Carving Invariants

Andreas Zeller
Saarland University

CREST Open Workshop "The Oracle Problem for Automated Software Testing", London, May 2012

Concrete Tests

Concrete Tests

```
y = sqrt(4)
assert(y == 2)
```

Concrete Tests

```
y = sqrt(9)
assert(y == 3)
```

```
y = sqrt(4)
assert(y == 2)
```

Concrete Tests

```
y = sqrt(9)
assert(y == 3)
```

```
y = sqrt(0)
assert(y == 0)
```

```
y = sqrt(4)
assert(y == 2)
```

Concrete Tests

```
y = sqrt(9)
assert(y == 3)
```

```
y = sqrt(0)
assert(y == 0)
```

```
y = sqrt(4)
assert(y == 2)
```

```
y = sqrt(1)
assert(y == 1)
```

Concrete Tests

```
y = sqrt(9)
assert(y == 3)
```

```
y = sqrt(0)
assert(y == 0)
```

```
y = sqrt(4)
assert(y == 2)
```

```
y = sqrt(1)
assert(y == 1)
```

```
y = sqrt(–1)
assert(y == NaN)
```

Generated Tests

```
assert (y*y == x)
```

Generated Tests

assert (y*y == x)

Program Proofs

assert (y*y == x)

Specifying is Hard

```
assert (y*y == x)
```

ERC Advanced Grant — Research Proposal

SPECMATE: Specification Mining and Testing

Principal Investigator (PI): Andreas Zeller
PI's host institution: Saarland University

Project duration: 60 months

SPECMATE Project Summary

In the past decade, automated validation of software systems has made spectacular progresses. On the testing side, it is now possible to automatically generate test cases that effectively explore the entire program structure; on the verification side, we can now formally prove properties for software as complex as operating systems. To push validation further, however, we need *specifications* of what the software actually should do. But writing such specifications has always been hard—and so far significantly inhibited the deployment of rigorous development methods.

The SPECMATE methodology automatically extracts such specifications from existing systems, effectively leveraging the knowledge encoded into billions of code lines. SPECMATE starts with just an executable program and automatically produces an *incremental specification,* starting with the most relevant properties; and a *set of test cases* fully covering the specification.

SPECMATE will boost quality and productivity in all software development activities, in particular:

Verification and modeling as the specifications mined are high-level and incremental, and thus form an ideal starting point for compositional modeling and verification—**enabling the rigorous construction and derivation of new, safe, dependable software systems**;

Testing as SPECMATE produces a full-fledged test suite for free: rather than manually exploring the system and its concrete executions, the programmer only needs to validate the mined high-level specifications against the (implicitly) intended behavior;

Defect detection since the mined specifications also reveal *undesired* properties: every such property comes with a test case demonstrating it;

Program maintenance as it eases program understanding and change impact assessment: every aspect of the program behavior will be described in a high-level, abstract specification.

ERC Advanced Grant — Research Proposal

SPECMATE: Specification Mining and Testing

Principal Investigator (PI): Andreas Zeller
PI's host institution: Saarland University

```
public class XMLElement implements IXMLElement
{
    // The name.
    private String name;

    // The child elements.
    private Vector children;

    // Returns an enumeration of all child elements.
    public Enumeration enumerateChildren() { ... }

    // Returns the number of children.
    public int getChildrenCount() { ... }

    // Removes a child element.
    public void removeChild(IXMLElement child) { ... }

    // More methods and attributes...
}
```

(a) Executable Program

removeChild

$\Delta XMLElement$
$child? : XML\_ELEMENT$

$child? \in enumerateChildren$
$child? \neq null$
$enumerateChildren' = enumerateChildren \setminus child?$
$getChildrenCount' = getChildrenCount - 1$

(b) Specification

```
public void testRemoveChild()
{
    child = element.getChildAtIndex(0);
    element.removeChild(child);
    assertEquals(element.getChildrenCount(),
                 old_getChildrenCount - 1);
}
```

(c) Test

Defect detection since the mined specifications also reveal *undesired* properties: every such property comes with a test case demonstrating it;

Program maintenance as it eases program understanding and change impact assessment: every aspect of the program behavior will be described in a high-level, abstract specification.

executions

Generate test cases to systematically explore behavior

Assess executions to learn about software behavior

specifications

Challenges

Mine specifications that are

Challenges

Mine specifications that are

complete

Challenges

Mine specifications that are

complete	**real**

Challenges

Mine specifications that are

complete	**real**	**proven**

Challenges

Mine specifications that are

complete	real	proven

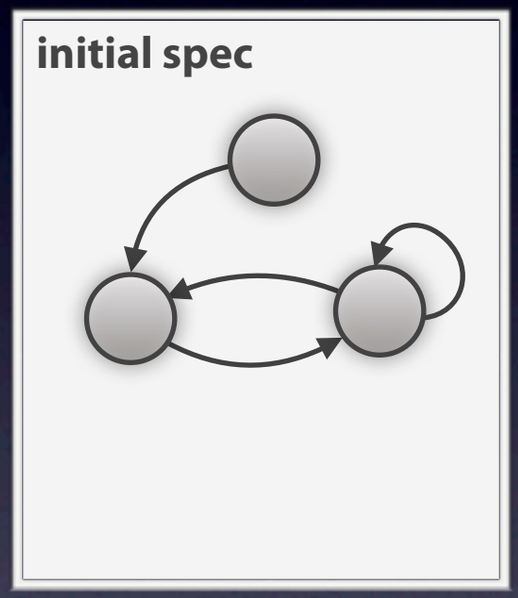
Enriching specifications

Enriching specifications

```
void ProtocolTest() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.quit();
}
```

Enriching specifications

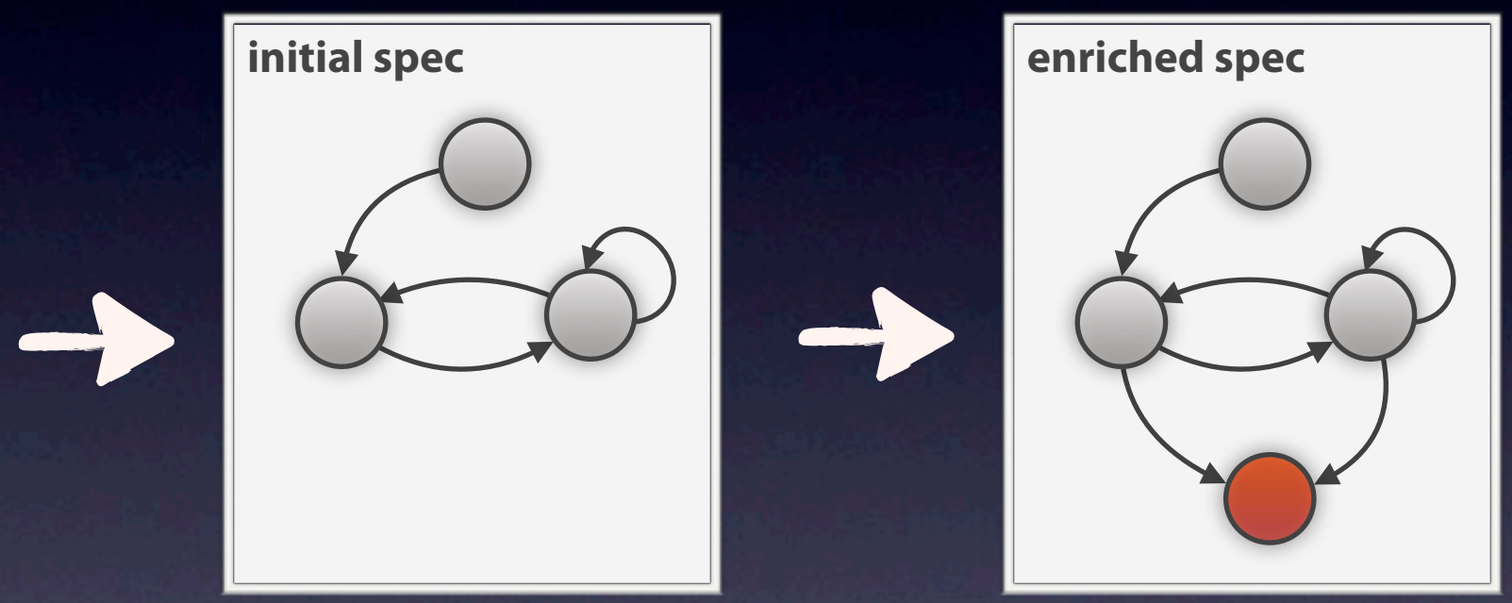
```
void ProtocolTest() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.quit();
}
```



Execute and extract initial spec

Enriching specifications

```
void ProtocolTest() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.quit();
}
```



Execute and extract initial spec

Generate test mutants and enrich specs

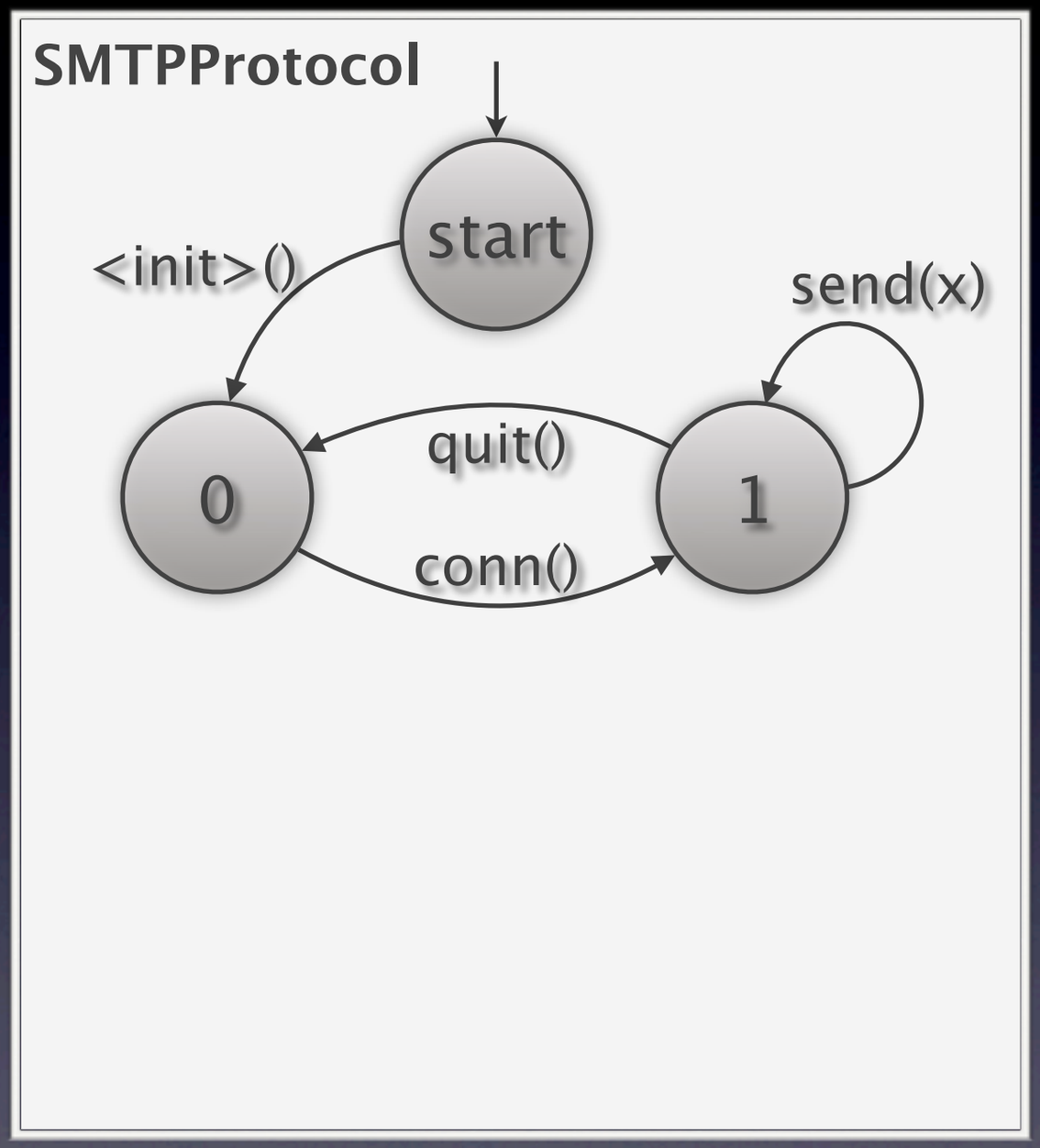
Dallmeier et al: "Generating Test Cases for Specification Mining", ISSTA 2010

Dallmeier et al: “Generating Test Cases for Specification Mining”, ISSTA 2010

```
void ProtocolTest() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.quit();
}
```

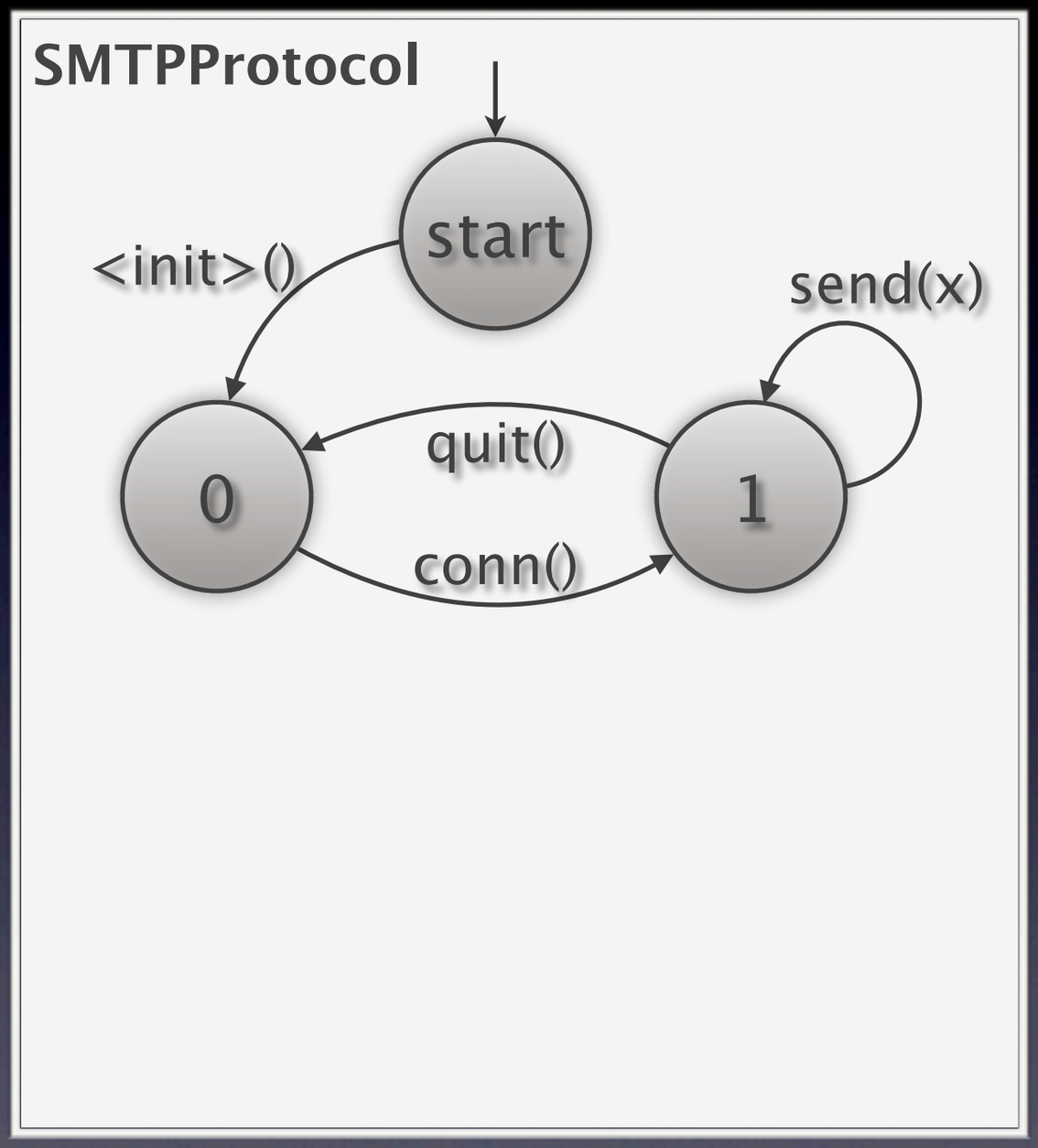
Dallmeier et al: “Generating Test Cases for Specification Mining”, ISSTA 2010

```
void ProtocolTest() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.quit();
}
```



Dallmeier et al: “Generating Test Cases for Specification Mining”, ISSTA 2010

```
void ProtocolTest() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.quit();
}
```



Uncovered

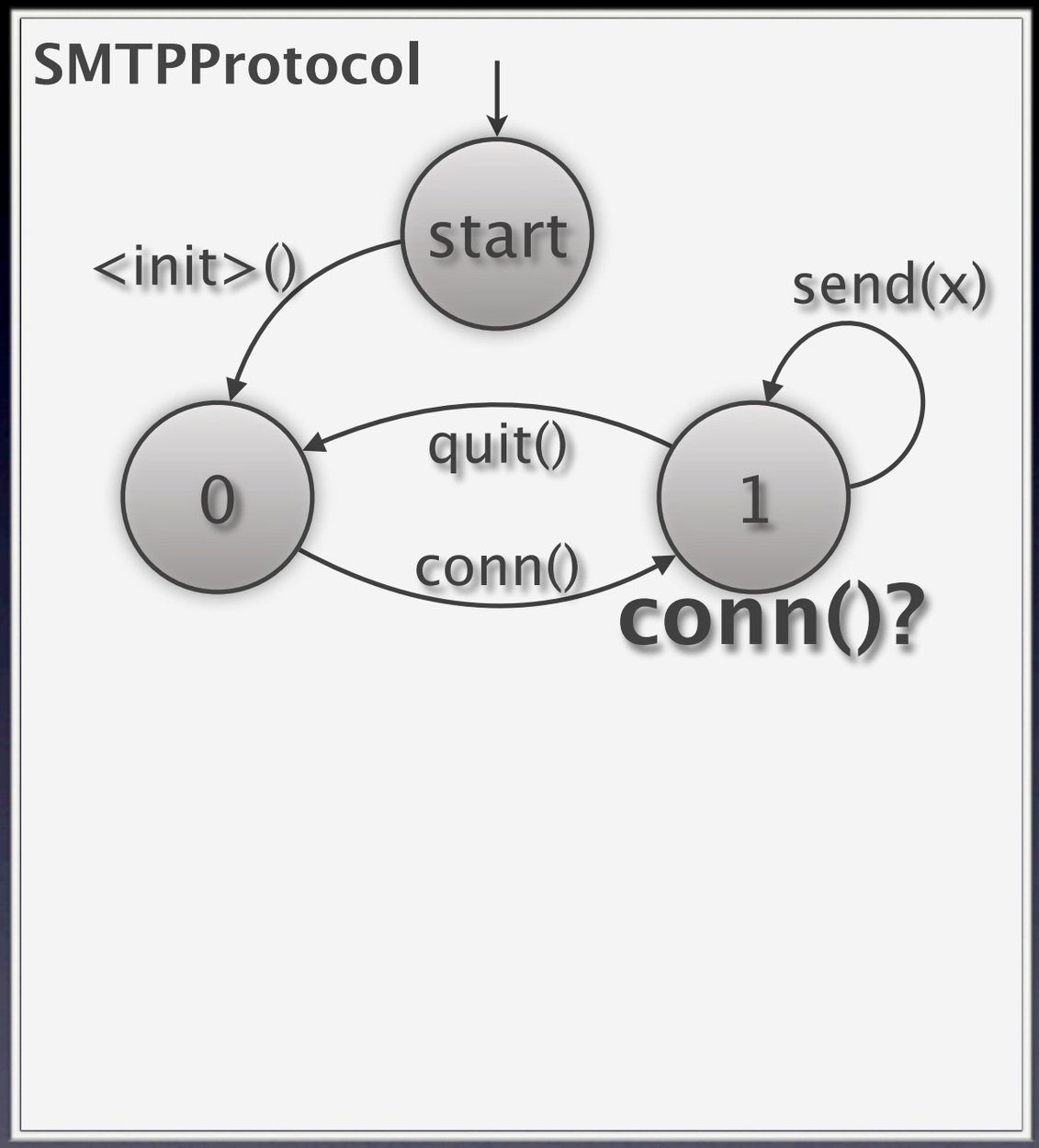
0: send(x)
   quit()
1: conn()

Dallmeier et al: "Generating Test Cases for Specification Mining", ISSTA 2010

```
void ProtocolTest() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.quit();
}
```

Uncovered

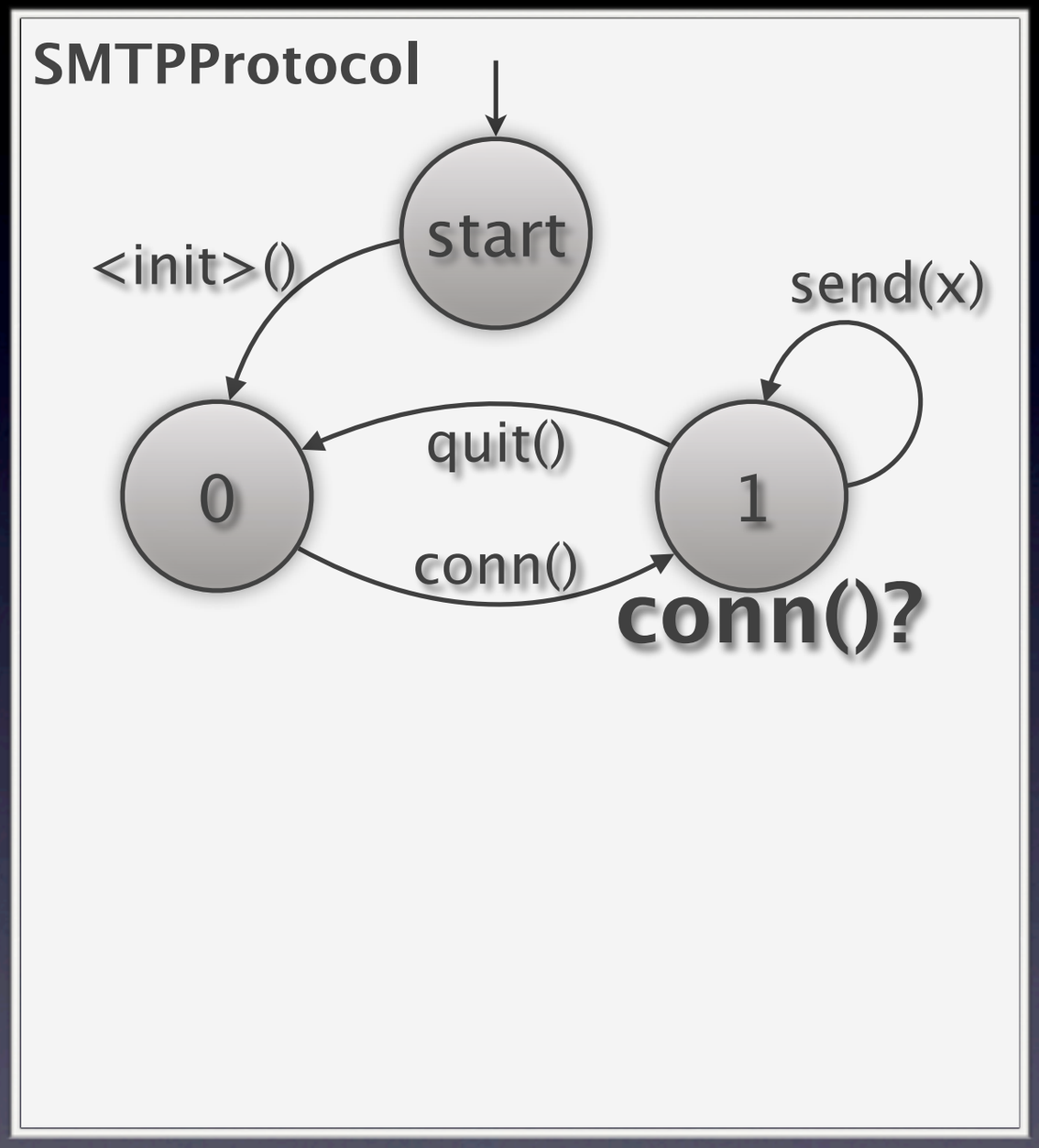
0: send(x)
   quit()
1: conn()



Dallmeier et al: "Generating Test Cases for Specification Mining", ISSTA 2010

```
void ProtocolTest() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.quit();
}
```

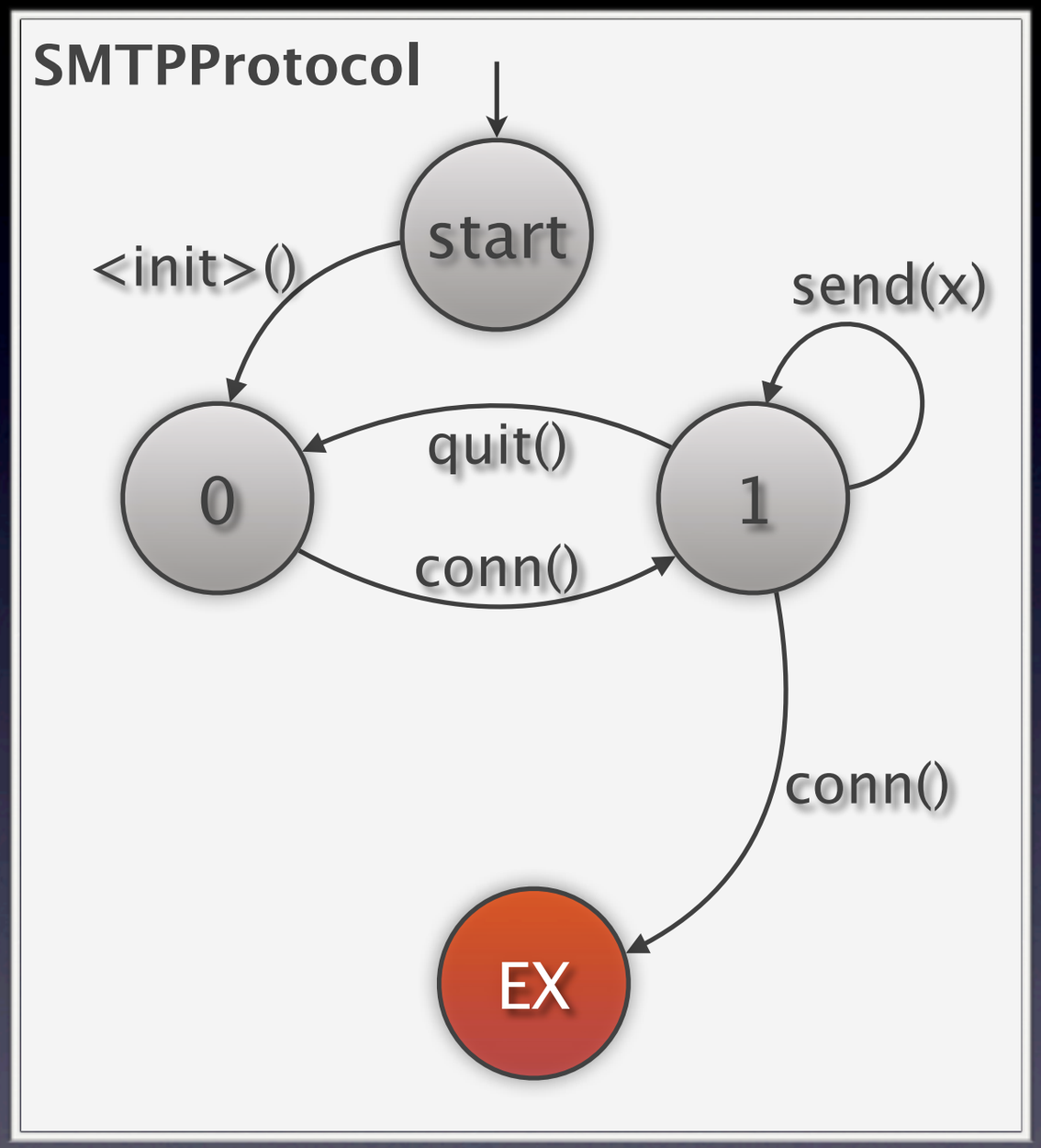
```
void TestMutant1() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.conn();
    p.quit();
}
```



Dallmeier et al: "Generating Test Cases for Specification Mining", ISSTA 2010

```
void ProtocolTest() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.quit();
}
```

```
void TestMutant1() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.conn();
    p.quit();
}
```



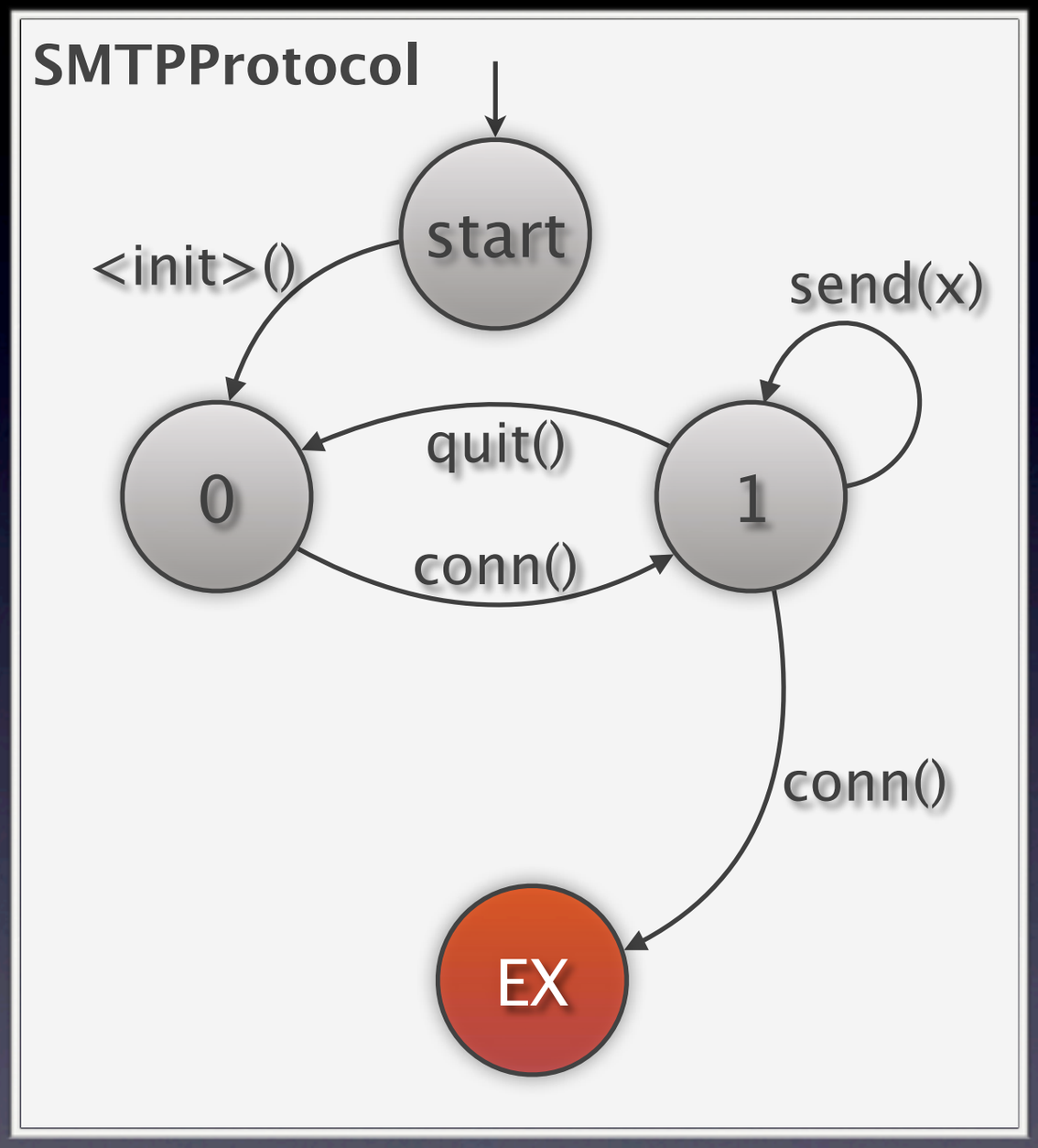
Dallmeier et al: "Generating Test Cases for Specification Mining", ISSTA 2010

```
void ProtocolTest() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.quit();
}
```

Uncovered

0: send(x)
   quit()

~~1: conn()~~



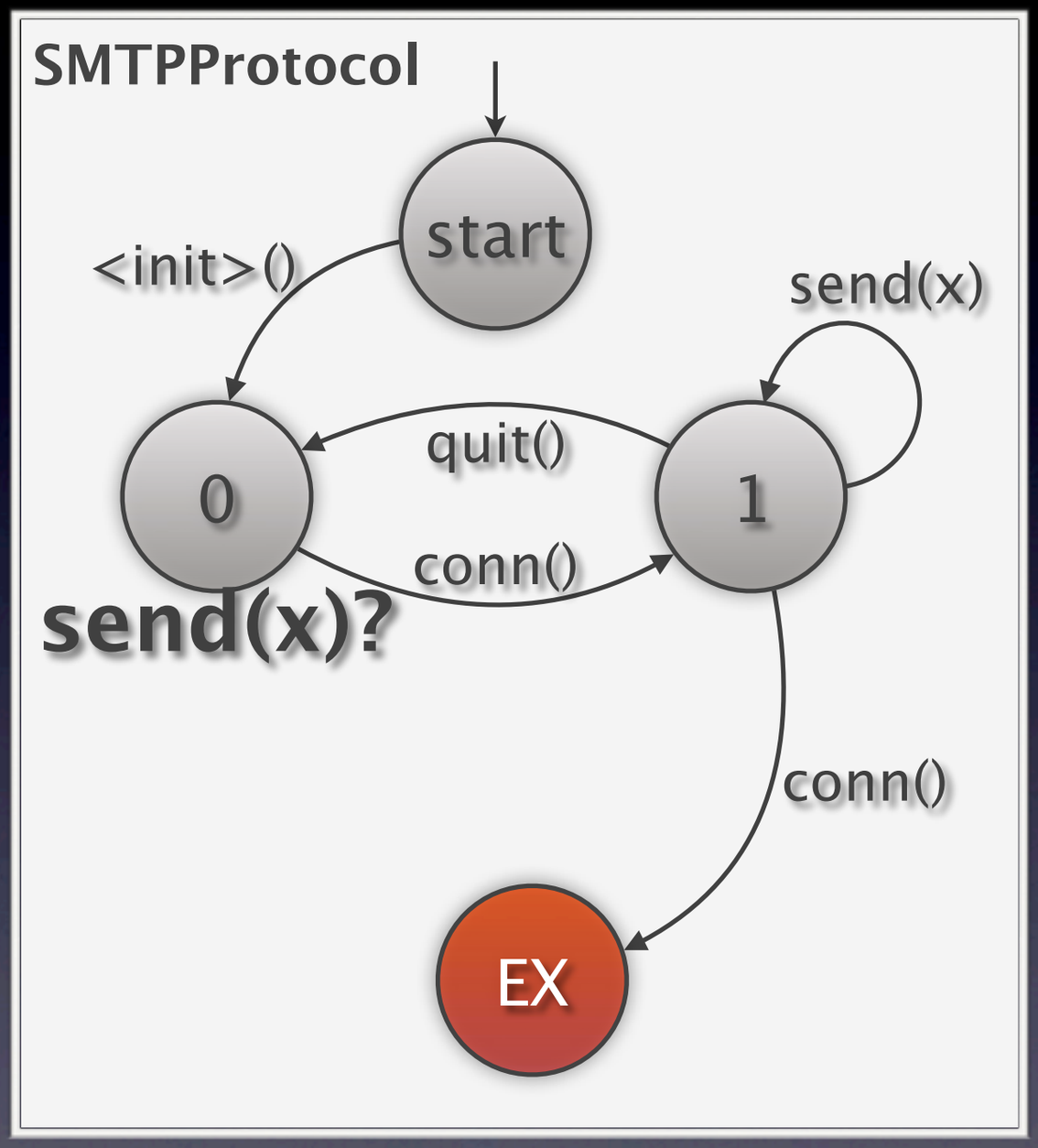
Dallmeier et al: "Generating Test Cases for Specification Mining", ISSTA 2010

```
void ProtocolTest() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.quit();
}
```

Uncovered

0: send(x)
   quit()

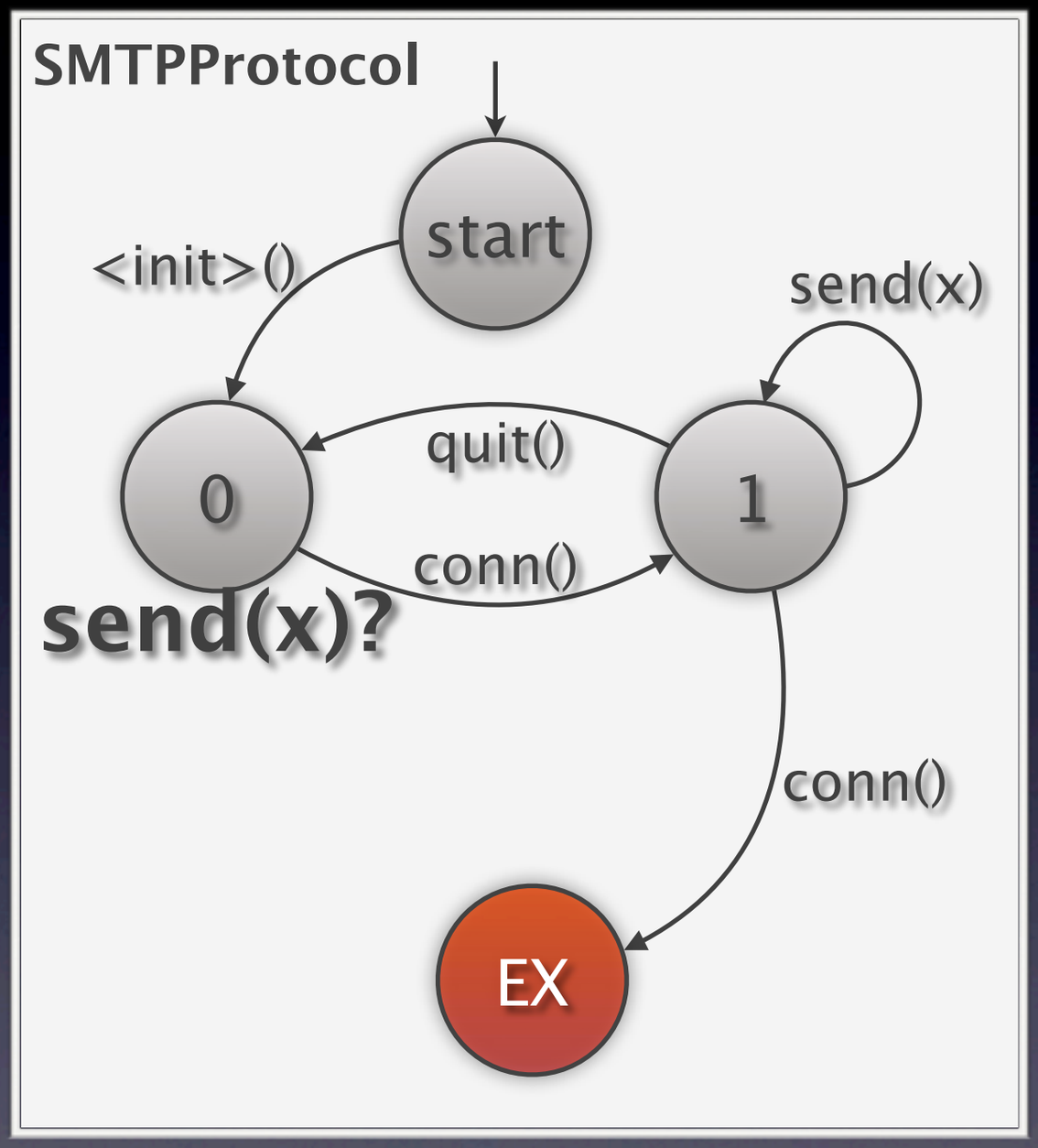
~~1: conn()~~



Dallmeier et al: "Generating Test Cases for Specification Mining", ISSTA 2010

```
void ProtocolTest() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.quit();
}
```

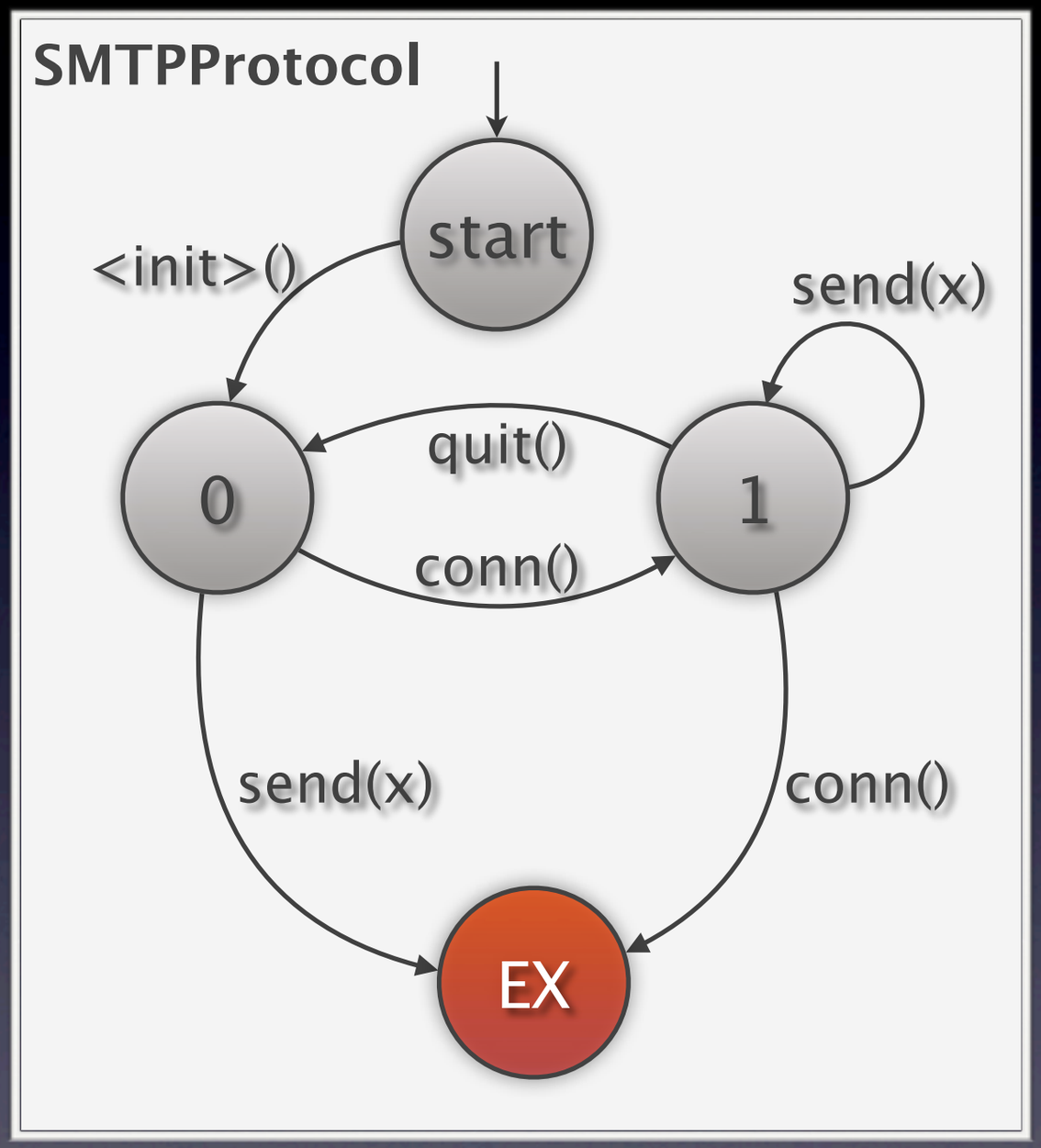
```
void TestMutant2() {
    Protocol p = new ...
    //p.conn();
    p.send(x);
    p.quit();
}
```



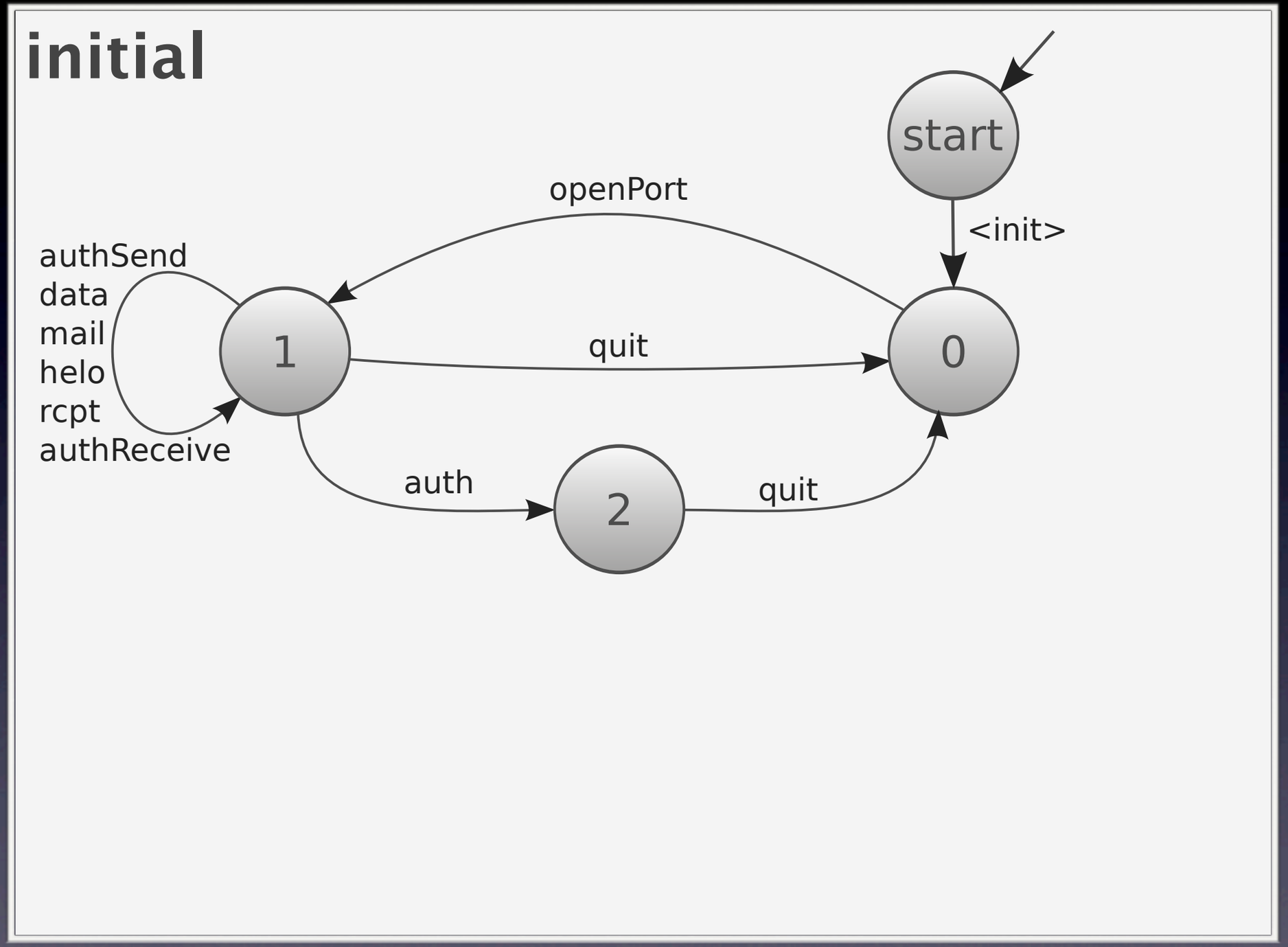
Dallmeier et al: "Generating Test Cases for Specification Mining", ISSTA 2010

```
void ProtocolTest() {
    Protocol p = new ...
    p.conn();
    p.send(x);
    p.quit();
}
```

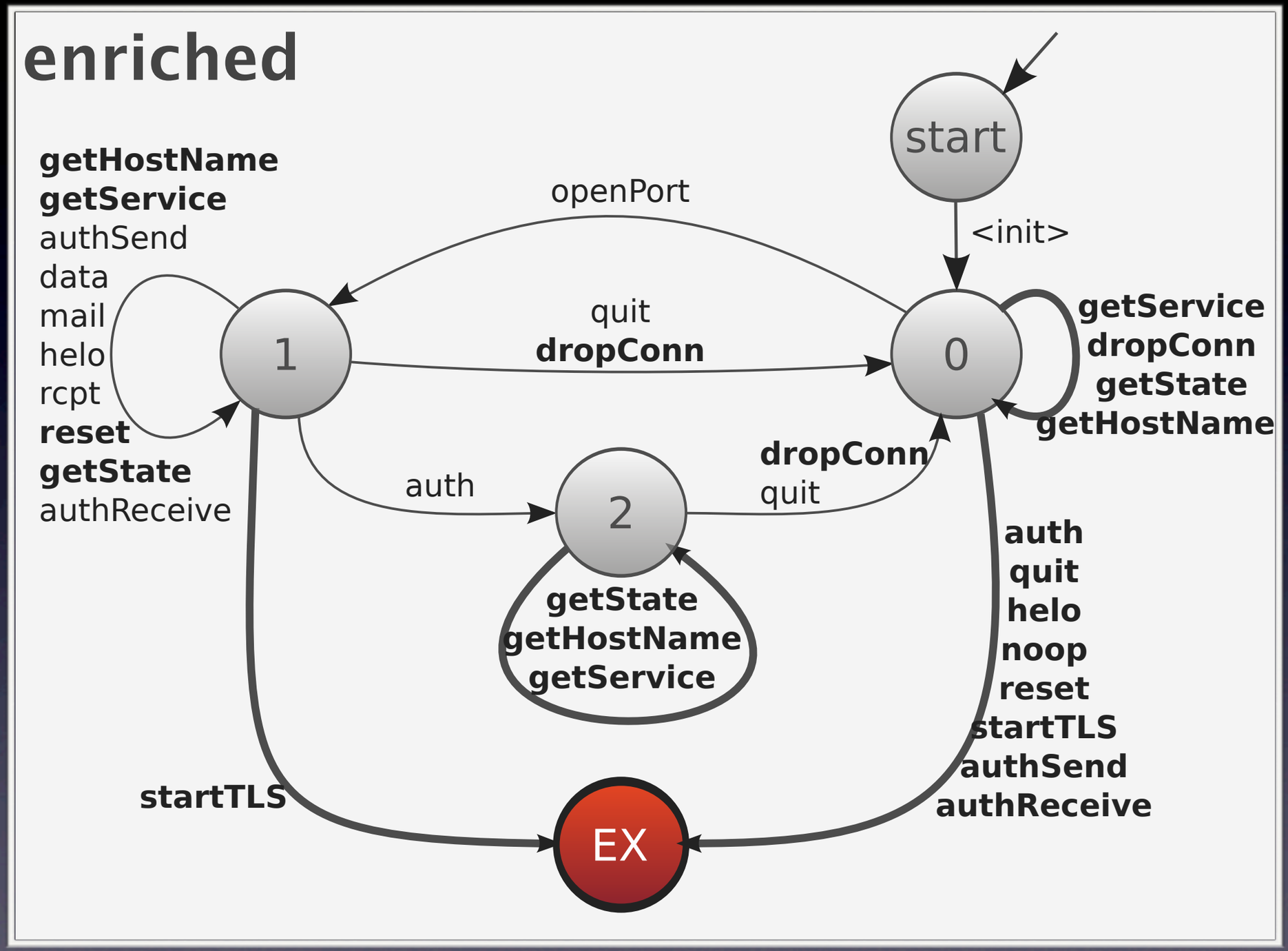
```
void TestMutant2() {
    Protocol p = new ...
    //p.conn();
    p.send(x);
    p.quit();
}
```



Dallmeier et al: "Generating Test Cases for Specification Mining", ISSTA 2010



Dallmeier et al: “Generating Test Cases for Specification Mining”, ISSTA 2010



Dallmeier et al: "Generating Test Cases for Specification Mining", ISSTA 2010

Challenges

Mine specifications that are

complete	real	proven

Challenges

Mine specifications that are

complete	real	proven

Address Book

New contact

New category

First name	Last name	E-mail	Phone	Mobile
James S.	Roebuck	JamesSRoe...	561-888-...	561-888-...
Naomi D.	Long	NaomiDLo...	390-12-5...	390-12-1...
Karen L.	**Lloyd**	**KarenLLlo...**	**228-76-1...**	**228-76-...**
Jean R.	Voigt	JeanRVoigt...	610-344-...	610-344-...
Douglas L.	Green	DouglasLG...	612-615-...	612-615-...

- All
  - Contractors
  - Customers
  - Employees
- Suppliers
  - Europe
  - U.S.

First name: Karen L.

Last name: Lloyd

Phone: 228-76-1230

Mobile: 228-76-8710

E-Mail: KarenLLloyd@ex

Second e-mail: Karen@CreditCa

URL: http://www.crec

Apply

Notes:
1673 Jehovah Drive
Fredericksburg, VA 22408

Random Testing

```
public class RandoopTest0 extends TestCase {
  ...

  public void test8() throws Throwable {
    if (debug) System.out.printf("%nRandoopTest0.test8");

    AddressBook var0 = new AddressBook();
    EventHandler var1 = var0.getEventHandler();
    Category var2 = var0.getRootCategory();
    Contact var3 = new Contact();
    AddressBook var4 = new AddressBook();
    EventHandler var5 = var4.getEventHandler();
    Category var6 = var4.getRootCategory();
    String var7 = var6.getName();
    var0.addCategory(var3, var6);
    SelectionHandler var9 = new SelectionHandler();
    AddressBook var10 = new AddressBook();
    EventHandler var11 = var10.getEventHandler();
```

```
MainWindow var64 = new MainWindow(var0);
AddressBook var65 = new AddressBook();
EventHandler var66 = var65.getEventHandler();
Category var67 = var65.getRootCategory();
Contact var68 = new Contact();
Category[] var69 = var68.getCategories();
var65.removeContact(var68);
java.util.List var71 = var65.getContacts();
AddressBook var72 = new AddressBook();
EventHandler var73 = var72.getEventHandler();
Category var74 = var72.getRootCategory();
EventHandler var75 = var72.getEventHandler();
SelectionHandler var76 = new SelectionHandler();
actions.CreateContactAction var77 = new actions.CreateContactAction(var72, var76);
boolean var78 = var77.isEnabled();
AddressBook var79 = new AddressBook();
EventHandler var80 = var79.getEventHandler();
Category var81 = var79.getRootCategory();
String var82 = var81.getName();
var77.categorySelected(var81);
Category var85 = var65.createCategory(var81, "hi!");
String var86 = var85.toString();
Category var88 = var0.createCategory(var85, "exceptions.NameAlreadyInUseException");
}
```

```
MainWindow var64 = new MainWindow(var0);
AddressBook var65 = new AddressBook();
EventHandler var66 = var65.getEventHandler();
Category var67 = var65.getRootCategory();
Contact var68 = new Contact();
Category[] var69 = var68.getCategories();
var65.removeContact(var68);
java.util.List var71 = var65.getContacts();
AddressBook var72 = new AddressBook();
EventHandler var73 = var72.getEventHandler();
Category var74 = var72.getRootCategory();
EventHandler var75 = var72.getEventHandler();
SelectionHandler var76 = new SelectionHandler();
```

```
actions.CreateContactAction var77 = new actions.CreateContactAction(var72, var76);
boolean var78 = var77.isEnabled();
AddressBook var79 = new AddressBook();
EventHandler var80 = var79.getEventHandler();
Category var81 = var79.getRootCategory();
String var82 = var81.getName();
var77.categorySelected(var81);
Category var85 = var65.createCategory(var81, "hi!");
String var86 = var85.toString();
Category var88 = var0.createCategory(var85, "exceptions.NameAlreadyInUseException");
}
```

Simplified Test Case

```
public class RandoopTest0 extends TestCase {
  public void test8() throws Throwable {
    if (debug) System.out.printf("%nRandoopTest0.test8");

    AddressBook a1 = new AddressBook();
    AddressBook a2 = new AddressBook();
    Category a1c = a1.createCategory(a1.getRootCategory(), "a1c");
    Category a2c = a2.createCategory(a1c, "a2c");
  }
}
```

Address Book

New contact

New category

First name	Last name	E-mail	Phone	Mobile
James S.	Roebuck	JamesSRoe...	561-888-...	561-888-...
Naomi D.	Long	NaomiDLo...	390-12-5...	390-12-1...
Karen L.	**Lloyd**	**KarenLLlo...**	**228-76-1...**	**228-76-...**
Jean R.	Voigt	JeanRVoigt...	610-344-...	610-344-...
Douglas L.	Green	DouglasLG...	612-615-...	612-615-...

- All
  - Contractors
  - Customers
  - Employees
- Suppliers
  - Europe
  - U.S.

First name: Karen L.

Last name: Lloyd

Phone: 228-76-1230

Mobile: 228-76-8710

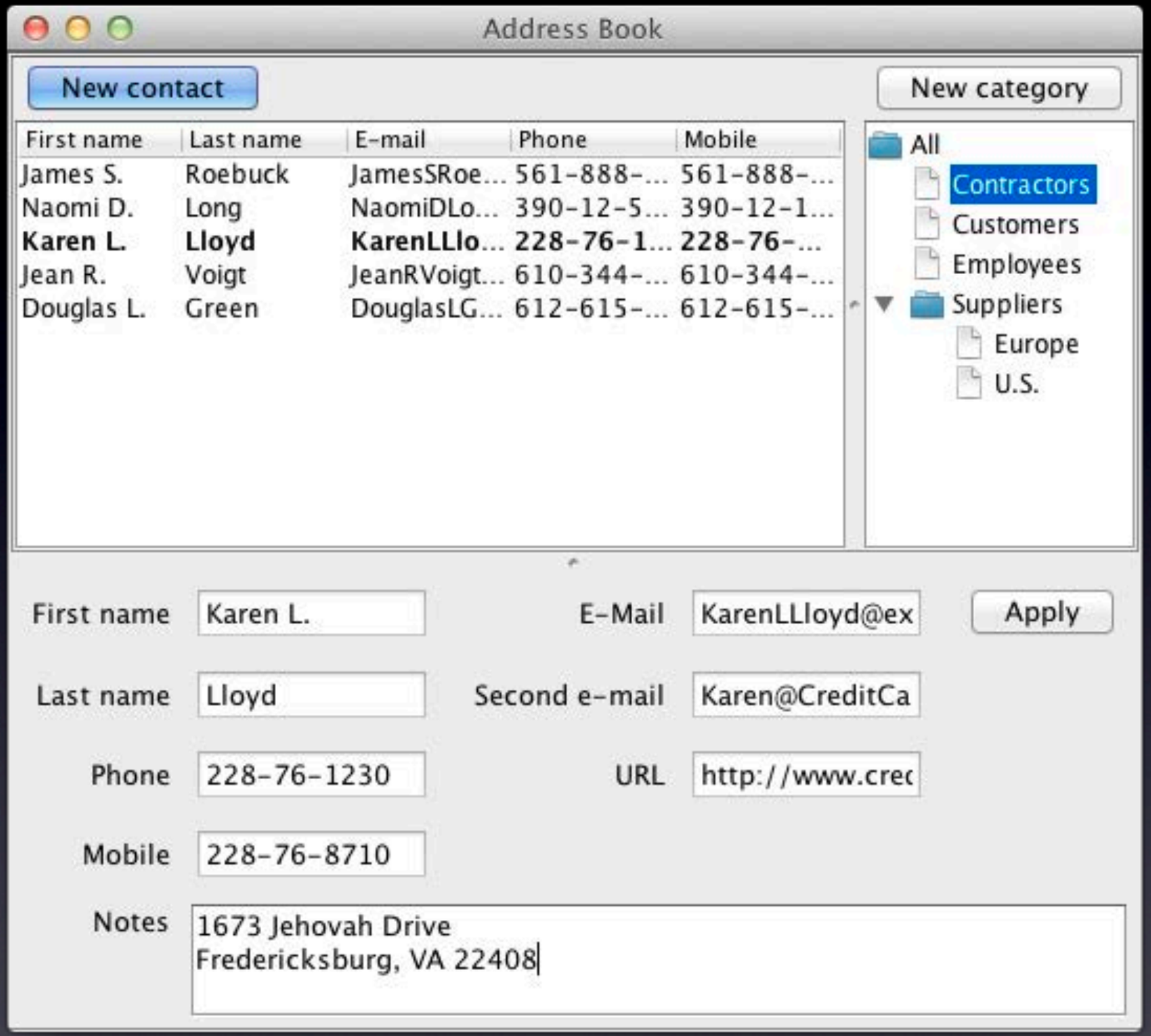
E-Mail: KarenLLloyd@ex

Second e-mail: Karen@CreditCa

URL: http://www.crec

Apply

Notes:
1673 Jehovah Drive
Fredericksburg, VA 22408



how many addressbooks?

Address Book

New contact

New category

First name	Last name	E-mail	Phone	Mobile
James S.	Roebuck	JamesSRoe...	561-888-...	561-888-...
Naomi D.	Long	NaomiDLo...	390-12-5...	390-12-1...
Karen L.	**Lloyd**	**KarenLLlo...**	**228-76-1...**	**228-76-...**
Jean R.	Voigt	JeanRVoigt...	610-344-...	610-344-...
Douglas L.	Green	DouglasLG...	612-615-...	612-615-...

- All
  - Contractors
  - Customers
  - Employees
- Suppliers
  - Europe
  - U.S.

First name: Karen L.

Last name: Lloyd

Phone: 228-76-1230

Mobile: 228-76-8710

E-Mail: KarenLLloyd@ex

Second e-mail: Karen@CreditCa

URL: http://www.crec

Apply

Notes:
1673 Jehovah Drive
Fredericksburg, VA 22408

Address Book

New contact

New category

First name	Last name	E-mail	Phone	Mobile
James S.	Roebuck	JamesSRoe...	561-888-...	561-888-...
Naomi D.	Long	NaomiDLo...	390-[illegible]	390-[illegible]
Karen L.	**Lloyd**	**KarenLLlo...**	**228-76-...**	**228-76-...**
Jean R.	Voigt	JeanR[illegible]	610-344-...	610-344-...
Douglas L.	Green	[illegible]	612-615-...	612-615-...

- All
  - Contractors
  - Customers
  - Employees
- Suppliers
  - Europe
  - U.S.

112 FAILURES

First name: Karen L.

Last name: Lloyd

Phone: 228-76-1230

Mobile: 228-76-8710

E-Mail: KarenLLloyd@ex

Second e-mail: Karen@CreditCa

URL: http://www.crec

Apply

Notes: 1673 Jehovah Drive
Fredericksburg, VA 22408

Address Book

New contact

New category

First name	Last name	E-mail	Phone	
James S.	Roebuck	JamesSRoe...	561-888-...	561-888-...
Naomi D.	Long	NaomiDLo...	390-[illegible]	390-[illegible]
Karen L.	**Lloyd**	**KarenLLlo...**	**228-76-...**	**228-76-...**
Jean R.	Voigt	JeanR[illegible]	610-344-...	610-344-...
Douglas L.	Green	[illegible]	612-615-...	612-615-...

- All
  - Contractors
  - Customers
  - Employees
- Suppliers
  - Europe
  - U.S.

112 FAILURES

First name: Karen L.

Last name: Lloyd

Phone: 228-76-1230

Mobile: 228-76-8710

E-Mail: KarenLLloyd@[illegible]

Second e-mail: [illegible]

Notes: 1673 [illegible] Dri[illegible]
Frederi[illegible]rg, VA 22408

0 PROBLEMS

Search-based System Testing

Florian Gross, Gordon Fraser, Andreas Zeller • ICSE 2012, ISSTA 2012

Search-based System Testing

Florian Gross, Gordon Fraser, Andreas Zeller • ICSE 2012, ISSTA 2012

- Generate tests at the user interface level

Search-based System Testing

Florian Gross, Gordon Fraser, Andreas Zeller • ICSE 2012, ISSTA 2012

- Generate tests at the user interface level
- Aim for *code coverage* and *GUI coverage*

Search-based System Testing

Florian Gross, Gordon Fraser, Andreas Zeller • ICSE 2012, ISSTA 2012

- Generate tests at the user interface level
- Aim for *code coverage* and *GUI coverage*
- Synthesize artificial input events

Search-based System Testing

Florian Gross, Gordon Fraser, Andreas Zeller • ICSE 2012, ISSTA 2012

- Generate tests at the user interface level
- Aim for *code coverage* and *GUI coverage*
- Synthesize artificial input events
- Any test generated is a valid input

Coverage Progress

0.00 %

Coverage Compared

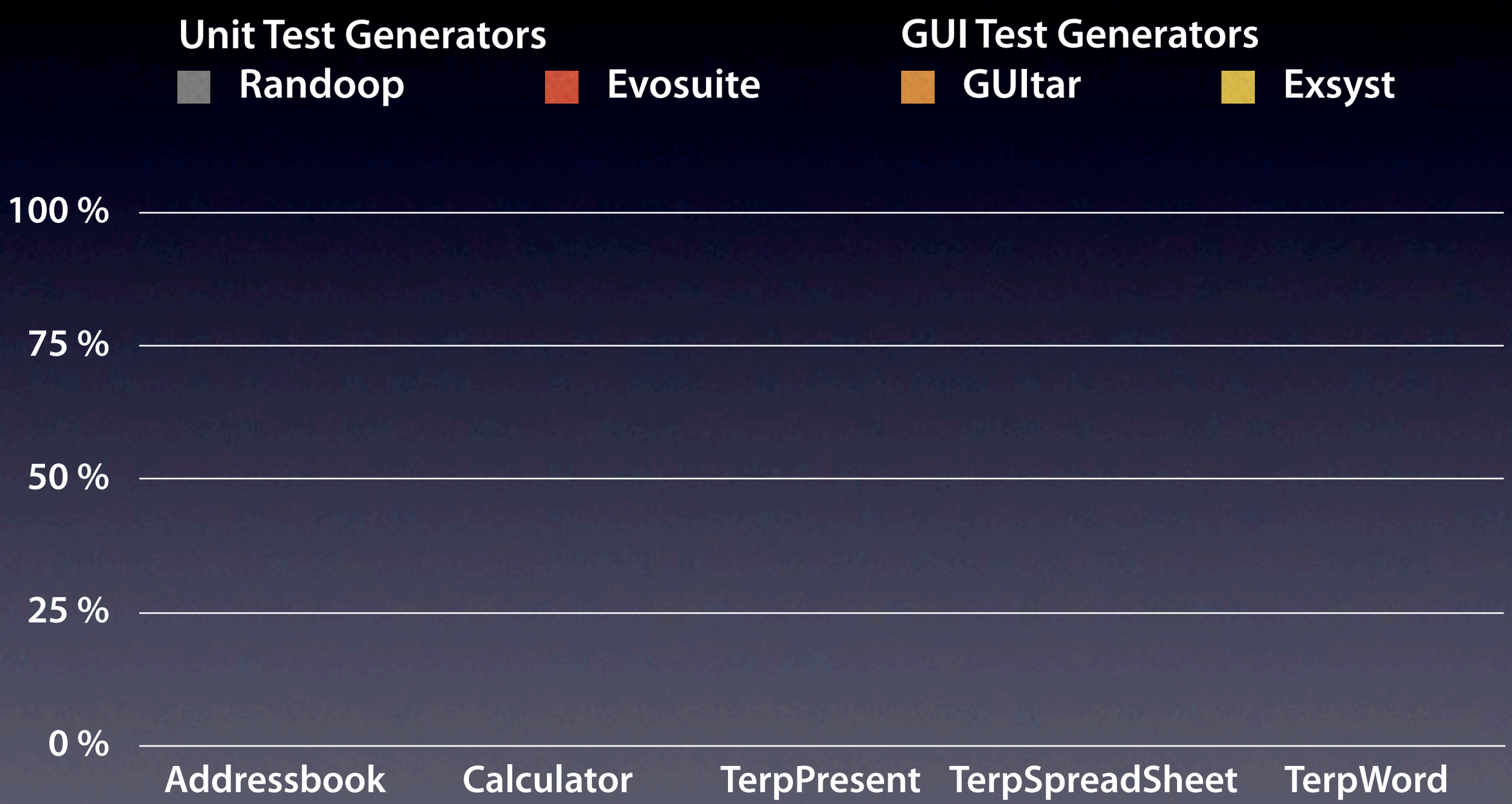
Unit Test Generators

- Randoop
- Evosuite

GUI Test Generators

- GUItar
- Exsyst

Coverage Compared



Coverage Compared

Unit Test Generators

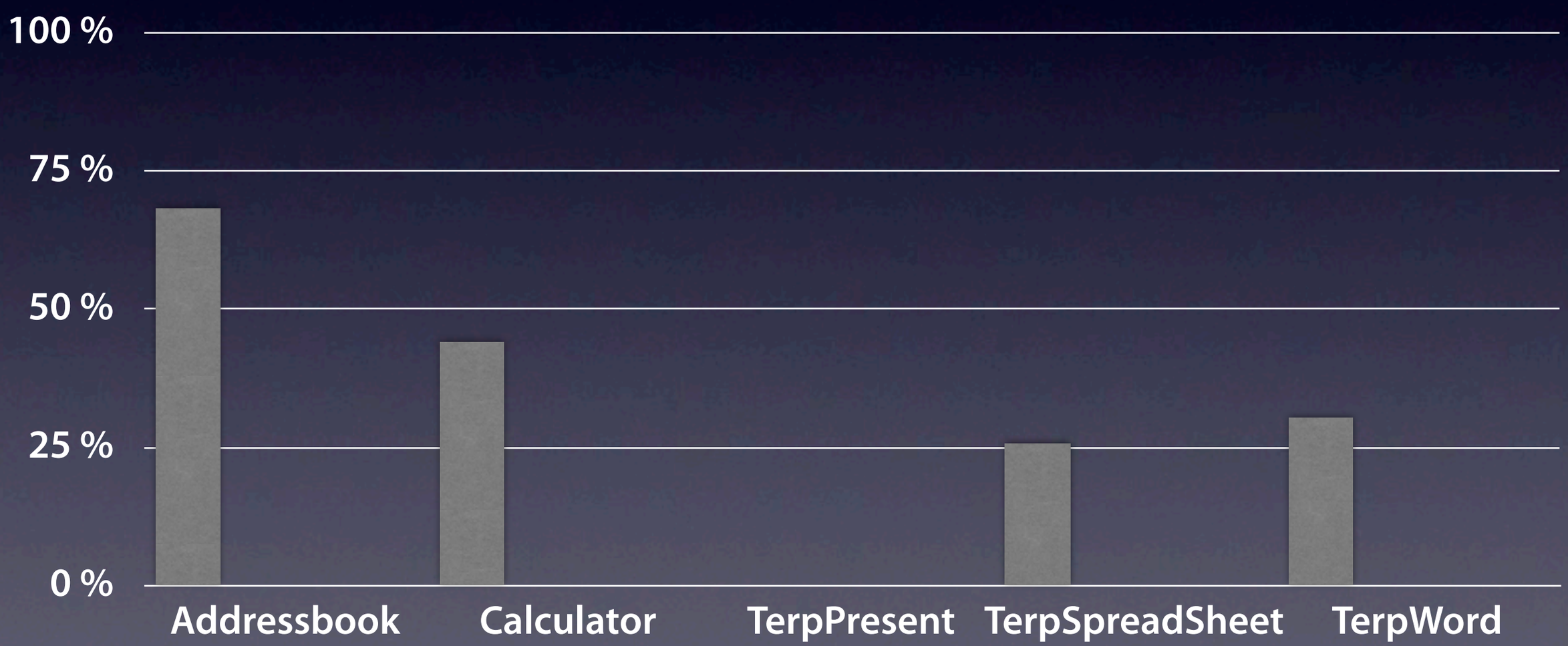
Randoop

Evosuite

GUI Test Generators

GUItar

Exsyst



Coverage Compared

Unit Test Generators

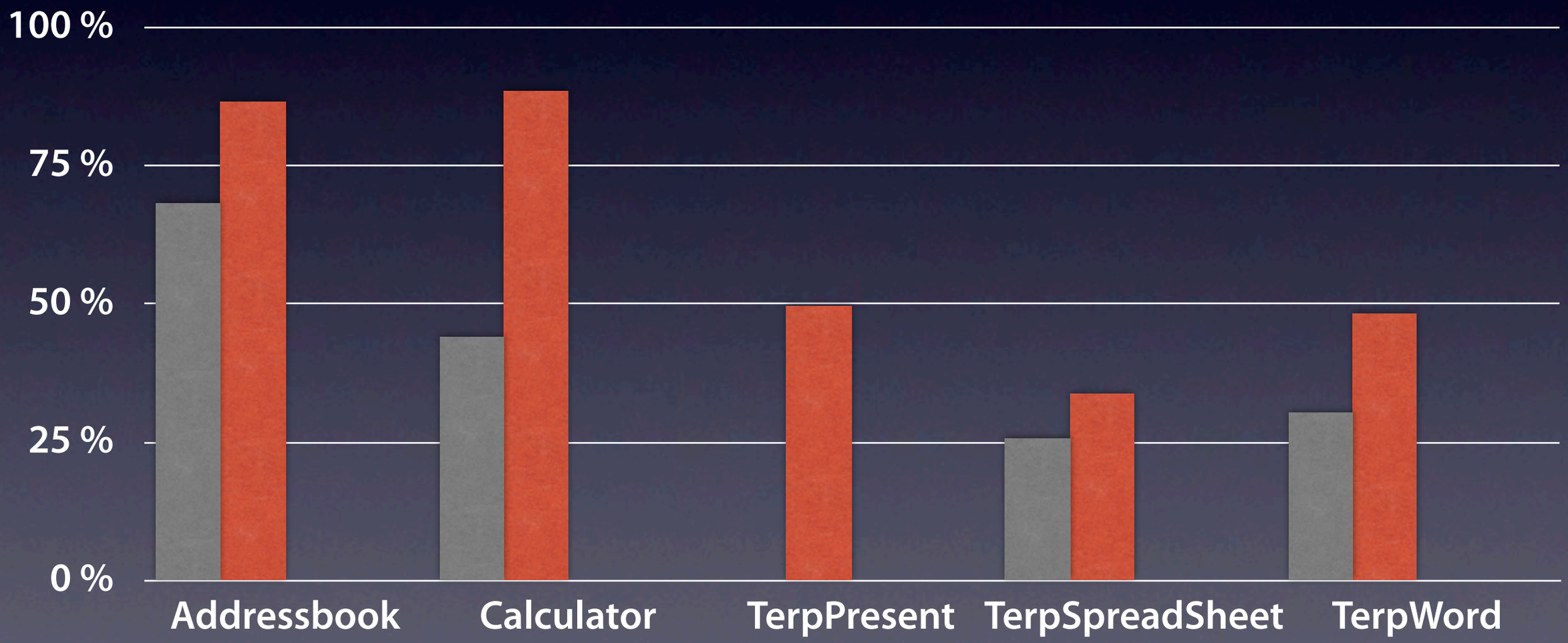
Randoop

Evosuite

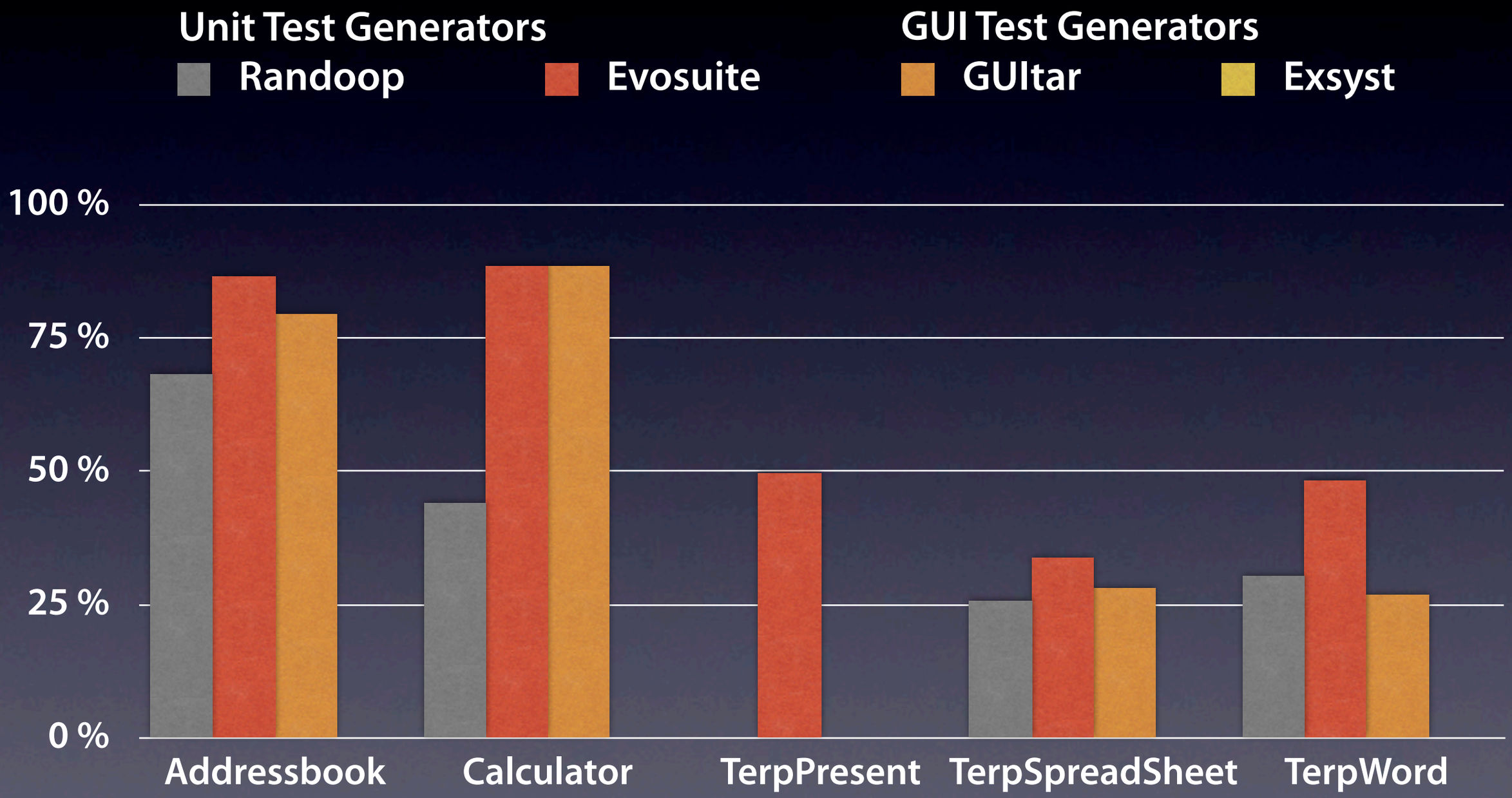
GUI Test Generators

GUItar

Exsyst



Coverage Compared



Coverage Compared

Unit Test Generators

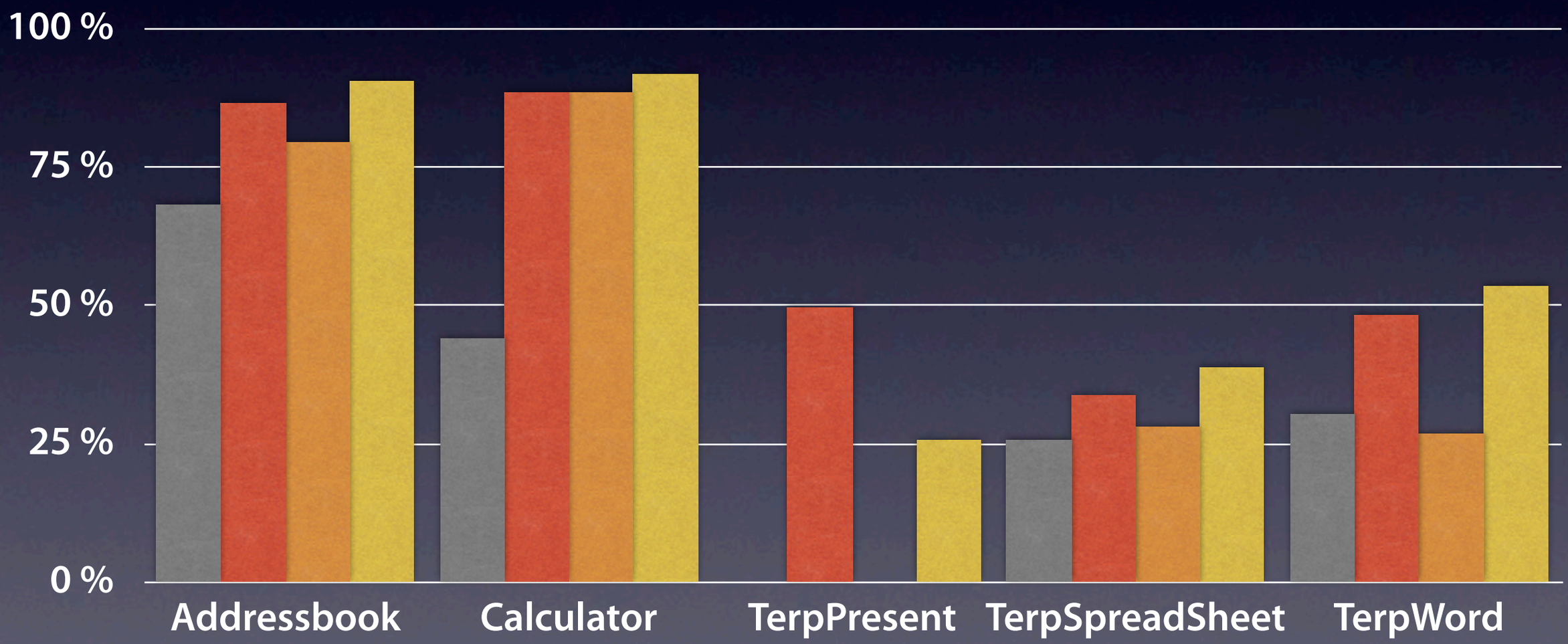
Randoop

Evosuite

GUI Test Generators

GUItar

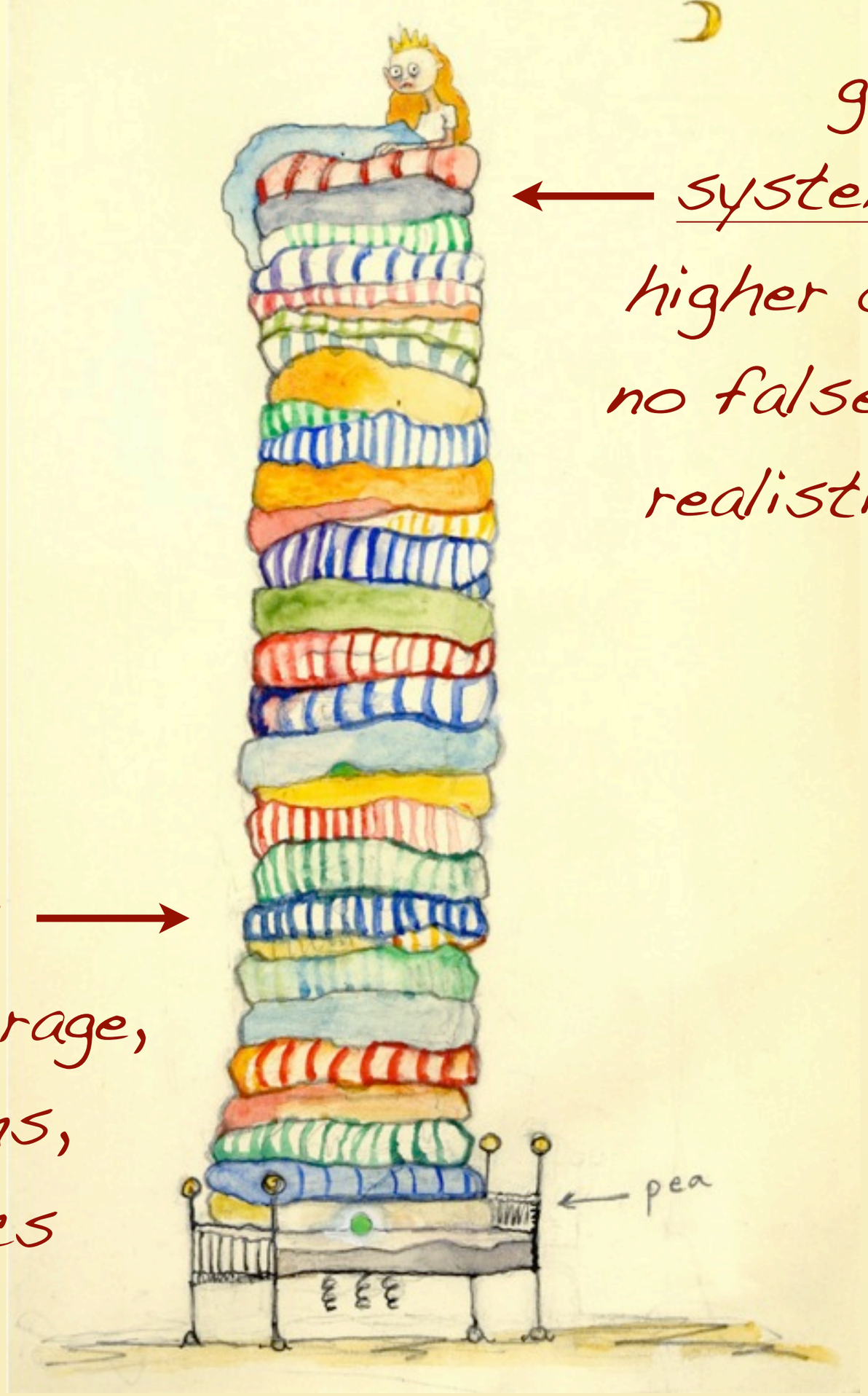
Exsyst




pea

generating
system tests:
higher coverage,
no false alarms,
realistic specs

generating
unit tests:
lower coverage,
false alarms,
fuzzy specs



WEBMATE

Martin Burger, Valentin Dallmeier, Andreas Zeller

Willkommen bei Facebook - anmelden, registrieren oder mehr erfahren - Mozilla Firefox

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

Willkommen bei Facebook - anmelden, regist...

www.facebook.com/index.php

facebook

E-Mail

Passwort

Anmelden

Angemeldet bleiben

Passwort vergessen?

Facebook ermöglicht es dir, mit den Menschen in deinem Leben in Verbindung zu treten und Inhalte mit diesen zu teilen.

Registrieren

Facebook ist und bleibt kostenlos.

Vorname:

Nachname:

Deine Email-Adresse:

E-Mail nochmals eingeben:

Neues Passwort:

Ich bin: Geschlecht auswählen:

Geburtstag: Tag: Monat: Jahr:

Warum muss ich meinen Geburtstag angeben?

Wenn du auf „Registrieren" klickst, akzeptierst du unsere Nutzungsbedingungen und erklärst unsere Datenverwendungsrichtlinien gelesen und verstanden zu haben.

Registrieren

Erstelle eine Seite für eine Berühmtheit, eine ...oder ein

WebDriver

Challenges

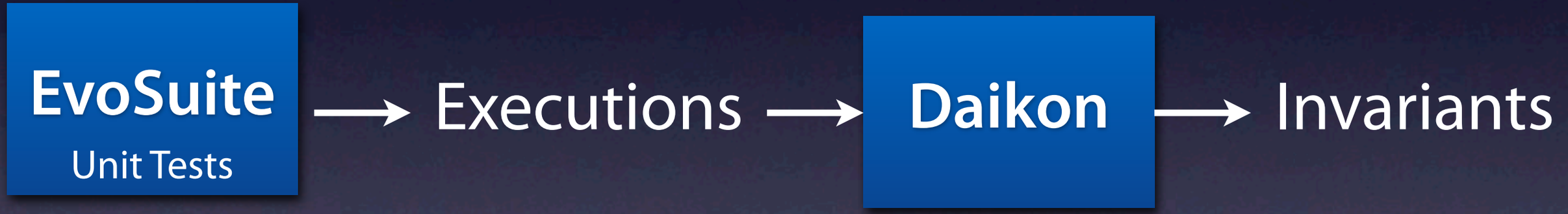
Mine specifications that are

complete	real	proven

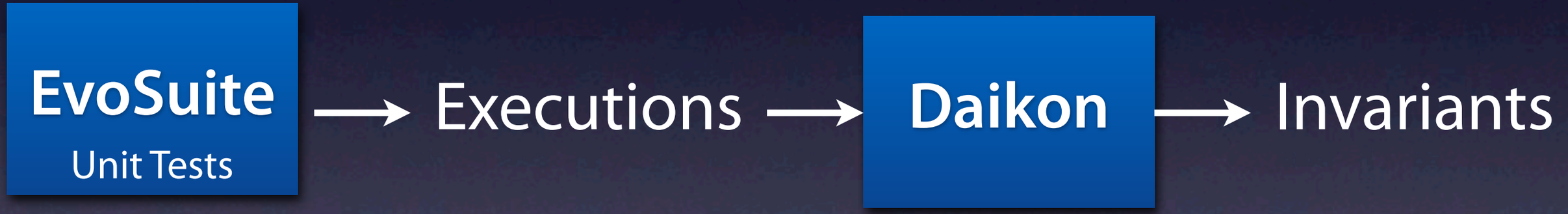
Brand new!

Carving Invariants

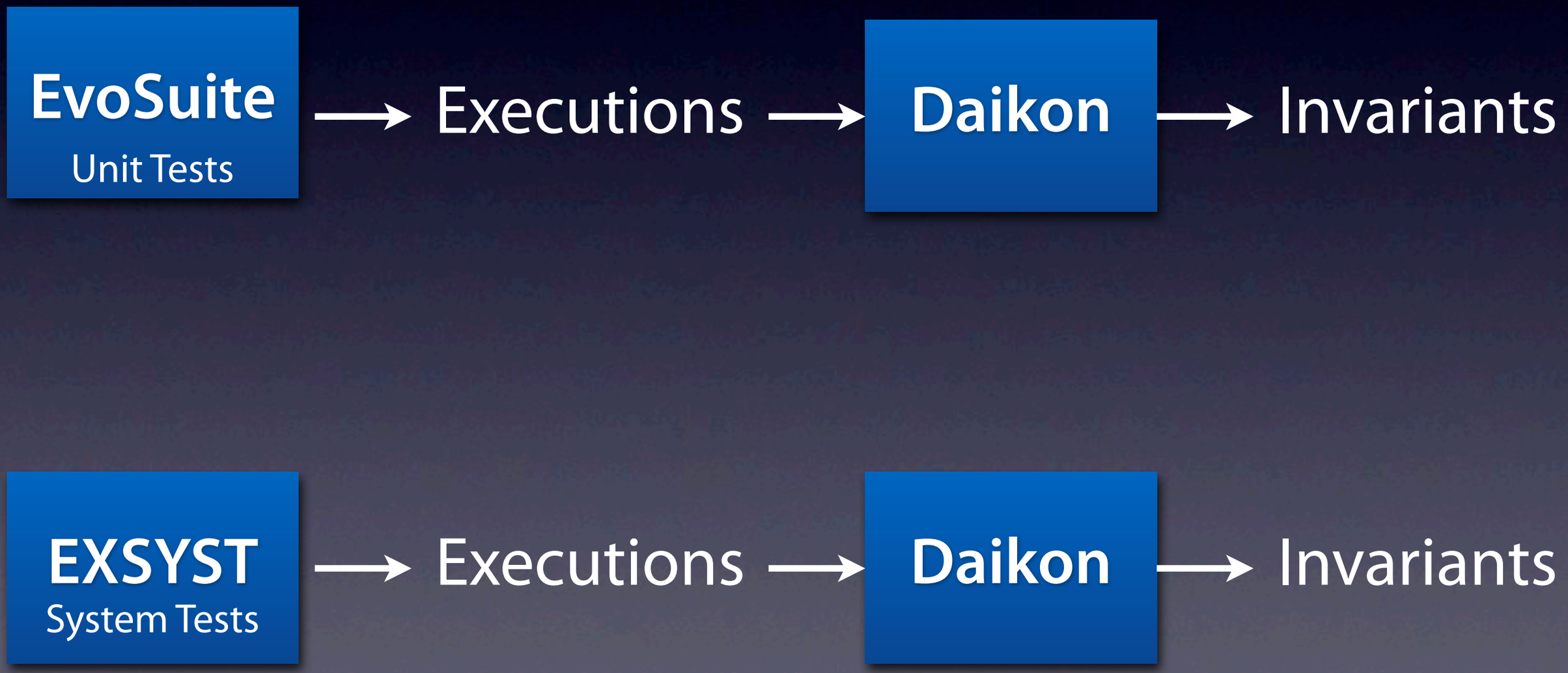
Carving Invariants



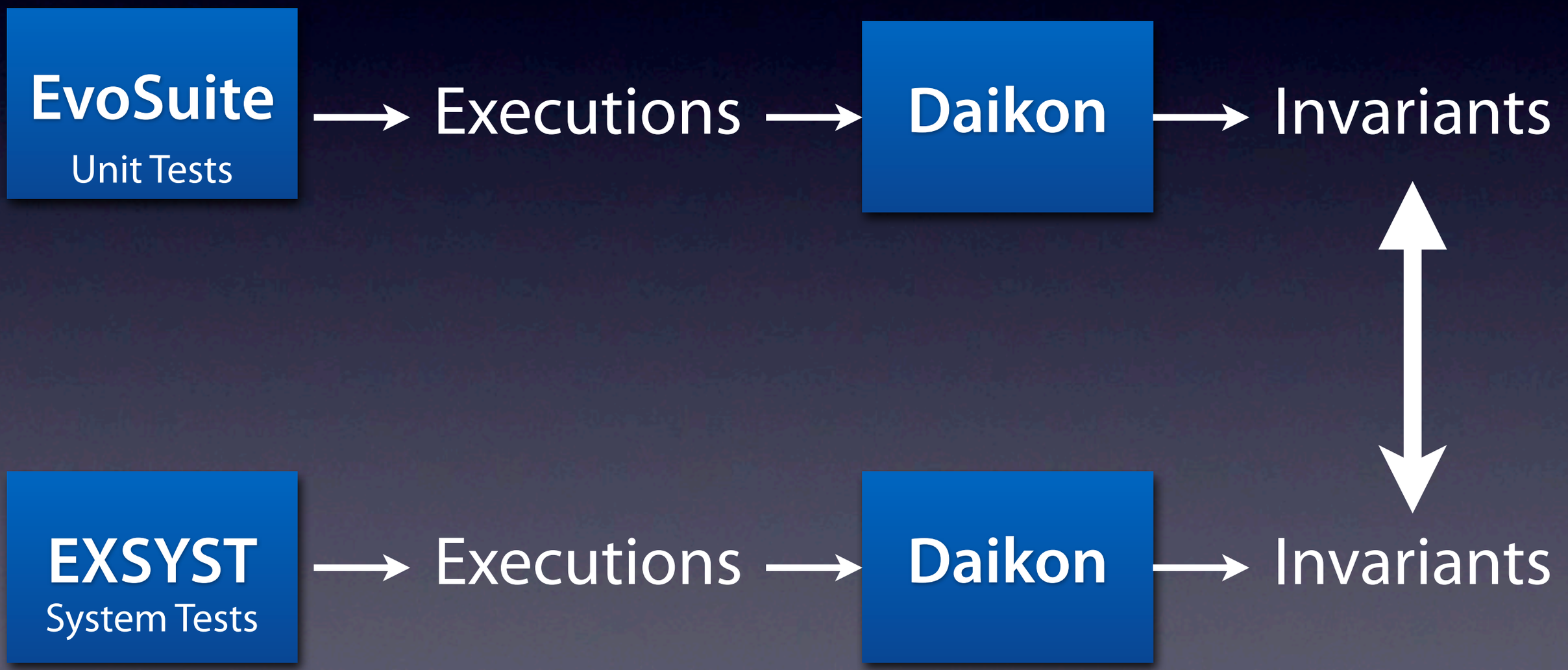
Carving Invariants



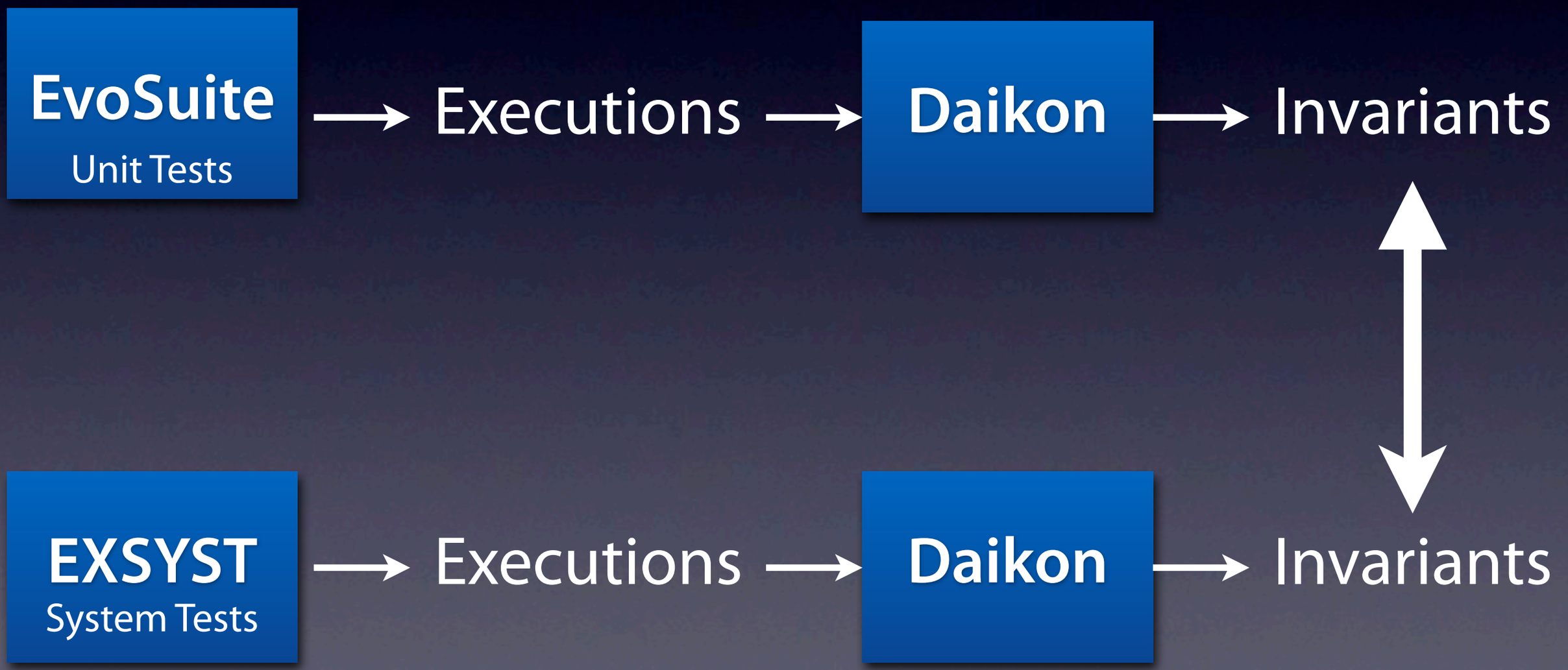
Carving Invariants



Carving Invariants

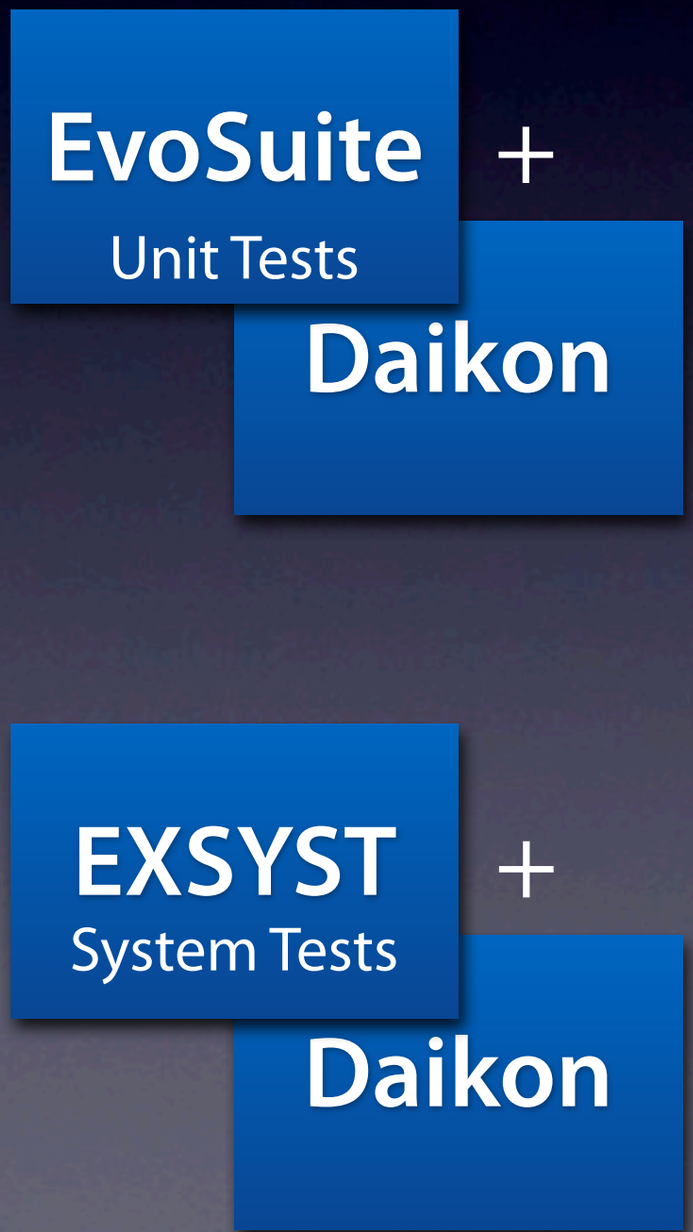


Carving Invariants



Invariants Compared

Florian Gross, Andreas Zeller



Invariants Compared

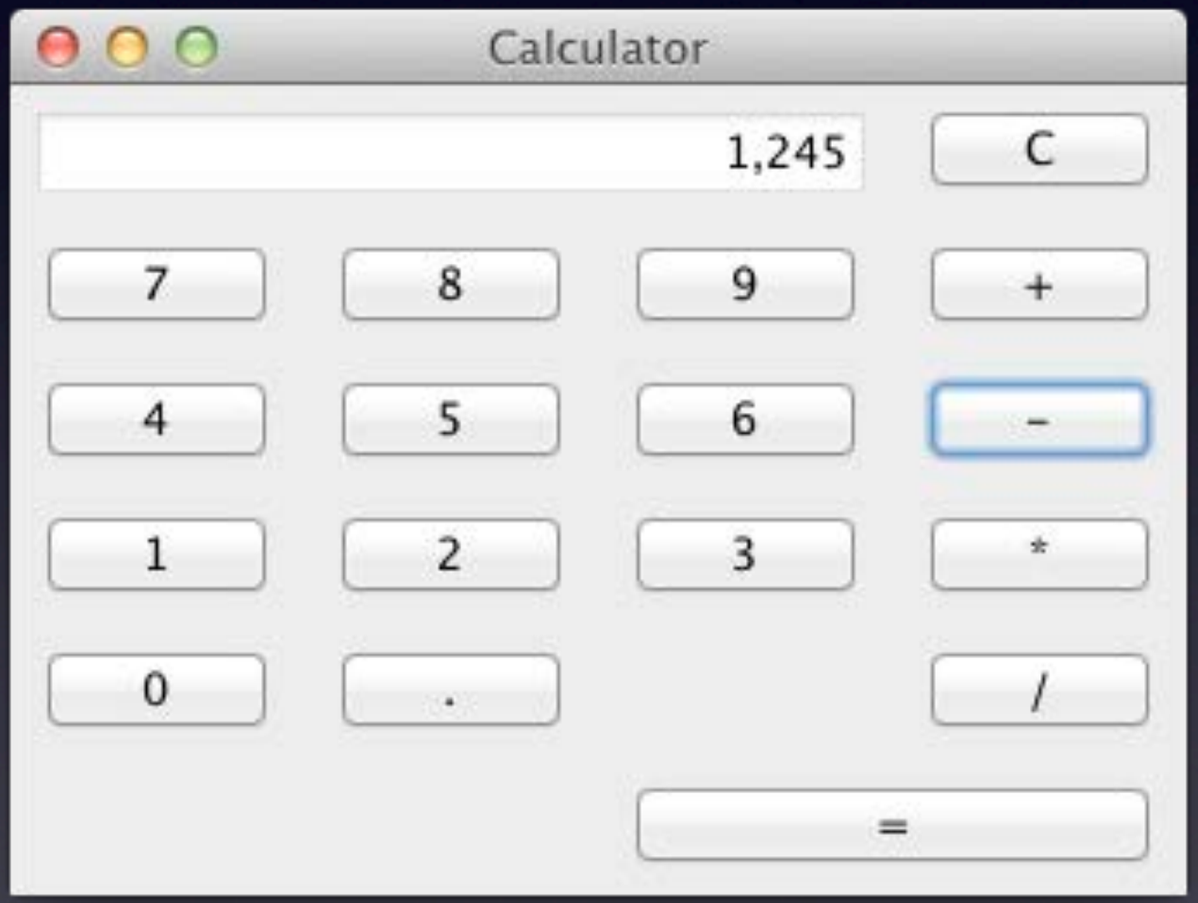
Florian Gross, Andreas Zeller

EvoSuite + **Daikon**

Unit Tests

EXSYST + **Daikon**

System Tests

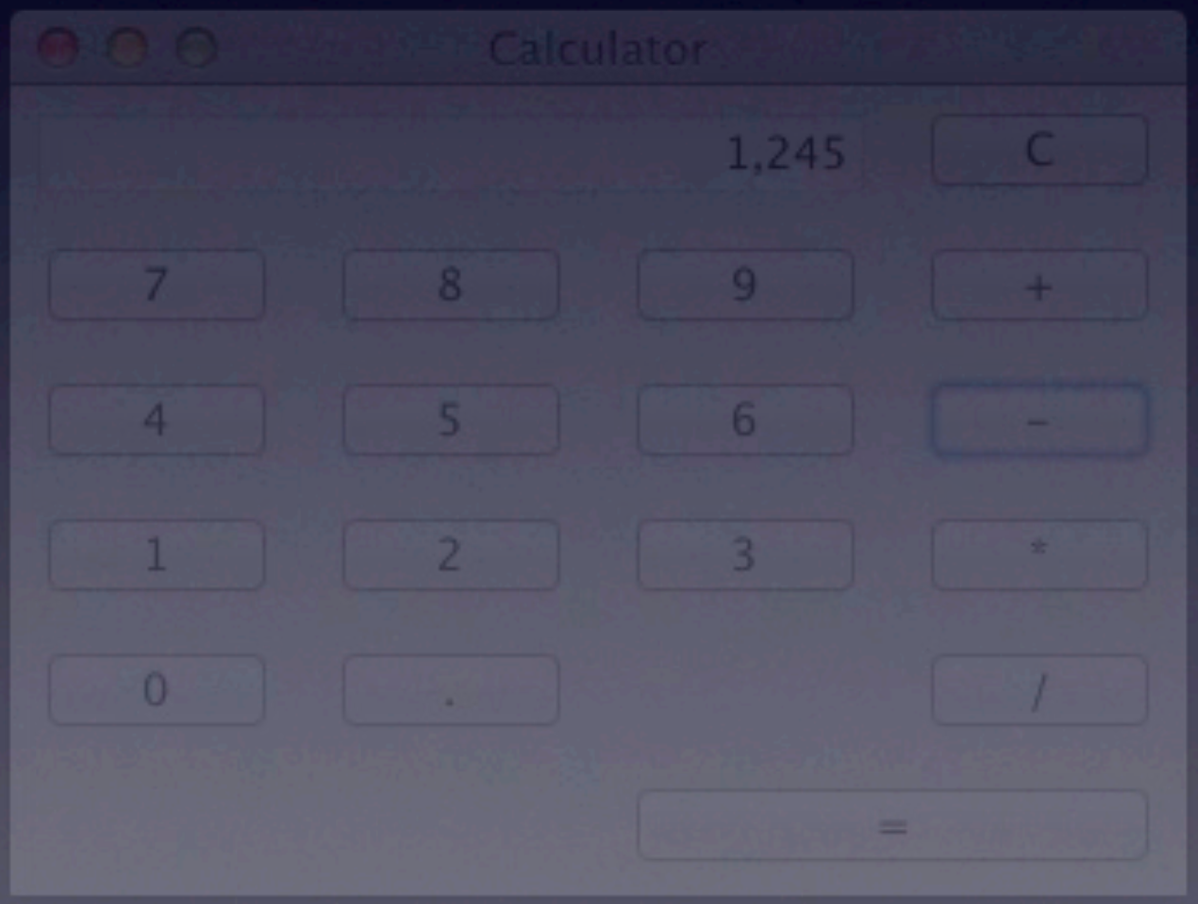


CalculatorPanel

Object Invariants

EvoSuite +
Unit Tests
Daikon

EXSYST +
System Tests
Daikon



CalculatorPanel

Object Invariants

EvoSuite
Unit Tests

+

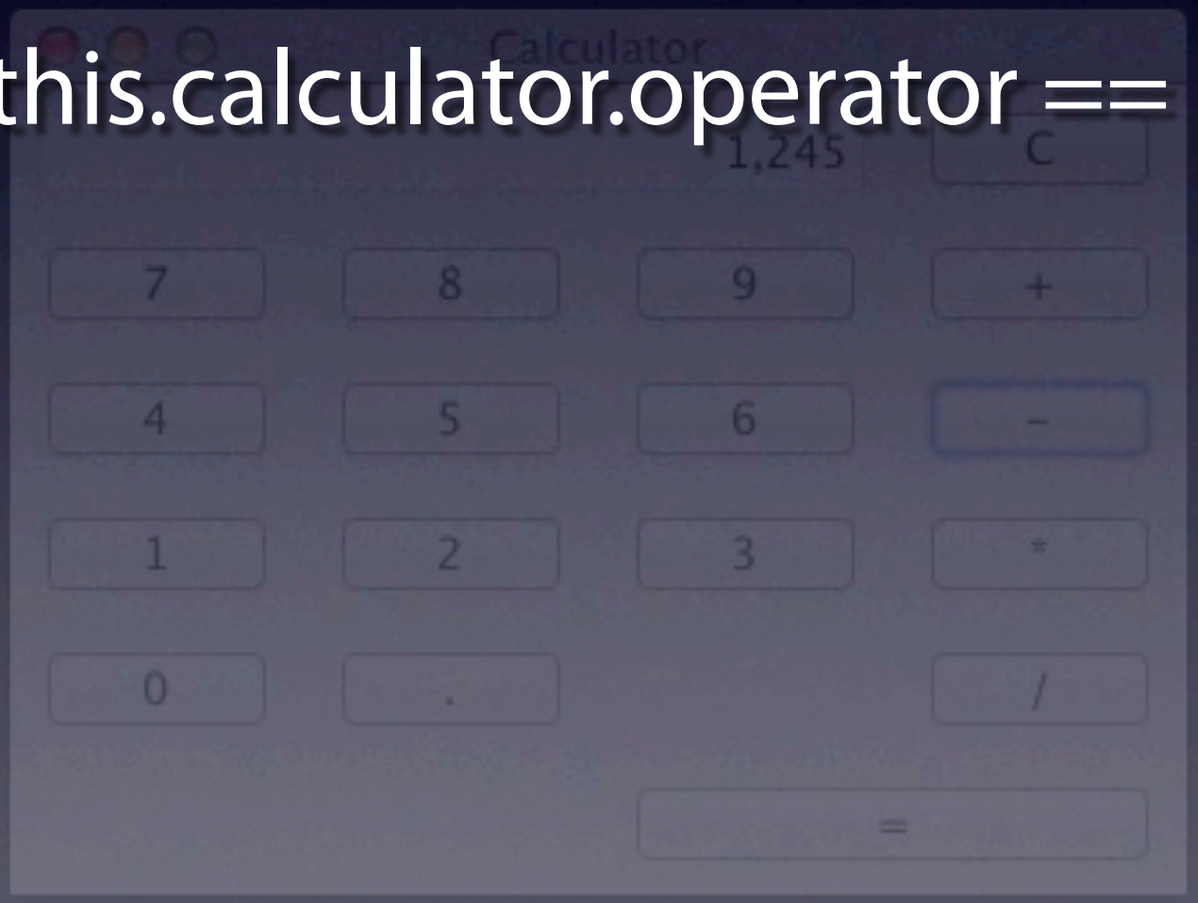
Daikon

EXSYST
System Tests

+

Daikon

this.calculator.operator == null



CalculatorPanel

Object Invariants

EvoSuite
Unit Tests

\+

Daikon

this.calculator.operator == null

EXSYST
System Tests

\+

Daikon

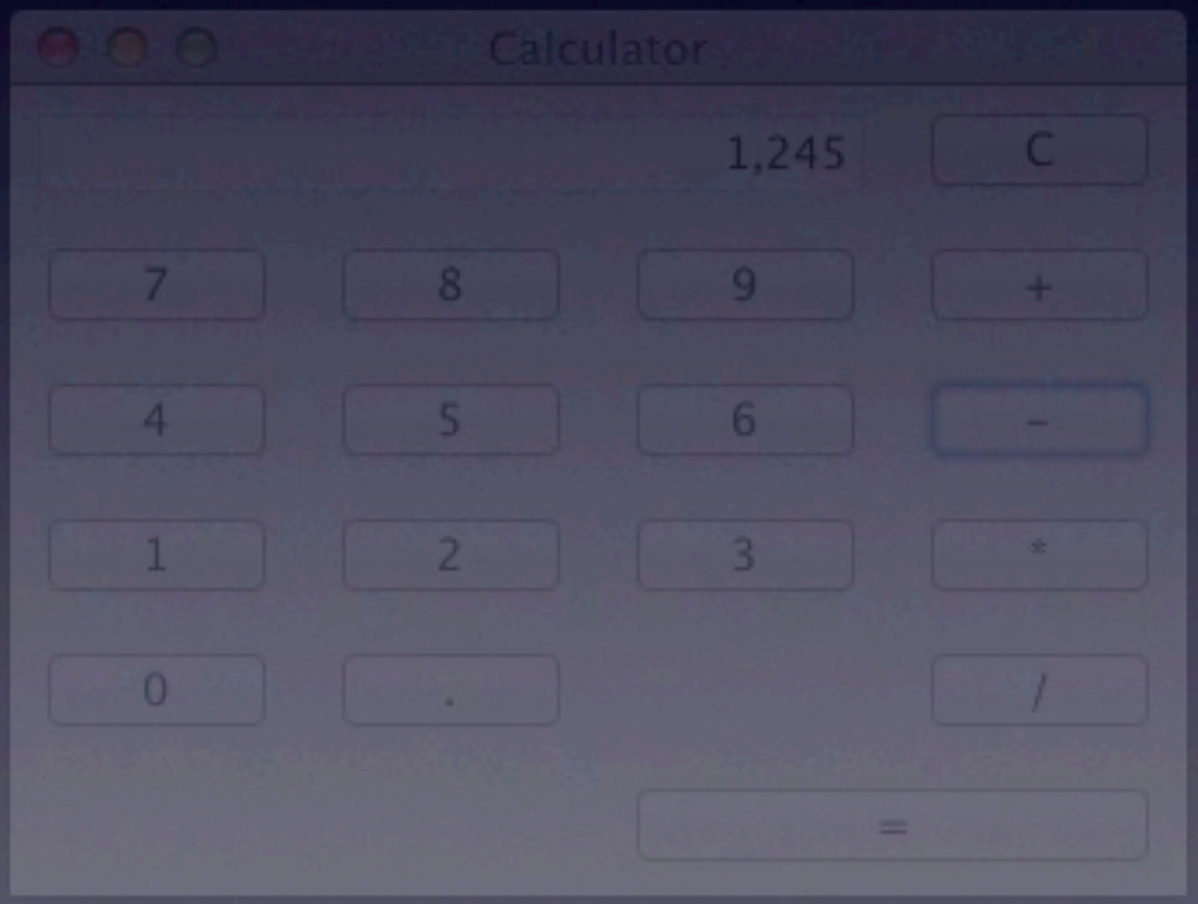
(no such invariant:
explores multiple operators)

CalculatorPanel

Object Invariants

EvoSuite +
Unit Tests
Daikon

EXSYST +
System Tests
Daikon

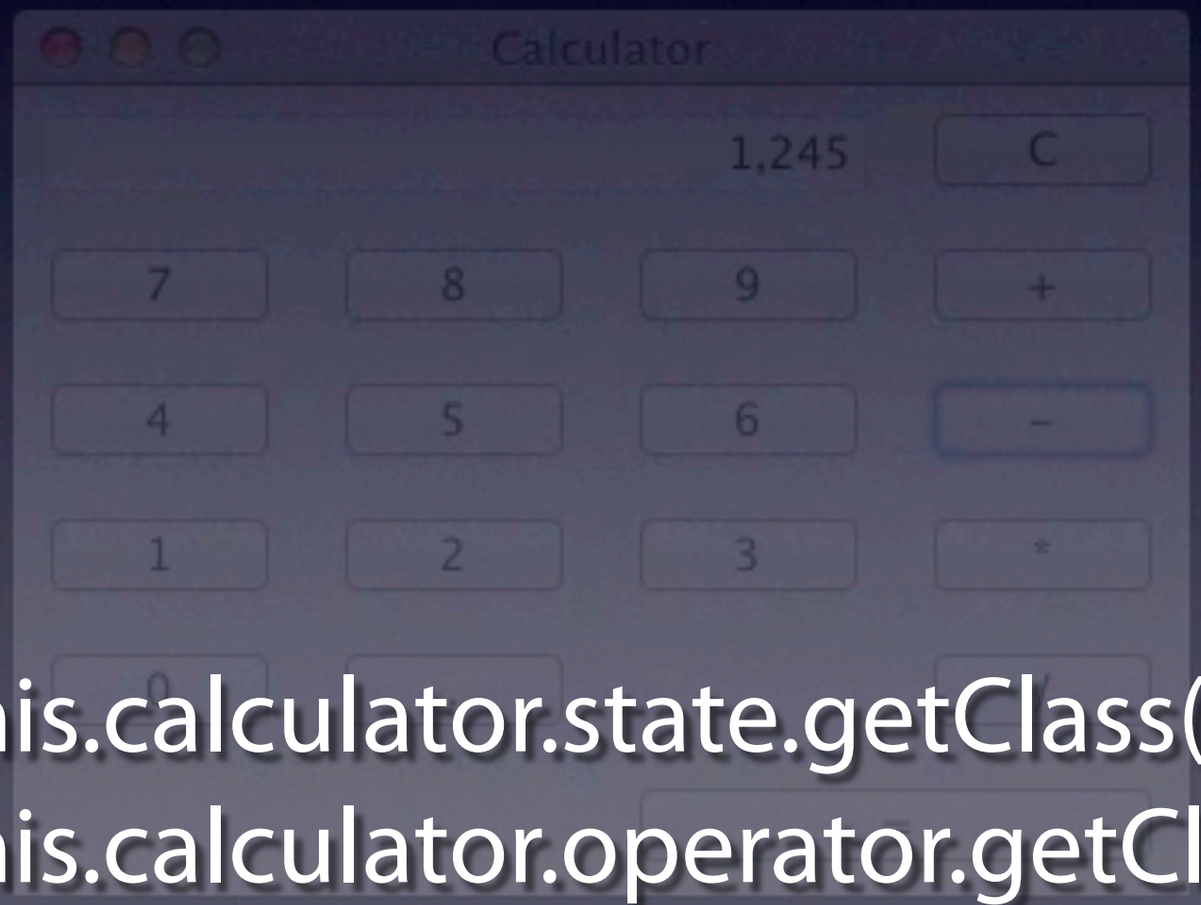


CalculatorPanel

Object Invariants

EvoSuite +
Unit Tests
Daikon

EXSYST +
System Tests
Daikon



this.calculator.state.getClass() !=
this.calculator.operator.getClass()

CalculatorPanel

Object Invariants

EvoSuite + **Daikon**
Unit Tests

(no such invariant:
this.calculator.operator == null)

EXSYST + **Daikon**
System Tests

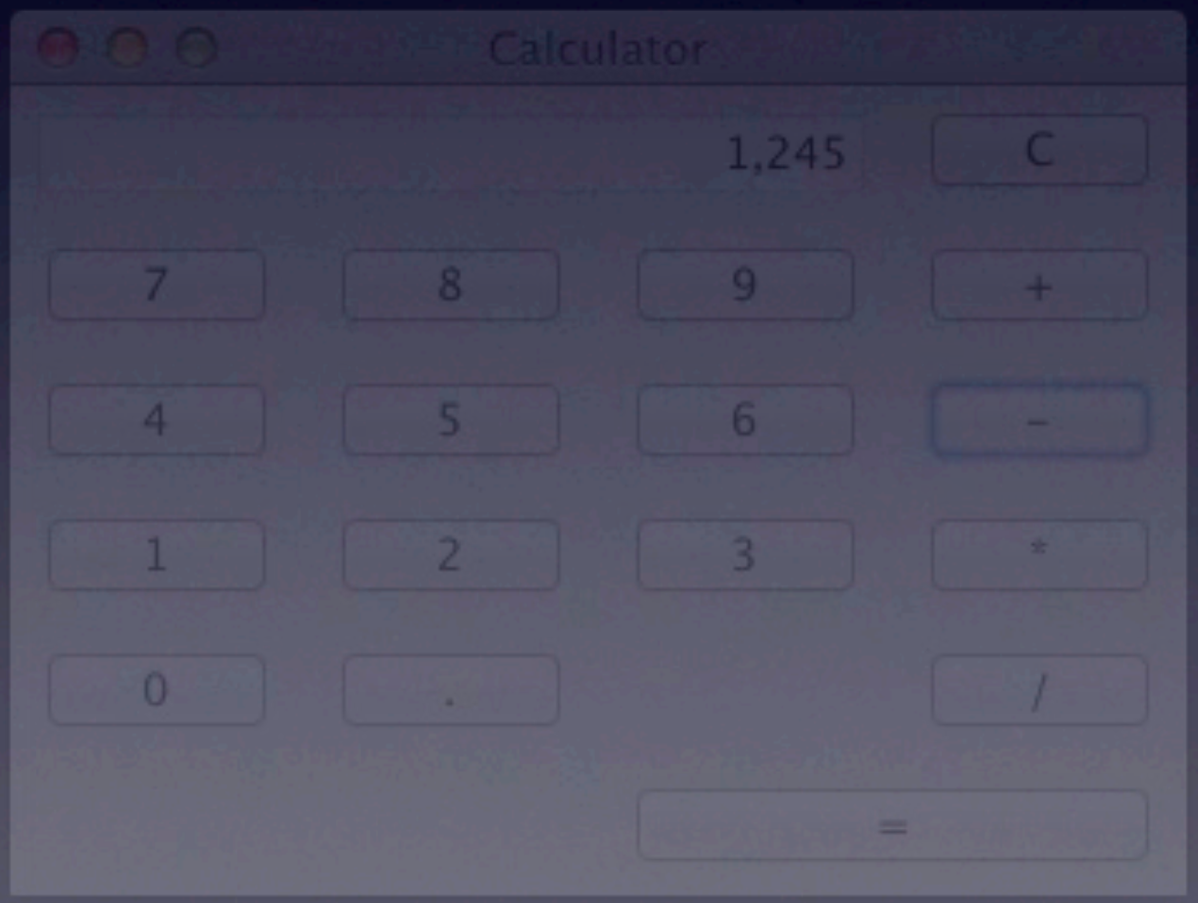
this.calculator.state.getClass() !=
this.calculator.operator.getClass()

Calculator Operand

EnteringFirstOperandState(Calculator, char c)

EvoSuite + **Daikon**
Unit Tests

EXSYST + **Daikon**
System Tests

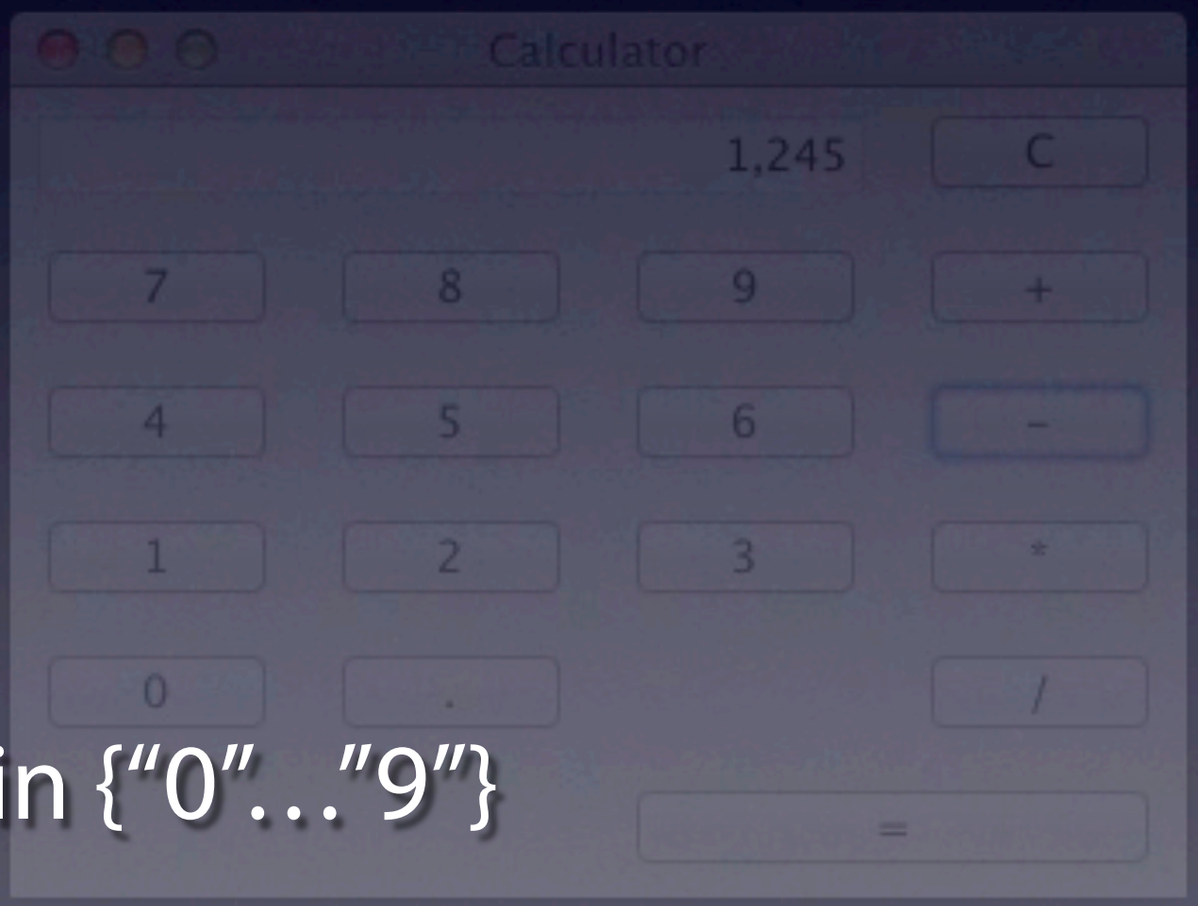


Calculator Operand

EnteringFirstOperandState(Calculator, char c)

EvoSuite + **Daikon**
Unit Tests

EXSYST + **Daikon**
System Tests



c in {“0”…”9”}

Calculator Operand

EnteringFirstOperandState(Calculator, char c)

EvoSuite Unit Tests + **Daikon**

(no such invariant: c takes random values)

EXSYST System Tests + **Daikon**

c in {“0”…”9”}

Challenges

Mine specifications that are

complete	real	proven

Challenges

Mine specifications that are

complete	real	proven

Proving a Multiplier

Proving a Multiplier

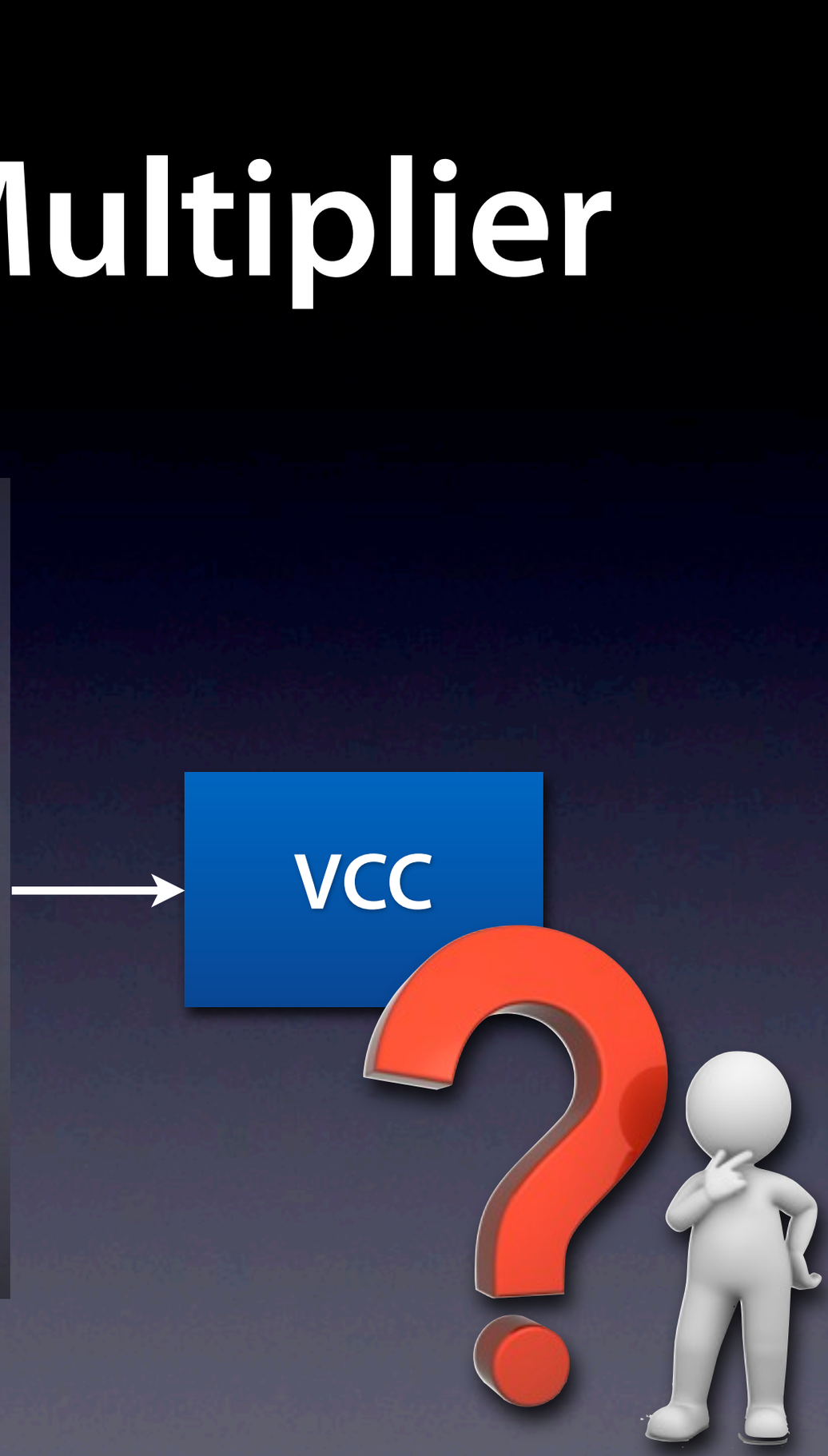
```
_(requires 0 ≤ x < 65535)
_(requires 0 ≤ y < 65535)
_(ensures \result == x*y)
mult = i = 0;
while (i<y) {
  mult += x; i++;
}
return mult;
```

Proving a Multiplier

```
_(requires 0 ≤ x < 65535)
_(requires 0 ≤ y < 65535)
_(ensures \result == x*y)
mult = i = 0;
while (i<y) {
  mult += x; i++;
}
return mult;
```

Proving a Multiplier

```
_(requires 0 ≤ x < 65535)
_(requires 0 ≤ y < 65535)
_(ensures \result == x*y)
mult = i = 0;
while (i<y) {
  mult += x; i++;
}
return mult;
```



Mining Loop Invariants

Juan Pablo Galeotti, Andreas Zeller

Mining Loop Invariants

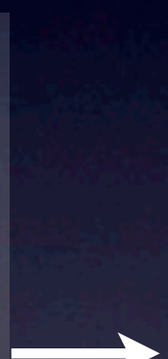
Juan Pablo Galeotti, Andreas Zeller

```
mult = i = 0;
while (i<y) {
  mult += x; i++;
}
return mult;
```

Mining Loop Invariants

Juan Pablo Galeotti, Andreas Zeller

```
mult = i = 0;
while (i<y) {
  mult += x; i++;
}
return mult;
```



EvoSuite

Unit Tests

+

Daikon

Mining Loop Invariants

Juan Pablo Galeotti, Andreas Zeller

```
mult = i = 0;
while (i<y) {
  mult += x; i++;
}
return mult;
```

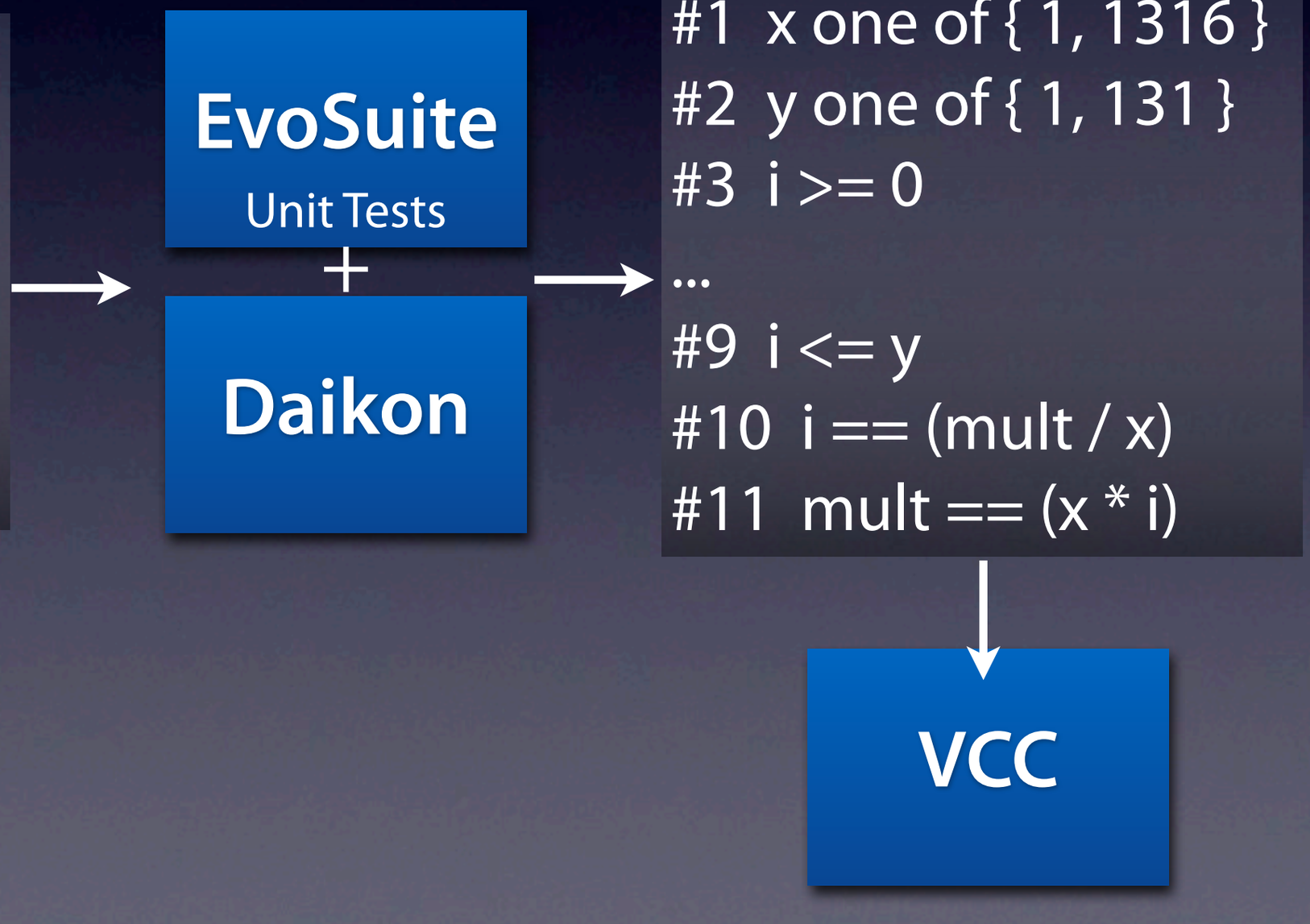
EvoSuite
Unit Tests
+
Daikon

#1 x one of { 1, 1316 }
#2 y one of { 1, 131 }
#3 i >= 0
...
#9 i <= y
#10 i == (mult / x)
#11 mult == (x * i)

Mining Loop Invariants

Juan Pablo Galeotti, Andreas Zeller

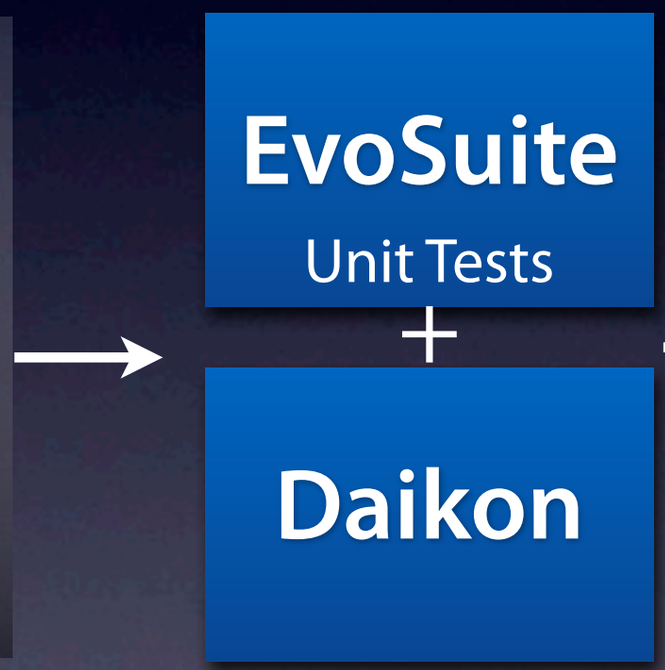
```
mult = i = 0;
while (i<y) {
  mult += x; i++;
}
return mult;
```



Mining Loop Invariants

Juan Pablo Galeotti, Andreas Zeller

```
mult = i = 0;
while (i<y) {
  mult += x; i++;
}
return mult;
```




```
#1  x one of { 1, 1316 }
#2  y one of { 1, 131 }
#3  i >= 0
...
#9  i <= y
#10  i == (mult / x)
#11  mult == (x * i)
```

Mining Loop Invariants

Juan Pablo Galeotti, Andreas Zeller

```
mult = i = 0;
while (i<y) {
  mult += x; i++;
}
return mult;
```

EvoSuite

Unit Tests

+

Daikon

~~#1 x one of { 1, 1316 }~~
~~#2 y one of { 1, 131 }~~
#3 i >= 0
...
#9 i <= y
#10 i == (mult / x)
#11 mult == (x * i)

VCC

Mining Loop Invariants

Juan Pablo Galeotti, Andreas Zeller

```
mult = i = 0;
while (i<y) {
  mult += x; i++;
}
return mult;
```

EvoSuite

Unit Tests

+

Daikon

~~#1 x one of { 1, 1316 }~~
~~#2 y one of { 1, 131 }~~
#3 i >= 0
...
#9 i <= y
#10 i == (mult / x)
#11 mult == (x * i)

VCC

Mining Loop Invariants

Juan Pablo Galeotti, Andreas Zeller

```
mult = i = 0;
while (i<y) {
  mult += x; i++;
}
return mult;
```

EvoSuite

Unit Tests

+

Daikon

~~#1 x one of { 1, 1316 }~~
~~#2 y one of { 1, 131 }~~
~~#3 i >= 0~~
...
#9 i <= y
#10 i == (mult / x)
#11 mult == (x * i)

VCC

Mining Loop Invariants

Juan Pablo Galeotti, Andreas Zeller

```
mult = i = 0;
while (i<y) {
  mult += x; i++;
}
return mult;
```

EvoSuite
Unit Tests
+
Daikon

~~#1 x one of { 1, 1316 }~~
~~#2 y one of { 1, 131 }~~
~~#3 i >= 0~~
...
#9 i <= y
~~#10 i == (mult / x)~~
#11 mult == (x * i)

VCC

Mining Loop Invariants

Juan Pablo Galeotti, Andreas Zeller

```
mult = i = 0;
while (i<y) {
  mult += x; i++;
}
return mult;
```

EvoSuite
Unit Tests
+
Daikon

~~#1 x one of { 1, 1316 }~~
~~#2 y one of { 1, 131 }~~
~~#3 i >= 0~~
...
#9 i <= y
~~#10 i == (mult / x)~~
#11 mult == (x * i)

VCC

Proven Specifications

```
_(requires 0 ≤ x < 65535)
_(requires 0 ≤ y < 65535)
_(ensures \result == x*y)
mult = i = 0;
while (i<y) {
  mult += x; i++;
}
return mult;
```

Challenges

Mine specifications that are

complete	real	proven

Challenges

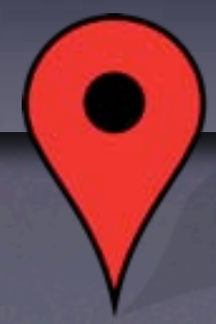
Mine specifications that are

complete	real	proven

Challenges

Mine specifications that are

complete	real	proven



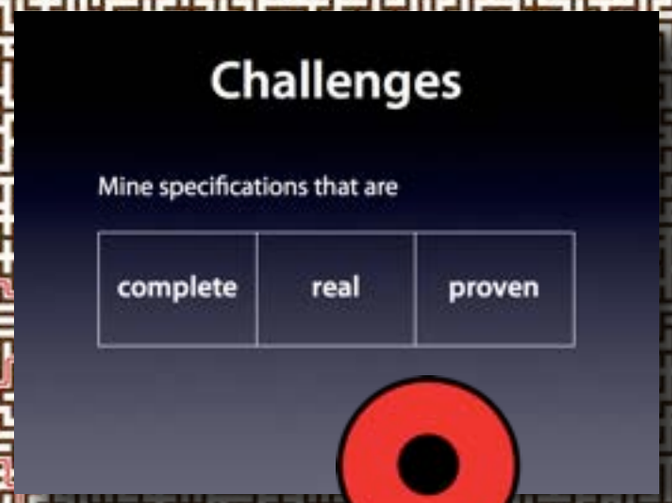
Specifying is hard

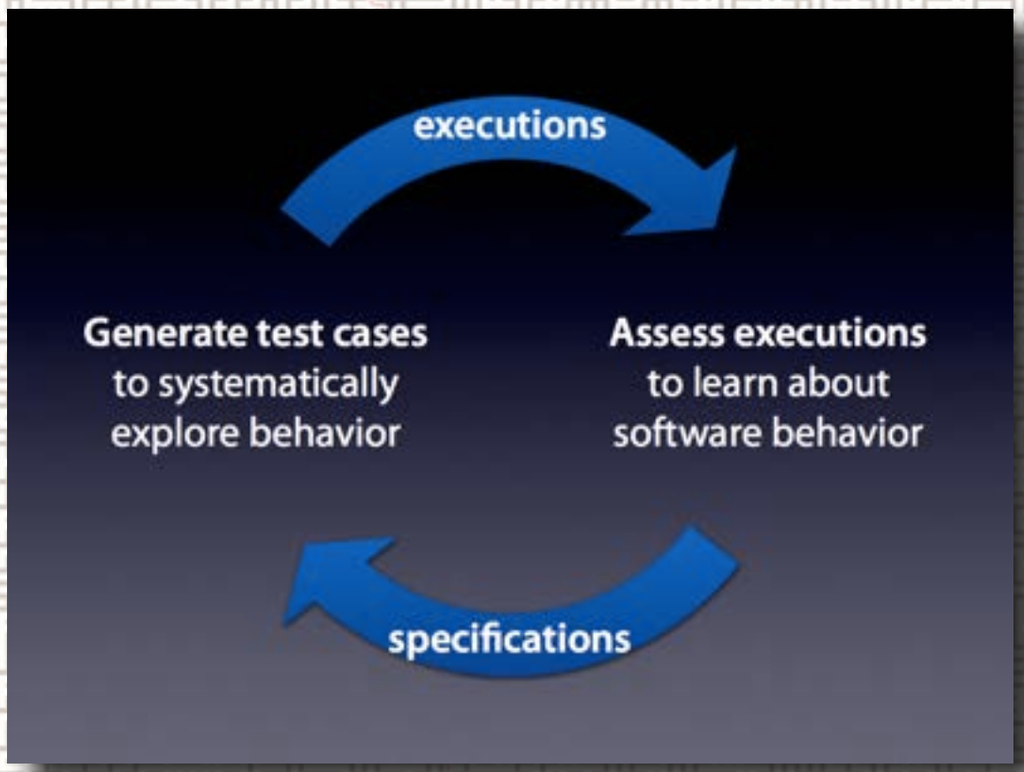
Challenges

Mine specifications that are

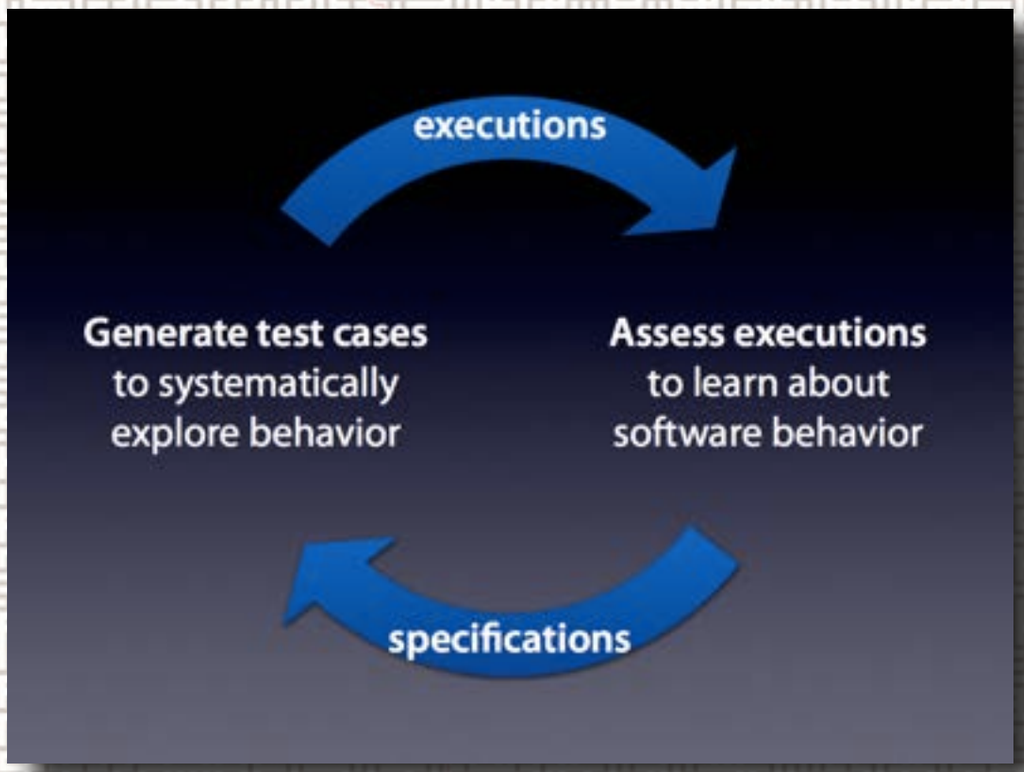
complete	real	proven


Specifying is hard
assert (y*y == x)

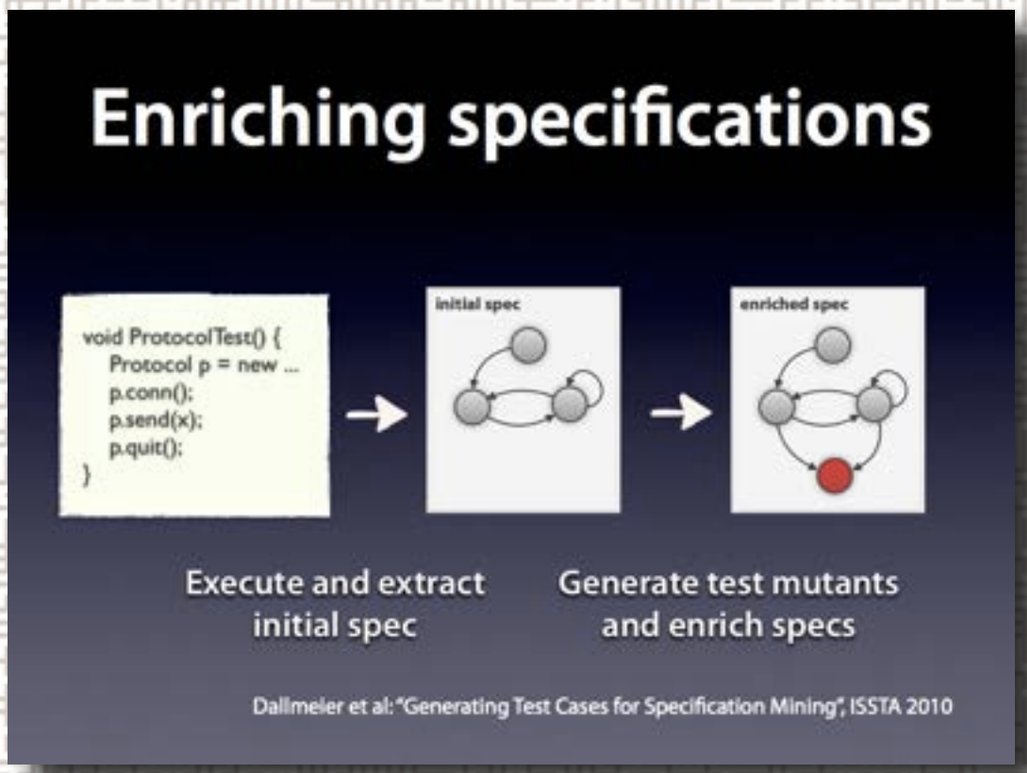

Challenges
Mine specifications that are
complete
real
proven



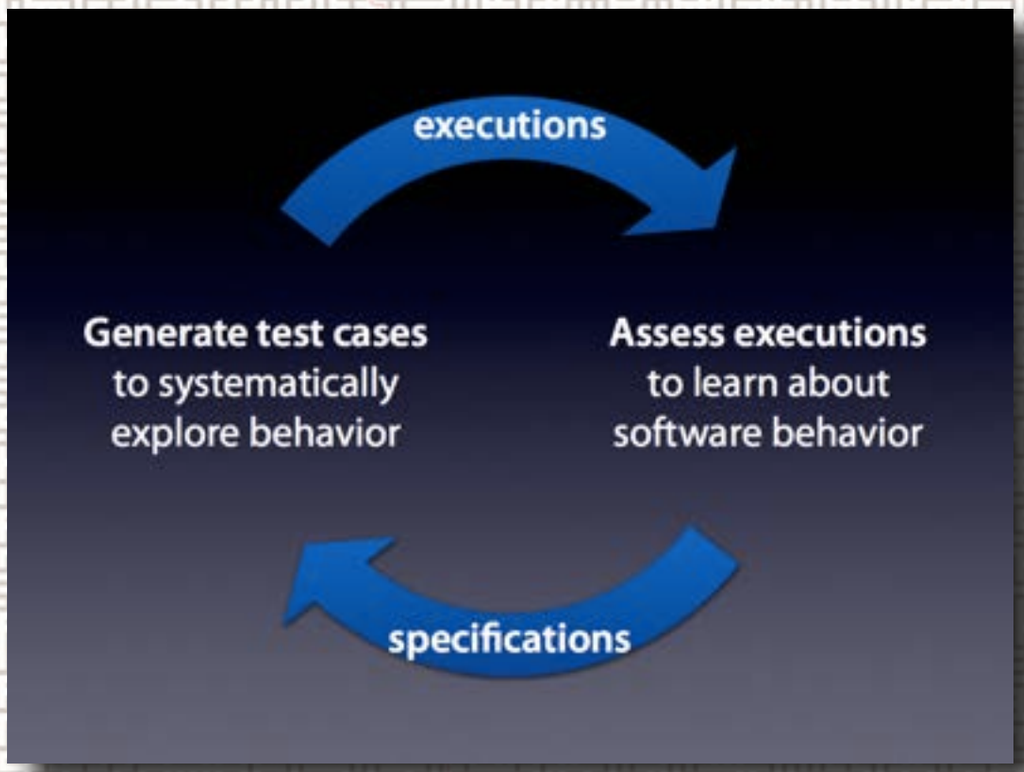
Mining specs



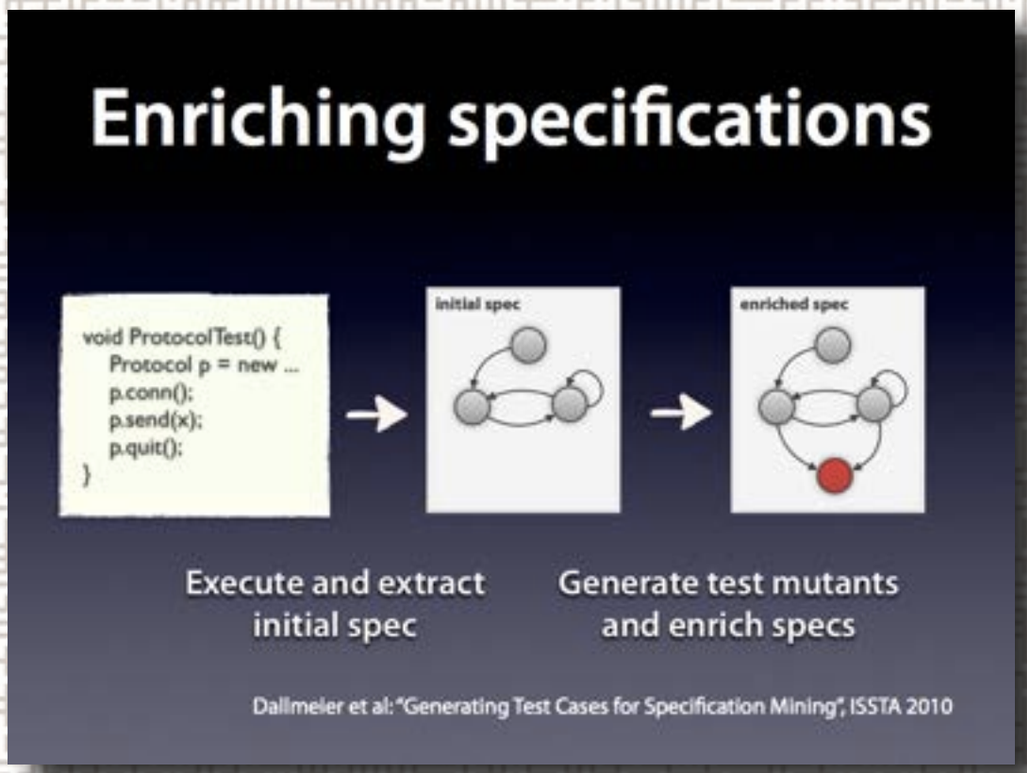
Mining specs



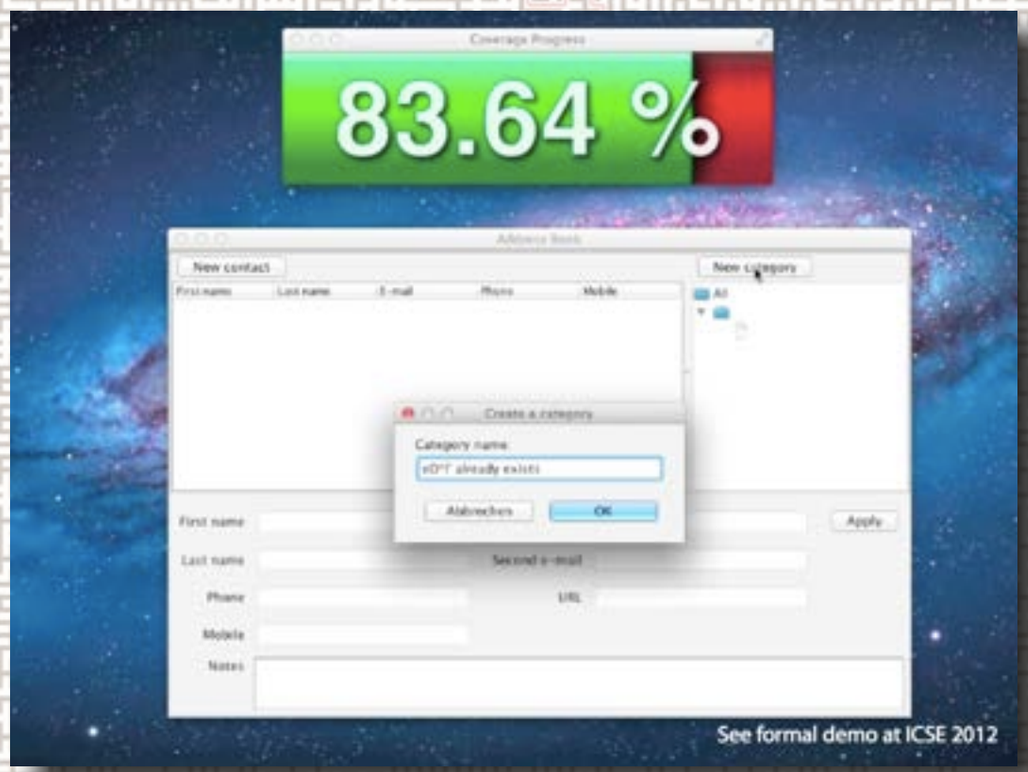
Complete specs



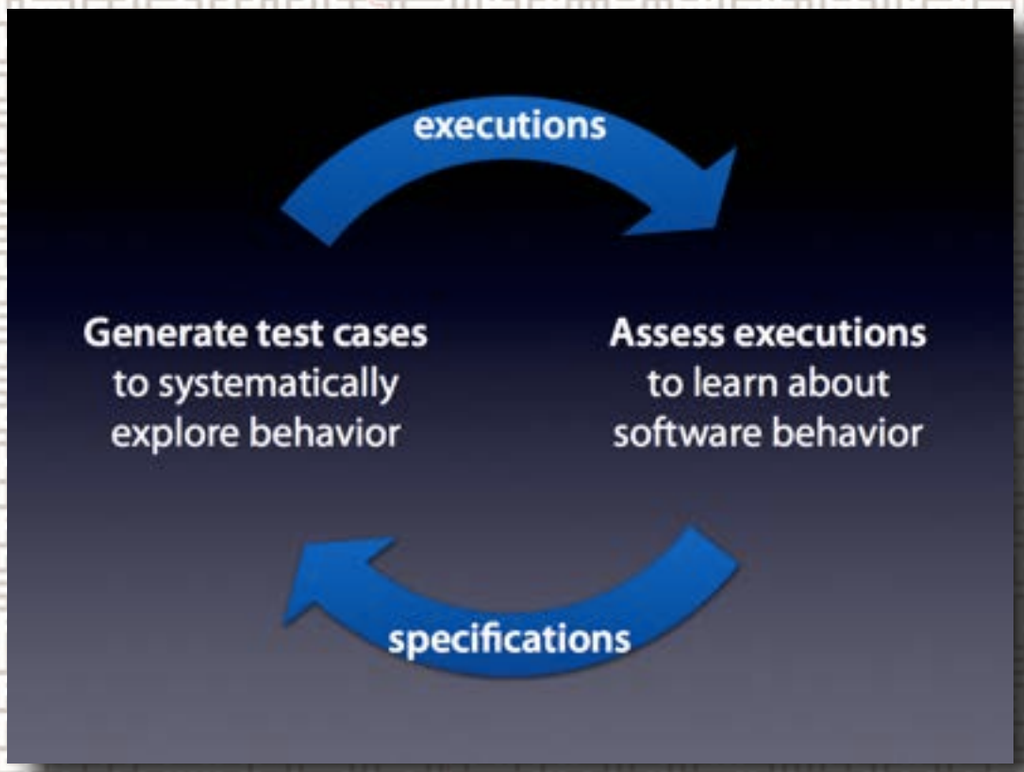
Mining specs



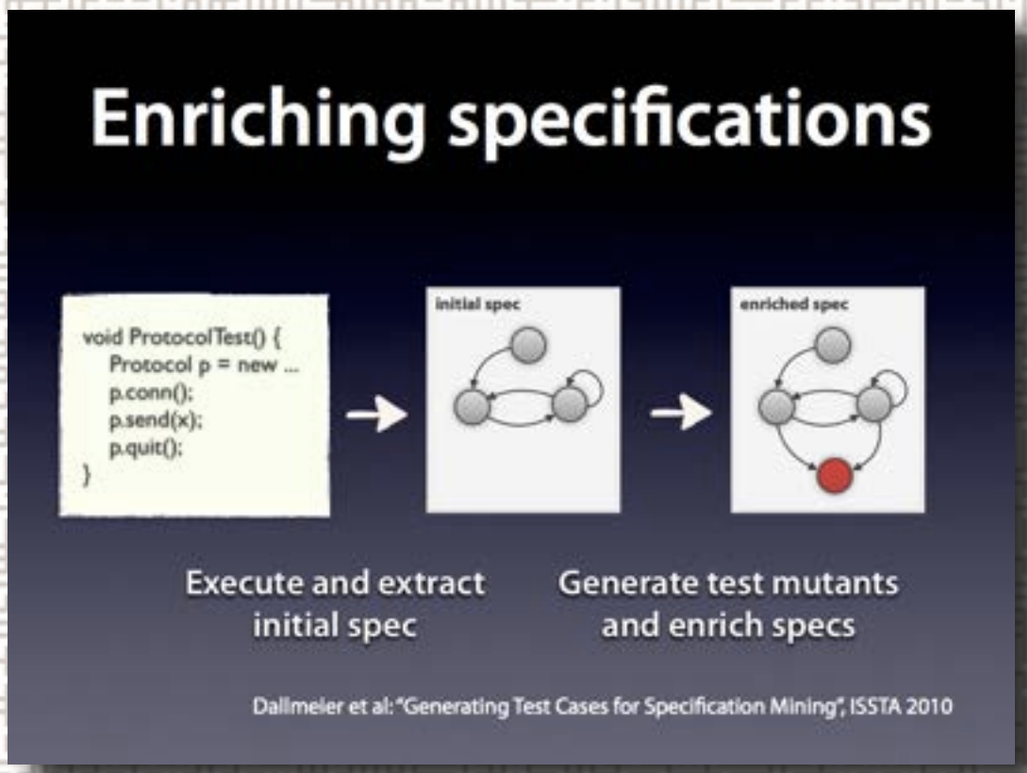
Complete specs



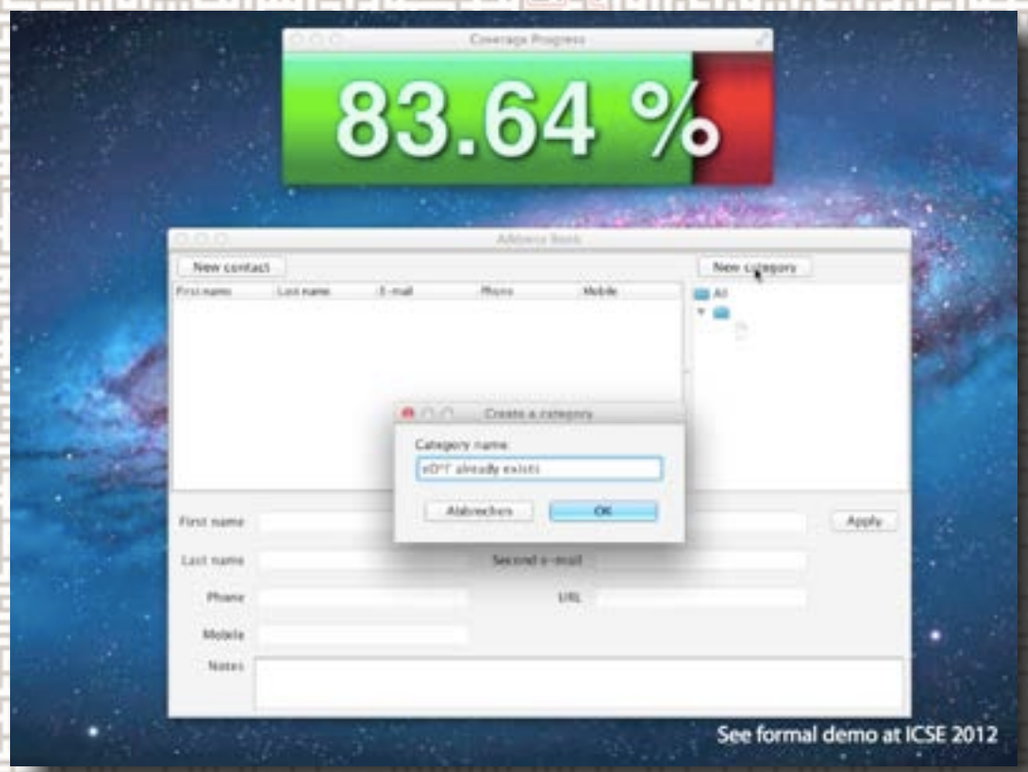
Real specs



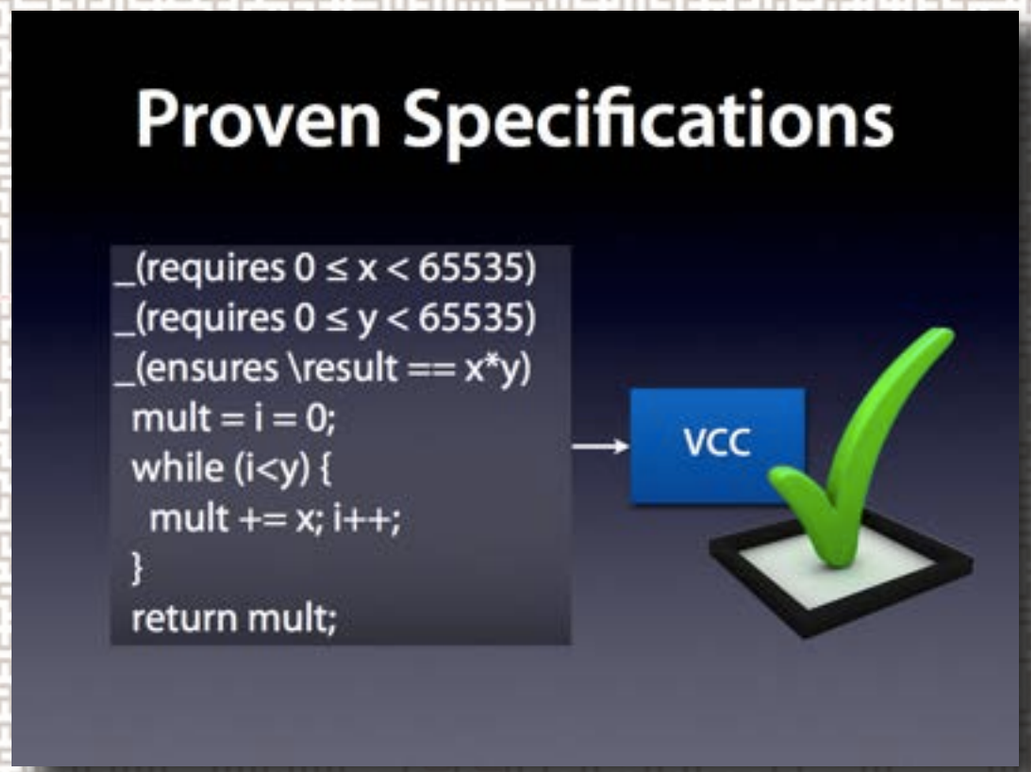
Mining specs



Complete specs



Real specs



Proven specs