IBM RESEARCH - INDIA, NATIONAL UNIVERSITY OF SINGAPORE, INDIAN INSTITUTE OF SCIENCE

Workshop on the Future of Debugging

Mysore Park Workshop Series held at Infosys Mysore Campus

Satish Chandra (IBM), K.V. Raghavan (IISc), Abhik Roychoudhury (NUS), Saurabh Sinha (IBM)

February 28 – March 2, 2012

This document is a summary of the discussions that took place at the International Workshop on the Future of Debugging, held under the Mysore Park Workshop Series from 28 February to March 2, 2012.

1. Preamble

The workshop on the Future of Software Debugging attracted around 40 participants, with a substantial participation from Indian universities and companies. The topics ranged from fault localization in software, testing, fixing of errors, as well as research on building the overall infra-structure for software debugging. The workshop keynote was delivered by Andreas Zeller (Saarland University), who focused on the future of software fixing via automated and semi-automated methods. The workshop ran from Tuesday Feb 28th morning to Friday March 2nd noon. On Wednesday afternoon, the participants enjoyed an excursion to the Mysore Palace.

The workshop webpage is located at http://researcher.ibm.com/view_project.php?id=2581

2. Technical contents of individual talks and ensuing discussions

Day 1

Implicit Fault Localization for Avoiding Failures

Mauro Pezzè from University of Lugano gave the first technical talk in the workshop. He raised several interesting directions related to fault localization, which attracted questions from the audience. One of the issues mentioned by Mauro relates to work-arounds in locating and fixing software faults. Thus, instead of building methods to precisely pin-point the errors, researchers could focus on approximate methods to get a rough idea of the error location, and then could find a way of simply avoiding the "faulty" software component. Notably, this idea of software workarounds was also discussed and elaborated further by Alessandra Gorla, also from the University of Lugano – later in the workshop.

Mauro's talk mentioned the possibility of using behavioral anomalies for approximate localization of errors (and subsequent failure recovery). The overall approach tries to capture program behavior using methods similar to dynamic specification mining. In particular, automata model mining was discussed. By following such a behavior capture method, anomalies can simply be "uncommon" method call sequences. One of the intuitions presented in Mauro's presentation was that anomalies are empirically found to occur in clusters and close together. For failure recovery, Mauro mentioned the possibility of taking code fragments with known faults and wrapping them with work-around code.

Can We Debug with Traces?

Venkatesh Prasad Ranganath from Microsoft Research India discussed methods for mining of temporal rules. Temporal rules are simply specific patterns of event occurrences – as observed in execution traces. Venkatesh hinted at the possibility of execution trace based debugging being achieved by specification mining. In particular, observable behavioral equivalence

between two program versions can be checked by mining models for the two versions, and comparing the models.

Explaining Error Traces Using Craig Interpolants

The talk by Martin Schäf from United Nations University (Macau) related to abstracting of error paths, which could have been found as counter-example to a temporal property. Thus, it is assumed that model checking of a temporal logic property is done in advance. This leads to a counter-example which is subjected to further analysis to find error root-cause. The notion of error invariants was discussed in this talk. Roughly speaking, error invariants guarantee that the reason for reaching an error location remains unchanged, as long as the invariant holds.

Comparing Executions to Explain Software Failures

Nick Sumner from Purdue dealt with methods for execution trace comparison. The idea here is to compare a failing execution trace with a successful execution trace so as to identify the error root cause. The key question in such works is how to find a "suitable" execution trace. Nick's talk focused on predicate switching – generating a correct execution by flipping a branch instance of a failing trace. An alert reader will note that this does not immediately amount to fault localization to the flipped branch, since only a branch instance is being flipped. Once such a successful execution is generated, one could compare more than the control flow across executions – e.g. we can compare the memory behavior across execution traces.

Lightweight Log-Based Code Coverage

Madan Musuvathi from Microsoft Research Redmond discussed useful infrastructures to enable software debugging. In particular, he focused on lightweight code coverage – methods to achieve basic block coverage in a program's control flow graph. This was achieved by using hardware mechanism used by breakpoints in a clever way.

Debugging of Data Centric Programs

Diptikalyan Saha from IBM Research – India presented an advanced fault-localization technique for data-centric ABAP programs that iterate over persistent data records, apply some business logic, and produce a report as the output. Overall, the technique is an example of spectra-based, or statistical, fault localization, which compares passing and failing test executions to identify suspicious statements, albeit with a generalized notion of differencing. The key idea is to extract multiple passing and failing execution slices (for comparison) from a single program execution by essentially treating the processing of each input data record as a separate execution.

Debugging of Evolving Programs

Abhik Roychoudhury from National University of Singapore gave the last talk on Day 1. His talk focused on regression debugging, or debugging of evolving programs. The aim here is to locate errors that creep in introducing new features to programs. Symbolic execution based debugging methods were suggested to find the root-cause of such regression errors. The central intuition

here is that symbolic execution of the earlier passing programs can help provide us with a reference intended behavior, with which the buggy behavior is compared.

Day 2

Reusing Debugging Knowledge via Semantic Bug Search

On the second day of the workshop – Earl Barr from University of California Davis gave a talk on debugging via searching in bug databases. One question here is how to build the bug database in the first place. Earl mentioned the possibility of defining bugs as deviant behavior – such that potential bugs in execution profiles can be identified. Then, given a bug, a programmer can search for "similar" bugs using a bug query language. Earl's research direction has synergies with the direction presented by Sriram Rajamani later in the workshop.

Debugging in Absence of Code

Pavan Chittimalli from Tata Research Development and Design Centre discussed the situation where testers may not have access to program source code. In such cases – the tests are written directly from the software requirements. Pavan's talk mentioned the possibility of running these tests on a business process model to get an idea about the real software execution of these tests.

Angelic Debugging

Satish Chandra from IBM T.J. Watson Research Center presented a computer assisted debugging method called angelic debugging. The method starts with identifying expressions in the program that can make failing tests pass; this phase uses symbolic execution to find such expressions. The method then tries to rule out those candidate expressions that are not good fix locations. It uses the principle that if any modification to such expressions would cause at least one passing test to fail, then it is not a good place to fix the program.

Day 3

Cooperating Testing and Analysis: Implications for Debugging (KEYNOTE)

Andreas Zeller gave the keynote of this workshop. His talk focused on computer assisted methods for fixing program errors. Andreas proposed the possibility of using dynamic specification mining to generate specifications for passing and failing tests. The difference of such mined specifications can then be used to generate program fixes.

Andreas dealt at length on support for specification mining. Dynamic mining methods rely on a test suite which provides execution traces. Andreas mentioned the possibility of having specification mining to guide the test suite generation itself. Thus, starting from a test suite, we can mine specifications. These specifications can lead us to test corresponding to corner cases, which are then added to the test suite. The augmented test suite then lends itself to a better software specification via dynamic specification mining.

Reproducing Field Failures

Alex Orso from Georgia Tech presented methods for debugging of field failures. Given the crash log, often we have the stack trace plus some other static information. Alex's talk focused on generating an actual program input which will generate an execution similar to the execution data captured in the crash log. This is useful for reproducing field failures at the developer's end. The main technique is to identify a sequence of control locations the execution should go through (the "breadcrumbs") and then use symbolic execution to find an input going through these breadcrumbs in sequence.

Debug Advisor

Sriram Rajamani from Microsoft Research India presented the DebugAdvisor tool which has found acceptance among developers. The observation here is to build a bug repository from previously resolved bugs. Given a new bug report – this is searched in the repository. Bug reports may contain plain text describing the error, stack traces as well as memory dumps. If the developer puts the effort of including execution traces in the bug report – there exist additional possibilities in terms of debugging. For example, one could use dynamic specification mining on all such collected traces in the bug reports for a given application.

Debugging of Concurrent Programs

Vijay Saraswat from IBM T.J. Watson Research Center focused on debugging of concurrent programs running on platforms with extremely large number of cores. His focus was on X10, a concurrent programming language recently developed at IBM.

Performance Modeling and Debugging of Distributed Queries

Kapil Vaswani from Microsoft Research India focused on root-causing performance bugs, rather than functionality errors. The work is in the domain of performance modeling for distributed systems.

Debugging with MaxSAT

Rupak Majumdar from Max Planck Institute presented an approach for software debugging using SAT solving. For a failing input, the logical incompatibility of the expected output with the processing of the input (as captured logically via strongest post-conditions) can be highlighted via max-SAT solving – the maximum number of clauses that can be satisfied even though the whole formula is unsatisfiable.

Rethinking the User Experience of Debugging

Rob Deline from Microsoft Research Redmond gave the last talk on Day 3. His talk focused on user experiences in software debugging, and the design of several innovative development tools to alleviate the user experience.

Day 4

Automatic Workarounds

Alessandra Gorla from University of Lugano gave the first talk on Day 4. Her talk focused on the observation that software is inherently redundant, as is evidenced from the presence of software clones. Thus, this redundancy could be exploited to find work-arounds to an observable error – so that the manifested error disappears (even though the error root-cause may not have been fixed). Suggested method for exploiting redundancy is via dynamic analysis – where "equivalent" operation sequences are observed at run-time and summarized as rewrite rules.

Automatic Repair of HTML Generation Bugs of PHP

Max Schafer from IBM T.J. Watson Research Center gave the penultimate talk in the workshop. His talk focused on program repair, specifically the repair of PHP programs. Repair is achieved via constraint solving driven by a dynamic approach. Given a test suite, for each test a constraint is generated (to capture the intuition that the observed and expected outputs should match). The constraints from all tests are solved together to generate repair.

Driving Debugging based on Proof Systems

R. K. Shyamasundar from Tata Institute of Fundamental Research gave the last talk of the workshop. His presentation was focused on using proof systems to drive software debugging.

3. Summary

Overall, we feel that the workshop achieved various goals in terms of research dissemination and manpower enhancement.

From the research point of view, we had engaging discussions on emerging trends in software debugging. Symbolic execution and constraint solving techniques were found to have an increasing presence in modern-day debugging methods. The importance of dynamic specification mining to drive testing and debugging came up several times during the presentations. Last but not the least, the importance of software repair and workaround generation was emphasized by several presentations, including the keynote by Zeller.

From manpower training point of view, we note that several PhD students from IISc and other Indian institutions attended the workshop. Several of them also made brief presentations on their ongoing PhD research and actively sought feedback from the workshop attendees – not only during their talks, but also during the coffee breaks, lunches, and through informal conversations. We believe this is an extremely important component of holding such a workshop.

The workshop organizers thank the Mysore Park Series for sponsoring this workshop, as well as Infosys for hosting the workshop in their campus. Thanks are also to G. Ramalingam and Sriram Rajamani for their advice on logistics in holding such a workshop.