Automatic Defect Detection

Andrzej Wasylkowski

Overview

Automatic Defect Detection

"Horizontal" Techniques

Specification-checking Techniques

Mining-based Techniques

Mining Repositories

Mining Traces

Mining Source Code

Mining Repositories (1): DynaMine

```
public void createPartControl(Composite parent) {
    ...
}

public void dispose() {
    ...
}
```

Benjamin Livshits, Thomas Zimmermann, "*DynaMine: Finding common error patterns by mining software revision histories*", ESEC/FSE 2005

Mining Repositories (1): DynaMine

```
public void createPartControl(Composite parent) {
    ...
    // add listener for editor page activation
    getSite().getPage().addPartListener(partListener);
}

public void dispose() {
    ...
    getSite().getPage().removePartListener(partListener);
}
```

Benjamin Livshits, Thomas Zimmermann, "*DynaMine: Finding common error patterns by mining software revision histories*", ESEC/FSE 2005

Mining Repositories (1): DynaMine

```
public void createPartControl(Composite parent) {
    ...
    // add listener for editor page activation
    getSite().getPage().addPartListener(partListener);
}

public void dispose() {
    ...
    getSite().getPage().removePartListener(partListener);
}
```

co-added

Benjamin Livshits, Thomas Zimmermann, "*DynaMine: Finding common error patterns by mining software revision histories*", ESEC/FSE 2005

Mining Repositories (1): DynaMine

```
public void createPartControl(Composite parent) {
    ...
    // add listener for editor page activation
    getS
}

public void dispose() {
    ...
    getSite().getPage().removePartListener(partListener);
}
```

Co-changed items form usage patterns
Patterns are validated at runtime

Benjamin Livshits, Thomas Zimmermann, "*DynaMine: Finding common error patterns by mining software revision histories*", ESEC/FSE 2005

Mining Repositories (1): DynaMine

- Evaluated on Eclipse and jEdit
- Preprocessing CVS history takes a few days
- Mining for co-changed items takes minutes
- Found 56 previously unknown patterns
- Found 263 pattern violations

Benjamin Livshits, Thomas Zimmermann, "*DynaMine: Finding common error patterns by mining software revision histories*", ESEC/FSE 2005

Mining Repositories (2): BugMem

```
public void computePackageFragmentRoots(...) {
    ...
    if (requiredProjectRsc.exists() &&
        requiredProjectRsc.isOpen()) {
    ...
}
```


```
public void computePackageFragmentRoots(...) {
    ...
    if (JavaProject.hasJavaNature(requiredProjectRsc))
    ...
}
```

Sunghun Kim, Kai Pan, E. James Whitehead, Jr., "*Memories of Bug Fixes*", FSE 2006

Mining Repositories (2): BugMem

```
public void computePackageFragmentRoots(...) {
    ...
    if (requiredProjectRsc.exists() &&
        requiredProjectRsc.isOpen()) {
    ...
}
```


```
public void computePackageFragmentRoots(...) {
    ...
    if (JavaProject.hasJavaNature(requiredProjectRsc))
    ...
}
```

Sunghun Kim, Kai Pan, E. James Whitehead, Jr., "*Memories of Bug Fixes*", FSE 2006

Mining Repositories (2): BugMem

```
public void computePackageFragmentRoots(...) {
    ...
    if (requiredProjectRsc.exists() &&
        requiredProjectRsc.isOpen()) {
    ...
}
```

Previous bugs form bug patterns
Previous fixes form fix patterns

```
public void computePackageFragmentRoots(...) {
    ...
    if (JavaProject.hasJavaNature(requiredProjectRsc))
    ...
}
```

Sunghun Kim, Kai Pan, E. James Whitehead, Jr., "*Memories of Bug Fixes*", FSE 2006

Mining Repositories (2): BugMem

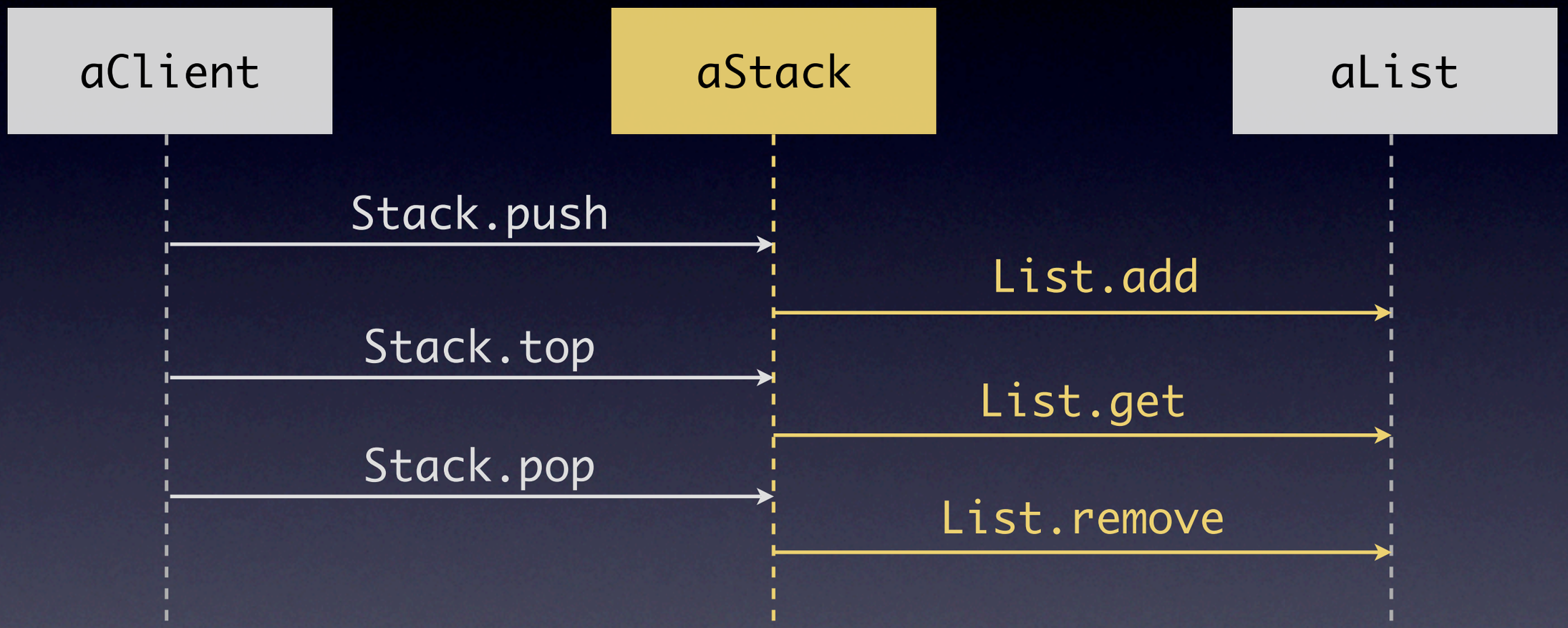
- Evaluated on 5 open source projects: ArgoUML, Columba, Eclipse, jEdit, Scarab
- 19.3%–40.3% of bugs appear in the memories
- 7.9%–15.5% of bug and fix pairs appear in the memories

Sunghun Kim, Kai Pan, E. James Whitehead, Jr., "*Memories of Bug Fixes*", FSE 2006

Mining Traces (1): AMPLE

Valentin Dallmeier, Christian Lindig, Andreas Zeller,
"*Lightweight defect localization for Java*", ECOOP 2005

Mining Traces (1): AMPLE



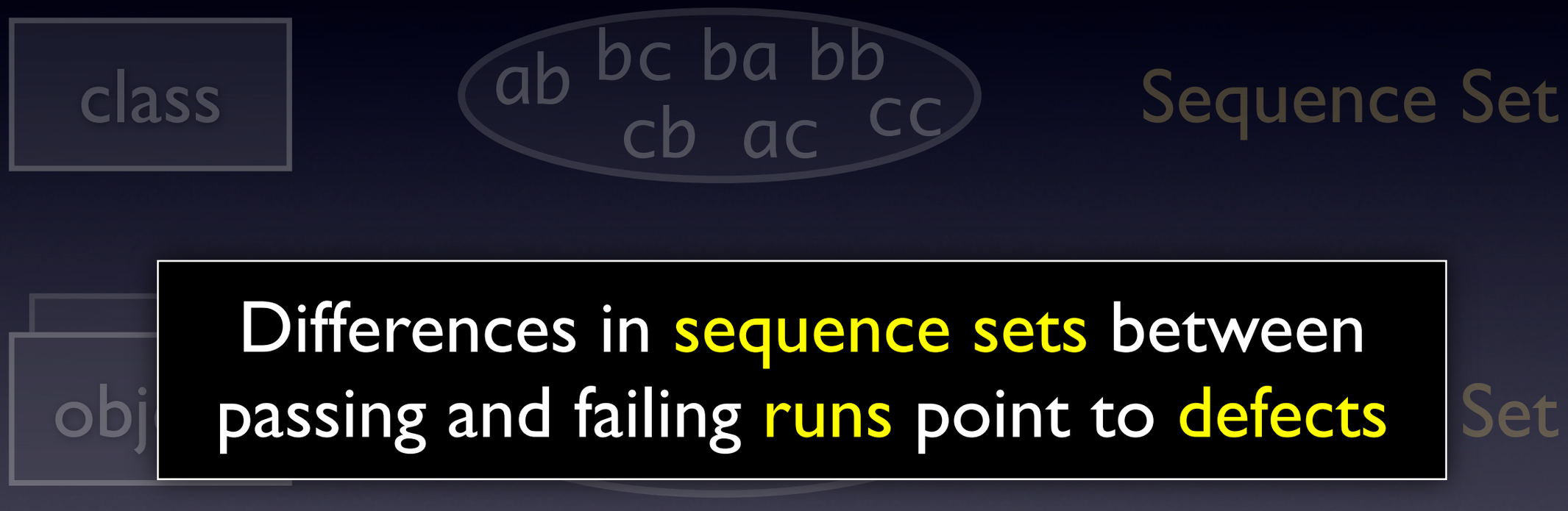
aStack: List.add, List.get, List.remove

Valentin Dallmeier, Christian Lindig, Andreas Zeller,
"*Lightweight defect localization for Java*", ECOOP 2005

Mining Traces (1): AMPLE

Valentin Dallmeier, Christian Lindig, Andreas Zeller,
"*Lightweight defect localization for Java*", ECOOP 2005

Mining Traces (1): AMPLE



Valentin Dallmeier, Christian Lindig, Andreas Zeller,
"*Lightweight defect localization for Java*", ECOOP 2005

Mining Traces (1): AMPLE

- Evaluated on NanoXML (33 seeded defects)
- Identifies the defective class in 36% of cases
- On average, a programmer has to investigate 10% of all classes to find the defective one

Valentin Dallmeier, Christian Lindig, Andreas Zeller,
"*Lightweight defect localization for Java*", ECOOP 2005

Mining Traces (2): Perracotta

Trace

```
ENTRY: Producer.main
ENTRY: Buffer.add
EXIT:  Buffer.add
ENTRY: Buffer.add
EXIT:  Buffer.add
ENTRY: Buffer.stop
EXIT:  Buffer.stop
EXIT:  Producer.main
```

Properties instantiations

```
P=ENTRY: Buffer.add
S=ENTRY: Buffer.stop

P=ENTRY: Buffer.add
S=EXIT:  Producer.main

...
```

Temporal property patterns

```
[-P]*;(P;[-S]*;S;[-P]*)*
...
```

Jinlin Yang, David Evans, Deepali Bhardwaj, Thirumalesh Bhat, Manuvir Das, "*Perracotta: Mining Temporal API Rules from Imperfect Traces*", ICSE 2006

Mining Traces (2): Perracotta

Trace

```
ENTRY: Producer.main
ENTRY: Buffer.add
EXIT:  Buffer.add
ENTRY: Buffer.add
EXIT:  Buffer.stop
EXIT:  Producer.main
```

Properties instantiations

```
P=ENTRY: Buffer.add
S=ENTRY: Buffer.stop
```

```
P=ENTRY: Buffer.add
...
```

Properties instantiations form API rules

Temporal property patterns

```
[-P]*;(P;[-S]*;S;[-P]*)*
...
```

Jinlin Yang, David Evans, Deepali Bhardwaj, Thirumalesh Bhat, Manuvir Das, "*Perracotta: Mining Temporal API Rules from Imperfect Traces*", ICSE 2006

Mining Traces (2): Perracotta

- Evaluated on Daisy, JBoss and Windows kernel
- Analysis of traces takes several minutes
- Found a defect in the NTFS file system

Jinlin Yang, David Evans, Deepali Bhardwaj, Thirumalesh Bhat, Manuvir Das, "*Perracotta: Mining Temporal API Rules from Imperfect Traces*", ICSE 2006

Mining Source Code (1): PR-Miner

Function's source code

```
static void
getRelationDescription (...)
{
    HeapTuple relTup;
    ...
    relTup = SearchSysCache (...);
    if (!HeapTupleIsValid (relTup))
        elog (...);
    relForm = ...;
    ...
    ReleaseSysCache (relTup);
}
```

Zhenmin Li, Yuanyuan Zhou, "*PR-Miner: Automatically extracting implicit programming rules and detecting violations in large software code*", ESEC/FSE 2005

Mining Source Code (1): PR-Miner

Function's source code

```
static void
getRelationDescription (...)
{
    HeapTuple relTup;
    ...
    relTup = SearchSysCache (...);
    if (!HeapTupleIsValid (relTup))
        elog (...);
    relForm = ...;
    ...
    ReleaseSysCache (relTup);
}
```

Itemset

```
T: HeapTuple
F: SearchSysCache
F: HeapTupleIsValid
F: elog
T: Form_pg_class
F: ReleaseSysCache
...
```

Zhenmin Li, Yuanyuan Zhou, "*PR-Miner: Automatically extracting implicit programming rules and detecting violations in large software code*", ESEC/FSE 2005

Mining Source Code (1): PR-Miner

Itemset #1

```
T: HeapTuple
F: SearchSysCache
F: HeapTupleIsValid
F: elog
T: Form_pg_class
F: ReleaseSysCache
...
```

Itemset #2

```
T: StringInfoData
F: getObjectClass
T: HeapTuple
F: SearchSysCache
F: NameStr
T: Relation
F: ReleaseSysCache
...
```

Itemset #3

```
T: Form_pg_class
F: SearchSysCache
F: elog
F: RelationIsVisible
F: ReleaseSysCache
...
```

Zhenmin Li, Yuanyuan Zhou, "*PR-Miner: Automatically extracting implicit programming rules and detecting violations in large software code*", ESEC/FSE 2005

Mining Source Code (1): PR-Miner

Itemset #1

```
T: HeapTuple
F: SearchSysCache
F: HeapTupleIsValid
F: elog
T: Form_pg_class
F: ReleaseSysCache
...
```

Itemset #2

```
T: StringInfoData
F: getObjectClass
T: HeapTuple
F: SearchSysCache
F: NameStr
T: Relation
F: ReleaseSysCache
...
```

Itemset #3

```
T: Form_pg_class
F: SearchSysCache
F: elog
F: RelationIsVisible
F: ReleaseSysCache
...
```

Zhenmin Li, Yuanyuan Zhou, "*PR-Miner: Automatically extracting implicit programming rules and detecting violations in large software code*", ESEC/FSE 2005

Mining Source Code (1): PR-Miner

Itemset #1	Itemset #2	Itemset #3
T: HeapTuple	T: StringInfoData	T: Form_pg_class
F: SearchSysCache	F: getObjectClass	F: SearchSysCache
F: HeapT		
F: elog		sVisible
T: Form_		sCache
F: ReleaseSysCache	T: Relation	...
...	F: ReleaseSysCache	
	...	

Frequent itemsets form programming rules
Missing items point to defects

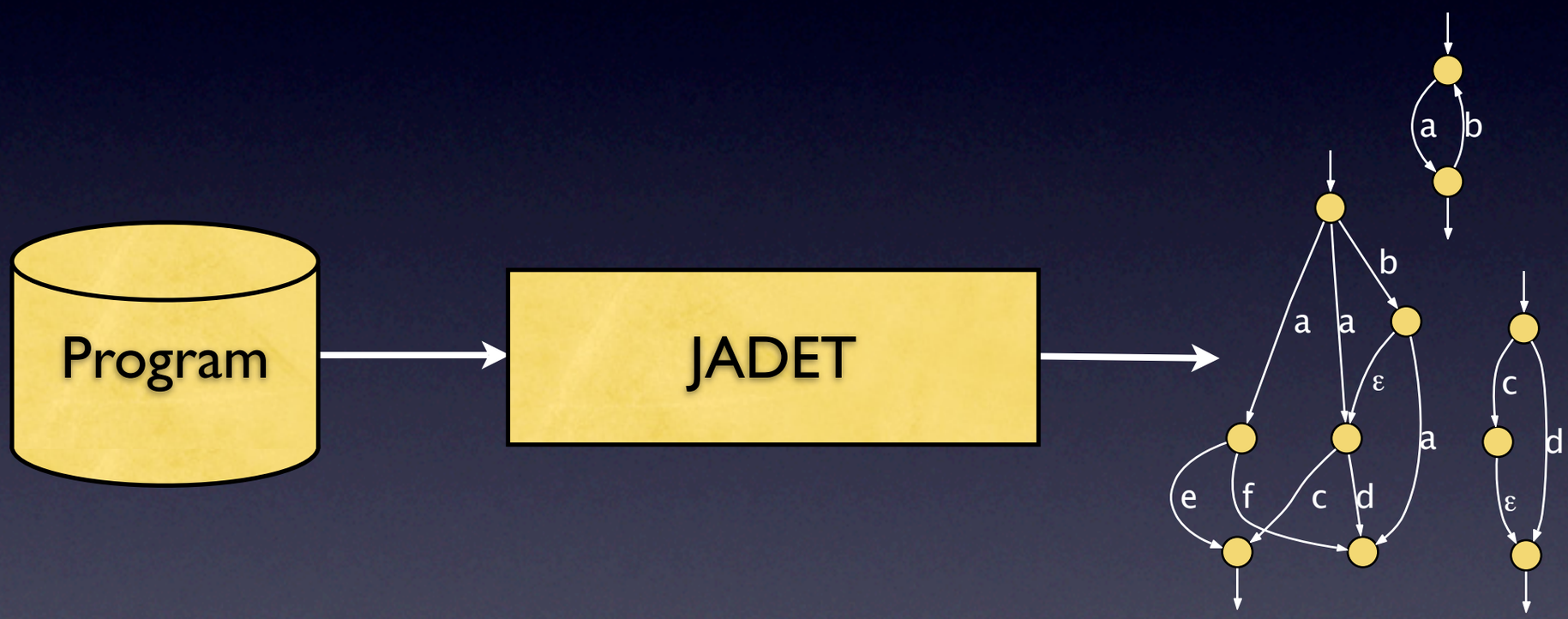
Zhenmin Li, Yuanyuan Zhou, "*PR-Miner: Automatically extracting implicit programming rules and detecting violations in large software code*", ESEC/FSE 2005

Mining Source Code (1): PR-Miner

- Evaluated on Linux, PostgreSQL and Apache
- Finds rules and violations within 2 minutes
- Found 32,283 rules for Linux
- Found 23 bugs in 180 violations

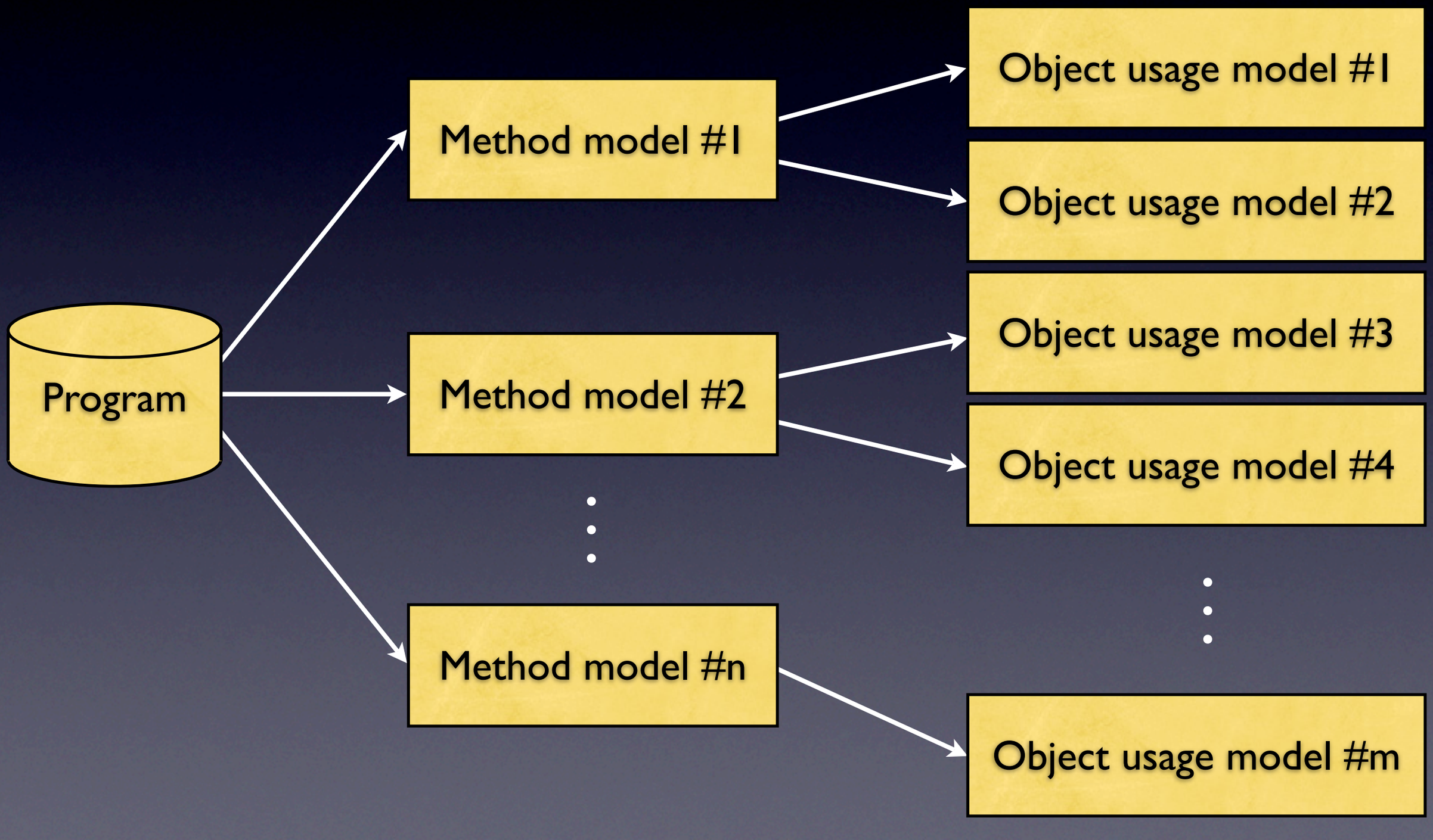
Zhenmin Li, Yuanyuan Zhou, "*PR-Miner: Automatically extracting implicit programming rules and detecting violations in large software code*", ESEC/FSE 2005

Mining Source Code (2): JADET



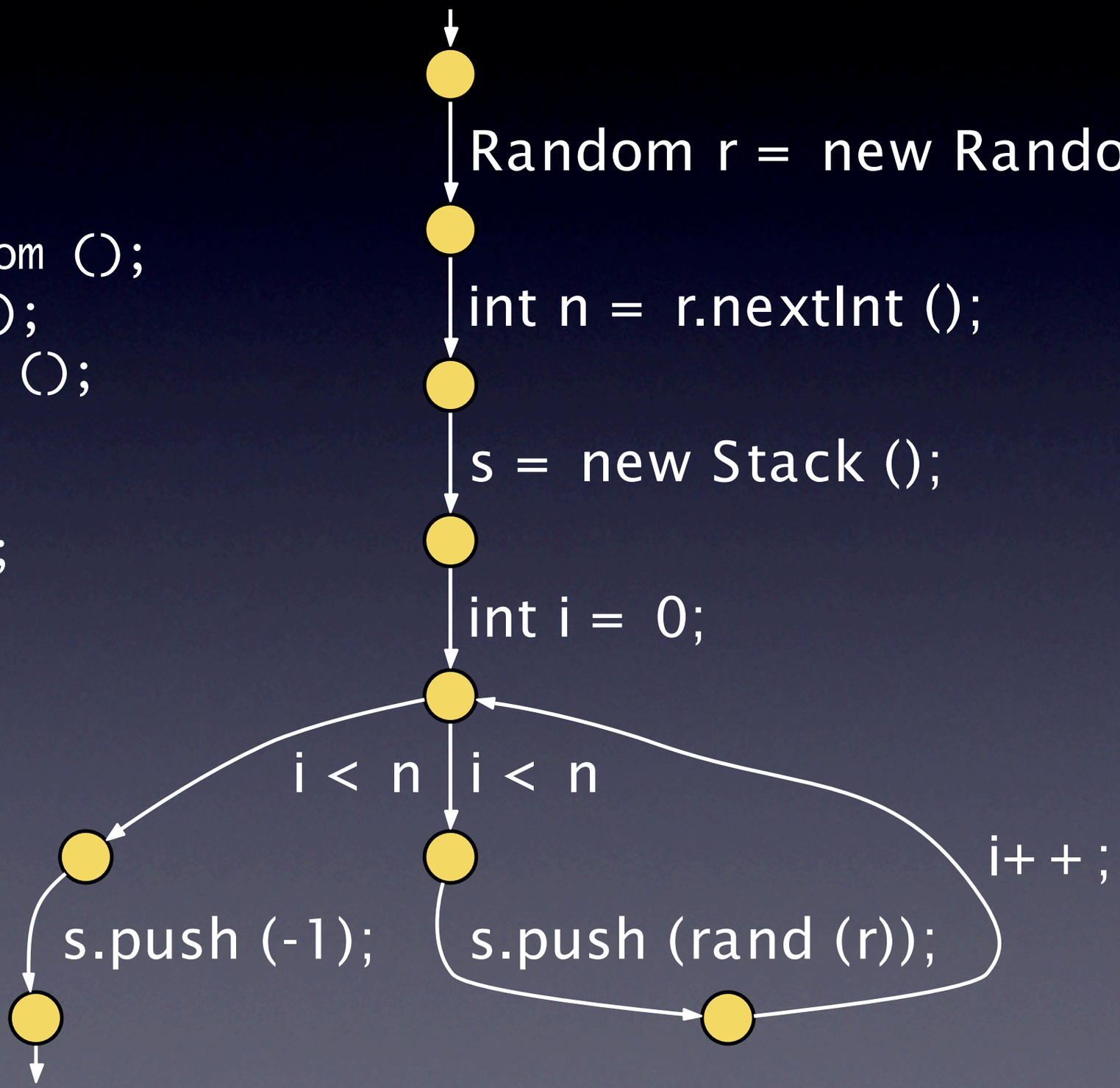
Modeling objects' behavior using finite state automata

The mining process: overview

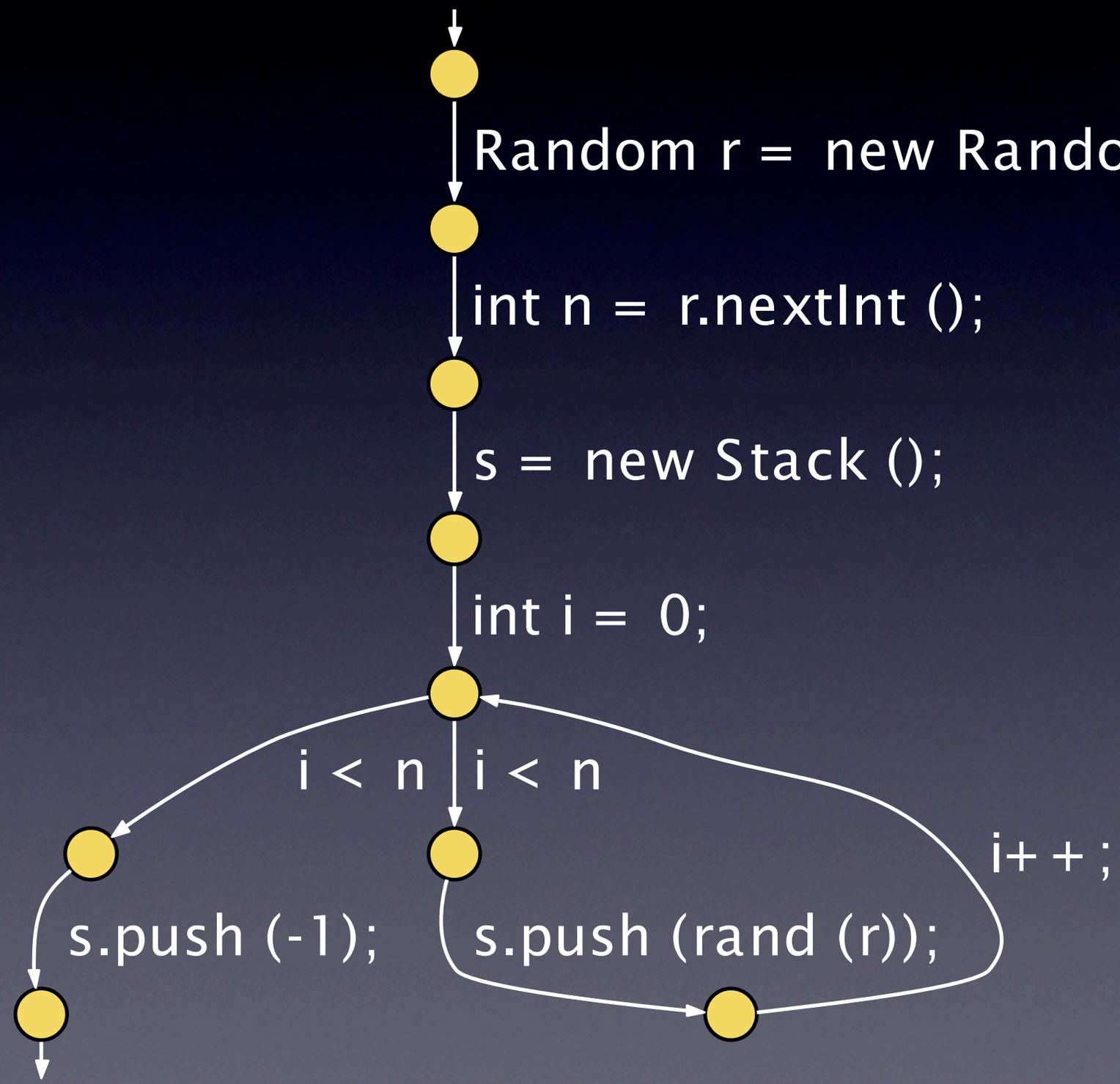


Building a method model

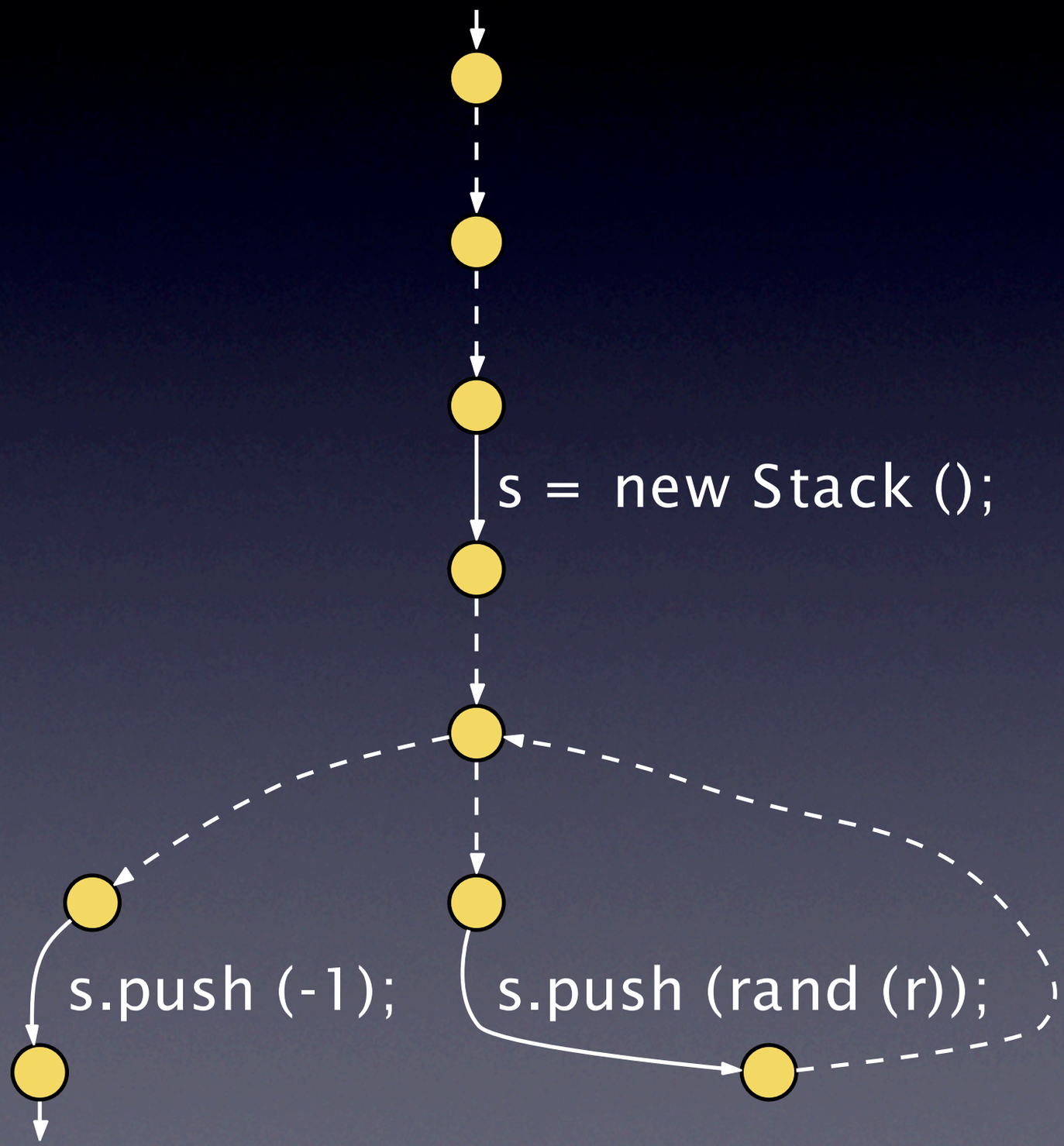
```
public Stack cS () {
  Random r = new Random ();
  int n = r.nextInt ();
  Stack s = new Stack ();
  int i = 0;
  while (i < n) {
    s.push (rand (r));
    i++;
  }
  s.push (-1);
  return s;
}
```



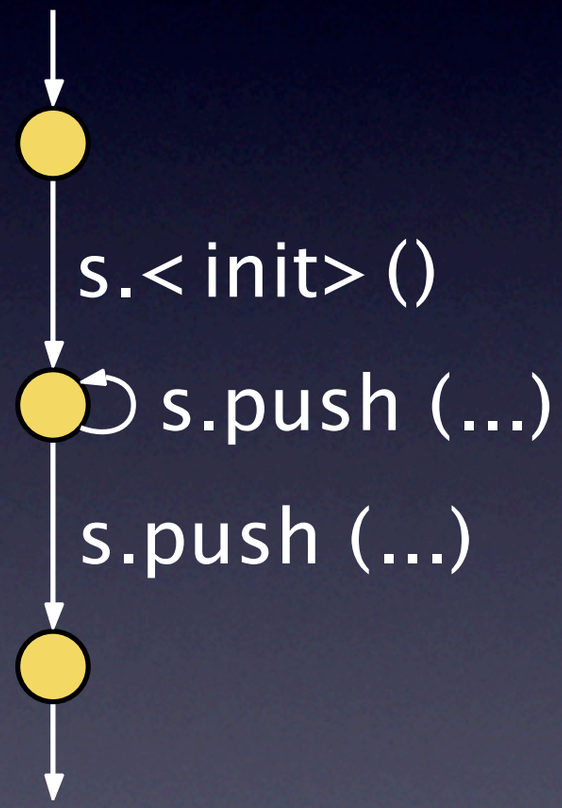
Creating a usage model



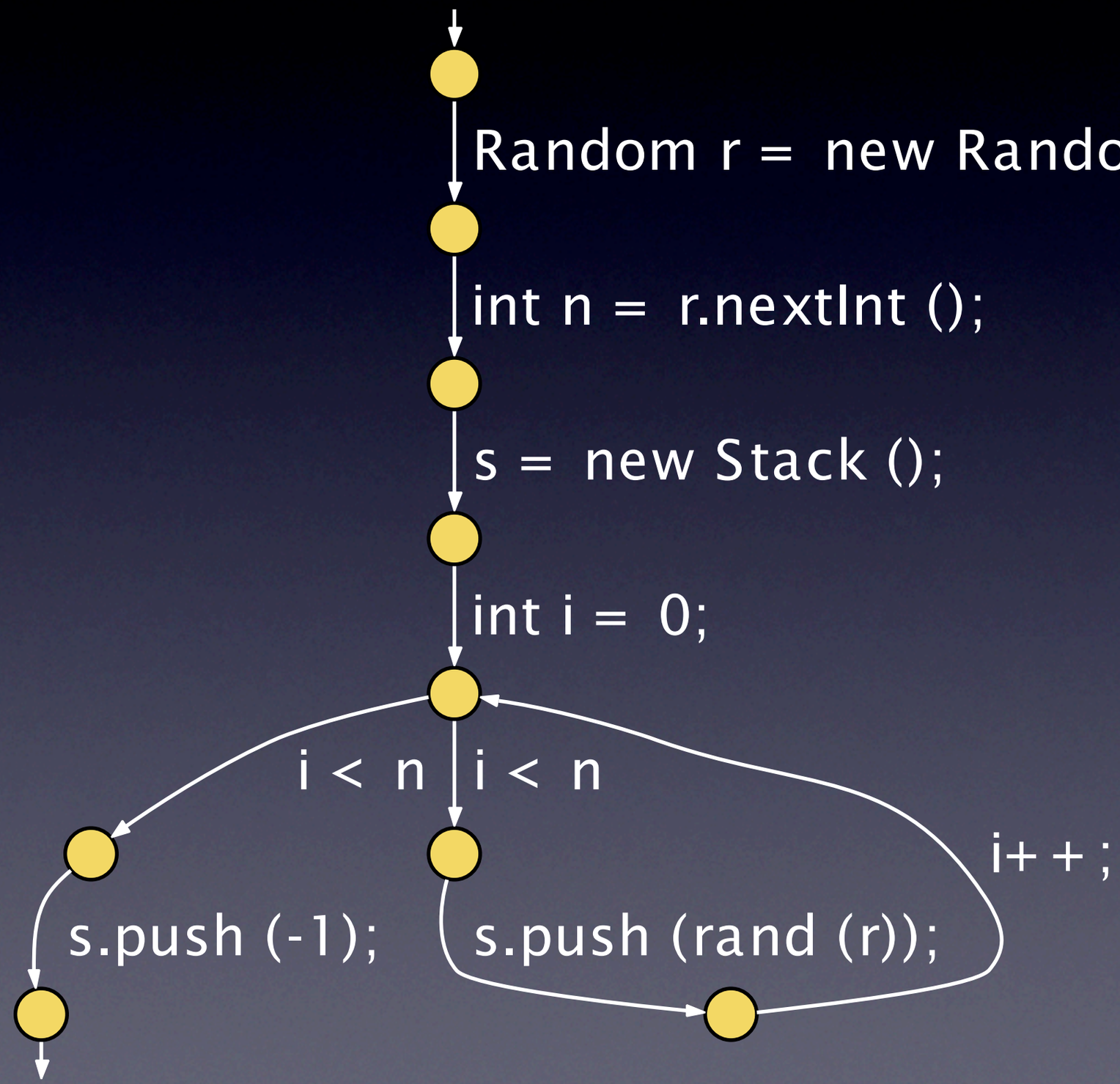
Creating a usage model



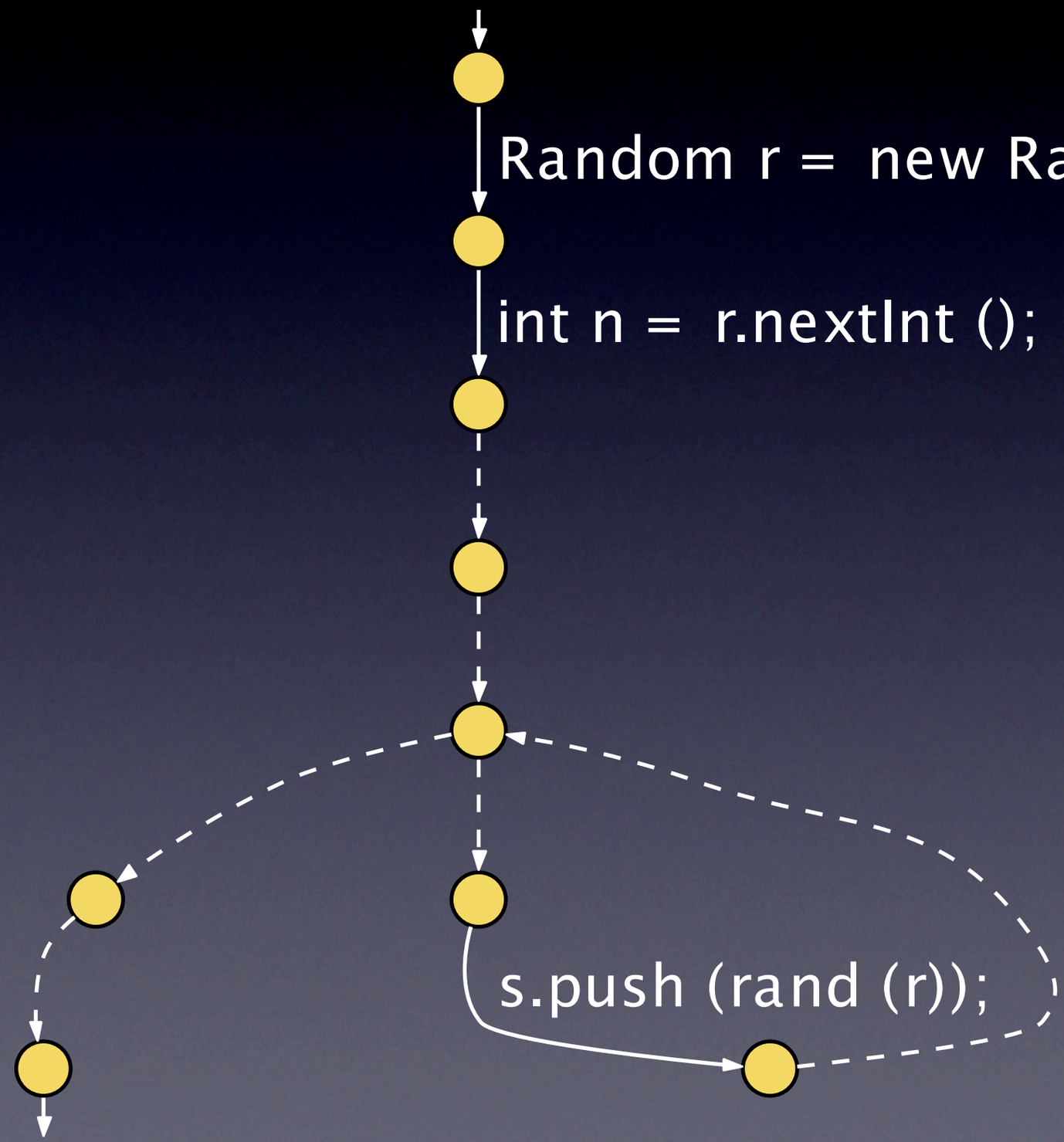
Creating a usage model



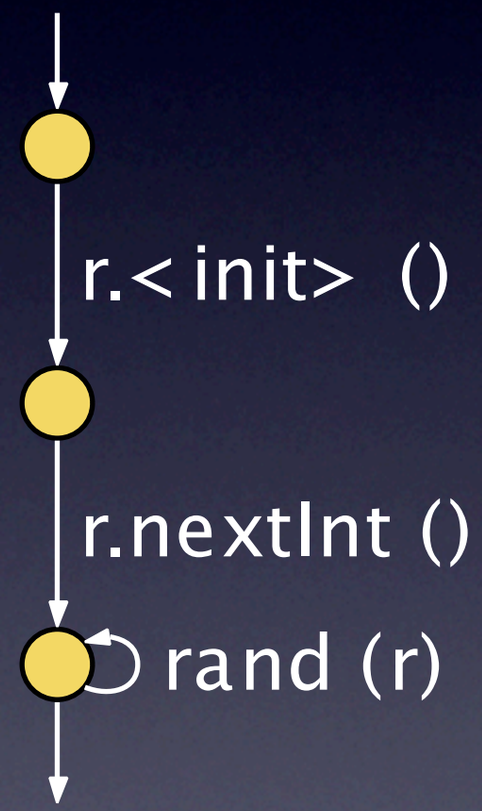
Creating a usage model



Creating a usage model



Creating a usage model

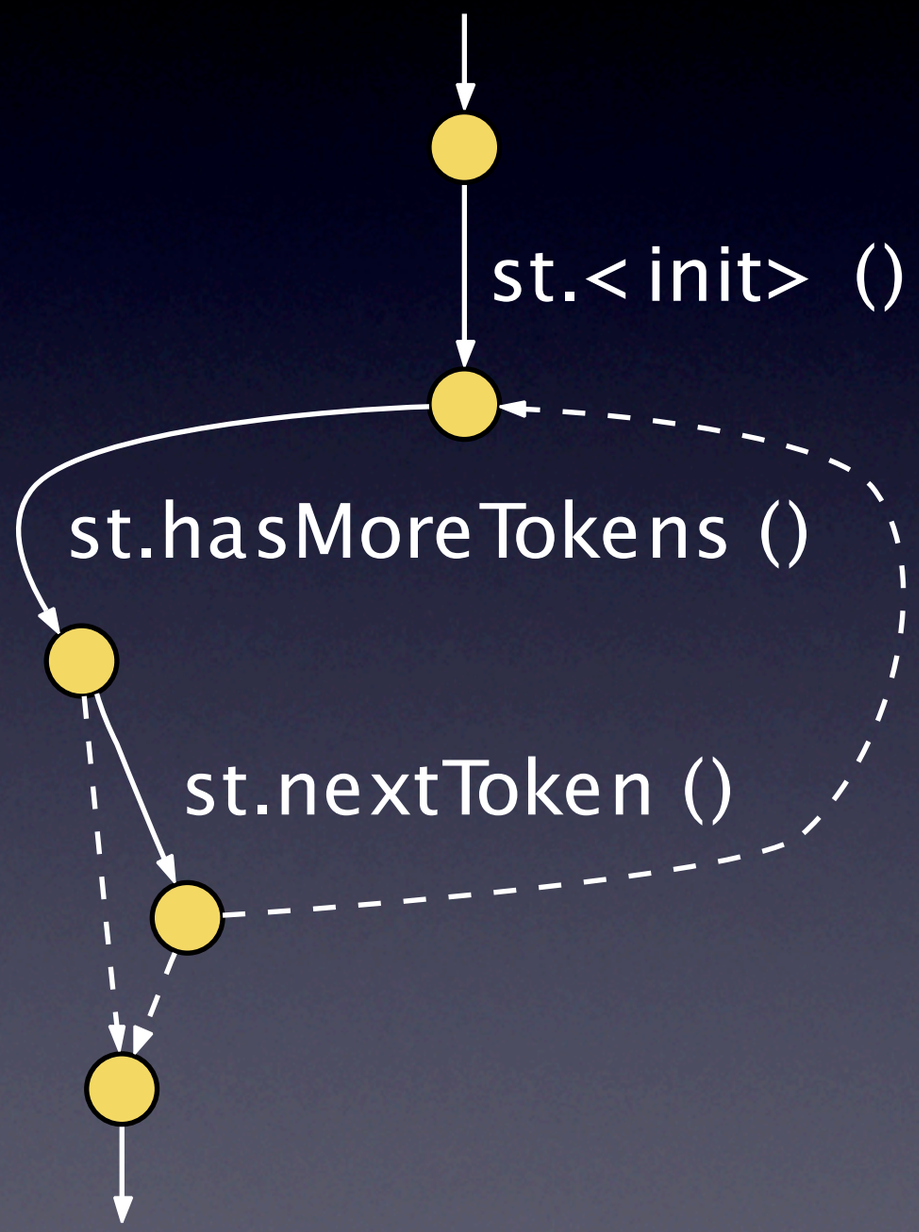


Time & Models

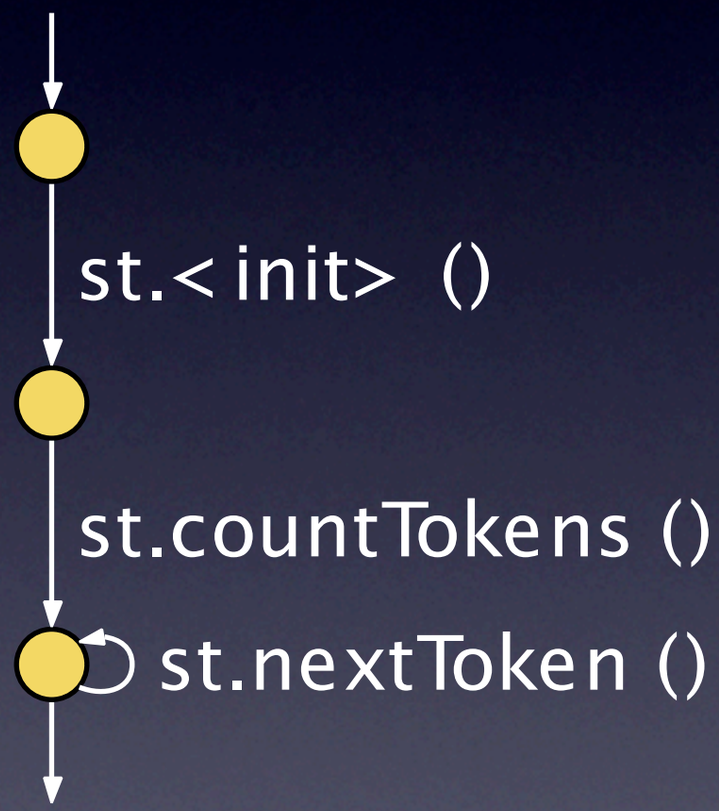
Program	Classes	Methods	# Models	Time
ACT–RBOT 0.8.2	2,492	26,501	187,137	9:03
AspectJ 1.5.3	2,957	36,044	253,084	13:19
Azureus 2.5.0.0	3,585	22,359	157,313	7:28
Columba 1.2	1,165	6,894	54,371	2:06
MusiComp 1.0 beta	347	2,078	17,321	1:02

StringTokenizer st

in AdviceSignatureImpl.toAdviceName()

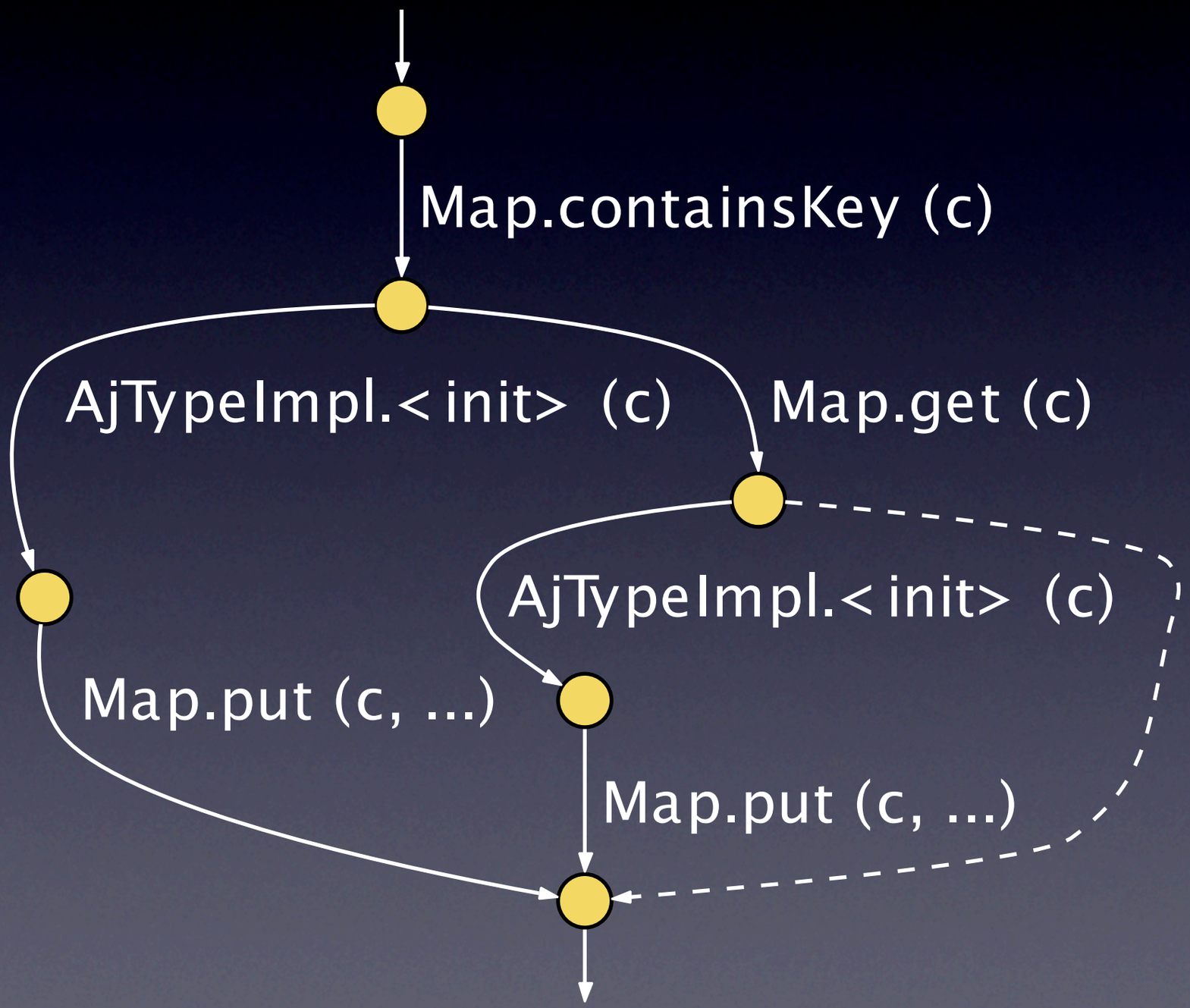


StringTokenizer st
in Factory.makeConstructorSig()

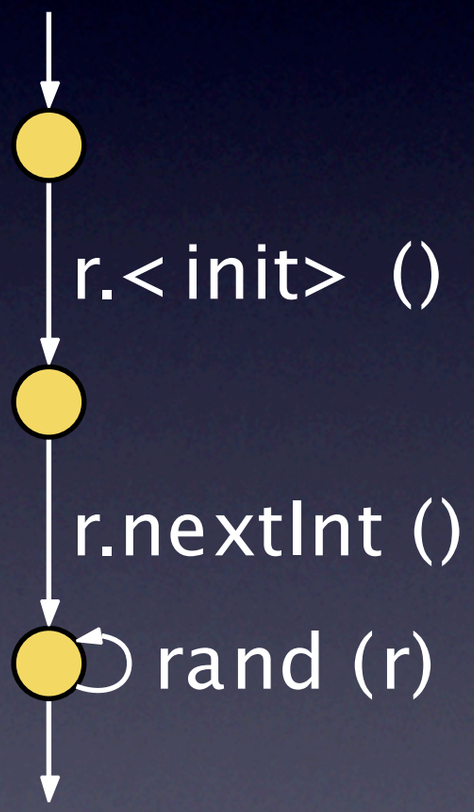


Class c

in AjTypeSystem.getAjType()



Temporal properties



```
<init> ≺ nextInt
<init> ≺ rand
nextInt ≺ rand
rand ≺ rand
```

Programming patterns

in Dump.println()

```
hasNext < hasNext
hasNext < next
getProperty < getProperty
<init> < append
…
```

Programming patterns

Method	hasNext < next	hasNext < hasNext
Dump.println	✔	✔
ClassPathManager.<init>	✔	✔
OrderedLock.updateCO	✘	✘
BcelObjectType.getA	✔	✔
DeclareParents.verifyNIAP	✔	✘

Missing properties point to anomalies

The anomalous method

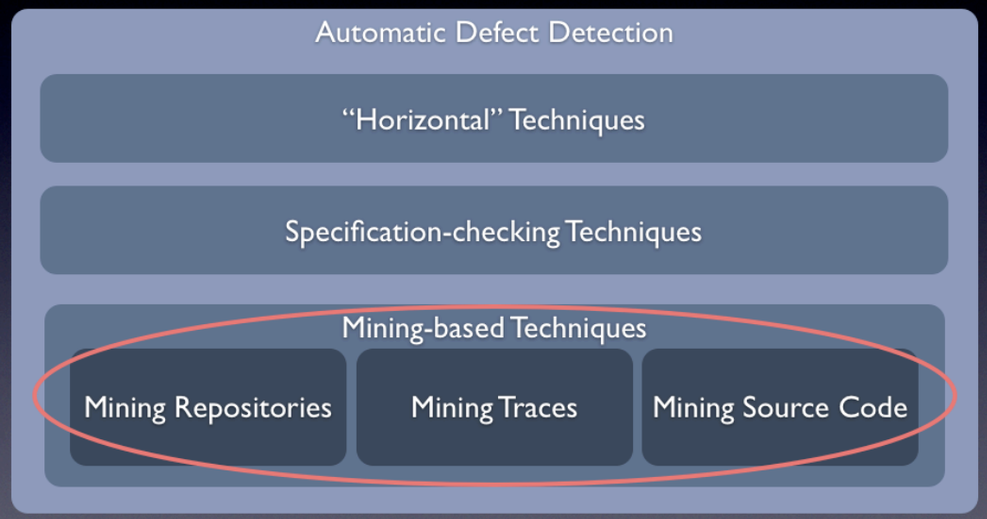
```
private boolean verifyNIAP (…) {
    …
    Iterator iter = …;
    while (iter.hasNext()) {
        … = iter.next();
        …
        return verifyNIAP (…);
    }
    return true;
}
```

Time & Anomalies

Program	# Models	# Anomalies	Time
ACT–RBOT 0.8.2	187,137	192	0:53
AspectJ 1.5.3	253,084	276	3:47
Azureus 2.5.0.0	157,313	365	0:50
Columba 1.2	54,371	64	0:11
MusiComp 1.0 beta	17,321	3	0:04

Top 10 contained 5 previously unknown defects

Overview



Mining Repositories (1): DynaMine

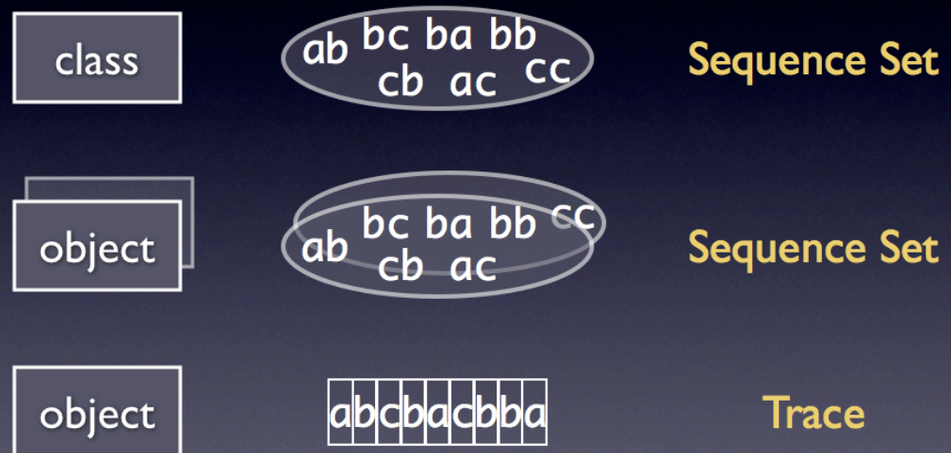
```
public void createPartControl(Composite parent) {
    ...
    // add listener for editor page activation
    getSite().getPage().addPartListener(partListener);
}

public void dispose() {
    ...
    getSite().getPage().removePartListener(partListener);
}
```

co-added

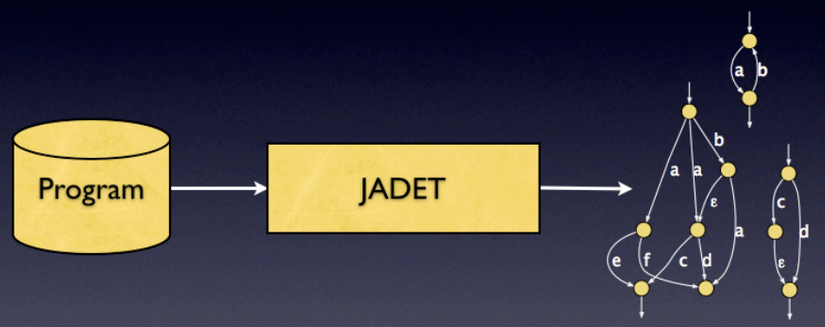
Benjamin Livshits, Thomas Zimmermann, "*DynaMine: Finding common error patterns by mining software revision histories*", ESEC/FSE 2005

Mining Traces (1): AMPLE



Valentin Dallmeier, Christian Lindig, Andreas Zeller, "*Lightweight defect localization for Java*", ECOOP 2005

Mining Source Code (2): JADET



Modeling objects' behavior using finite state automata