UMassAmherst

Making Offline Analyses Continuous

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

slide author names omitted for FERPA compliance

Outline

- Definitions
- Challenges
- Key Idea
- Research Questions
- Discussion

Snapshot

- Developer's point of view
- Program at certain point in time

Snapshot 1

</> enter your source code or insert template or sample or your template

```
/* package whatever; // don't place package name! */

import java.util.*;
import java.lang.*;
import java.io.*;

/* Name of the class has to be "Main" only if the class is public. */
class Hello
{
    public static void main (String[] args) throws java.lang.Exception
    {
        System.out.println("Hello World");
    }
}
```

Snapshot 2

</> enter your source code or insert template or sample or your template

```
1  /* package whatever; // don't place package name! */
2
3  import java.util.*;
4  import java.lang.*;
5  import java.io.*;
6
7  /* Name of the class has to be "Main" only if the class is public. */
8  class Hello
9  {
10     public static void main (String[] args) throws java.lang.Exception
11     {
12         System.out.println("Hello CS521/621 Students!");
13     }
14 }
```

Analysis

- Runs on a snapshot
- Gives feedback
- 2 types:
  - Offline
  - Continuous

Offline Analysis

- No human input after execution
- FindBugs
  - Eclipse plug-in
  - Static code analyzer
  - Helps detect bugs
  - Gives feedback to snapshot

FindBugs

FindbugsExample.java

```
package com.infosupport.peterhe;

import javax.annotation.Nonnull;

public class FindbugsExample {

    public static String sayHello(@Nonnull String message) {
        return "Hello " + message;
    }

    public static void main(String[] args) {
        System.out.println(sayHello(null));
    }
```

H C NP: Method call in com.infosupport.peterhe.FindbugsExample.main(String[]) passes null to a nonnull parameter of sayHello(String)

Continuous Analysis

- Runs constantly and informs developers with up-to-date feedback
- Continuous FindBugs
  - Faster results
  - Constant
  - Not as distracting to developer

Continuous FindBugs

VectorClock.java

```
@Override
public boolean equals(Object object) {
    if(this == object)
        return true;
    if(object == null)
        return false;
    if(!object.getClass().equals(VectorClock.class))
        return false;
    VectorClock clock = (VectorClock) object;
    return versions.equals(clock.versions);
}
```

FindBugs Results

```
Findbugs standard output:
M B Eq: voldemort.versioning.VectorClock.equals(Object) fails for subtypes  At Vec
M B Eq: voldemort.store.socket.SocketDestination.equals(Object) fails for subtypes
M B Eq: voldemort.serialization.SerializerDefinition.equals(Object) fails for subtypes
M B Eq: voldemort.store.StoreDefinition.equals(Object) fails for subtypes  At StoreD
```

VectorClock.java

```
@Override
public boolean equals(Object object) {
    if(this == object)
        return true;
    if(object == null)
        return false;
    if(!(object instanceof VectorClock))
        return false;
    VectorClock clock = (VectorClock) object;
    return versions.equals(clock.versions);
}
```

FindBugs Results

```
Findbugs standard output:
M B Eq: voldemort.store.socket.SocketDestination.equals(Object) fails for subtypes  At SocketDestination.j
M B Eq: voldemort.serialization.SerializerDefinition.equals(Object) fails for subtypes  At SerializerDefinition.
M B Eq: voldemort.store.StoreDefinition.equals(Object) fails for subtypes  At StoreDefinition.java:[line 355]
M B Eq: voldemort.store.slop.Slop.equals(Object) fails for subtypes  At Slop.java:[line 131]
```

Continuous vs Offline

- Runs constantly
- Runs in background
- Does not delay or block developer code
- Simplifies developer's workflow

- Require more work from developer
- Interferes with workflow
- Delay or block developer code

Challenges

- *Isolation*
  - should not prevent developer from making new changes
  - should not alter code while developer is working on it
- *Currency*
  - when analysis is optionally restarted, old results marked “stale”
  - should make results available as soon as analysis completes

Key Idea

- **Codebase Replication**
  - A novel approach
  - Turns offline into continuous
  - Incorporates 4 principles

Research Question 1

How does Codebase Replication solve the challenges of isolation and currency?

Solve the Challenges

Codebase Replication has 4 principles:

- Replication
- Buffer-level Synchronization
- Exclusive Ownership
- Invalidation Detection

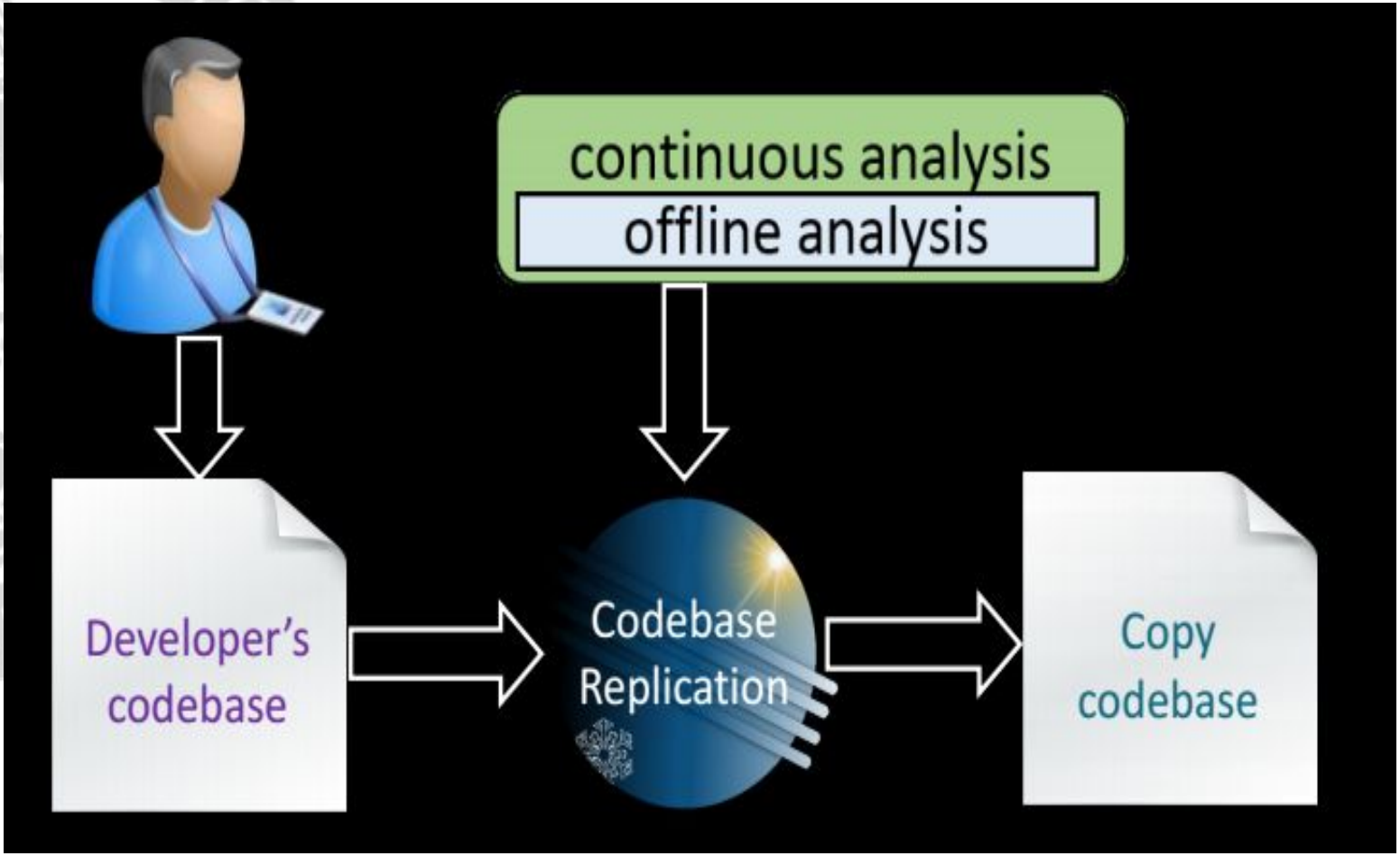
Overcoming Isolation

- Replication - copy of code
- Buffer-level Synchronization - run tool on latest copy of code

Overcoming Currency

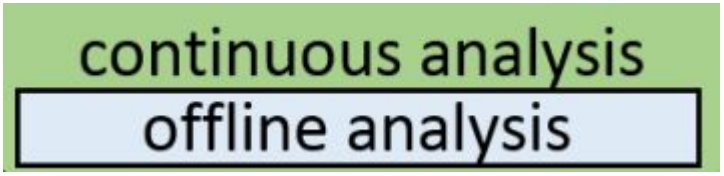
- Exclusive Ownership - request write access to program
- Invalidation Detection - identify stale changes

Codebase Replication



Solstice

- An open source implementation of Codebase Replication within Eclipse
- A wrapper to convert offline analyses to continuous
- FindBugs into Continuous FindBugs



Previous Approaches

- Manually managed copy codebase
- Trigger-based analysis
- Re-architect an offline analysis

Research Question 2

How efficient is Codebase Replication compared to re-architecting the offline analyses to work continuously?

- Overhead <= 2.5 ms
- Initial synchronization <= 2.5 ms

Case Study

Study on SolsticeCT - continuous testing plug-in and a buggy program

- Speeds up discovery of unknown bugs
- Makes debugging information available sooner
- Unobtrusive

Research Question 3

- How difficult is it to implement Solstice wrappers?

- Just 800 lines of code!
- Just 18 hours to implement!


easy

Discussion Question 1

How likely would a new programmer use this tool compared to a more experienced one? Why?

Discussion Question 2

Would this scale well to larger and more complex analysis tools?

Discussion Question 3

Would all offline tools benefit from being converted to continuous?

Discussion Question 4

Will this change the way we look at development and analysis tools?

Discussion Question 5

Can this approach work outside of an IDE?

Thank you!

References

http://homes.cs.washington.edu/~mernst/pubs/offline-continuous-esecfse2013-slides.pdf