Supporting Test Oracle Construction

Matt Staats

KAIST

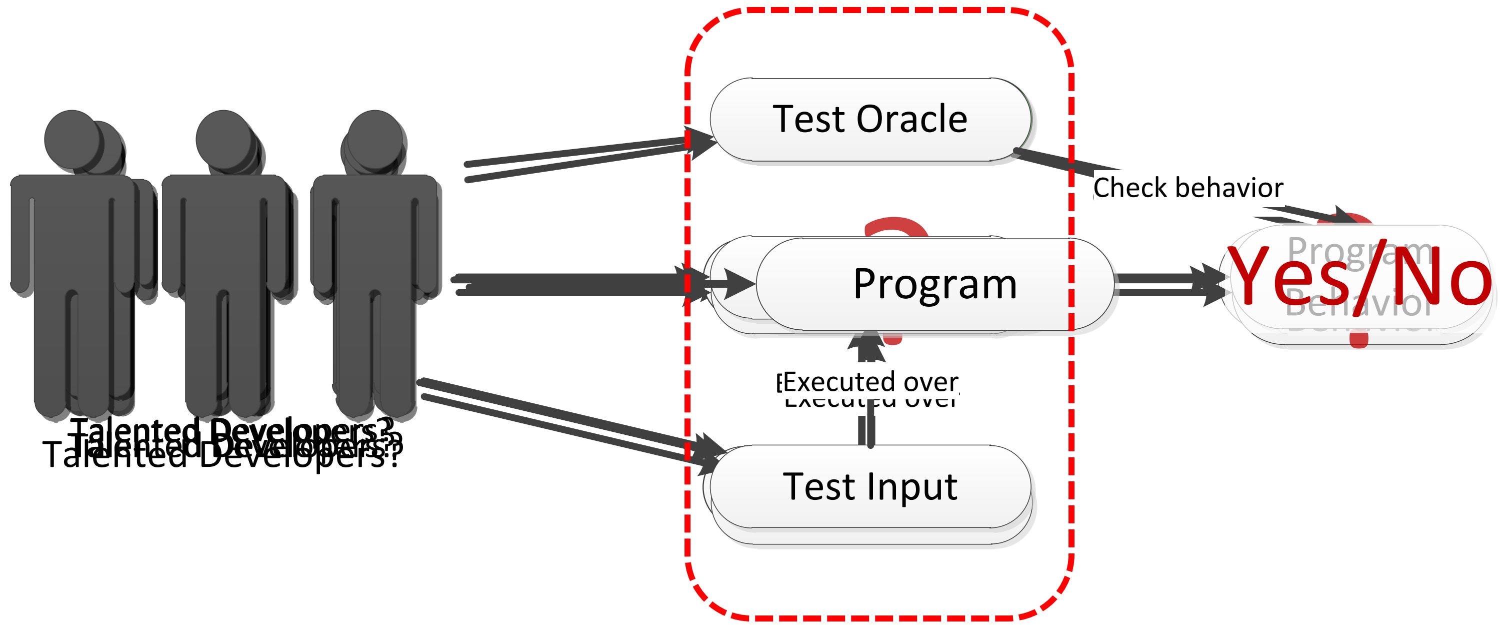
Shin Hong, Moonzoo Kim,
Gregg Rothermel

KAIST / University of Nebraska-Lincoln

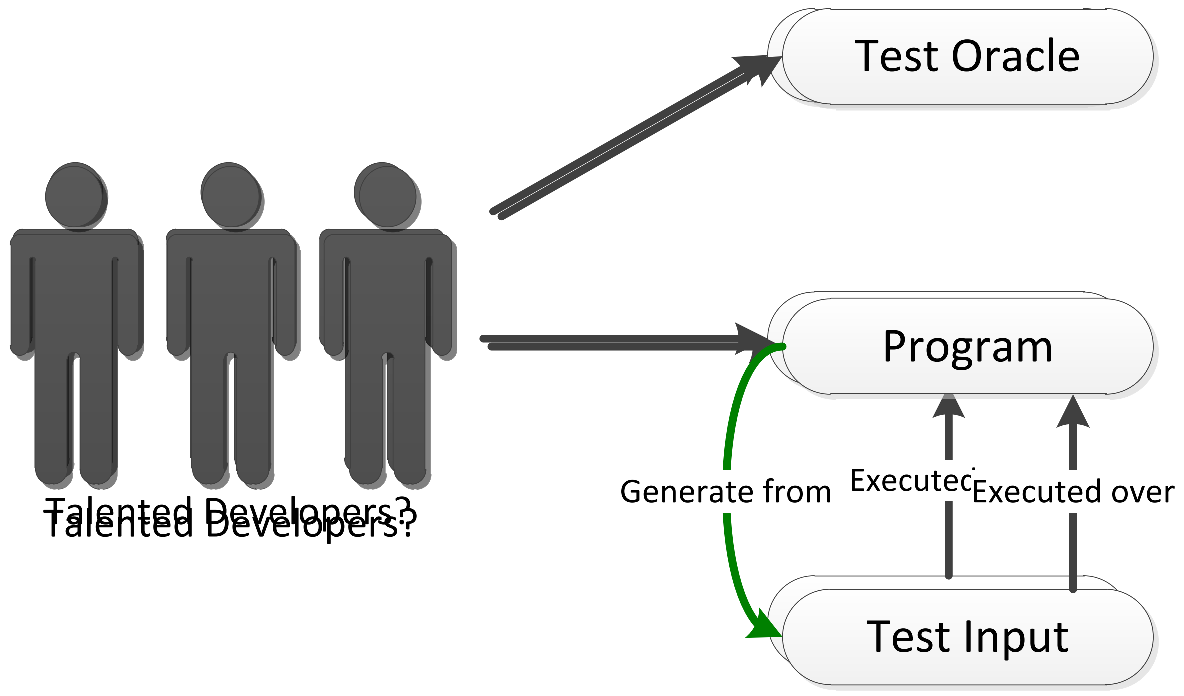
Gregory Gay, Mats Heimdahl

University of Minnesota Twin-Cities

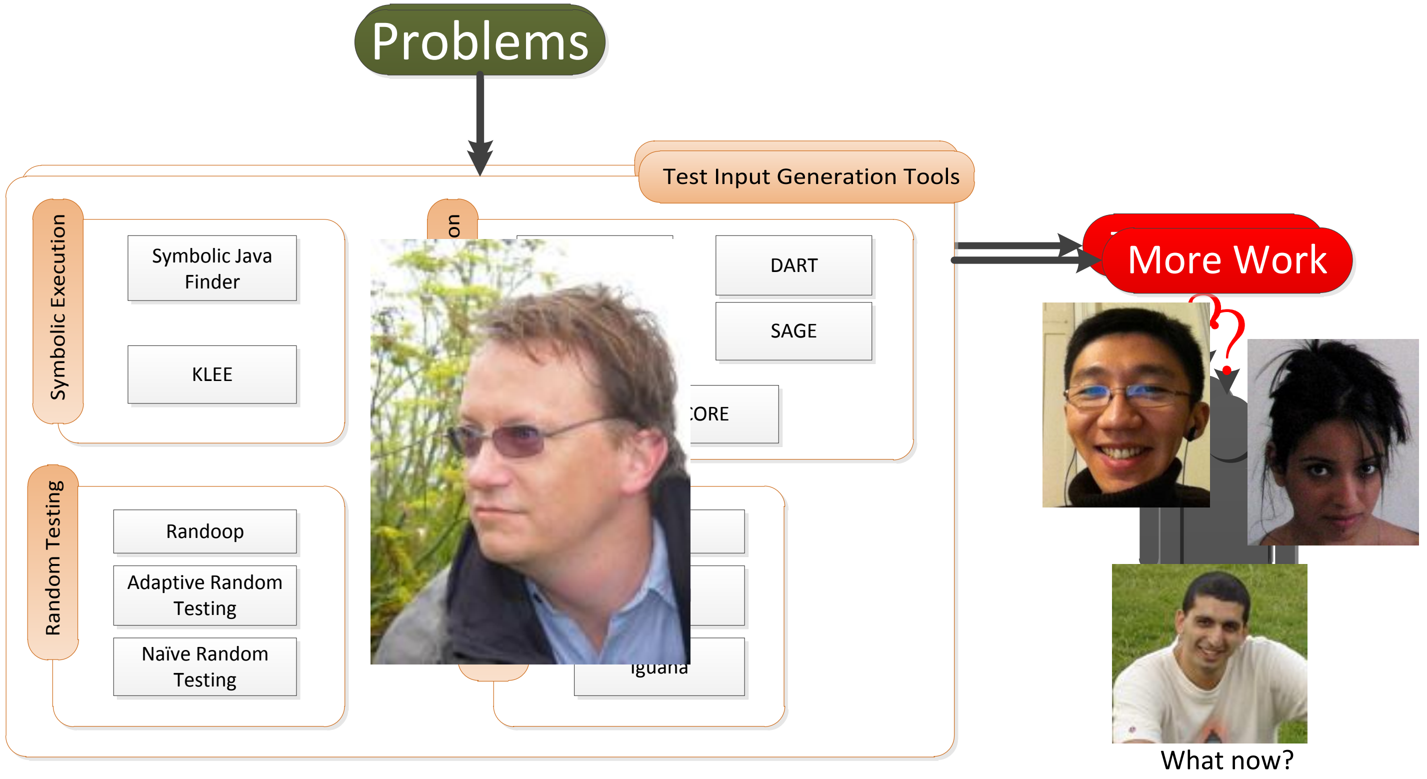
The Testing Problem



The Automated Testing Problem



The Automated Testing Problem



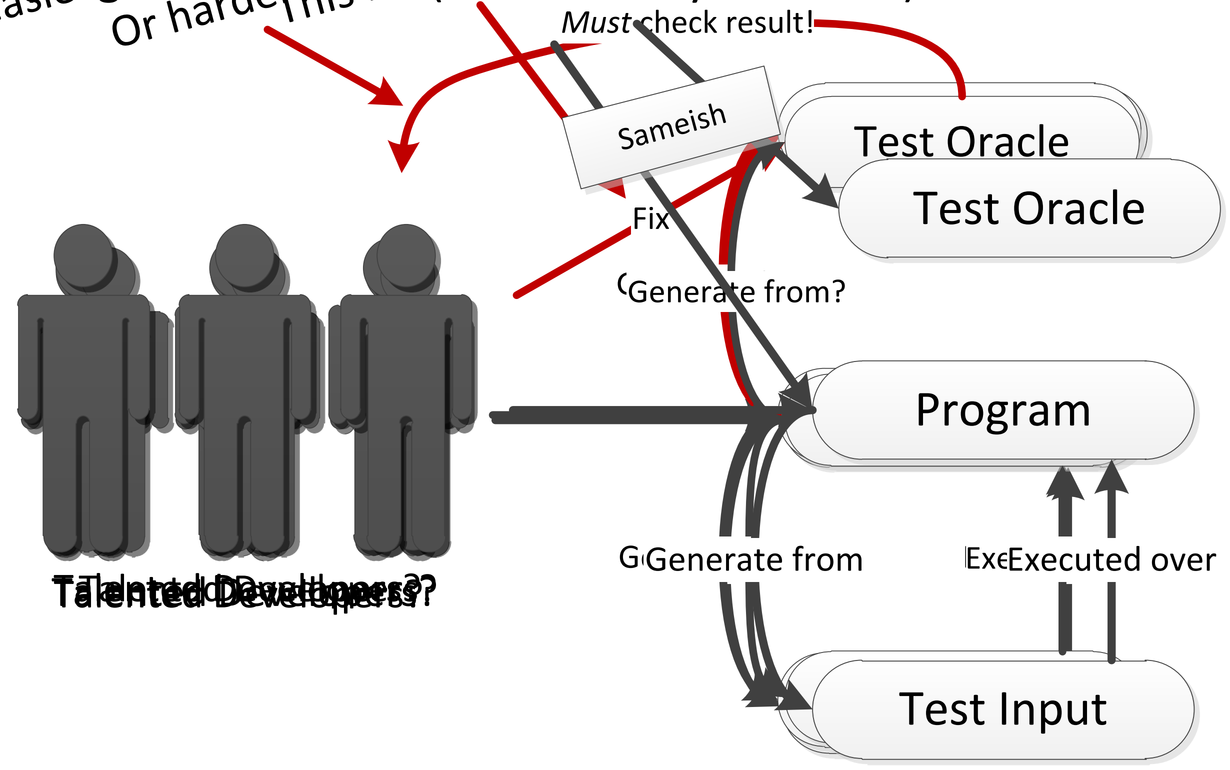
- Huge numbers of tools for this!

The Automated Testing Problem

Goal: get developer to state what the program *should* do (but correctly this time)

Different representation, easier to find mistakes? Or harder?

Differential: proto This is kinda weird.

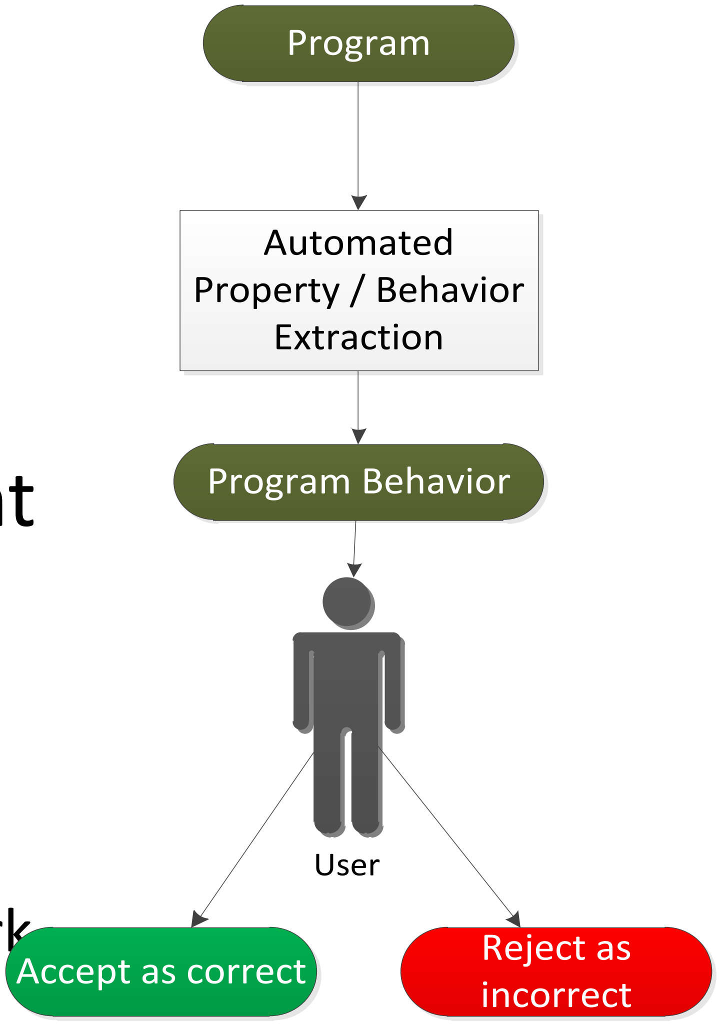


Two Big Research Questions

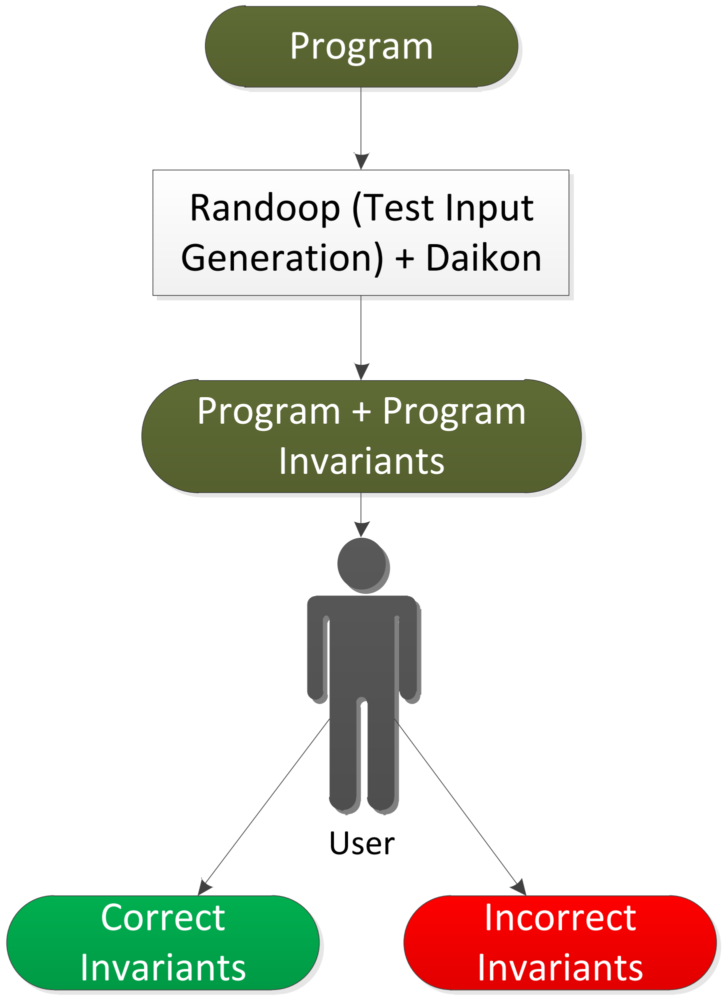
- What types of test oracles can developers easily/correctly understand and build?
  - What tasks are people actually good at?
  - *What should be trying to deliver?*
- How can we help construct such test oracles?
  - Techniques, algorithms, tools, etc.
  - Empirical studies (with users, necessarily)
  - Industrial case studies

Automatic Invariant Generation

- Idea: automatically generate invariants from the system
- User then (necessarily) evaluates result
- Remaining invariants represent test oracle
- Several approaches, varying result
    - *Daikon*, *AutoInfer*, Xie/Notkin work



Automatic Invariant Generation



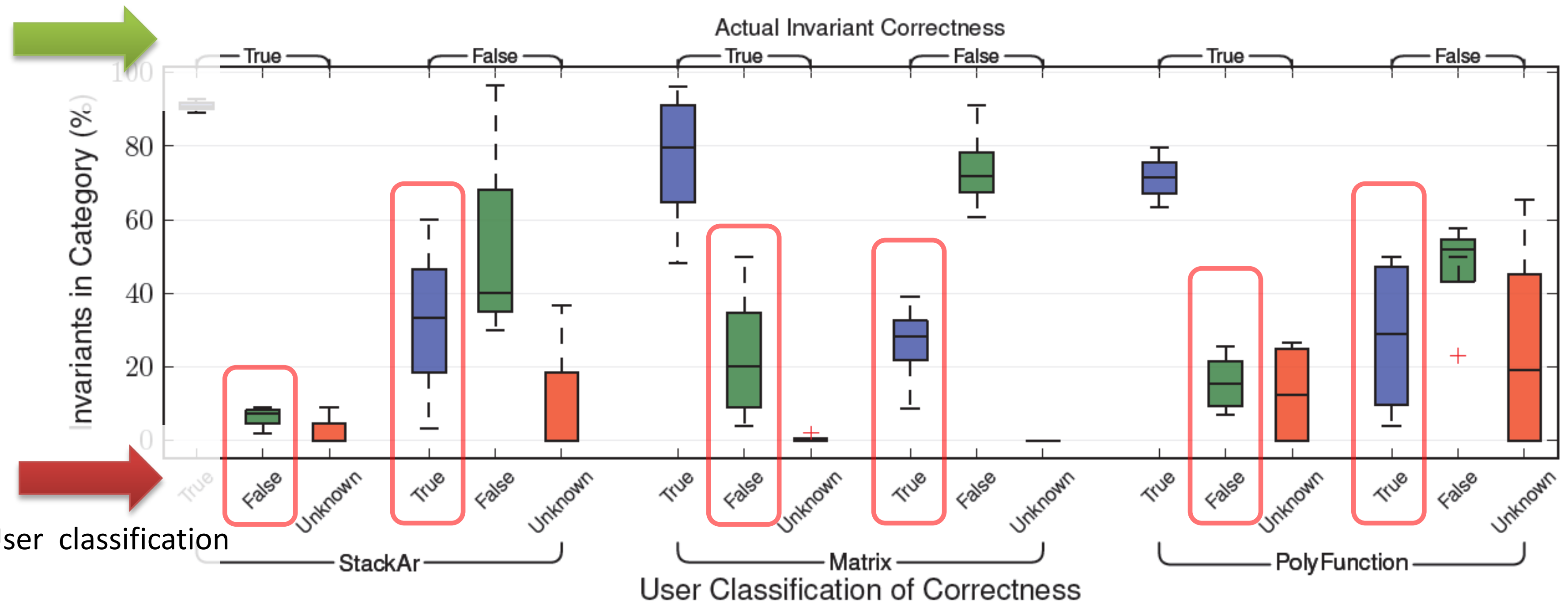
Program
Randoop (Test Input Generation) + Daikon
Program + Program Invariants
User
Correct Invariants
Incorrect Invariants

- Unclear how effective users are at classifying results
  - Problems if poor
  - Little evidence in favor of use
- Study: Daikon dynamic invariant generator
  - 2 case studies, approx. 30 students total
  - 3 programs
  - Each student classified an invariant as true, false, or unknown (unclassified)

Automatic Invariant Generation

KAIST User Study Classification Results

Our classification



User classification

Automatic Invariant Generation

- Questions:
  - Why does this occur?
  - Impact of this on actual testing process?
- Answers:
  - Why? Not really sure
  - Impact? No idea at all (but we guess negative)

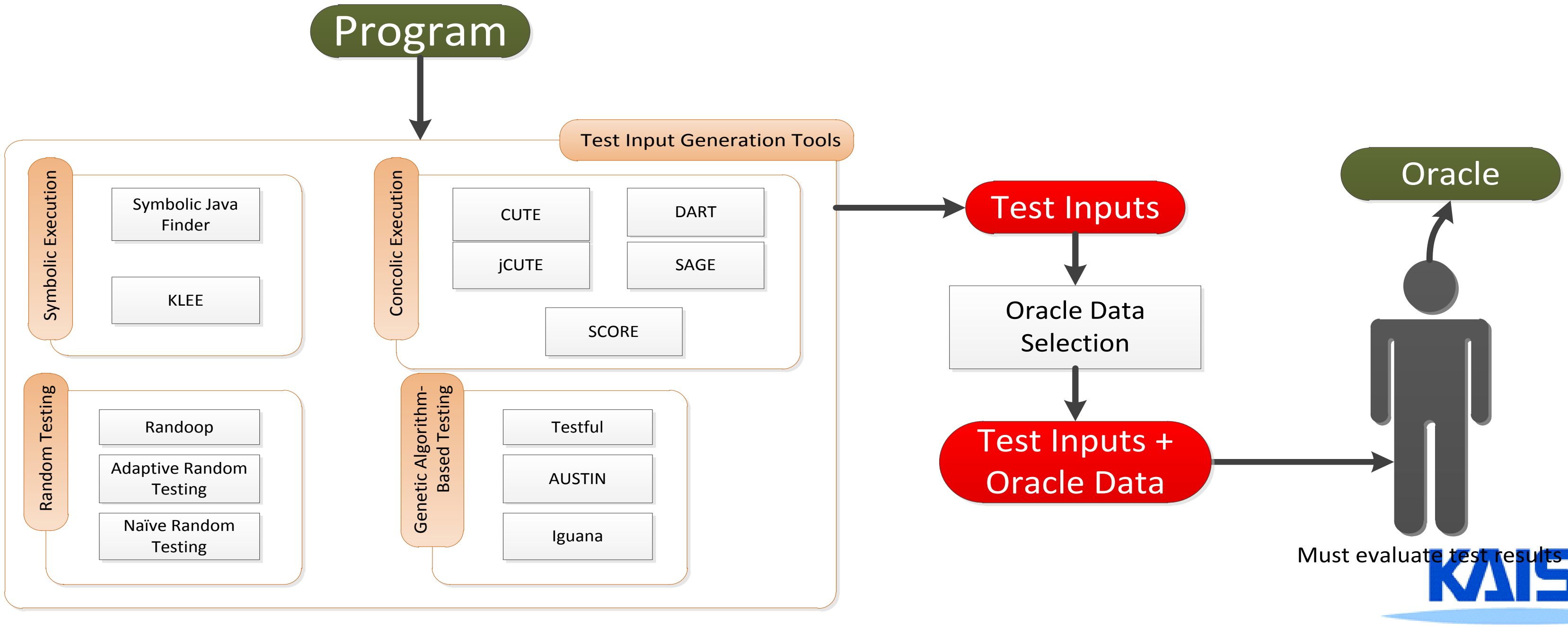
To be presented at **ISSTA 2012**

Coauthors: Shin Hong, Moonzoo Kim, Gregg Rothermel

KAIST

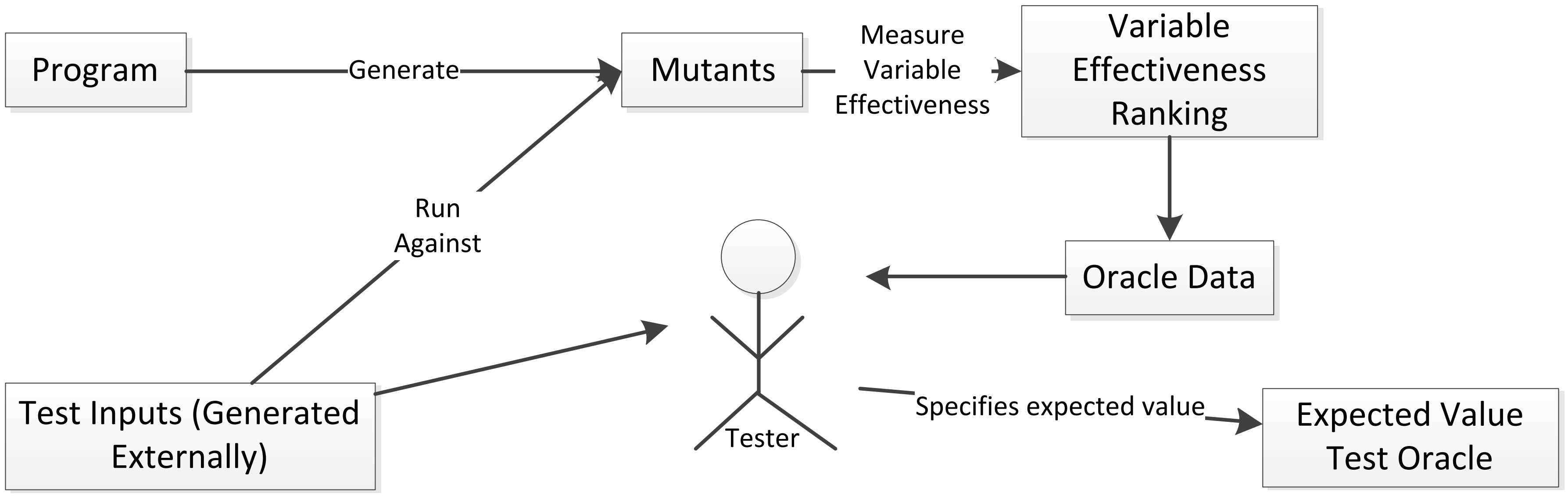
Test Oracle Generation Support

- Uncomfortable with complete automation for oracles
  - Evidence is suspect
  - Requires change in user behavior
- As an alternative to complete construction, we thought we could support users in making oracles
- Select *oracle data*: part of system oracle defined over
- User still has to define oracle

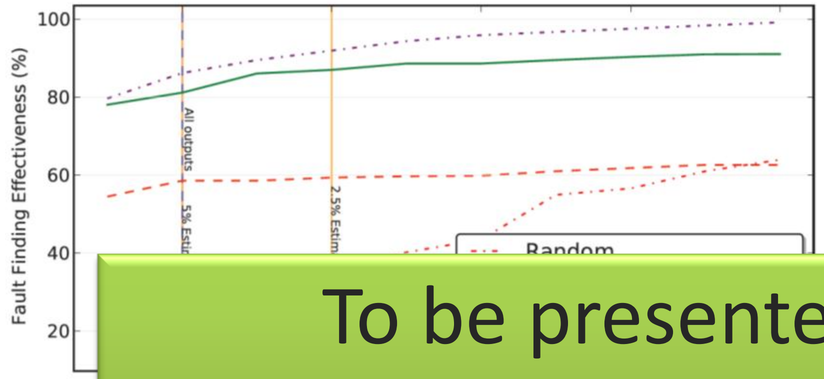


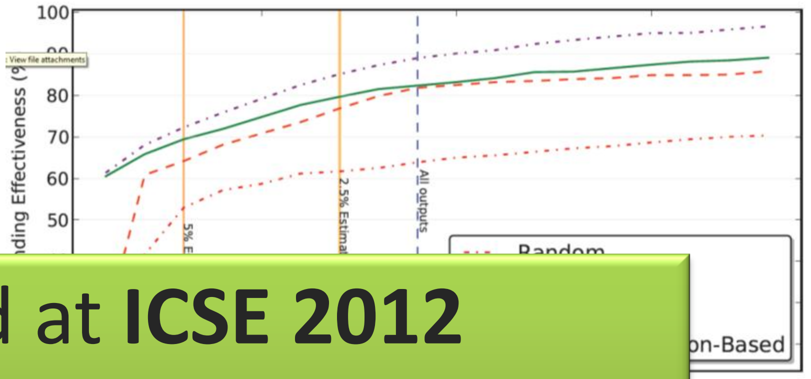
Test Oracle Generation Support

- Mutation testing was used to determine where and when we can detect changes
- Result is that for a set of test inputs, person has a list of useful variables
  - Note: domain is critical avionics, so problems of heap, etc. go away
- Goal: do better than other methods of selecting oracle data


Program
Generate
Mutants
Measure Variable Effectiveness
Variable Effectiveness Ranking
Run Against
Oracle Data
Tester
Test Inputs (Generated Externally)
Specifies expected value
Expected Value Test Oracle

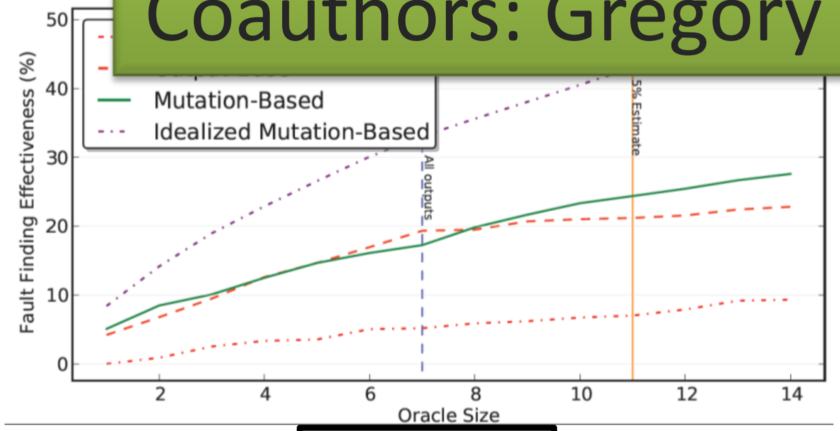
Test Oracle Generation Support



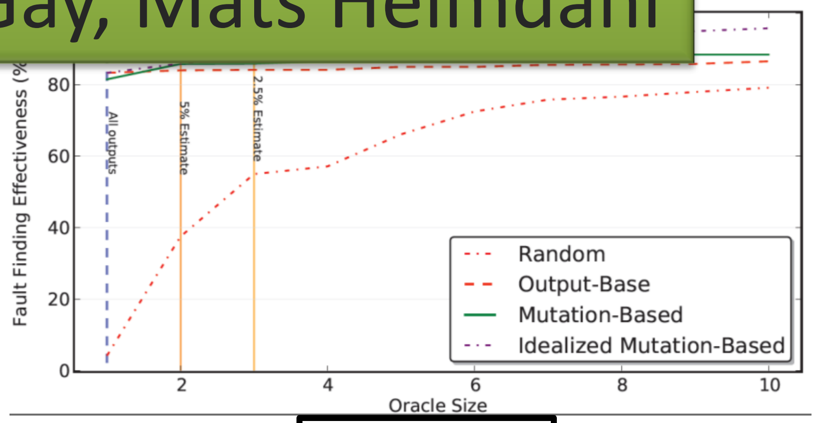


To be presented at **ICSE 2012**

Coauthors: Gregory Gay, Mats Heimdahl



DWM_2



DWM_1

Questions

Future Work

Program: Gatekeeper
Input: “Keymaster”

Automatically extract test oracle

Expected Value: “Zuel”