Making Offline Analyses Continuous

Kıvanç Muşlu, Yuriy Brun, Michael D. Ernst, David Notkin

♣ University of Washington
❄ University of Massachusetts, Amherst
Compilation: Continuous vs. Offline Analysis

Continuous Analysis
• Invoked without developer interaction
• Updates the result as input program changes

Offline analysis: invoked by the developer manually
Continuous Analysis Feedback is Good

- Manual invocation interrupts development
- Research: continuous feedback is useful
  - Continuous testing reduces development time 15%
    [SaffE 2003]
Goal: Let’s build tons of continuous analyses!

Wait! Building continuous analyses is hard
Ways to Build a Continuous Analysis

Re-architect an offline analysis:
• Incrementalization [Eclipse compilation]
  – Extremely complex, not possible for some analyses

Wrap an offline analysis:
• Trigger-based analysis [Metrics, FindBugs, Check-style plug-ins]
  – Analysis must be fast
  – Analysis cannot observe buffer-level edits
• Manually-managed copy codebase (Quick Fix Scout [MusluBHEN 2012], Crystal [BrunHEN 2011])
  – Implementation is complex and difficult
Our approach: Making Offline Analyses Continuous

Wrap an offline analysis into an IDE-integrated continuous analysis easily and efficiently
Outline

• Motivation
• Wrapping offline analyses into continuous
• Evaluation and results
• Contributions
Outline

• Motivation
• Wrapping offline analyses into continuous
• Evaluation and results
• Contributions
Goal 1: Currency

Analysis should have access to most recent code

Most recent analysis results should be accessible to the developer

continuous analysis
offline analysis
Goal 2: **Isolation**

Analysis should run on a consistent codebase

Analysis should not block the developer
Approach: Codebase Replication

Achieves goals of currency and isolation
Codebase Replication: Architecture

![Diagram showing the architecture of codebase replication.]

- IDE
  - IDE API
    - developer's editor
    - developer edits
- Codebase Replication
  - developer edits
Codebase Replication: Architecture

- IDE
- IDE API
- developer's editor
- Codebase Replication
- copy codebase
- developer edits
Codebase Replication: Architecture

- **Developer’s Editor**: For initial code editing.
- **IDE API**: Interacts with the IDE for editing.
- **Codebase Replication**: Performs continuous and offline analysis of codebase.
- **Copy Codebase**: Used to make copies of the codebase.
- **Run Offline Analysis**: Activates offline analysis process.

Connections:
- Developer edits flow through IDE API to Codebase Replication.
- Codebase Replication sends continuous and offline analysis signals.
- Offline analysis leads back to copy codebase.
Codebase Replication: Architecture

- analysis
- output
- currency
- developer
- isolation
- analysis
- input
- currency

IDE API

developer’s editor

developer edits

IDE

developer edits

Codebase Replication

continuous analysis
offline analysis

run offline analysis

display offline
analysis results

copy codebase

analysis isolation

analysis output currency
Codebase Replication: Architecture

- Analysis output currency
- Developer isolation
- Analysis isolation
- Analysis input currency

Diagram showing the architecture of codebase replication with components such as developer’s editor, IDE API, ID E, and Codebase Replication event queue with continuous analysis offline analysis and developer edits.
Solstice:
Codebase Replication for Eclipse

Eclipse-specific design changes:

• Solstice runs a headless (w/o UI) copy Eclipse
• Copy Eclipse manages the copy workspace
  – One Eclipse is associated with one workspace
• Bidirectional link between two Eclipses
Solstice: Architecture

- Developer’s Eclipse
  - IDE API
  - DEDE
  - display offline analysis results
- Copy (headless) Eclipse
  - IDE
  - developer edits
  - copy codebase
  - run offline analysis
  - continuous analysis
  - pause(...) resume(...)
Outline

• Motivation
• Wrapping offline analyses into continuous
• Evaluation and results
• Contributions
Research Questions

Quantitative evaluation: Currency and isolation
• How fast does the analysis get access to changed code?
• How fast does the developer see new analysis results?
• Does the developer notice any IDE slowdown?
• How much is the analysis delayed?

Case study
• How hard is it to implement Solstice analysis wrappers?
• Are Solstice analysis wrappers useful?
  - Preliminary result: yes. Refer to the paper for details.
Efficient Currency & Isolation

Analysis Results Delay

3ms

display offline analysis results

Overhead on the Developer Edits: 2.5ms
File ops: 1.5ms

Penalty on Analysis
70ms

Analysis Input Delay
2.5ms
It is Easy to Implement an Analysis Wrapper

3 analysis wrappers:

- FindBugs
- PMD
- Testing

On average 800 LoC (500 LoC without UI), 18 hours.

Compare to:

- Eclipse FindBugs plug-in: 16 KLoC
- Quick Fix Scout: 7.4 KLoC
- Eclipse continuous testing plug-in: 3.5 KLoC
Example Analysis Wrapper Implementation

```java
@SolsticeServerPreferences
public class ContinuousFindBugsServer extends SolsticeServerNodeWithLogger {
    public ContinuousFindBugsServer() {
        super(true);
    }
    private String processAnalysisCompleted(FindbugsAnalysisCompletedMessage message) {
        // Pretty print results
    }
}

public class ContinuousFindBugs extends SCPurePreciseAnalysis {
    private volatile String findBugsExecutablePath_ = "";
    public ContinuousFindBugs() {
        super(AnalysisGranularity.PROJECT);
    }
    protected void preferenceChanged(String preferenceID, String preferenceValue) {
        if (preferenceID.equals(SharedOperations.EXECUTABLE_PATH_PREFERENCE_ID))
            findBugsExecutablePath_ = preferenceValue;
            resumeAnalysis();
    }
    protected @Nullable ClientAnalysisFailedMessage shallRunAnalysis() {
        return projectContainsCompilationErrors(getCurrentProject());
    }
    protected ClientAnalysisMessage runAnalysis() {
        // Run FindBugs & return results.
    }
```
Contributions

- Codebase Replication
  - New approach to implement continuous analyses
  - Analyses get currency and isolation

- Solstice
  - Evaluation: fast and responsive
  - Implement continuous analyses quickly and easily

http://bitbucket.org/kivancmuslu/solstice