REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
10/13/2011	Final	06/15/2010-06/14/2011

4. TITLE AND SUBTITLE
Helix Project Test Bed: Towards the Self-Regenerative Incorruptible Enterprise

5a. CONTRACT NUMBER
N/A

5b. GRANT NUMBER
FA9550-10-1-0277

5c. PROGRAM ELEMENT NUMBER
N/A

6. AUTHOR(S)
Forrest, Stephanie, Ph.D.

5d. PROJECT NUMBER
N/A

5e. TASK NUMBER
N/A

5f. WORK UNIT NUMBER
N/A

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of New Mexico
1700 Lomas Blvd. NE, Ste. 2200
Albuquerque, NM 87106-3807

8. PERFORMING ORGANIZATION REPORT NUMBER
35278

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
USAF, AFRL
AF Office of Scientific Research
875 N. Randolph Street, Room 3112
Arlington, VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

13. SUPPLEMENTARY NOTES
None

14. ABSTRACT
Funds provided by this grant allowed us to purchase several large compute-servers, which supported intensive experimentation conducted under the auspices of MURI FA9550-07-1-0532. We conducted extensive mutational robustness studies. Given a population of variant programs, created from an original program by applying random mutations, we measure which variants still pass all available test cases and call them "neutral." The fraction of all variants that are neutral is defined as the program's "mutational robustness." Even when the mutations are restricted to statements executed by the test cases, mutational robustness is surprisingly high, 36.75\% on a corpus of programs taken from 22 production software projects, the Siemens benchmark suite, and a few specially constructed programs. Next, we conducted experiments on the assembly-level automated program repair algorithm using a benchmark test suite of ten open source programs. The ASM repair method successfully repaired all of the ten programs on at least one run, with an overall success rate of 0.54 per trial. We also conducted experiments on a recent extension of the automated program repair framework to elf files. This extension would be useful, for example, when working on embedded devices (continued on next page)

15. SUBJECT TERMS
Major equipment purchase to support ongoing MURI supported research.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Stephanie Forrest, PhD
U	U	U	UU	2	19b. TELEPHONE NUMBER *(Include area code)* 505-277-7104

SF298

14. Abstract (Continued)

and either the source code is unavailable or there isn't room to store the complete tool chain required for repairs at the abstract syntax tree level. We used the DURIP-supplied computers to conduct many of the elf level experiments, discovering that on the same benchmark set that we can repair five out of the ten programs at the elf level without any access to the source code.

(DURIP-10) HELIX Project Testbed
Towards the Self-Regnerative Incorruptible Enterprise
FA9550-10-1-0277
Final Report

Stephanie Forrest, PI
University of New Mexico
October 13, 2011

Under the auspices of this grant, we purchased the following equipment:

1. 1 Dell 224-8310 Power Edge R815
2. 1 Dell 224-9502 Power Vault NX3000
3. 1 Dell 224-8310 Power Edge R815
4. 1Apple Z0M4 - Mac Pro Two 6-Core Intel Xeon
5. 1 Dell 224-8310 Power Edge R815
6. 2 Apple 27" Monitors
7. Computer parts from New Egg(eg. video card, Cooler Master)
8. Video cards from New Egg
9. Video cards from New Egg

This equipment has been used to further the goals of our MURI-supported Helix project. Our focus has been on developing methods for automatically repairing programs and studying mutational robustness. Our method uses genetic programming to automatically search for repairs to documented bugs in programs with test cases. In earlier work we demonstrated the method on abstract syntax trees derived from C source code. Over the past year, we conducted experiments at the assembly-level using a benchmark test suite of ten open source programs. The ASM repair method successfully repaired all of the ten programs on at least one run, with an overall success rate of 0.54 per trial. Using this advance we demonstrated repairs of programs compiled to assembly code from several different languages, a major extension of our earlier work.
We also conducted experiments on a recent extension of the automated program repair framework to elf files. This extension would be useful, for example, when working on embedded devices and either the source code is unavailable or there isn't room to store the complete tool chain required for repairs at the abstract syntax tree level. We used the DURIP-supplied computers to conduct many of the elf level experiments, discovering that on the same benchmark set that we can repair five out of the ten programs at the elf level without any access to the source code.
Finally, we conducted extensive experiments on mutational robustness at all three program representation levels: abstract syntax tree, assembly, elf format binaries. Given a population of variant programs, created from an original program by applying random mutations, we measure which variants still pass all available test cases and call them "neutral." The fraction of all variants that are neutral is defined as the program's "mutational robustness." Even when the mutations are restricted to statements executed by the test cases, mutational robustness is surprisingly high, 36.75% on a corpus of programs taken from 22 production software projects, the Siemens benchmark suite, and a few specially constructed programs.

In addition to the published papers documented on the form, we have submitted the results of these experiments for publication at the International Conference on Software Engineering. We

are also preparing a paper on the assembly and elf levels of representation for submission either to PLDI (programming language design and implementation) or ISSTA (International Symposium on Software Testing and Analysis,)