Poster Abstracts

Quality Assessment that uses Language Processing

Henry Feild (Loyola College in Maryland)

Many attempts have been made to establish a correlation between some static aspect of code and the measure of its quality. In this study, a new approach called Quality Assessment that uses Language Processing (QALP) is explored. QALP tries to correlate program quality to a strong comment-code relationship. The idea is that high quality programs will consist of high quality comments and code, which can be determined by using their identifiers. Comments should shed a good deal of understanding on the immediate code, and, likewise, the code should be readable by means of descriptive variable and function names. The study breaks each program into function sets. Each function set contains both the leading and internal comments of the function and the code. All stop words (i.e., common English words) are stripped from the comments and all the programming-language syntax is removed from the code, leaving only identifiers. The words left in the comments are then processed in several different ways to correlate them to the identifiers in the code. The different correlation methods are used in the event that the programmer used the full, English word in the comment and an abbreviation in the code (i.e., "Find the statistics" in the comments vs. the variable name "stat" in the code). An important part of this investigation is an empirical study of this technique's effectiveness. This study requires programmers to rate functions and compare their rankings with those of the prototype tool.

Using Language Clues to Discover Cross-cutting Concerns

David Shepherd (University of Delaware), Tom Tourwe (Centrum voor Wiskunde en Informatica), and Lori Pollock (University of Delaware)

Researchers have developed ways to describe a concern, to store a concern, and even to keep a concern's code quickly available while updating it. Work on identifying concerns (semi-) automatically, however, has yet to gain attention and practical use, even though it is a desirable prerequisite to all of the above activities, particularly for legacy applications. Of course, automating parts of concern identification will lead to less than 100% precision, but concern identification information can, even with errors, still save a developer considerable time. This poster describes a concern identification technique that leverages the natural language processing (NLP) information in source code. Developers often use NLP clues to help understand software, because NLP helps them identify concepts that are semantically related. However, few analyses use NLP to understand programs, or to complement other program analyses. We have observed that an NLP technique called lexical chains offers the NLP equivalent of a concern. In this poster, we investigate the use of lexical chaining to identify cross-cutting concerns, present the design and implementation of an algorithm that uses lexical chaining to expose concerns, and provide examples of concerns that our tool is able to discover automatically.

Interclass Test Dependence

Weilei Zhang and Barbara Ryder (Rutgers University)

In class testing for object-oriented program, interclass test dependence is important in deciding the class test order in order to reduce the number of test stubs. However, as a concept, interclass test dependence has not been formally defined per se but only used in an intuitive way in the existing techniques. We analyze the spurious test dependences present in the existing techniques and give a formal definition for interclass test dependence

to prune them out. A definition in terms of program analysis is also given and a generalized algorithm is designed to approximate the interclass test dependence according to the definition. The algorithm is parameterized by the precision of program analysis and three different program analysis algorithms are used in the implementation. The results are evaluated and compared.

CRISP: A Tool for Incremental Debugging of Java Programs
Ophelia Chesley and Xiaoxia Ren (Rutgers University)

CRISP has been created as an exploration tool that extends the functionality of our previous work on Chianti, program change analysis software developed as a plug-in to the Eclipse IDE. Chianti takes two versions (called the original P and new P') of an application written in Java and decomposes their differences into 13 atomic changes. Integrated with both static and dynamic call graphs based on a programmer's preference, Chianti also calculates the set of regression tests (called "affected tests") that exercise these changes and reports the group of atomic changes (called "affecting changes") associated with these affected tests. From our experiments on 52 weeks of Daikon data, an affected test could potentially be impacted by a large number of affecting changes. When these affected tests failed, or exhibited unexpected behavior such as throwing an exception, programmers needed additional information to assist them in analyzing and debugging their application. Using affecting changes generated by Chianti, CRISP provides the programmers with a graphical user interface where they can select a subset of affecting changes and automatically apply them to the original version of the application, thus creating an intermediate source program P+, versioned between P and P'. Programmers can then re-execute the affected test and examine its behavior concentrating only on the set of affecting changes being applied. Through this extension of Chianti, programmers can iteratively select, apply, and "un-apply" individual affecting changes and effectively converge on a small set of failure-inducing changes.

Relating User Session Clusters to Dynamic Web Application Behavior
Sreedevi Sampath, Sara Sprenkle, Emily Gibson, Lori Pollock (University of Delaware), and Amie Souter (Drexel University)

User sessions provide valuable insight into the dynamic behavior of web applications as well as play a key role in user-session-based testing, which gathers user sessions in the field and replays selected sessions to test an evolving application. To reduce the testing and analysis effort, testers reduce the set of collected user sessions by either clustering user sessions by their shared URL attributes or by program coverage requirements-based reduction techniques. Clustering based on URL attributes can be a considerably less expensive approach; however, the tradeoff may be that the clustering is not representative of dynamic behavior similarities. This poster describes our analysis of user session data to reveal any correlations between user sessions clustered on attributes of the user sessions themselves and the relative dynamic behavior of the program for those user sessions. Our results show that clustering based on attributes provides similar clustering of program coverage and fault detection capabilities, and thus the expense of test suite reduction using mappings between user sessions and program coverage requirements can be eliminated, and on-the-fly test suite reduction can be performed with respect to attributes of user sessions alone. The results of our analysis of user session clustering can be used to formulate test suite reduction techniques as well as to learn more about how clusters of web application use cases are related in terms of the underlying user session attributes, program coverage, and fault detection.

An Empirical Comparison of Test Suite Reduction Techniques for User-session-based Testing of Web Applications
Sara Sprenkle, Sreedevi Sampath, Emily Gibson, Lori Pollock (University of Delaware), and Amie Souter (Drexel University)

Automated cost-effective test strategies are required to provide reliable, secure, and usable web applications to consumers, government agencies, and businesses who rely on web applications to perform daily tasks. User-session-based testing is an automated approach to enhancing an initial test suite with real user data, enabling additional testing during maintenance as well as adding test data that represents real usage as operational profiles evolve. Test suite reduction techniques are critical to the cost effectiveness of user-session-based testing because a key issue is the cost of collecting, analyzing, and replaying the large number of test cases generated from user session data. We designed and performed an empirical study comparing the test suite size, program coverage, and fault detection effectiveness of three different requirements-based reduction techniques and three variations of concept analysis reduction applied to two web applications. Our poster describes our experiments and presents the results for one application.

A Fault Detection Framework for Comparing Test Suite Reduction Techniques
Emily Gibson, Sreedevi Sampath, Sara Sprenkle, Lori Pollock (University of Delaware), and Amie Souter (Drexel University)

As organizations increasingly rely on web applications for routine tasks, testing these applications becomes crucial. In addition to gathering and/or generating test cases, a tester must reduce the number of test cases in a test suite to make running the suite computationally feasible. Because a number of test suite reduction techniques have been developed, fault detection has been used as a metric to evaluate and compare reduced suites. We present an automated framework to replay reduced suites and detect seeded faults in the subject web application.

An Automated Approach to Improving Communication-Computation Overlap in Clusters
Lewis Fishgold, Anthony Danalis, Martin Swany, and Lori Pollock (University of Delaware)

Parallel clusters have become common platforms for programmers to achieve desired runtime performance for applications with high processing demands. Unfortunately, scaling these applications to larger numbers of CPUs for even higher performance gains often fails because the communication overhead increases at a similar rate. This poster describes our work in developing a program transformation that can significantly reduce the communication overhead of parallel MPI programs by restructuring a program towards maximizing communication-computation overlap. Our optimization specifically targets the large community of domain scientists that use MPI to implement their parallel algorithms, and RDMA-enabled network clusters. We describe a source-to-source optimizer that is capable of automatically identifying safe transformation and performing the necessary restructuring to pre-push data during computation to exploit the underlying capabilities of the RDMA-enabled network. Our initial experience indicates that while the current prototype is limited in its targeted application forms, the potential performance gains and generalization to a wider set of applications is quite promising.

Adding Security Controls to Dynamically Optimized Mobile Programs
Anteneh Addis Anteneh, Mike Jochen, Lori Pollock, and Lisa Marvel (University of Delaware)

Mobile programs can provide great functionality to modern computing systems. Allowing mobile programs to evolve during execution as they move through a network of hosts can improve program performance. Dynamic optimization systems can achieve this performance gain by taking into consideration the current runtime characteristics of the program to make better-informed optimization decisions. Performing this kind of program transformation on multiple computing hosts throughout a network will provide the same performance gains while reducing the overhead on the local machine. The decrease in overhead is gained by distributing different responsibilities of the dynamic optimization process to multiple hosts. The use of efficient mobile programs can greatly benefit computing systems; however, a host must be able to confirm the authenticity of a program before it can proceed to run the software.

A serious issue concerning mobile programs is that the code may be forged or altered enroute from a server to the local machine. Existing program validation and authentication methods such as digital signatures and checksums are not adequate when programs are allowed to evolve or change as they move through a network. Other validation methods involve the execution of a program, which is not desirable if the authenticity of the program is not yet confirmed. Therefore, there exists a need for new security measures that enable users to reap the benefits of dynamically evolving mobile code while mitigating the risks of the use of these mobile programs. We propose a first step in developing a security framework that will restrict how a program can change as it passes through a network of hosts. The system will allow transformations to occur based on a defined program transformation policy. Restricting what parts of a program can change as it is being transformed will make mobile programs a safe and efficient technology.

Chapter 1 Bridging Parsing and Lambda Lifting
Barbara Mucha and Marco T. Morazan (Seton Hall University)

Lambda lifting transforms a program with local function definitions into a program containing only global function definitions that are combinators. The technique works well for languages that support curried functions, but must be adapted for Scheme-like languages in which functions are not curried. Recent observations made by Danvy and Schultz have improved the complexity of lambda lifting from $O(n^3)$ to $O(n^2)$. Despite the progress made in improving the complexity of lambda lifting, it remains a non-intuitive algorithm that has been separated from syntactic analysis. Lambda lifting can be made more intuitive by exploiting the process of parsing. In this article, we present an $O(n^2)$ lambda lifting algorithm for a pure subset of Scheme, called MT-Scheme, that exploits syntactic analysis and that incorporates the insights of Danvy and Schultz. Exploiting syntactic analysis to perform lambda lifting yields a more intuitive algorithm that makes lambda lifting accessible to non-experts in the implementation of functional languages, that provides a basis from which improvements can be made, and that has pedagogical value by facilitating the teaching of lambda lifting to students. The MT lambda lifting algorithm is implemented as part of the MT-Scheme compiler.

Design and Implementation of Compiler and Toolchain for Cellular Architecture

Ziang Hu, Geoff Gerfin, Brice Dobry, Guang R. Gao, Weirong Zhu, and Juan Cuvillo (University of Delaware)

Cellular architectures (IBM Cyclops-64 as an example) expose many challenges to compiler and toolchain design. The architecture has the following features: 1) large number of hardware thread units (160 integer thread units and 80 floating point units on a single chip); 2) heterogeneous, explicit, and complex memory hierarchy (with on-chip scratchpad memory for each thread unit, on-chip shared SRAM, off-chip DRAM, and large register file for each thread unit); 3) on-chip memory banks (160 banks) and thread units (160) are connected with one level switch bar, which provides very large on-chip memory access bandwidth. This architecture has very large computation power and very large on-chip memory access bandwidth. Comparatively, the off-chip memory bandwidth is limited (4 memory banks, each 4GB/s). The off-chip memory access latency is to be hidden by mutlithreading. Among the many challenges to compiler and toolchain, the following are considered the most important: multithreaded programming model; compiler, assembler, linker that can allocate data objects to different memory banks with different size and latency; dynamic memory allocation and management. In this poster, major design decisions on compiler and toolchain will be presented; emphasis will be on how the compiler and toolchain support the heterogeneous memory hierarchy. A preliminary performance study will be reported. This study may also be applied to other architectures, including multi-core architecture, DSP with heterogeneous memories, chip multiprocessors, and other single-chip parallel architectures.