Discussion Summary of the IWPD 2014 Panel:
Program Debugging – Research and Practice
November 3, 2014
11:00 am – 12:37 pm
Panelists: W. Eric Wong, Franz Wotawa

The panel started with brief presentations by each panelist about their experiences in program debugging, research challenges, and issues that have to be resolved before research results can be applied to industrial settings. PowerPoint slides of these presentations have been posted at the IWPD 2014 web site (http://paris.utdallas.edu/iwpd14) under the "Panel" section. The floor was then opened for comments and discussion.

The first question was how to conduct effective program debugging for cloud computing and SOAs. The main challenge is that the overall systems can be very large and complicated, including both software and hardware. The general idea is to use a combination of different debugging techniques. The first step is to use an appropriate technique with a low computation footprint to identify a small part of the system that may be responsible for the detected failure. Then, other techniques are used to further reduce the search domain, which may contain the root cause(s) of the failure. An alternative approach is to use a *hierarchical diagnosis* if additional specification details of the underlying system are also available. The third approach reduces the search domain during fault localization by taking advantage of special domain expert knowledge that programmers may have for the systems being debugged.

The second question was whether the software development process has an impact on the number of post-release bugs. The discussion was focused on how to estimate the number of remaining bugs in a system using process- and/or product-oriented metrics, and the importance of introducing a more formal process (referring to the ISSRE 2013 invited talk from IPL/NASE). The latter requires rules (such as zero warning compilation) to be fulfilled during the development. It is also affected by specific programming languages or restrictions.

One participant voiced his concern of how to reduce effort spent on manual debugging. Possible solutions include the use of reverse engineering and data analysis to identify suspicious code. Another participant emphasized that the exact execution environment of a program failure might not be known to programmers, which makes it extremely difficult to reproduce the failure.

An important research challenge is that programming over time becomes more and more dynamic. Various libraries are used during development as well as in the deployed systems. Memory dump has too much data that may not provide useful hints to aid in debugging. Programmers have to spend time screening such data to filter out possible cases for failures. Moreover, a failure might have different root causes, which further makes software fault localization even more complicated.

Also discussed was the debugging of systems consisting of both hardware and software. A major challenge is that sometimes it is not easy to determine whether a failure is due to hardware faults or software faults. This issue is somewhat related to multi-core environments and/or concurrent programming. An additional question is whether there is appropriate tool support for debugging these systems. All the participants agreed that more research should be conducted along this direction.

There are other issues that need to be resolved before results from academic research can become industry practices in real-life settings. One issue is to better understand the challenges that

practitioners face at their daily work. Assumptions made for a controlled experiment at a university laboratory may not be realistic in practice. Case studies should use data from real-life applications to demonstrate the feasibility and potential benefits of techniques developed based on academic research. Unfortunately, such collaboration between industry and universities in most cases does not exist, as companies are reluctant to share their data. Another issue is to how to appropriately include software testing and program debugging as part of the computer science and software engineering curriculum. It is too late to wait until students are in their senior year. These important topics should be introduced from the first programming course and repeated whenever appropriate with more advanced content throughout the entire undergraduate curriculum. Some academic programs have already started to adopt this approach.

The panel also discussed whether it would be more cost-effective to take action to prevent bugs from being introduced during development rather than spending effort detecting and fixing these bugs. Our industry participants emphasized that any techniques which can help them avoid bugs are of particular interest.

There was also unanimous agreement that it is very important to provide the research community with a benchmark infrastructure (including programs with real-life bugs and test cases used for in-house and field testing) to support studies on software testing, fault localization, and bug fixing. One of the panelists, Professor Eric Wong, mentioned his research group's ongoing effort to create a repository with a list of papers on software fault localization based on a comprehensive survey the group recently completed at the University of Texas at Dallas. The repository is also to be extended to include source code, test cases, and tool support. In addition to research papers, reports by industry practitioners discussing the problems and challenges they have faced at work are especially welcome.

The panelists and the audience agreed on the necessity of using all means for a closer interaction between academia and industry. Establishing a forum such as IWPD to provide a platform for interested parties to get together to discuss ideas, exchange experiences, and review current practices and tools support is the right direction to go.