Making Software Correctness Profitable: Leveraging comparative advantage in software engineering tools

Todd W. Schiller, Kellen Donohue, and Michael D. Ernst
{tws,kellend,mernst}@cs.washington.edu

Software bugs cost the US economy an estimated $59.5 billion annually

- Testing and verification are labor intensive; skilled labor is expensive
- The uncertainty of future bug costs makes establishing correctness during development unattractive

An Abundance of Cheap(er) Labor

<table>
<thead>
<tr>
<th>Role</th>
<th>Average Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Software developer</td>
<td>$90K/yr</td>
</tr>
<tr>
<td>Computer-science related graduates</td>
<td>$59K/yr</td>
</tr>
<tr>
<td>Est. trade school IT specialist</td>
<td>$66K/yr</td>
</tr>
<tr>
<td>Liberal arts graduates</td>
<td>$34K/yr</td>
</tr>
<tr>
<td>Average Mechanical Turk worker</td>
<td>$1.40/hr</td>
</tr>
</tbody>
</table>

Adaptive Semi-Automated (ASA) tools use less-skilled, less-expensive, workers

Decompose large tasks into automated and human-performed subtasks

- Tools adaptively guide less-skilled users through the task
  - Create or redistribute subtasks as needed
  - Similar to wizards, but with complex control flow

ASA Tool Design Principles:
1. Target well-defined skill sets
2. Exploit parallelism between subtasks
3. Create labor markets to minimize costs

ASA Program Verification
Goal: statically verify the absence of run-time exceptions in C# programs

4 Basic Phases:

- Purity
- Pure Method Specs
- Method Specs
- Object Invariants

Static and dynamic inference are combined to minimize the required skill-level

Learn more in our paper “Rethinking the Economics of Software Engineering” available at http://bit.ly/c8zN9X

Supported in part by IBM through a John Backus Award and by an NSF Graduate Research Fellowship