Fixing software bugs in 10 minutes or less using evolutionary computation

University of New Mexico
Stephanie Forrest
ThanhVu Nguyen
University of Virginia
Claire Le Goues
Westley Weimer

Summary of method

- Assume:
  - Access to C source code
  - Negative test case that executes the buggy code
  - Positive test cases to encode required program functionality
- Construct Abstract Syntax Tree (AST)
- Evolve repair that avoids negative test case and passes positive test case
- Minimize repair using program analysis methods

THE UNIVERSITY of NEW MEXICO

Repairing the Zune bug using GP

- Infinite loop when input is last day of a leap year.
- Microsoft sold about 1.2 million units of Zune 30, generating thousands of complaints.
- Repair is not trivial. Microsoft's recommendation was to let Zune drain its battery and then reset.
- GP discovered the repair in 42 seconds.

```
1   void zunebug_repair(int days) {
2     int year = 1980;
3     while (days > 365) {
4       if (isLeapYear(year)){
5         if (days > 366) {
6           // days -= 366; // repair deletes
7           year += 1;
8         }
9         else {
10        }
11        days -= 366;       // repair inserts
12      } else {
13        days -= 365;
14        year += 1;
15      }
16    }
17    printf("current year is %d\n", year);
18  }
```

Downloaded from http://pastie.org/349916 (Jan. 2009).

THE UNIVERSITY of NEW MEXICO

Example repairs

Program	Version	LOC	Time to Repair	Program Description	Fault
gcd	example	22	153s	handcrafted example	infinite loop
zune		28	42s	media player	infinite loop
uniq	ultrix 4.3	1146	34s	duplicate text processing	segfault
look	ultrix 4.3	1169	45s	dictionary lookup	segfault
look	svr4.0 1.1	1363	55s	dictionary lookup	infinite loop
units	svr4.0 1.1	1504	109s	metric conversion	segfault
deroff	ultrix 4.3	2236	131s	document processing	segfault
indent	1.9.1	9906	546s	source code processing	infinite loop
flex	2.5.4a	18775	230s	lexical analyzer generator	segfault
atris	1.0.6	21553	80s	graphical tetris game	loc. stack buffer exploit
nullhttpd	0.5.0	5575	578s	webserver	rem. heap buffer exploit
openldap io.c	2.3.41	6519	665s	directory protocol	non-overflow DOS
lighthttpd fastcgi.c	1.4.17	13984	49s	webserver	rem. heap buf overflow
php string.c	5.2.1	26044	6s	scripting language	int overflow
wu-ftp	2.6.0	35109	2256s	FTP server	format string
total		144933			

THE UNIVERSITY of NEW MEXICO

Why is this human competitive?

- Software is THE (indisputably difficult) problem
- Time to discover a repair
- Quality of repair

THE UNIVERSITY of NEW MEXICO

Software is THE problem

- Software faults and debugging are expensive:
  - US corporate development organizations spend $5.2 - $22 million annually fixing software defects (IDC Software Quality Survey, 2008)
  - Cost of repairing bugs increases throughout the development process. A $25 fix while the program is under development increases to $16,000 after the software has gone live (IBM Rational group, 2008)
- Security violations are expensive:
  - Average total per-incident costs in 2008 were $6.65 million, compared to an average per-incident cost of $6.3 million in 2007.
  - Monetary loss by 639 companies in 2005 totaled $130 million (FBI 2005)

THE UNIVERSITY of NEW MEXICO

Software is THE problem

- Bugs are plentiful:
  - Mozilla project received 51,154 bug reports in 2002-2006.
  - In 2005, a Mozilla developer reported that “almost 300 bugs appear every day that need triaging.”
- Fixing bugs is time-consuming:
  - Industrial software repair: 1/2 of all fixed bugs in Mozilla from 2002-2006 took more than 29 days for developers to fix; Median repair time for ArgoUML project in 2002-2003 was 190 days; Median repair time per bug for PostgreSQL was 200 days.

Time to discover repair

- To date, we have repaired 15 programs totaling nearly 150,000 lines of code
- Average time to repair: 3 minutes (for first 11 programs shown)
- Time includes:
  - GP algorithm (selection, mutation, calculating fitness, etc.)
  - Running test cases
  - Pretty printing and memoizing ASTs
  - gcc (compiling ASTs into executable code)

THE UNIVERSITY of NEW MEXICO

Quality of repair

- Manual checks for repair correctness.
- Microsoft requires that security-critical changes be subjected to 100,000 fuzz inputs (randomly generated structured input strings).
- Used SPIKE black-box fuzzer (immunitysec.com) to generate 100,000 held-out fuzz requests for web server examples.
  - In no case did GP repairs introduce errors that were detected by the fuzz tests, and in every case the GP repairs defeated variant attacks based on the same exploit.
  - Thus, the GP repairs are not fragile memorizations of the input.
- GP repairs also correctly handled all subsequent requests from indicative workload.

THE UNIVERSITY of NEW MEXICO

Why should we win the prize?

- The idea of computational evolution (genetic algorithms) was introduced nearly 50 years ago (by JHH).
  - Led to many successes in engineering and science.
- Yet, the dream of “automatic programming” is still largely unfilled.
  - Why does the evolutionary approach to design work throughout nature and engineering, but not in software?
- A gap in the evolutionary record that needs to be filled.

THE UNIVERSITY of NEW MEXICO