Answering reachability questions
Thomas LaToza
School of Computer Science, Carnegie Mellon University
Tuesday, January 12, 2010
3:00pm Rm 110 Purdy-Kresge Library

Abstract:
What are the most frequent, time-consuming, hard-to-answer, and error-prone questions professional software developers ask about programs? Reachability questions. A reachability question is a search upstream or downstream across paths from a statement for target statements. For example, a developer debugging a deadlock searched downstream for calls acquiring resources.

My studies indicate that reachability questions are pervasive throughout coding tasks. In one study, half of the bugs developers inserted were associated with reachability questions developers asked or should have asked. Developers report asking these questions more than 9 times a day, and 82% agree at least one is hard to answer. Neither increased professional experience nor even increased familiarity with a codebase make reachability-related questions easier or less frequent. In another study, 9 of the 10 longest investigation and debugging activities involved answering a single reachability question.

Using existing tools, developers traverse paths across method calls in search of target statements. Reachability questions are hard to answer because developers must guess both which paths lead to targets and which paths are feasible and may execute. To help developers more effectively answer reachability questions, I am designing a new kind of reverse engineering technique in which developers search across paths for target statements. Starting at a statement in a program, developers enter search strings that are matched against identifiers or comments along paths. Specific situations can be considered by posing “What if?” questions such as “What happens when this data table is uninitialized?”

A static analysis for answering reachability questions determines the feasible paths through conditionals. Existing approaches either do not eliminate infeasible paths or are too slow to be used in an interactive tool. However, examples of reachability questions suggest that many common infeasible paths are caused by conditionals evaluating variables that may only contain constants (e.g., dynamic dispatch, flags). I am designing a fast feasible path analysis to eliminate infeasible paths caused by constant-controlled conditionals. A preliminary implementation is able to eliminate many common infeasible paths through a 50 KLOC Java program in just 13 seconds of analysis time.

Biography:
Thomas LaToza is a Ph.D. candidate in Software Engineering in the School of Computer Science at Carnegie Mellon University. He is co-advised by Professors Brad Myers and Jonathan Aldrich. His research interests span both Human-Computer Interaction and Software Engineering and include studies of developers, software visualization, and program analysis. He received BS degrees in Computer Science and Psychology from the University of Illinois at Urbana-Champaign.