Debugging Reinvented

Asking and Answering Why and Why Not Questions about Program Behavior
Debugging with Whyline
Implementation Overview

Steps

- Record trace
- Load trace
- Create I/O history
- Derive questions

Outcome

- Sequence of events
- Complete order of events
- I/O events
- Why-did-questions

- Class and source files
- List of output instructions
- Output affecting fields and invocations
- Call graph
- Why-didn't-questions
## Debugging Concepts

<table>
<thead>
<tr>
<th>Classic Debugging</th>
<th>Debugging with Whyline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developers translate questions about behavior into questions about code</td>
<td>Developers select a question from a set of why-did and why-didn’t questions</td>
</tr>
<tr>
<td>Debugging at runtime</td>
<td>Post mortem approach: Debugging after execution</td>
</tr>
<tr>
<td></td>
<td>Questions about things that did not happen</td>
</tr>
</tbody>
</table>
Results and Outlook

- Novice programmers with the Whyline were *twice as fast* as experts without it.
- Experts using the Whyline were *twice as fast* compared to experts using classic debugging tools.

Program understanding and debugging represent up to 70% of the time required to ship a software product.