Visualizing the Results of Field Testing

Brian Chan and Ying Zou
Dept. of Elec. and Comp. Engineering
Queen's University
Kingston, Ontario, Canada
{2byc, ying.zou}@queensu.ca

Anand Sinha
Handheld Software
Research In Motion (RIM)
Waterloo, Ontario, Canada
asinha@rim.com

Abstract

Software must endure rigorous field testing before market deployment. Field testing helps uncover problems due to unexpected user behaviors and unforeseen usage patterns in a natural setting. Instrumented versions of the application are used during field testing. This enables the collection of field data in real-time without the developer's interference. A problem report is sent and stored in a central repository when an unexpected situation occurs. Common information in the report includes error messages and call stacks for a problem. The report also records the user who had the problem.

Developers must analyze and resolve these reported problems. Problems reported by a larger number of users are a high priority for developers to resolve. To verify the repair procedure of a problem or to understand its peculiarity, developers must often replicate the scenarios which trigger the problem. By understanding the characteristics of the users reporting a problem and its peculiarities, developers should be able to gain insight into ways of resolving it. However, the large number of users involved in field testing along with the variety of problems reported by them increases the complexity of managing the field testing process. Most field testing processes are monitored using ad-hoc techniques and simple metrics (e.g., the number of reported problems). The developers often examine problems individually without a global view on the relation between the different problems and the relation between the users reporting these problems. For instance, it may be the case that:

1. A particular set of problems co-occur frequently together; therefore studying and resolving any one of these problems might lead to the resolution of all these problems. With some problems easier to replicate than others, such information is likely to lead to faster resolution of these problems. We developed a *problem graph* to highlight such information.
2. Several users often report the same set of problems (i.e., share the same problem profile); therefore recruiting a few of these users to verify any fix is often a faster option instead of deploying the fix across the field blindly and awaiting problem reports. We developed a *user graph* to identify users reporting common problems. We can cross reference the problems reported and recruit users to replicate them.
3. A large number of problems are reported, however very few of these problems have a wide impact on many users. Fixing problems with high impact is usually a high priority effort. We developed an *interaction graph* to give a global view of the relation between the participants of field testing (i.e., users) and the outcome of the testing (i.e., problems).

To help developers perform deeper analysis of field testing results, our proposed three types of graphs visualize the relations between users and their reported problems. The graphs help provide a high level view of the large amount of field testing results. Using our visualization, developers can analyze massive amounts of data reported during field testing. We also provide techniques to automatically highlight patterns that demonstrate important user and problem interactions. In particular, such patterns help developers categorize, prioritize, and replicate problems, allowing developers to observe new relationships that are often overlooked in practice. By visualizing the field testing results across multiple product releases, developers can compare the progress of field testing efforts for different releases. We demonstrate the effectiveness of our proposed techniques using the field testing results for two releases of a large scale enterprise application used by millions of users worldwide.