Predicting Effectiveness of Automatic Testing Tools

http://mir.cs.uiuc.edu/predictcoverage/

Brett Daniel
University of Illinois at Urbana-Champaign
bdaniel3@illinois.edu

Marat Boshernitsan
Coverity, Inc.
maratb@acm.org

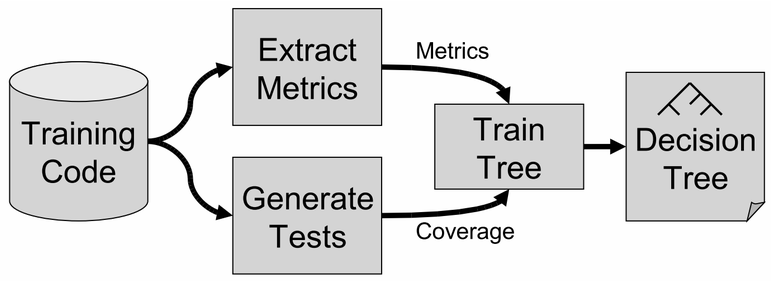
Problem

The structure of an input program often impacts the effectiveness of an automatic testing tool in ways tool users and designers may not expect or understand.

"Why did this testing tool get low coverage on my code?"

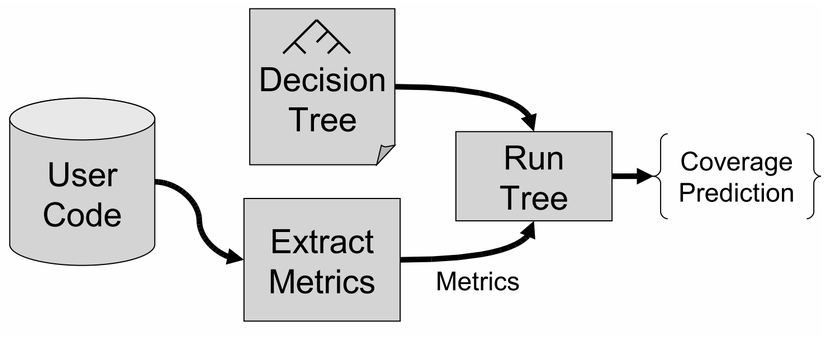
Technique

1 **Tool designers train a decision tree to predict high or low coverage using structural metrics extracted from corpus of software with known coverage.**



Training Process

2 **Tool users predict coverage by "executing" the decision tree on their code, testing against the same set of metrics as in the training code.**

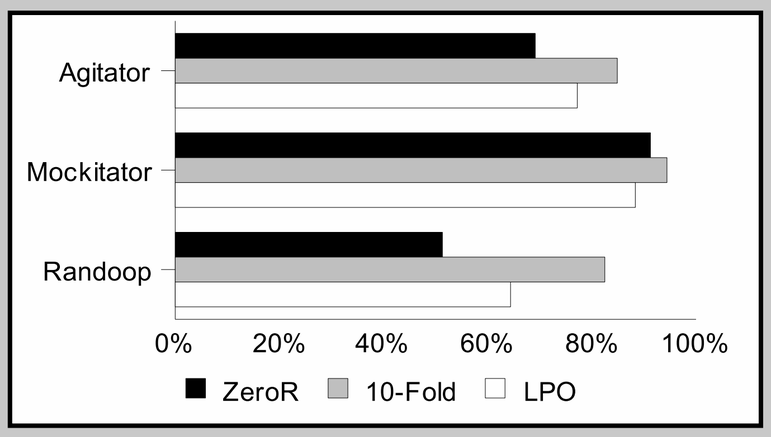


Decision Tree Usage

Results

- **Trained decision trees on 21,000 methods in 11 open-source Java projects**
- **Extracted 49 metrics derived from a method's signature, body, or containing class**
- **Generated tests with three tools: Agitator and Mockitator from Agitar Software, Inc. and Randoop from Pacheco et al. at MIT**
- **Two measures of success: 10-fold and leave-project-out cross-validation, both compared against the majority class (ZeroR)**

*Decision tree classifiers can predict high or low coverage with success rates of **82% to 94%***



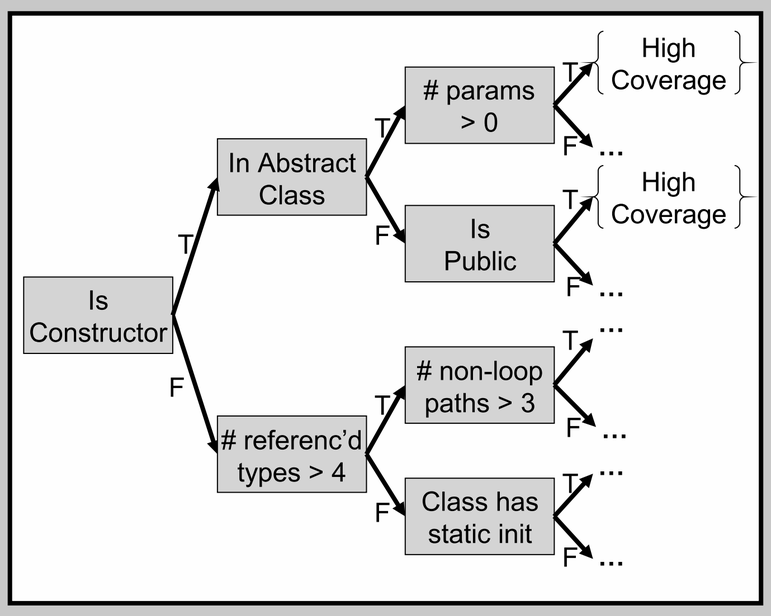
Prediction Success Rates

Future Work

Prediction — **Correlate program structure with tool effectiveness**

Explanation — **Show specific structures that impact tool effectiveness**

Improvement — **Change code to improve effectiveness**



Example Decision Tree