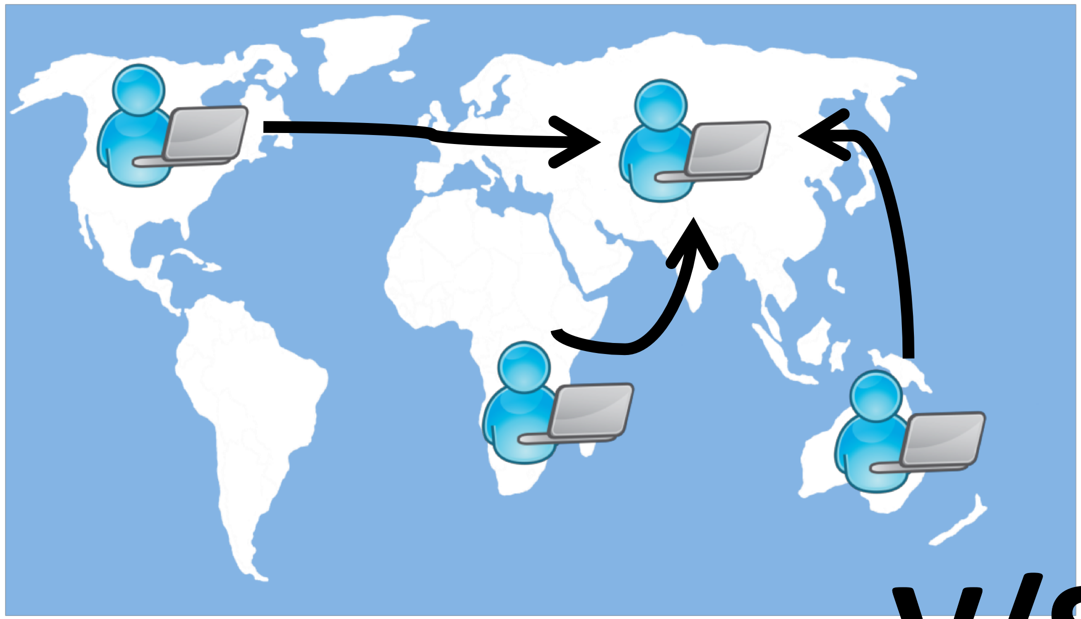
An empirical study on Commits in Collaborative Software Development

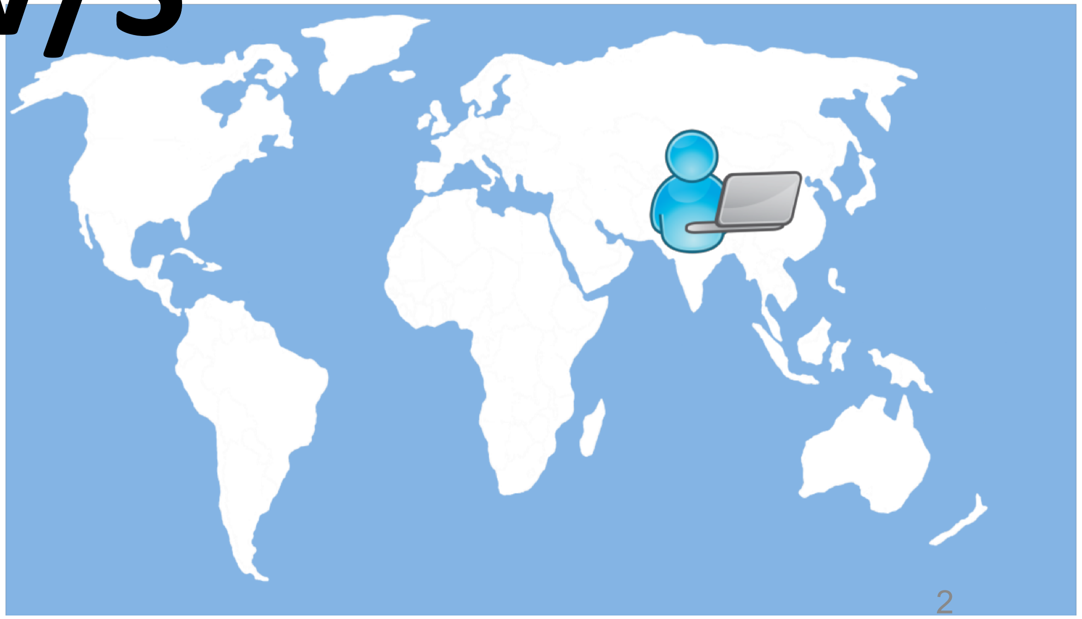
Advisor: Dr. Rahul Purandare
Presented By: Chaitanya Kumar

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

1. Motivation and Problem



V/S

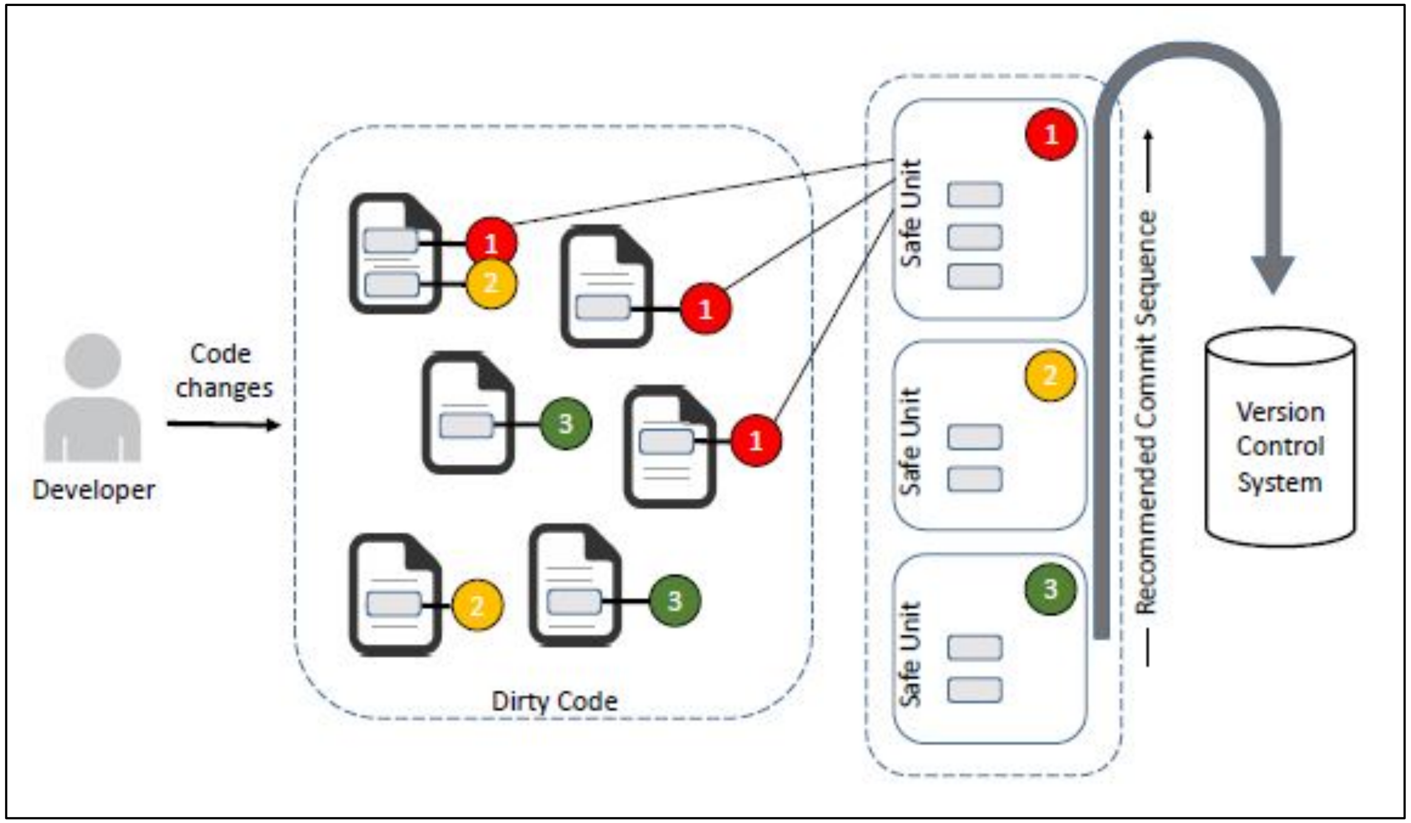


1.1 Questions we begin with

- Can merge conflicts be avoided if we organize check-ins (as an ordered set of safe units) and follow the practice of early commits?

- Given a pair of safe units, how do we assign a measure of potential for merge-conflict?

2. What we propose



Units containing code that can potentially cause conflicts are suggested to be committed early.

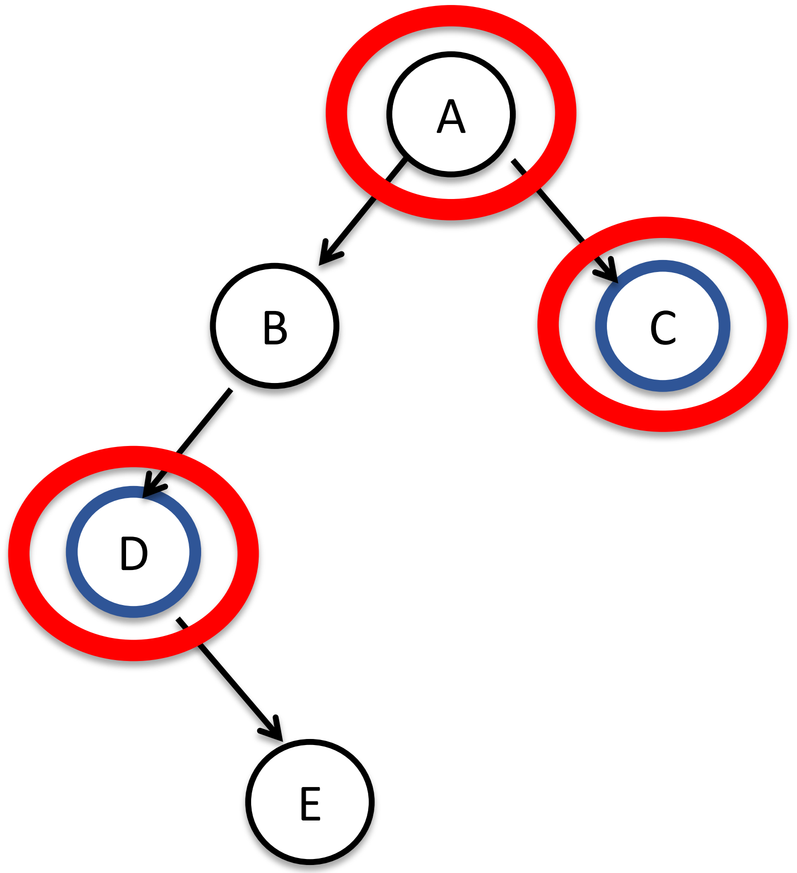
•**Unit:** A unit is a subset of source code that is edited by the developer in his working copy. A Unit cannot be empty. The contents of a Unit are committed together.
•**Safe Unit:** A safe unit is any unit which doesn't cause the build to break, i.e., no test case fails if a safe unit is checked in.

3. The Approach

- Identifying dependency paths
  - Modeled as method invocations

- Observing commit trends in Open Source Software
  - Commit long enough?
  - Why do we do this, at all?

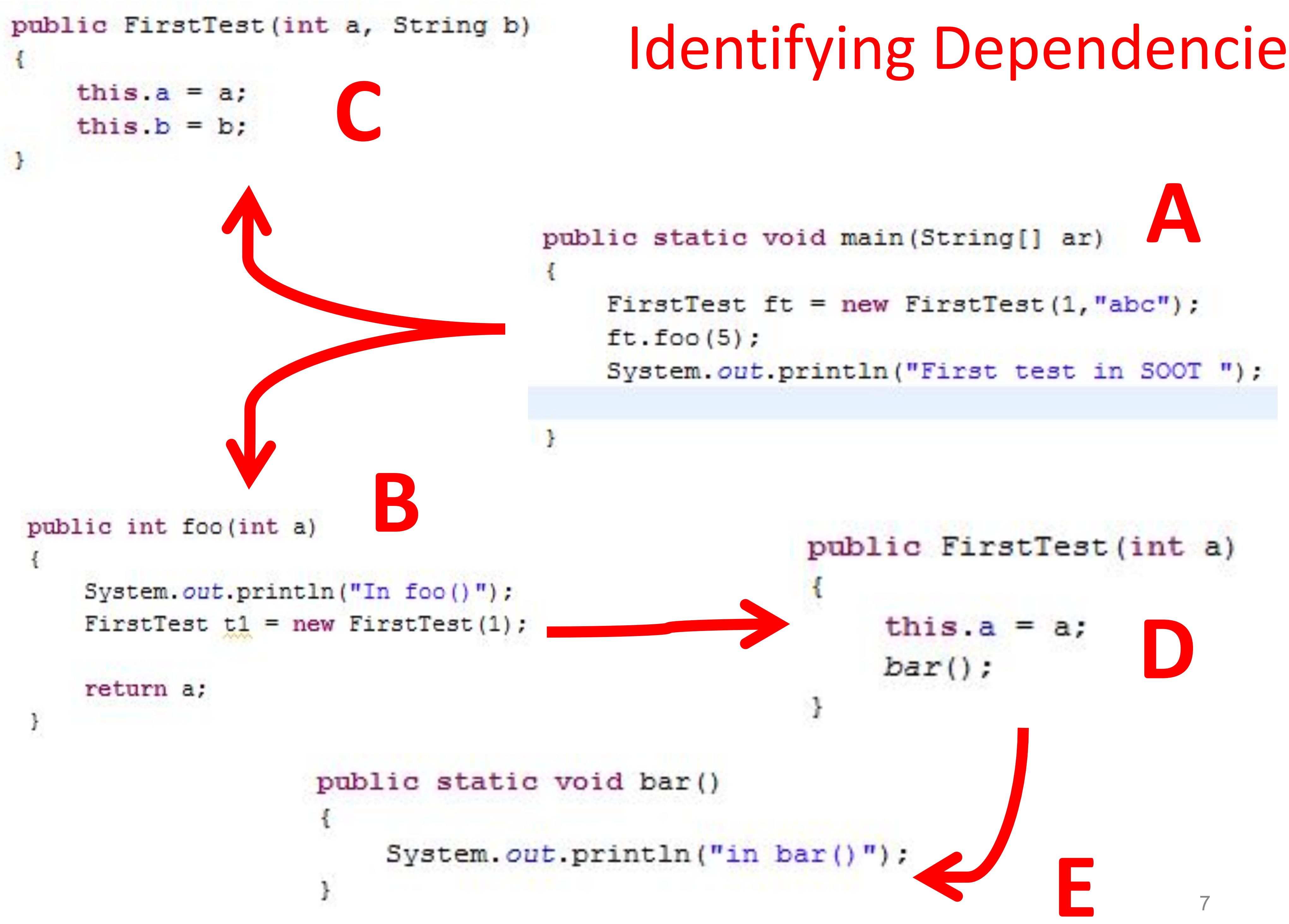
3.1 Defining a Dependency



A, C are an unsafe pair
A, D are an unsafe pair
C, D are a safe pair

➔ **Two developers can be given C and D to work on in parallel, being assured that they won't have conflicts during merge time.**

Identifying Dependencies



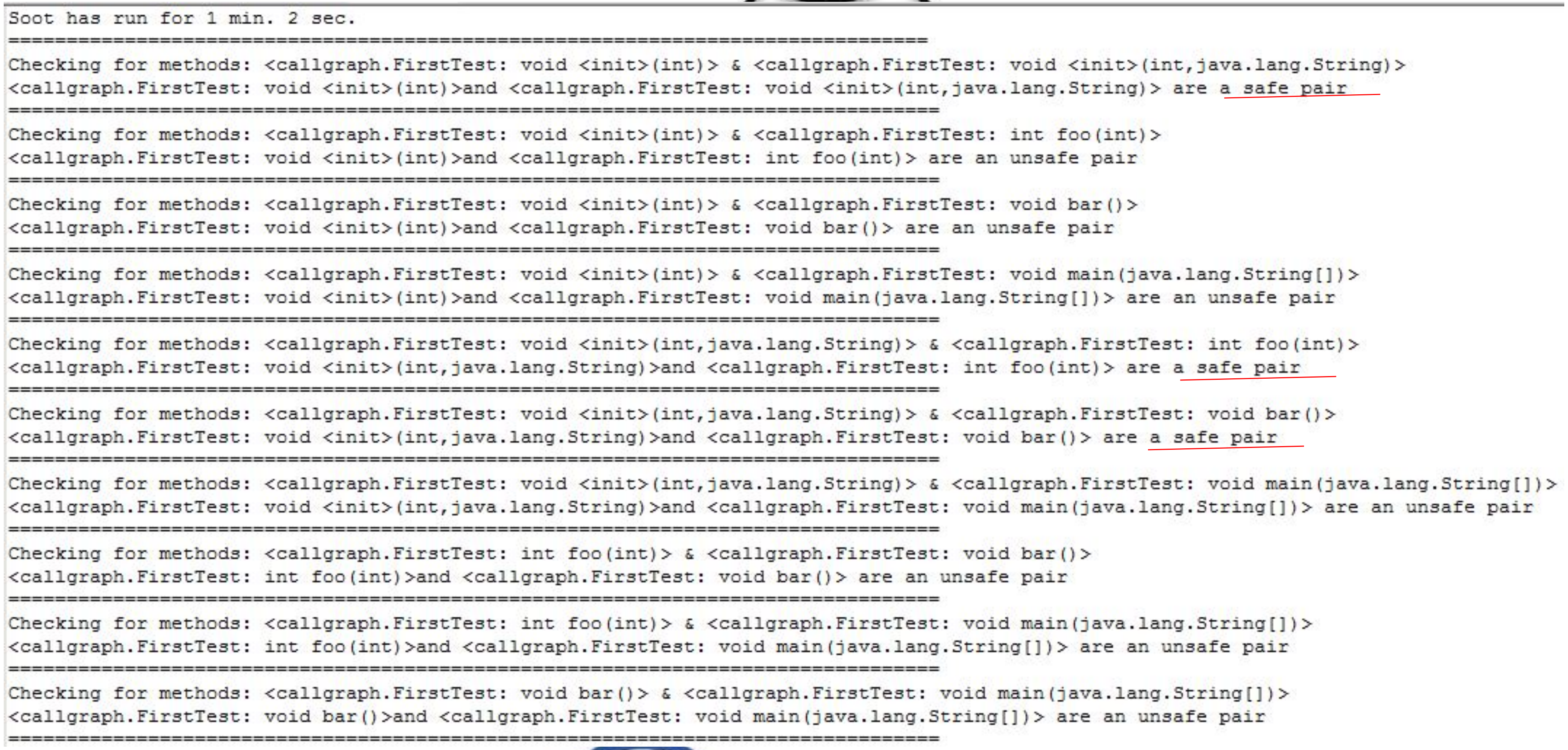
```
public FirstTest(int a, String b)
{
    this.a = a;
    this.b = b;
}
```

```
public static void main(String[] ar)
{
    FirstTest ft = new FirstTest(1,"abc");
    ft.foo(5);
    System.out.println("First test in SOOT ");

}
```

```
public int foo(int a)
{
    System.out.println("In foo()");
    FirstTest t1 = new FirstTest(1);

    return a;
}
```

```
public FirstTest(int a)
{
    this.a = a;
    bar();
}
```

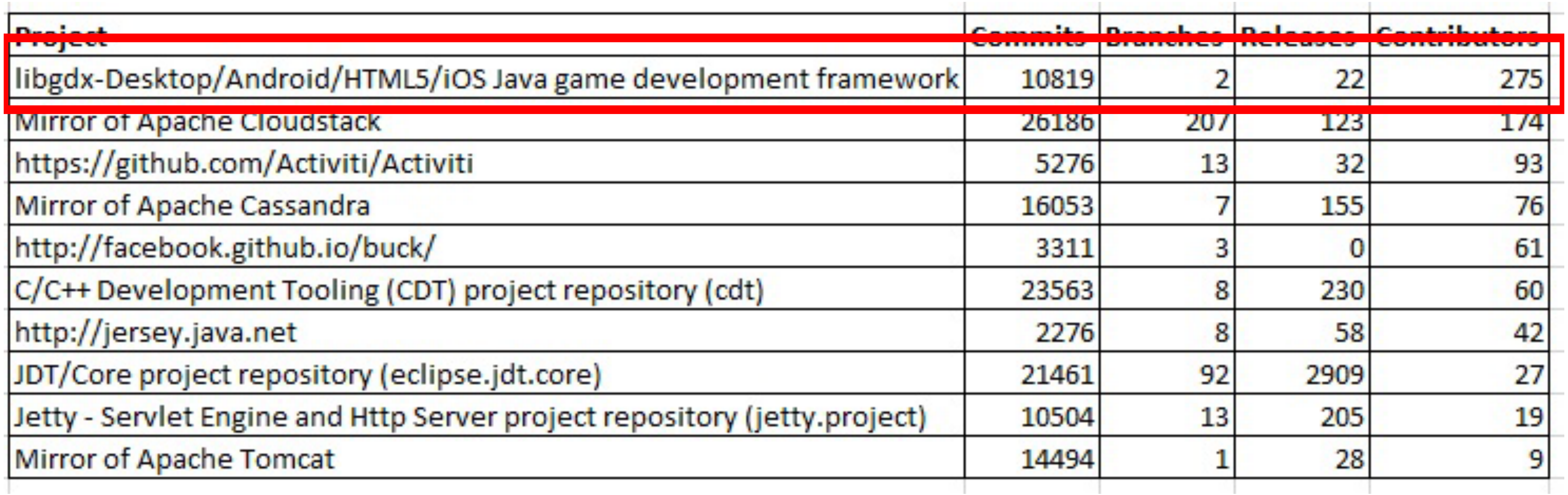
```
public static void bar()
{
    System.out.println("in bar()");
}
```



```
Soot has run for 1 min. 2 sec.
==============================================================================
Checking for methods: <callgraph.FirstTest: void <init>(int)> & <callgraph.FirstTest: void <init>(int,java.lang.String)>
<callgraph.FirstTest: void <init>(int)>and <callgraph.FirstTest: void <init>(int,java.lang.String)> are a safe pair
==============================================================================
Checking for methods: <callgraph.FirstTest: void <init>(int)> & <callgraph.FirstTest: int foo(int)>
<callgraph.FirstTest: void <init>(int)>and <callgraph.FirstTest: int foo(int)> are an unsafe pair
==============================================================================
Checking for methods: <callgraph.FirstTest: void <init>(int)> & <callgraph.FirstTest: void bar()>
<callgraph.FirstTest: void <init>(int)>and <callgraph.FirstTest: void bar()> are an unsafe pair
==============================================================================
Checking for methods: <callgraph.FirstTest: void <init>(int)> & <callgraph.FirstTest: void main(java.lang.String[])>
<callgraph.FirstTest: void <init>(int)>and <callgraph.FirstTest: void main(java.lang.String[])> are an unsafe pair
==============================================================================
Checking for methods: <callgraph.FirstTest: void <init>(int,java.lang.String)> & <callgraph.FirstTest: int foo(int)>
<callgraph.FirstTest: void <init>(int,java.lang.String)>and <callgraph.FirstTest: int foo(int)> are a safe pair
==============================================================================
Checking for methods: <callgraph.FirstTest: void <init>(int,java.lang.String)> & <callgraph.FirstTest: void bar()>
<callgraph.FirstTest: void <init>(int,java.lang.String)>and <callgraph.FirstTest: void bar()> are a safe pair
==============================================================================
Checking for methods: <callgraph.FirstTest: void <init>(int,java.lang.String)> & <callgraph.FirstTest: void main(java.lang.String[])>
<callgraph.FirstTest: void <init>(int,java.lang.String)>and <callgraph.FirstTest: void main(java.lang.String[])> are an unsafe pair
==============================================================================
Checking for methods: <callgraph.FirstTest: int foo(int)> & <callgraph.FirstTest: void bar()>
<callgraph.FirstTest: int foo(int)>and <callgraph.FirstTest: void bar()> are an unsafe pair
==============================================================================
Checking for methods: <callgraph.FirstTest: int foo(int)> & <callgraph.FirstTest: void main(java.lang.String[])>
<callgraph.FirstTest: int foo(int)>and <callgraph.FirstTest: void main(java.lang.String[])> are an unsafe pair
==============================================================================
Checking for methods: <callgraph.FirstTest: void bar()> & <callgraph.FirstTest: void main(java.lang.String[])>
<callgraph.FirstTest: void bar()>and <callgraph.FirstTest: void main(java.lang.String[])> are an unsafe pair
==============================================================================
```

3.2 Commit trends in Open Source s/w

- Choice of projects:
  - Searched for “Java” on Github- 156,931 repos, 100 million+ LoC
  - Factors considered to come up with the list of repos:
    - ✓Commits-to-date
    - ✓Number of branches
    - ✓Number of contributors



Project	Commits	Branches	Releases	Contributors
libgdx-Desktop/Android/HTML5/iOS Java game development framework	10819	2	22	275
Mirror of Apache Cloudstack	26186	207	123	174
https://github.com/Activiti/Activiti	5276	13	32	93
Mirror of Apache Cassandra	16053	7	155	76
http://facebook.github.io/buck/	3311	3	0	61
C/C++ Development Tooling (CDT) project repository (cdt)	23563	8	230	60
http://jersey.java.net	2276	8	58	42
JDT/Core project repository (eclipse.jdt.core)	21461	92	2909	27
Jetty - Servlet Engine and Http Server project repository (jetty.project)	10504	13	205	19
Mirror of Apache Tomcat	14494	1	28	9

3.2 Commit trends in Open Source s/w

- Having zeroed down on the project, walk through the commit history
  - Revisited 10,823 commits
  - 2003 were merges
    - For each of these merges

3.2 Commit trends in Open Source s/w

- Having zeroed down on the project, walk through the commit history
  - Revisited 10,823 commits
  - 2003 were merges
    - For each of these merges, checked:
      - If it resulted in a merge conflict

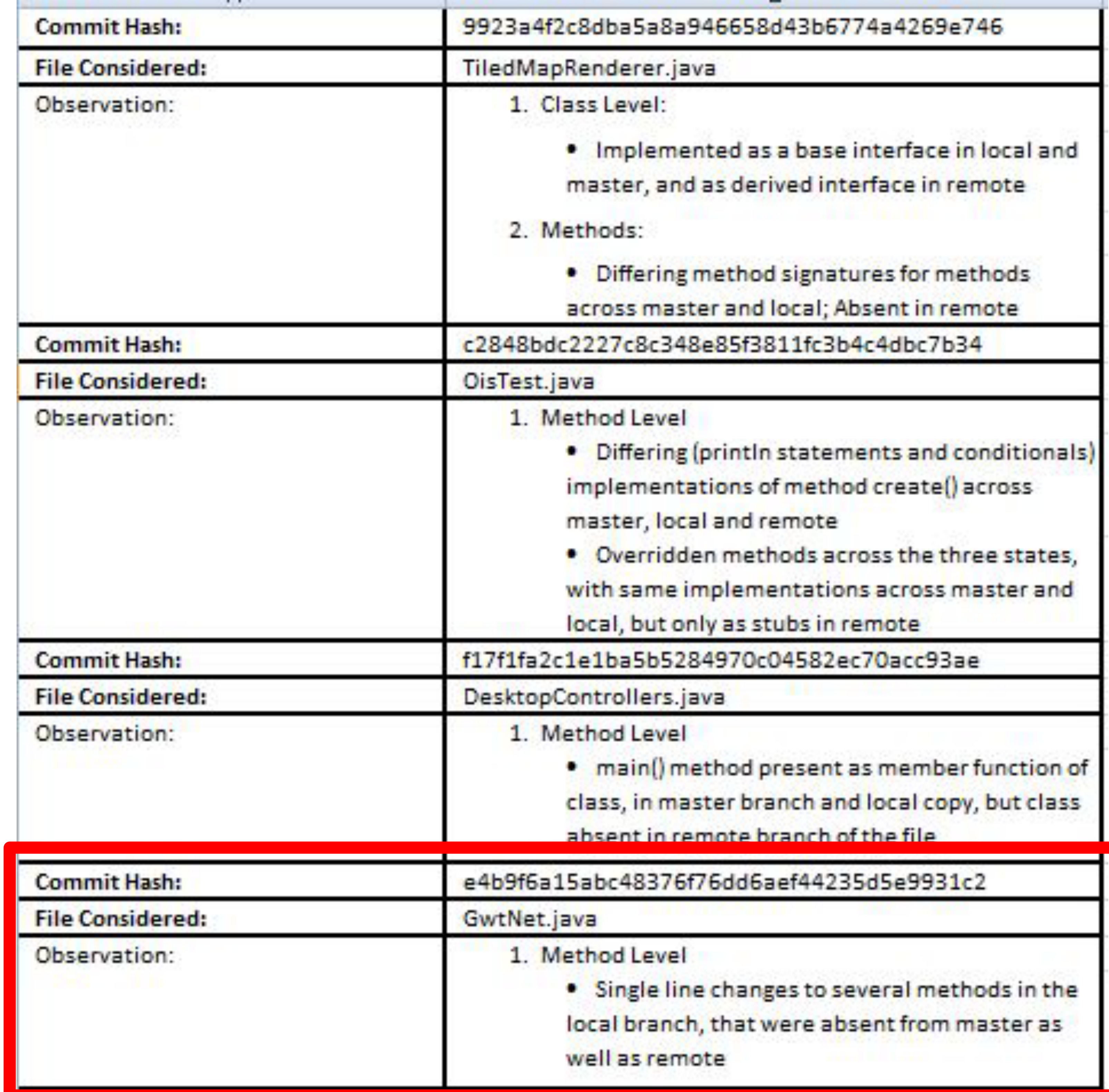
3.2 Commit trends in Open Source s/w

- Having zeroed down on the project, walk through the commit history
  - Revisited 10,823 commits
  - 2003 were merges
    - For each of these merges, checked:
      - If it resulted in a merge conflict
        - If the conflict affected multiple files, methods, etc (Only Java language considered)

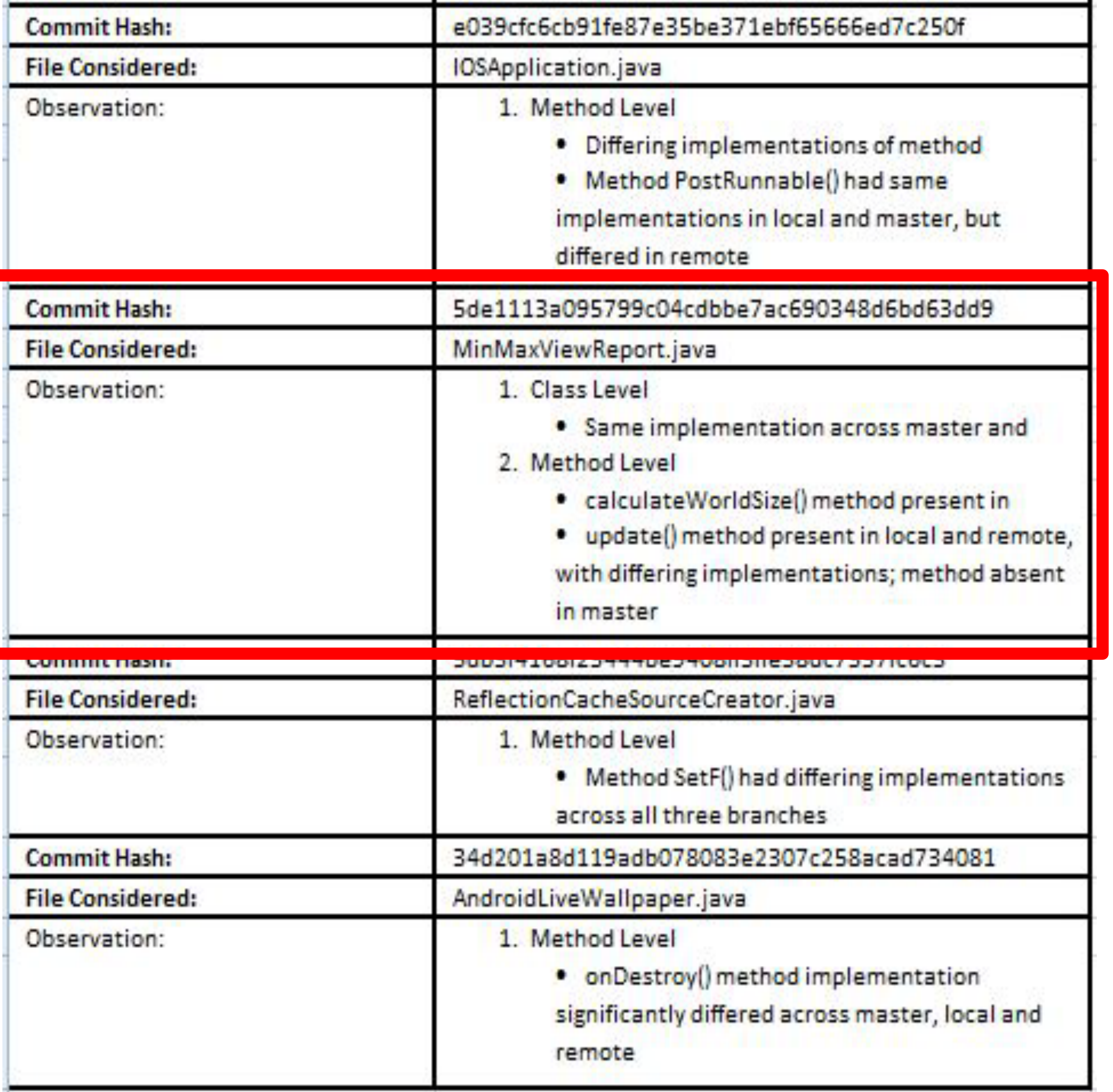
3.2 Commit trends in Open Source s/w

- Having zeroed down on the project, walk through the commit history
  - Revisited 10,823 commits
  - 2003 were merges
    - For each of these merges, checked:
      - If it resulted in a merge conflict
        - If the conflict affected multiple files, methods, etc (Only Java language considered)

- Only 8 merge conflict points could be identified



Commit Hash:	9923a4f2c8dba5a8a946658d43b6774a4269e746
File Considered:	TiledMapRenderer.java
Observation:	1. Class Level: • Implemented as a base interface in local and master, and as derived interface in remote 2. Methods: • Differing method signatures for methods across master and local; Absent in remote
Commit Hash:	c2848bdc2227c8c348e85f3811fc3b4c4dbc7b34
File Considered:	OisTest.java
Observation:	1. Method Level • Differing (println statements and conditionals) implementations of method create() across master, local and remote • Overridden methods across the three states, with same implementations across master and local, but only as stubs in remote
Commit Hash:	f17f1fa2c1e1ba5b5284970c04582ec70acc93ae
File Considered:	DesktopControllers.java
Observation:	1. Method Level • main() method present as member function of class, in master branch and local copy, but class absent in remote branch of the file
Commit Hash:	e4b9f6a15abc48376f76dd6aef44235d5e9931c2
File Considered:	GwtNet.java
Observation:	1. Method Level • Single line changes to several methods in the local branch, that were absent from master as well as remote



Commit Hash:	e039cfc6cb91fe87e35be371ebf65666ed7c250f
File Considered:	IOSApplication.java
Observation:	1. Method Level • Differing implementations of method • Method PostRunnable() had same implementations in local and master, but differed in remote
Commit Hash:	5de1113a095799c04cdbbe7ac690348d6bd63dd9
File Considered:	MinMaxViewReport.java
Observation:	1. Class Level • Same implementation across master and 2. Method Level • calculateWorldSize() method present in • update() method present in local and remote, with differing implementations; method absent in master
Commit Hash:	[illegible]
File Considered:	ReflectionCacheSourceCreator.java
Observation:	1. Method Level • Method SetF() had differing implementations across all three branches
Commit Hash:	34d201a8d119adb078083e2307c258acad734081
File Considered:	AndroidLiveWallpaper.java
Observation:	1. Method Level • onDestroy() method implementation significantly differed across master, local and remote

4. What we arrived at

- What the tool (prototype) can do
  - Input: A skeletal framework expected
  - Output: For all pairs in input, whether a pair is *safe* or *unsafe*

- Commit observations

4.1 Questions we began with

- Can merge conflicts be avoided if we organize check-ins (as an ordered set of safe units) and follow the practice of early commits?
- Given a pair of safe units, how do we assign a measure of potential for merge-conflict?

5. Limitations and Future Work

- The (dependency) analysis relies on the *skeletal input*
- Extension to higher abstraction levels
- Output the path; Levels rather than just binary safety

Thank you