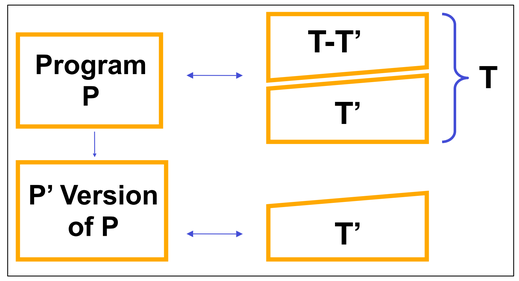
DejaVOO: A Regression Testing Tool for Java Software*

Alessandro Orso, Donglin Liang, Mary Jean Harrold and Manas Tungare — Georgia Institute of Technology
http://www.cc.gatech.edu/aristotle/

1. Regression Testing

- Performed on new versions of the software to provide confidence that:
  - the changed parts behave as intended
  - the unchanged parts are not adversely affected by the modifications
- Can be very expensive, especially in some contexts
  - daily re-testing after nightly builds
  - frequent re-testing during development
  - for the extreme programming paradigm

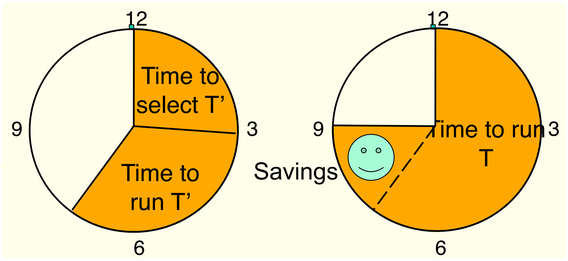
2. Regression Test Selection



A safe algorithm
- requires certain assumptions for safety
- ensures that T' contains all test cases in T that can expose a fault in P'
  - same execution environment
  - deterministic test runs
  - …

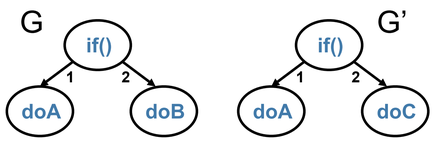
3. Potential Savings

Time to select T' + Time to run T'
< Time to run T



4. Our Safe Regression Test Selection Algorithm: Overview

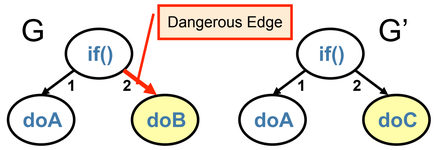
a. Construct graph representations for P and P'



b. Associate test cases in T with entities in P

TC / En	tc1	tc2	tc3
en1	X		
en2		X	X

c. Differentiate G, G' to find dangerous entities



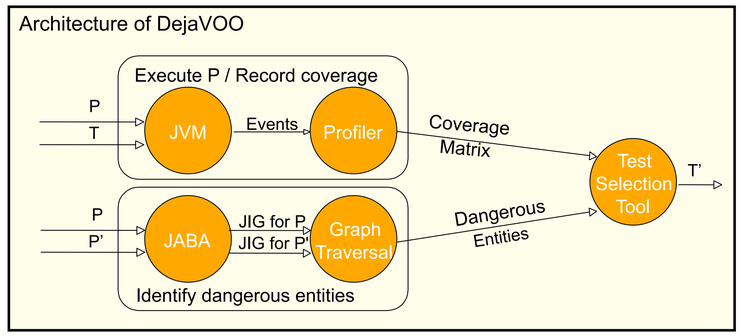
d. Select test cases based on dangerous entities

TC / En	tc1	tc2	tc3
en1	X		
en2		X	X

5. Our Safe Regression Test Selection Algorithm: Characteristics

- It handles Java object-oriented features resulting in simple components and complex interactions
  - inheritance
  - polymorphism and dynamic binding
  - exception handling
- It handles incomplete programs that play an important role in object-oriented software
  - components
  - applications using libraries (e.g., AWT)
  - subsystems

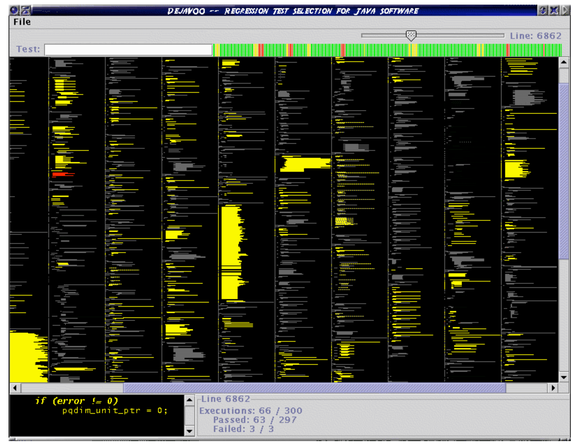
6. DejaVOO: An Implementation of our Regression Test Selection Algorithm for Java Software



JVM: Java Virtual Machine
Profiler: JVM Profiling Interface
JABA: Java Architecture for Bytecode Analysis

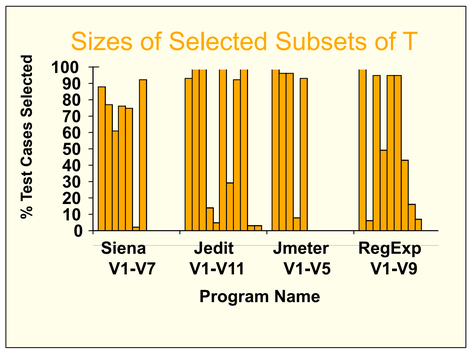
Prototype GUI for DejaVOO

- It shows a SeeSoft view of the source code of P
- It highlights the lines of code that have been modified
- It displays the test cases to be rerun
- It lets the user selectively shows which test cases traverse which parts of the code and vice-versa



7. Empirical Studies: Subjects

- **Siena**: Event notification system (185 methods), 7 versions, 138 test cases, 70% method coverage
- **Jedit**: Text editor (3495 methods), 11 versions, 189 test cases, 75% method coverage
- **Jmeter**: Web-application test tool (109 methods), 5 versions, 50 test cases, 67% method coverage
- **RegExp**: Regular-expression library (168 methods), 10 versions, 66 test cases, 46% method coverage



* This work was supported in part by a grant from Boeing Aerospace Corporation to Georgia Tech, by National Science Foundation award CCR-9707792 to Ohio State University, and awards CCR-9988294, CCR-0096321, and EIA-0196145 to Georgia Tech, and by the State of Georgia to Georgia Tech under the Yamacraw Mission.