Dynamic Data-Flow Testing

Mattia Vivanti - University of Lugano

Università
della
Svizzera
italiana

**Facoltà
di scienze
informatiche**

Data flow testing

```
public class DummyDivision {

    int i;

    public DummyDivision() {
        i=1;
    }

    public void resetI(){
        i=0;
    }

    public int dividePerI(int j){
        return j/i;
    }

}
```

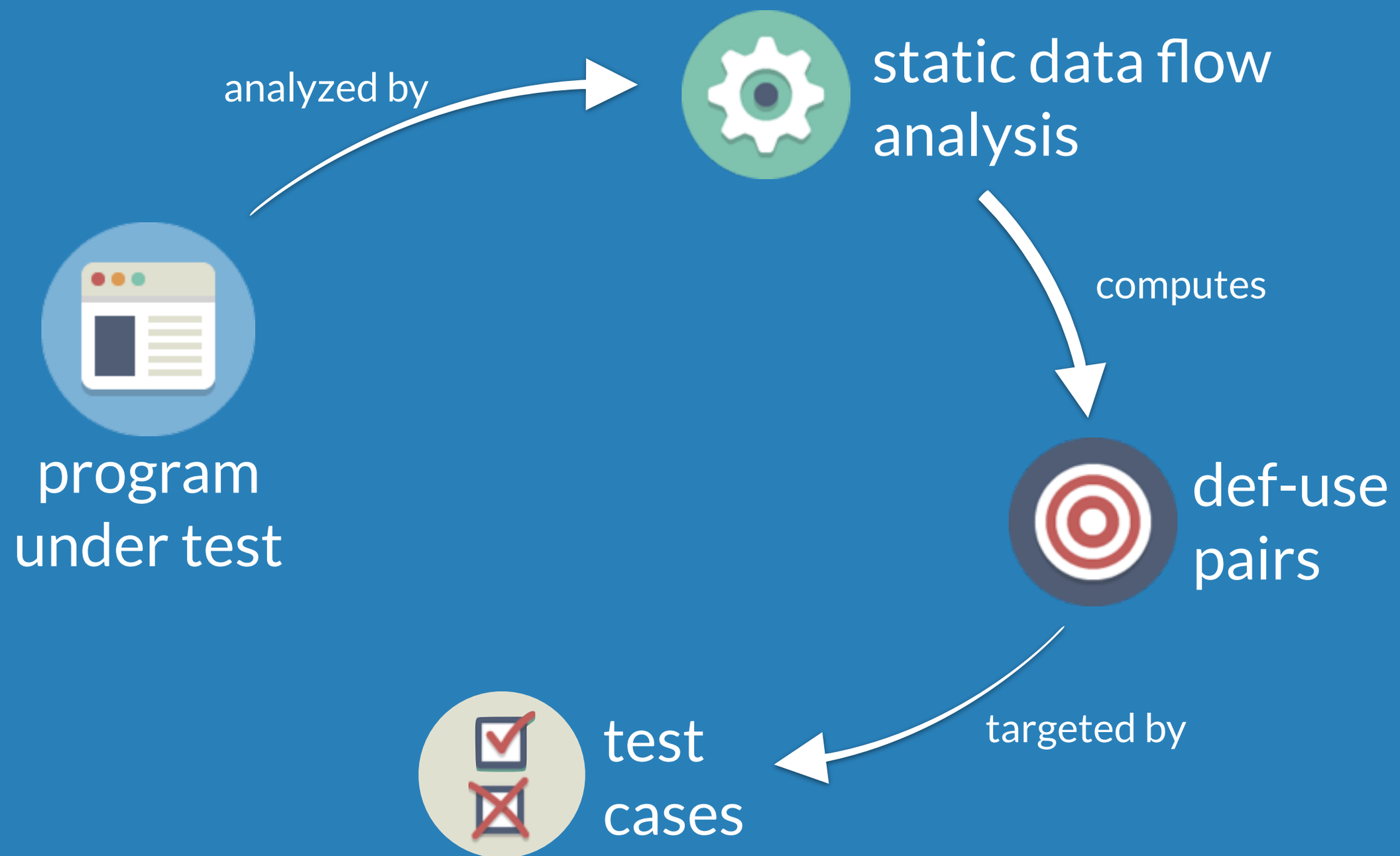
definition of i

definition of i

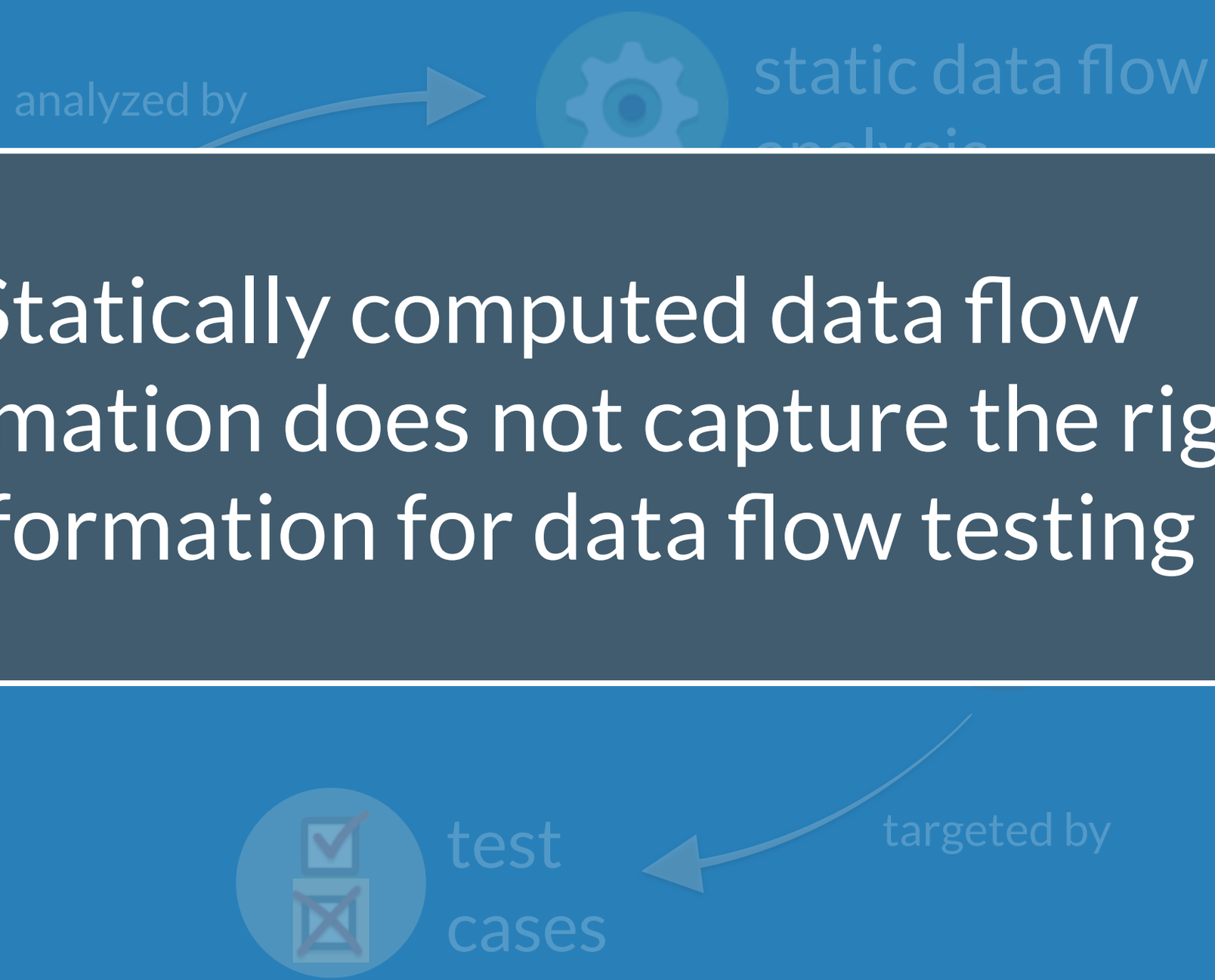
use of i

def-use pairs

Data flow testing



Data flow testing



Statically computed data flow information does not capture the right information for data flow testing

Data flow testing

static data flow analysis

computes

def-use pairs

targeted by

st
ses

Conservative choices and approximations to address **dynamic binding, inter-procedural control flow** and **scalability**

static data
flow analysis

infeasible def-use pairs

```
if(boolean){
    i = 1;        Def i
}
...
if(!boolean){
    read(i);      Use i
}
```

Frankl and Weiss, "An experimental comparison of the effectiveness of branch testing and data flow testing," *TSE 1993*
Hutchins et al. "Experiments of the effectiveness of dataflow- and controlflow-based test adequacy criteria," ICSE 1994

misses def-use pairs

```
Object first = ...;
List list = new ArrayList();
list.add(first);
list.get(0).setI(5);
```

Missed Def `first.i`

Denaro et al., "On the Right Objectives of Data Flow Testing", *ICST 2014*

Dynamic Data-Flow Testing

Detect precise data flow information by observing data flow events dynamically

Incremental computation of test targets by combining dynamic, static analysis and testing

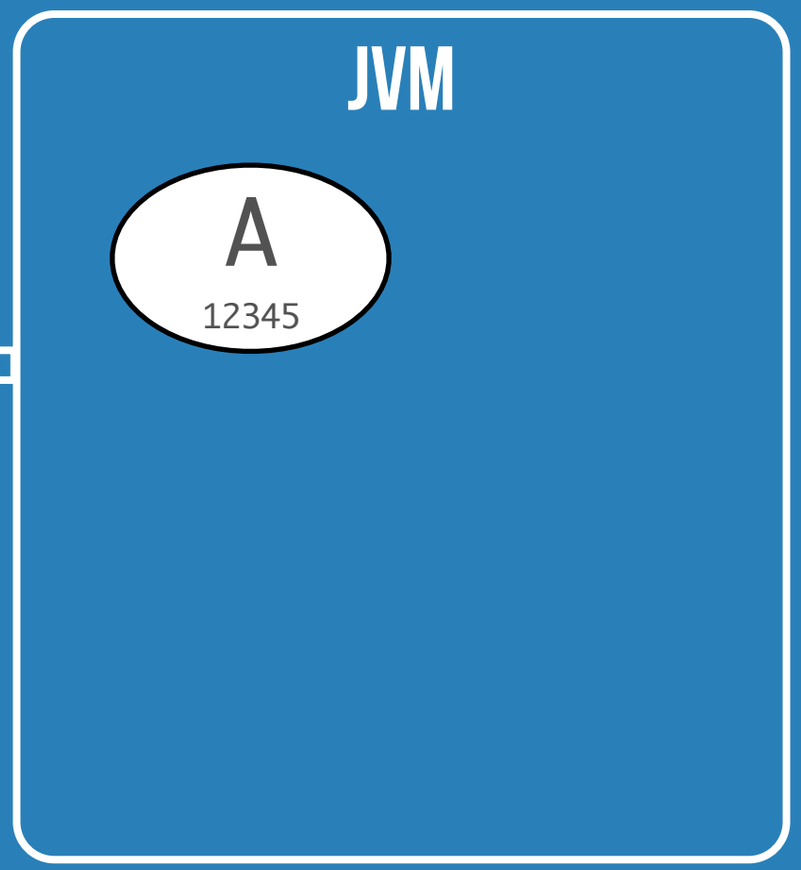
DReaDs: dynamic reaching definitions analysis

monitors definitions and uses at the memory level

monitors associations between instances to identify nesting of states

```
public class A {
  private I b;

  public void methOfA(){
    ...
    b = new B("msg");
    ...
  }
  ...
}
```



ACTIVE Definitions
{-}

Definition EVENTS
{-}

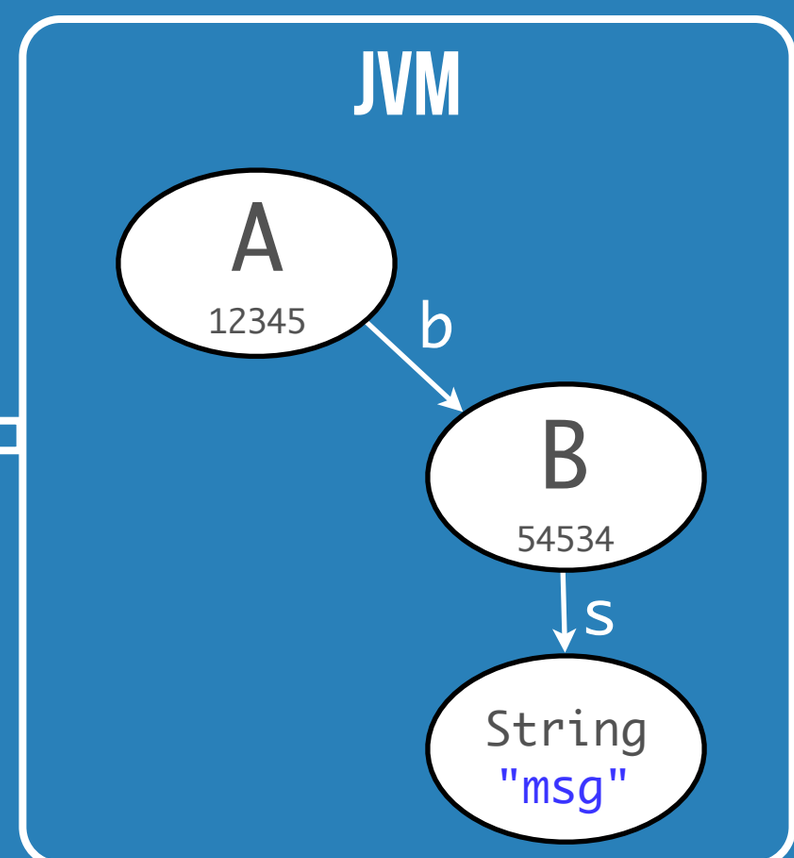
DReaDs: dynamic reaching definitions analysis

monitors definitions and uses at the memory level

monitors associations between instances to identify nesting of states

```
public class A {
  private I b;

  public void methOfA(){
    ...
    b = new B("msg");
    ...
  }
  ...
}
```



ACTIVE Definitions
{B.s, A.b, A.b.s}

Definition EVENTS
{B.s in B.<init>()
A.b in A.methOfA(),
A.b.s in A.methOfA()}

Identification of test targets

Test Case 1

```
public void setup(){
    this.b = "ciao";
}
```

definitions observed at method exit

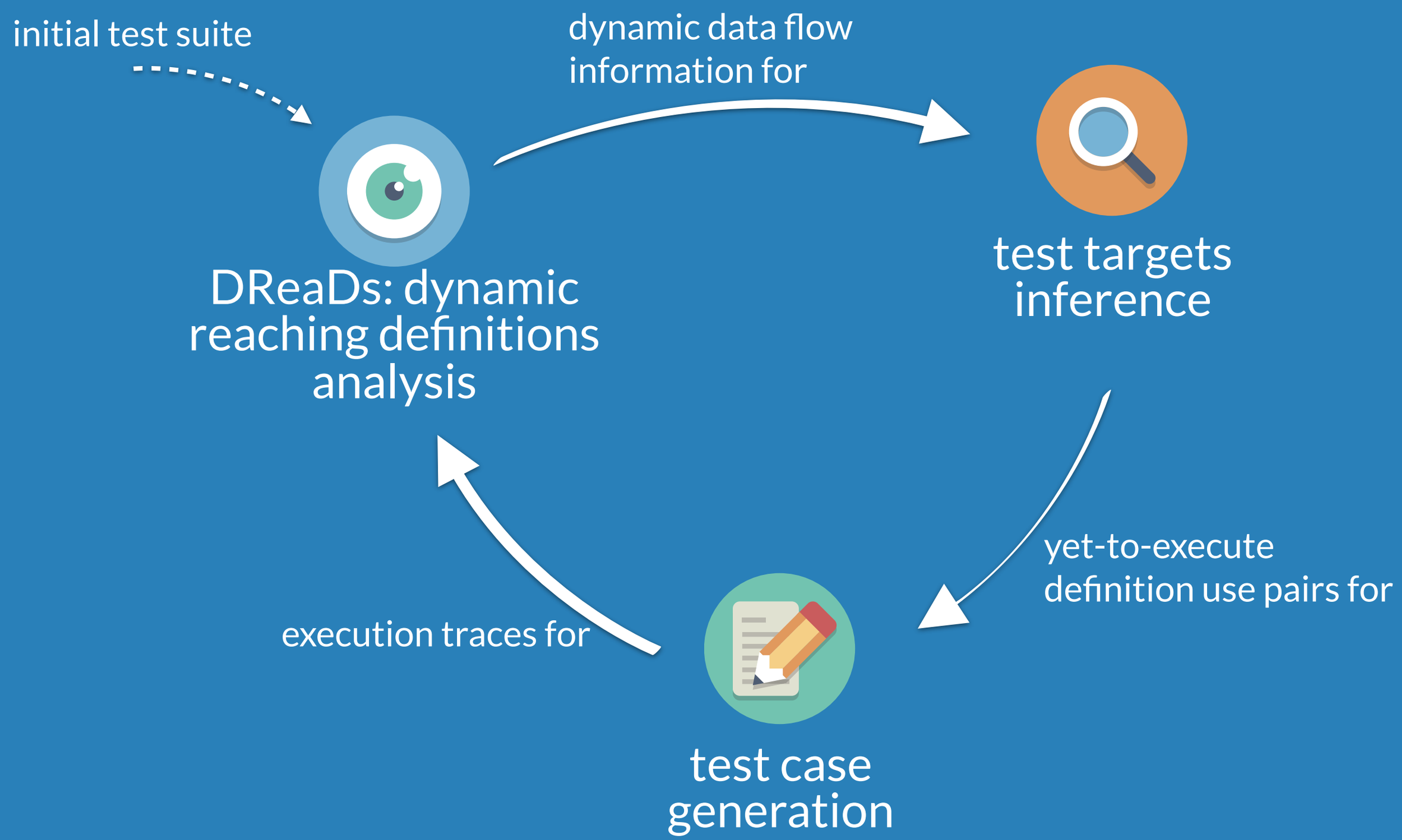
Test Case 2

```
public void read(){
    print(this.b);
}
```

uses observed in the method

Def: `this.b` ⟷ match ⟷ Use: `this.b`

Dynamic Data Flow Testing



Status of Research

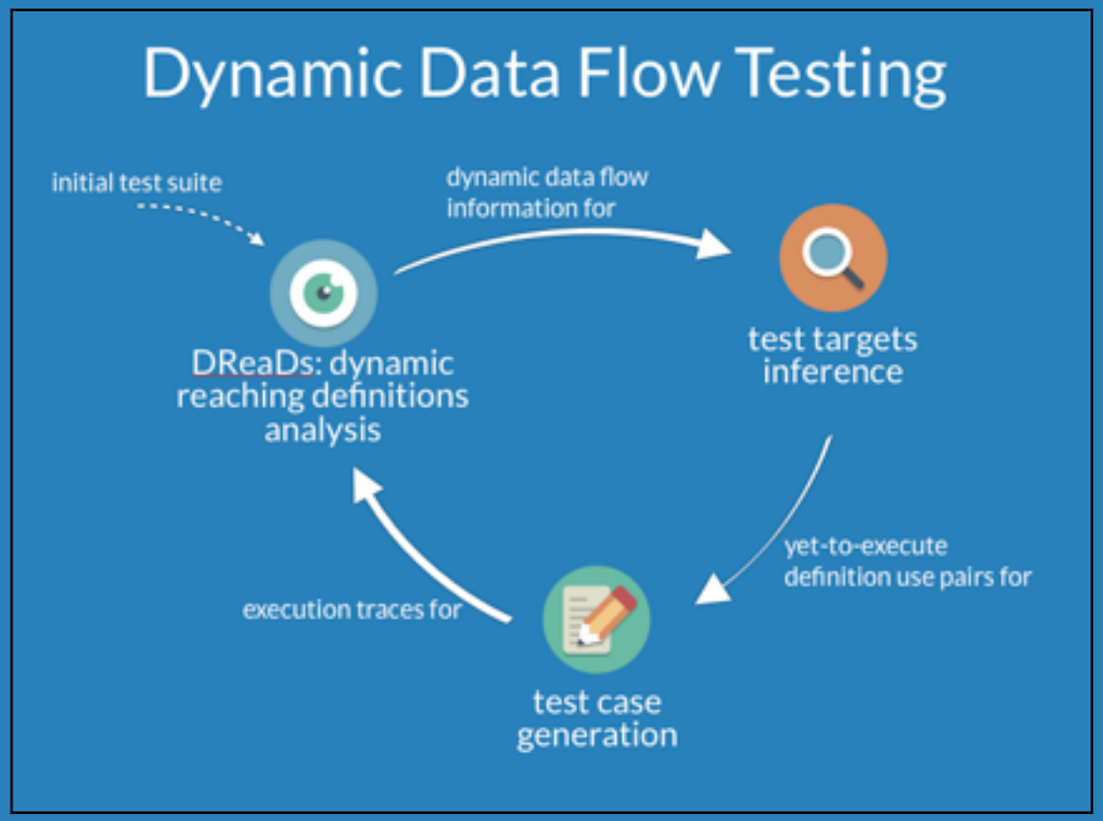
DReaDs: Dynamic Reaching Definitions Analysis for Java

Case study: 5 Java projects, 1531 classes, 88000 eloc.

	# Definitions	% Missed by the other technique
Dreads	169,495	**96%**
DaTeC	28,929	**23%**

Denaro et al., "On the Right Objectives of Data Flow Testing", *ICST 2014*

Current Work



Finalize implementation

Evaluation:

- can we discover a relevant set of definition use pairs using dynamic data flow testing?
- how effective are generated test cases?