Application to Participate in
National Science Foundation's "The Science of Cloud Computing"

Name: Mary Jean Harrold
Affiliation: Professor, School of Computer Science
ADVANCE Professor, College of Computing
Georgia Institute of Technology
Email: harrold@cc.gatech.edu

Topics of Interest:

4. Programming Models for the Cloud
7. Cloud Debugging, Certification, and Update
8. Cloud Self-Monitoring and Autonomic Control
11. Cloud Test-Beds

Current Research Activities Related to Cloud Computing

Several of my research activities can be applied to cloud computing; here I briefly overview a few of them. The first project, called Tarantula, provides new techniques for *automatic fault localization.* These techniques use information recorded about a program's execution on passing and failing test cases, along with statistical analysis, to automatically identify those parts of the program that are the most suspicious of being the program's fault. Studies with these techniques have shown that they are both effective and efficient.

The second project, called Matrix, provides techniques for change analysis and regression testing. *Change analysis* performs analysis to determine where changes are, what their impact is on the system, whether they interact, and how they are related to test cases. *Regression testing* is performed on the changed software to show that the changes behave as expected and that the changes have not had adverse effects on the unchanged parts of the system. Studies have shown that previous change analysis techniques were extremely imprecise (producing many false positives) and unsafe (producing many false negatives), but that Matrix performs significantly better than these techniques. Studies also show that the regression test cases created by Matrix are much more likely to uncover faults than test cases created by previous techniques. Results of our previous work on regression testing are now being used in industry.

The third project, called Gamma, monitors software after it has been deployed, and uses this information for tasks such as testing and debugging. Gamma uses a technique called *software tomography*, which analyzes the program to determine what to monitor, instruments the software to gather the determined information, collects the monitored information as the software executes, and combines the information to get a global picture of the execution. To date, we have used Gamma to determine and visualize the location of faults when the software crashes, and to predict when the software is likely to fail. Studies show the effectiveness of these techniques on real-world deployed software.

The fourth project, called Stinger, provides new techniques that can facilitate use of *symbolic execution* for analysis on real-world programs. Previous approaches cannot perform the symbolic execution on programs that interact with external components, such as Java libraries. Thus, techniques that use symbolic execution, such as automatic test-input generators, cannot be applied to real-world programs. Stinger uses a new approach called cloning that enables the symbolic execution to be performed in the presence of libraries. Thus, tasks that use symbolic execution, such as test-input generation, can be performed on real-world programs.

Abstract of Future Research Problems Related to Cloud Computing

Overall Goal

Develop software-engineering tools for development, testing, debugging, and monitoring of cloud software.

Monitoring Cloud Application for Detecting Potential Errors and Debugging

Monitoring software in the cloud is useful for detecting potential problems (e.g., detecting that an overloaded application is failing) so that steps to rectify it can be applied before the problem becomes wide-spread. Monitoring is also useful for debugging, and it can be used for debugging in the cloud. Typically, for debugging, developers try to reproduce the problem that caused the system to fail. However, reproducing the problem can be difficult for cloud software because of various reasons such as (1) the large number of users and multiple data centers that run the software, (2) the geographically distributed nature of the users and data centers, and (3) heterogeneity of the hardware in the data centers. Although the problem cannot be reproduced, can we debug the software using some other types of runtime data that could be recorded during runtime? To address this problem, we are interested in investigating (1) which program behavior to monitor, (2) which program behaviors are affected when the cloud parameters change, and (3) how these behaviors can be determined through static and dynamic analysis. Such information gathered through the monitoring can be used for detecting potential problems, detecting failures, and debugging errors.

Developing a Testbed for Unit and Regression Testing of Cloud Software

Testing of traditional software has been a challenging problem because of the complexity and size of the software. For cloud software, such testing is significantly more challenging because of the multitude of parameters (e.g., cloud API, and locations and numbers of data centers) of the software's operating environment. For example, suppose that there is a given set of standard manually-developed unit tests for the software. Can we use these tests effectively to determine whether a change in a parameter may cause the software to behave incorrectly? To address this problem, we are interested in adapting existing automatic unit testing techniques, and developing new techniques where needed, that can determine the subset of tests that may behave differently when a parameter of the cloud changes. Then, a testbed can be used to execute each such unit test by changing the parameter and checking whether the test still passes. For another example, when legacy systems are moved to the cloud, software that previously worked correctly may not work in cloud. How can we retest efficiently the software in the cloud to determine whether it still works as expected? To address this problem, we are interested in developing regression-testing techniques that will be specialized to discover regression errors arising from the differences between traditional and cloud computing environments.

Retrofitting Legacy Code to Programming Models for Cloud

There is much legacy software that exist and that will be required to be moved to the cloud so that those software can benefit from cloud computing. For example, many existing, important and widely-used applications, such as the information system of the Social Security Administration, which is predicted to collapse due to overloading by 2012, are written in Cobol. Thus, there will be significant research required to develop software-engineering tools that can perform tasks, such as rewriting (automatically or semi-automatically) legacy code using programming models (e.g., Map-Reduce) for the cloud. To address this problem, we are interested in developing tools to (1) automatically detect which parts of the application can be rewritten as a Map-Reduce operation, (2) help programmers detect those parts by providing relevant information gathered from static and dynamic analyses, and (3) verify that the developer's eventual rewriting strategy does not introduce errors into the system.