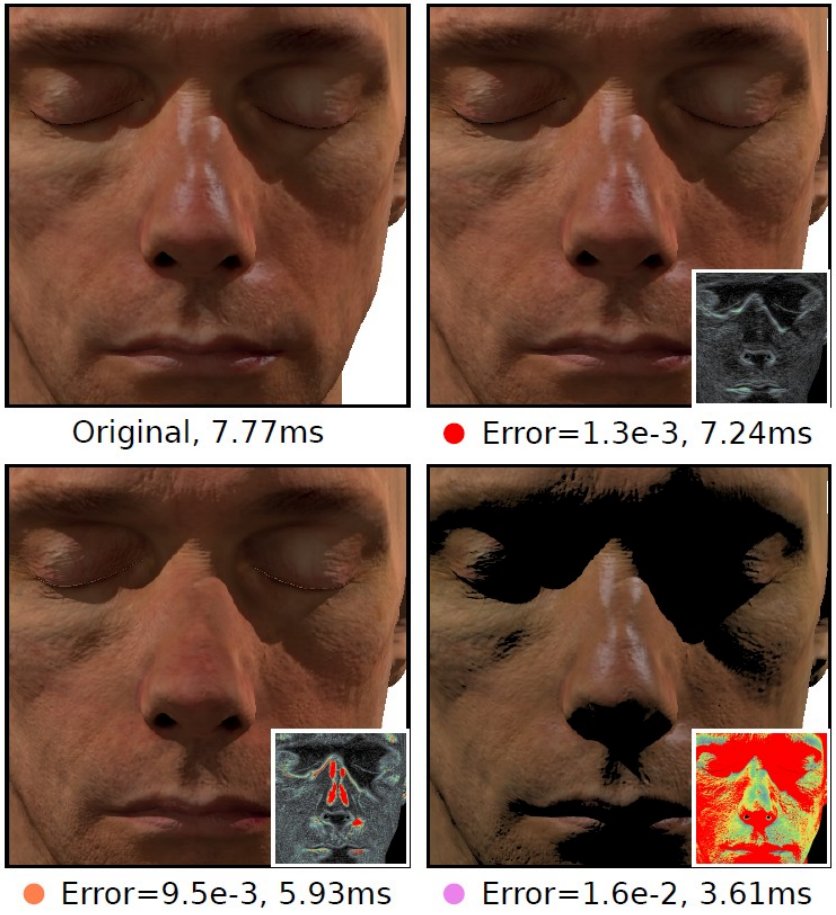
Repair Quality (1)

- Functional correctness
- Repairs always pass all test cases
- Repairs succeed on 10,000+ held-out tests
  - Experiments with multiple webservers
  - Yield same answer (bit-per-bit) in same time or less
- For security bugs, exploit fuzzing shows:
  - repairs defeat variant attacks
  - repairs not found to introduce new vulnerabilities

Extension: Graphics

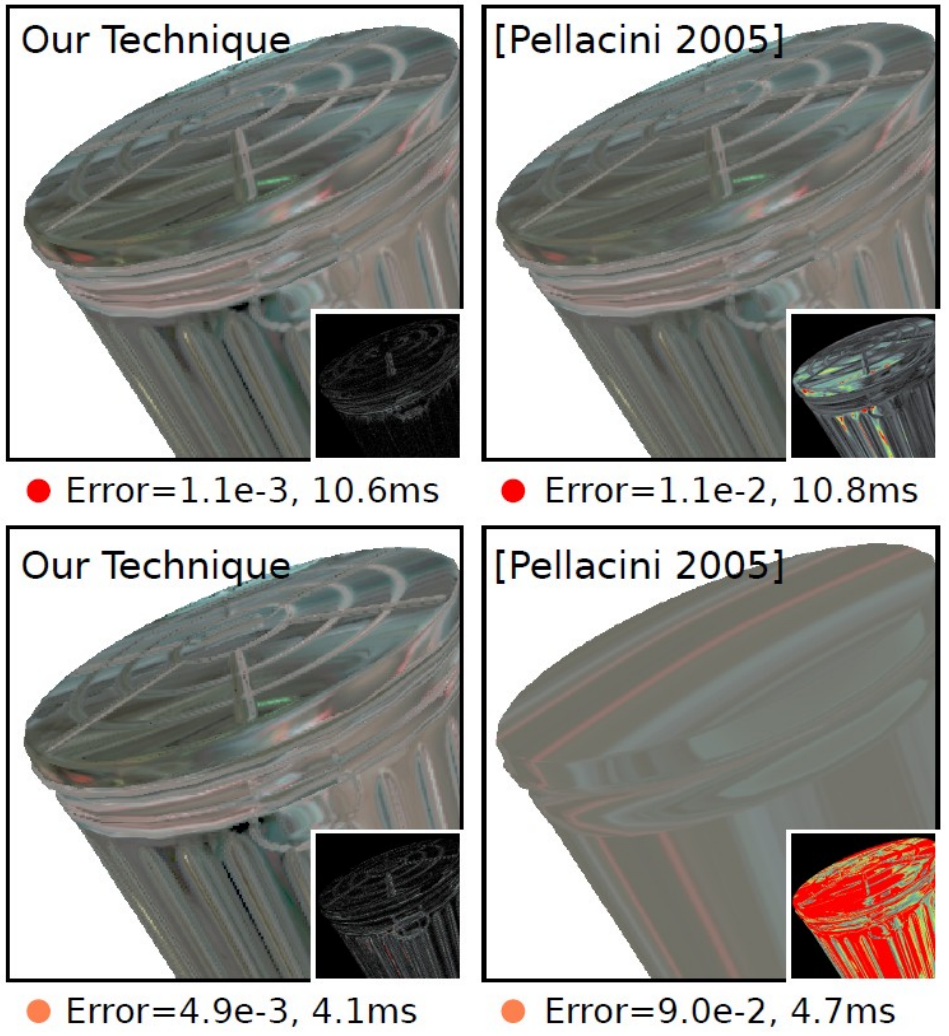
- Automatically simplify multipass shaders, retaining perceptual fidelity.





Extension: Graphics (2)

- Our multi-objective extension exceeds graphics-specific previous work (less error given constant time budget).
- Suggests “low power audio player”, “low heat video player”, etc.



Extension: Embedded Systems

- Challenge: reduce disk and memory requirements by an order of magnitude
- Representation: sequence of edits
- Fault localization: stochastic sampling
- Can repair assembly files and ELF binary executables (x86 and ARM)
  - Concrete experiments on Nokia N900 smartphones
  - Similar repair success rates as source-level (+- 10%)

Limitations

- Requires test suite and failing test
  - Future work on test suite generation and test oracle generation
- Requires fault localization
  - Future work on using non-naive fault localizer
- Requires deterministic defects
  - Others working on race conditions, etc.
- “Contains the seeds of its own repair”

Looking *Forward*

- Automatic repairs to half of high-severity defects would "take the heat off" maintenance software engineers, enabling a focus elsewhere (or on other bugs)
- High-quality, human-competitive repairs for large programs in 0.6 hours give packet filtering / read-only mode solutions something to wait for
- Apply GP transformations to create a diversity of implementations, trading off accuracy for speed / power / heat / etc.
- Pro-actively produce diverse variants, bolster immunity

Conclusion

- Repaired 55/105 bugs in over 5 MLOC with over 10,000 tests for $8 each
  - Patch quality and cost are human-competitive
- Technique works in other domains
  - Graphics, Embedded Systems
- Automated Program Repair using Evolutionary Computation
- http://genprog.cs.virginia.edu

Papers and Awards

- 7 conference papers, 3 journals, 2 workshops, 3 best paper awards, 2 other awards (since this MURI started).

Claire Le Goues, Michael Dewey-Vogt, Stephanie Forrest, Westley Weimer: A Systematic Study of Automated Program Repair: Fixing 55 out of 105 bugs for $8 Each. International Conference on Software Engineering (ICSE) 2012: to appear

Claire Le Goues, Stephanie Forrest, Westley Weimer: The Case for Software Evolution. Foundations of Software Engineering Working Conference on the Future of Software Engineering (FoSER) 2010: 205-209

Westley Weimer, Stephanie Forrest, Claire Le Goues, ThanhVu Nguyen: Automatic Program Repair With Evolutionary Computation. Communications of the ACM Vol. 53 No. 5, May 2010, Pages 109-116.

Zachary P. Fry, Bryan Landau, Westley Weimer: A Human Study of Patch Maintainability. International Symposium on Software Testing and Analysis (ISSTA) 2012: to appear

Pitchaya Sitthi-amorn, Nicholas Modly, Westley Weimer, Jason Lawrence: Genetic Programming for Shader Simplification. ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 30(6): 152 (2011)

Eric Schulte, Stephanie Forrest, Westley Weimer: Automated Program Repair through the Evolution of Assembly Code. Automated Software Engineering (ASE) Short Paper 2010: 313-316

Claire Le Goues, Westley Weimer, Stephanie Forrest: Representations and Operators for Improving Evolutionary Software Repair. Genetic and Evolutionary Computing Conference (GECCO) 2012: to appear

Ethan Fast, Claire Le Goues, Stephanie Forrest, Westley Weimer: Designing better fitness functions for automated program repair. Genetic and Evolutionary Computing Conference (GECCO) 2010: 965-972

Stephanie Forrest, Westley Weimer, ThanhVu Nguyen, Claire Le Goues. A Genetic Programming Approach to Automated Software Repair. Genetic and Evolutionary Computing Conference (GECCO) 2009: 947-954 (best paper award) (gold human-competitive award)

Claire Le Goues, ThanhVu Nguyen, Stephanie Forrest, Westley Weimer: GenProg: A Generic Method for Automated Software Repair. IEEE Trans. Software Engineering 38(1): 54-72 (January/February 2012) (featured paper award)

ThanhVu Nguyen, Westley Weimer, Claire Le Goues, Stephanie Forrest. Using Execution Paths to Evolve Software Patches. Workshop on Search-Based Software Testing (SBST) 2009 (best short paper award) (best presentation award)

Westley Weimer, ThanVu Nguyen, Claire Le Goues, Stephanie Forrest: Automatically Finding Patches Using Genetic Programming. International Conference on Software Engineering (ICSE) 2009: 364-374 (distinguished paper award) (IFIP TC2 Manfred Paul award)

Repair Quality (2)

(This IRB-approved human study was NSF-funded.)

- Are GenProg patches as easy to reason about and maintain as human patches?
- Grounded human study, 157 participants:
  - GenProg patches, human-accepted patches, human-reverted patches, GenProg+Doc patches … all for the same "$8" bugs!
- GenProg patches augmented with automatic documentation **are as maintainable** (same accuracy, 30% less time) as human-accepted patches.