Impact Analysis in the Presence of Dependence Clusters Using Static Execute After in WebKit

Lajos Schrettner

with J. Jász, T. Gergely, Á. Beszédes, T. Gyimóthy

SCAM, Riva del Garda, Italy, September 2012
Overview

- Motivation
- Static Execute After (SEA)
  - Definition
  - Impact analysis in WebKit
- SEA-based dependence clusters
  - Dependence clusters in WebKit
  - Application to regression test selection
- Summary and future plans
Motivation

- Change propagation in large systems
  - Indirect effects of changes are often neglected
  - Slicing is too expensive
  - Static Execute After may be applicable

WebKit

- is a large, actively developed system
- ~90000 procedures, frequent commits
- ~30000 regression test cases
Static Execute After (SEA)

- **Definition**
  - Relation on procedures of a program
  - Control flow based
  - \( P \rightarrow Q \) iff part of \( P \) may be executed before a part of \( Q \) is executed

- **Compared to slicing**
  - Faster to compute
  - Slightly less precise
  - May be suitable for large systems
Impact analysis in WebKit

- Test case(i)
  - revision m
  - Passed

- Test case(j)
  - Failed

- Test case(k)
  - ... Failed ...
  - Failed
  - Passed

- Changed procedures
- Impact set

- Changed procedures

- ?

2012.09.23
SCAM 2012
SEA prediction capability
SEA dependence clusters in WebKit

- Monotone Size Graph (MSG*) applied to SEA

*Binkley, Harman: Locating Dependence Clusters and Dependence Pollution, ICSM 2005
Regression test selection in WebKit

- Large regression test suite (~30000 test cases)
  - Full execution is expensive
  - Test prioritization using test case coverage frequencies
  - Test selection based on assigned priorities

- Cluster information can be used to improve inclusiveness (failed test cases found)
Summary and future plans

- Information about dependence clusters helped
  - in assessing gains from impact analysis
  - in improving test selection methods (WebKit EWS)

- What can we do about large clusters?
  - Eliminate
    - Detection → removal
  - Avoid
    - Tool support to give early warning to programmers
    - Can we advise best practices/design patterns?
    - What language features should be changed (eliminated)?