Cate: A System for Analysis and Test of Java Card Applications

Peter Pfahler, Universität Paderborn, Institut für Informatik

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Jürgen Günther, ORGA Kartensysteme GmbH, Paderborn

ORGA
The Smart Card Integrator

First International Workshop on Software Quality

SOQUA 2004, Erfurt, September 30

NET.OBJECTDAYS 2004

The Smart Card Market

Telecommunications

- Cards for GSM and UMTS (3G)

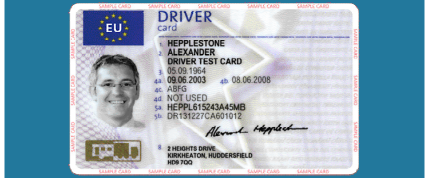
Banking

- Bank and credit cards

Health

- Health insurance cards
- Signature cards

Identification



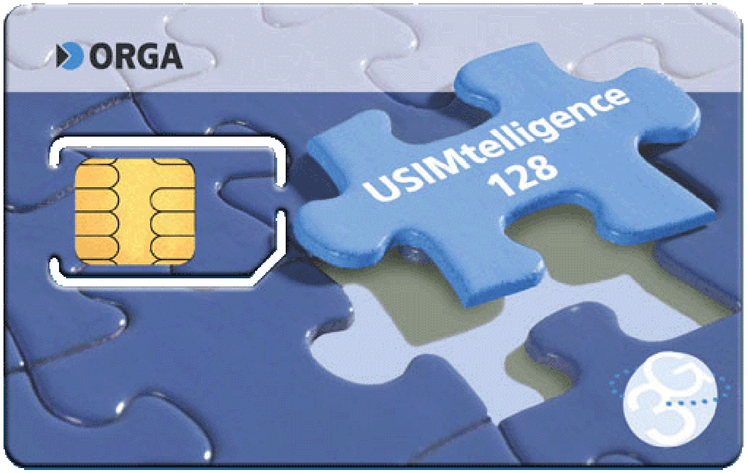
- ID- and signature cards

- Security and Authentication
- No Updates, Patches, Service Packs
- Software Quality
- Java Card

Cate: A System for Analysis and Test of Java Card Applications

Basic Idea:

By using Java as the programming language for card software, the usage of program analysis tools becomes feasible.

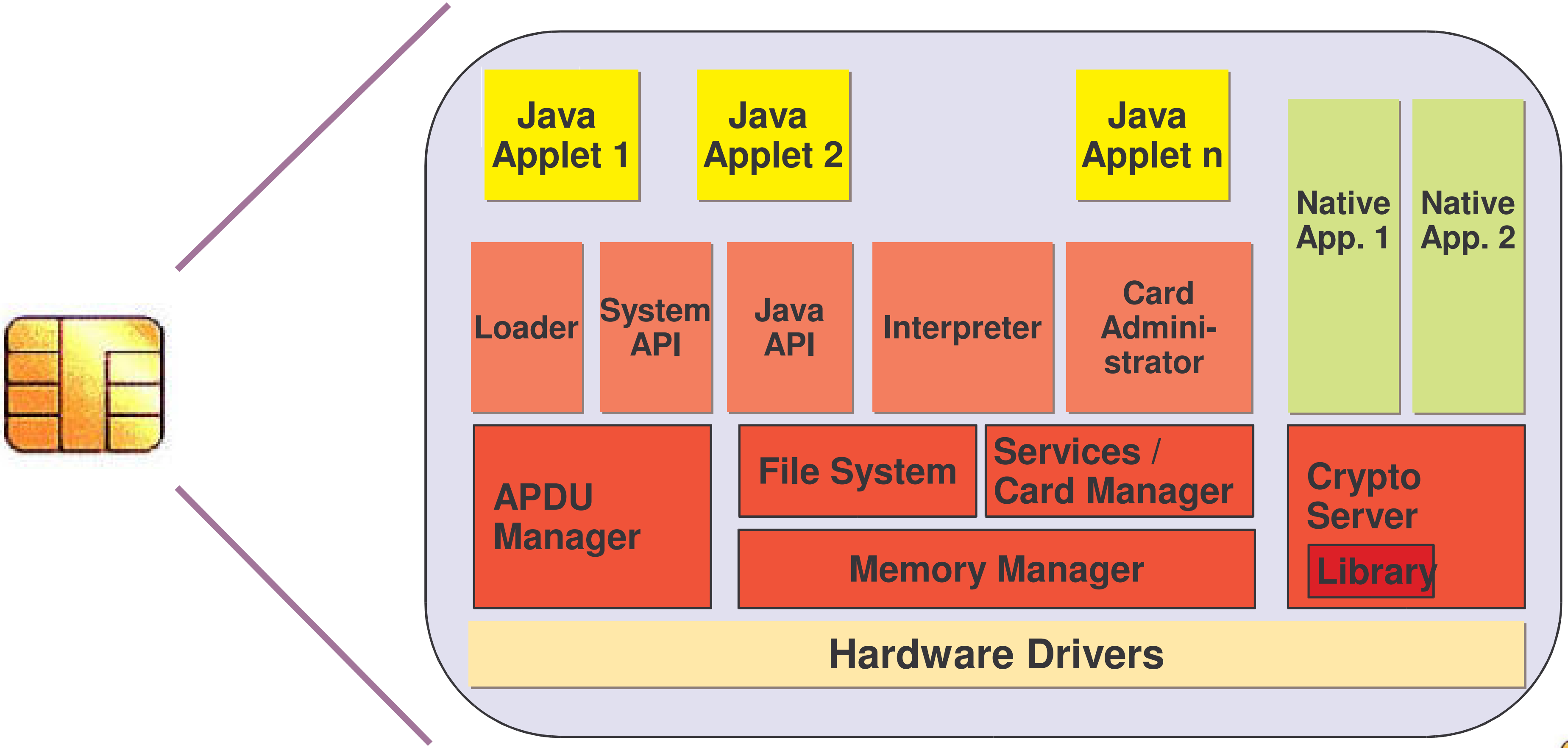


Overview:

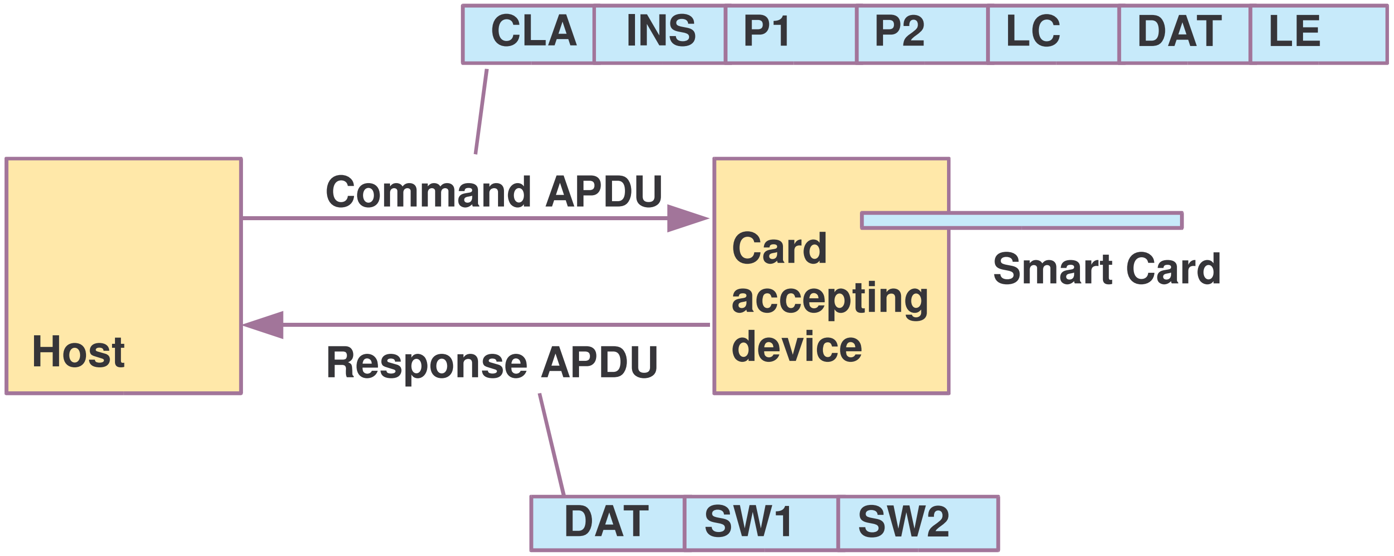
Smart card basics:	Master/Slave Communication, Java Card
Static Analysis:	Command-Response behavior
Dynamic Analysis:	Test coverage
The Cate System:	Practical experience

Java Card

Java Cards include a Java Virtual Machine (JVM) to run Java applications.



The smart card communication model: Master/Slave



CLA	INS	P1	P2	LC	DAT	LE

Command APDU

Host

Card accepting device

Smart Card

Response APDU

DAT	SW1	SW2

Static Analysis of Command/Response Behavior

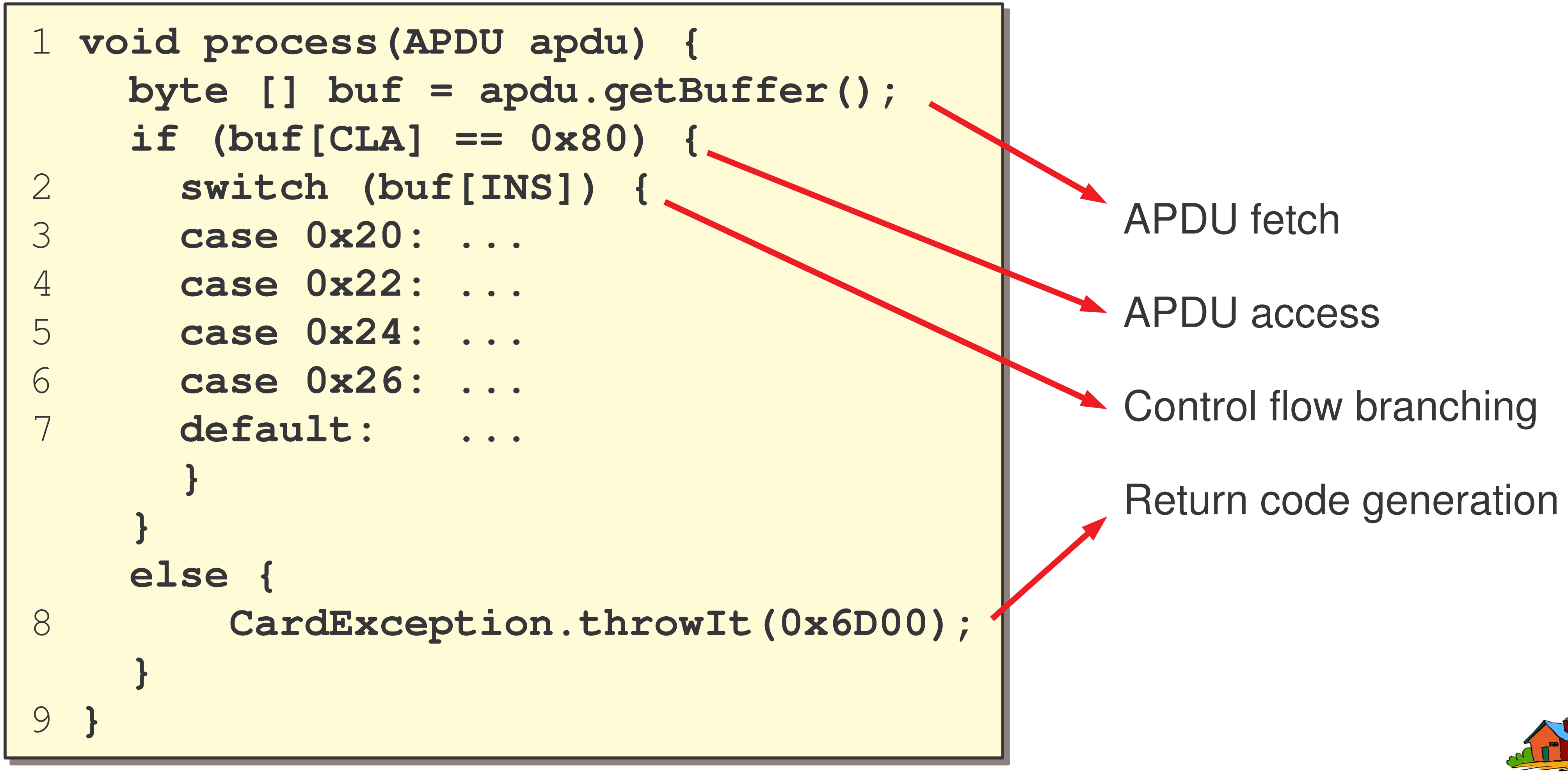
Typical Structure of a Java Card Applet

```
void process(APDU apdu) {
  byte [] buf = apdu.getBuffer();
  if (buf[CLA] == 0x80) {
    switch (buf[INS]) {
    case 0x20: ...
    case 0x22: ...
    case 0x24: ...
    case 0x26: ...
    default:   ...
    }
  }
  else {
      CardException.throwIt(0x6D00);
  }
}
```

Static Analysis of Command/Response Behavior

Typical Structure of a Java Card Applet

Code Clichés



```
1 void process(APDU apdu) {
    byte [] buf = apdu.getBuffer();
    if (buf[CLA] == 0x80) {
2     switch (buf[INS]) {
3     case 0x20: ...
4     case 0x22: ...
5     case 0x24: ...
6     case 0x26: ...
7     default:   ...
      }
    }
    else {
8       CardException.throwIt(0x6D00);
    }
9 }
```

APDU fetch

APDU access

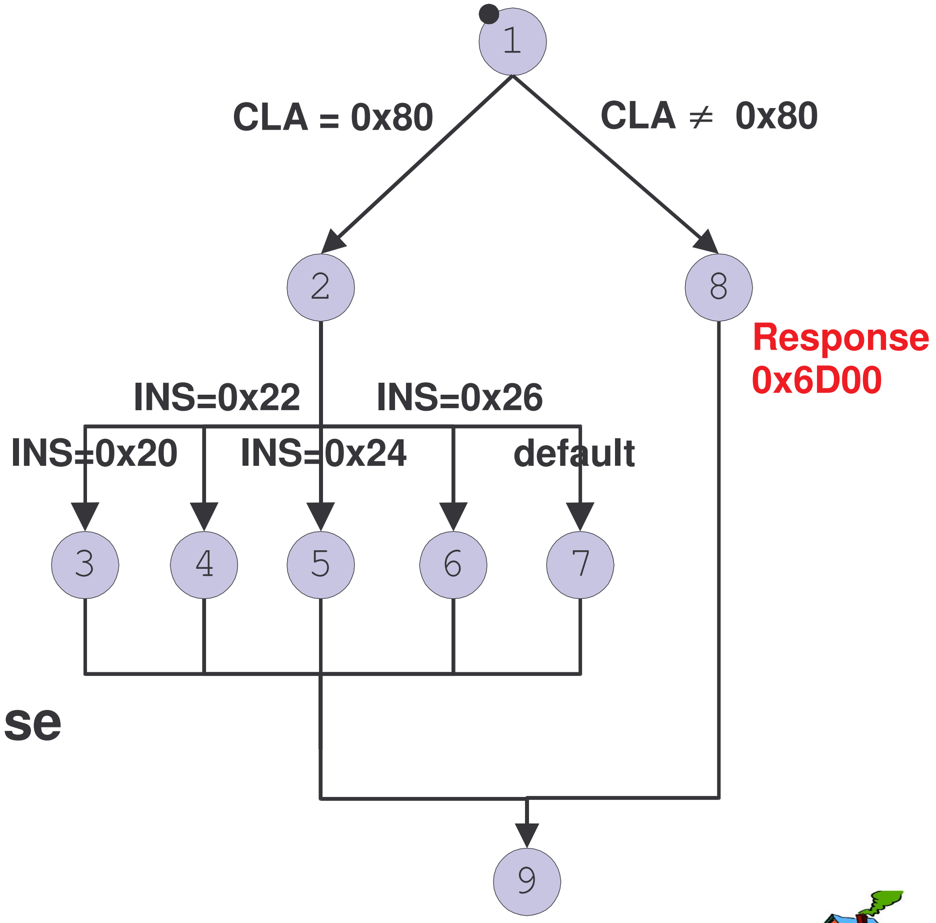
Control flow branching

Return code generation

Static Analysis of Command/Response Behavior

Control Flow Analysis

Data Flow Analysis based on Clichés

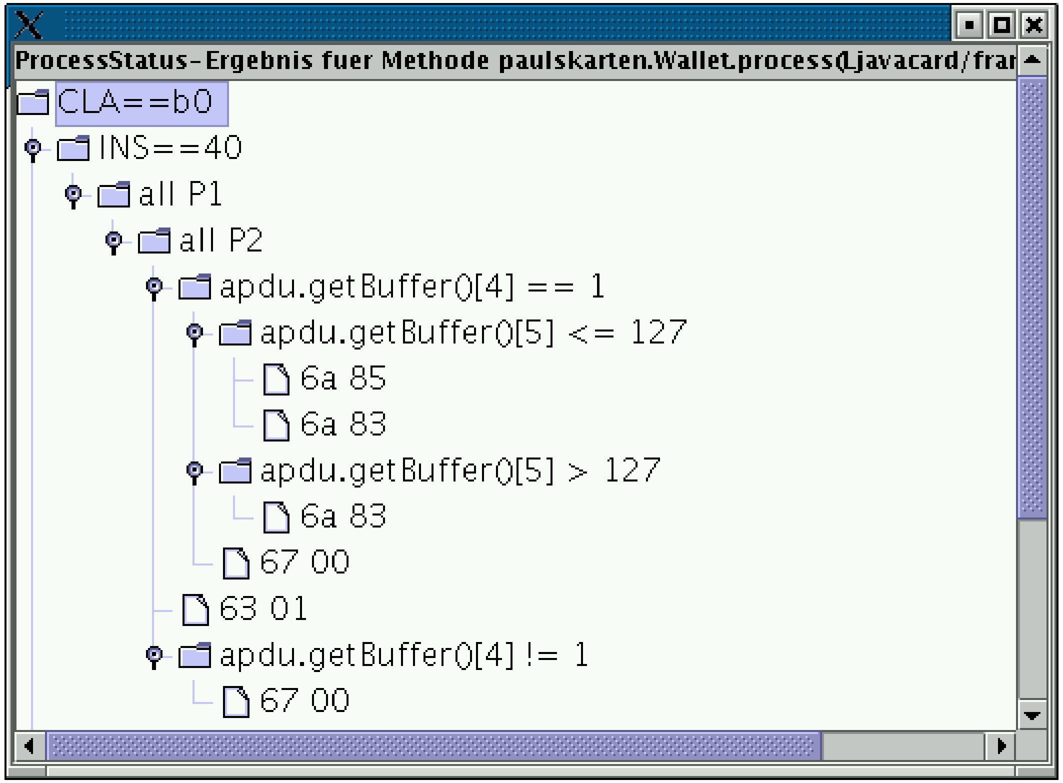


Results:

- Document listing the command/response combinations
- Annotated Control Flow Graph

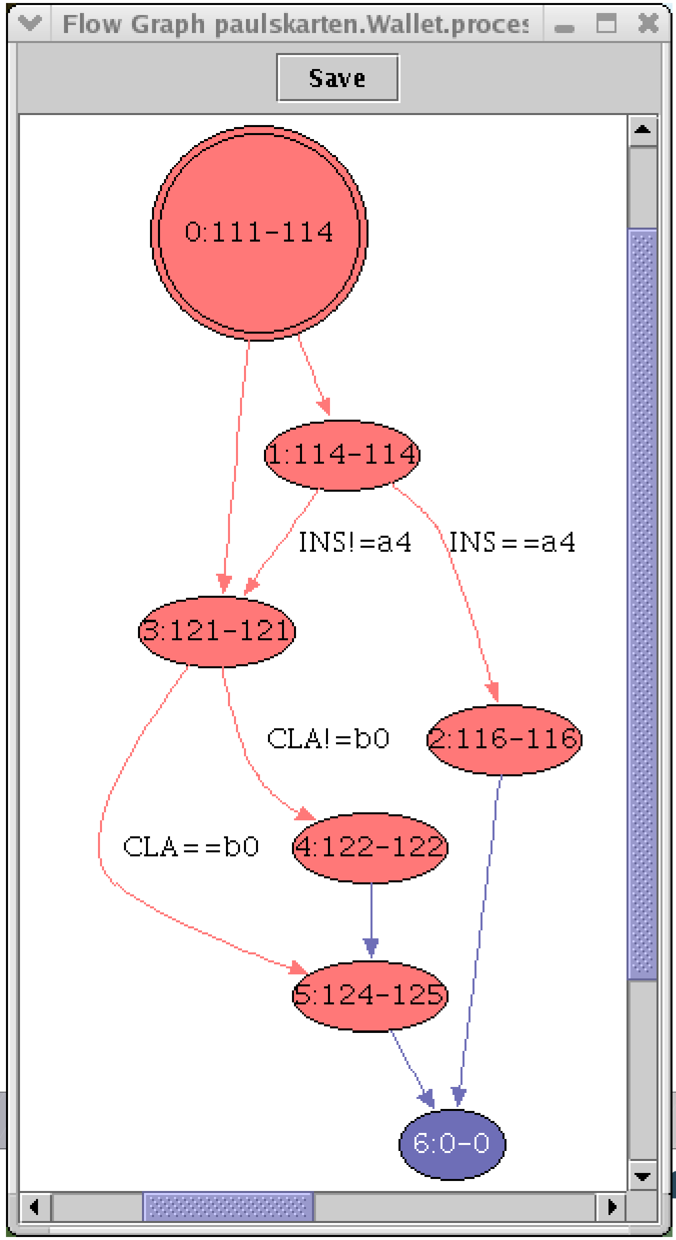
Static Analysis of Command/Response Behavior

Results of Static Analysis presented by Cate



```
ProcessStatus-Ergebnis fuer Methode paulskarten.Wallet.process(Ljavacard/fra
CLA==b0
  INS==40
    all P1
      all P2
        apdu.getBuffer()[4] == 1
          apdu.getBuffer()[5] <= 127
            6a 85
            6a 83
          apdu.getBuffer()[5] > 127
            6a 83
          67 00
        63 01
        apdu.getBuffer()[4] != 1
          67 00
```

Command/Response Combinations



Annotated Control Flow Graph

Dynamic Test Coverage Analysis

Test engineers need:

- information about untested program locations
- a measurement of test quality (e.g. C_0: basic block execution ratio)

Code coverage information can be gained by

- instrumentation of the card applet
- or profiling during card applet simulation

In practice coverage information turned out to be more valuable than the static analysis results.

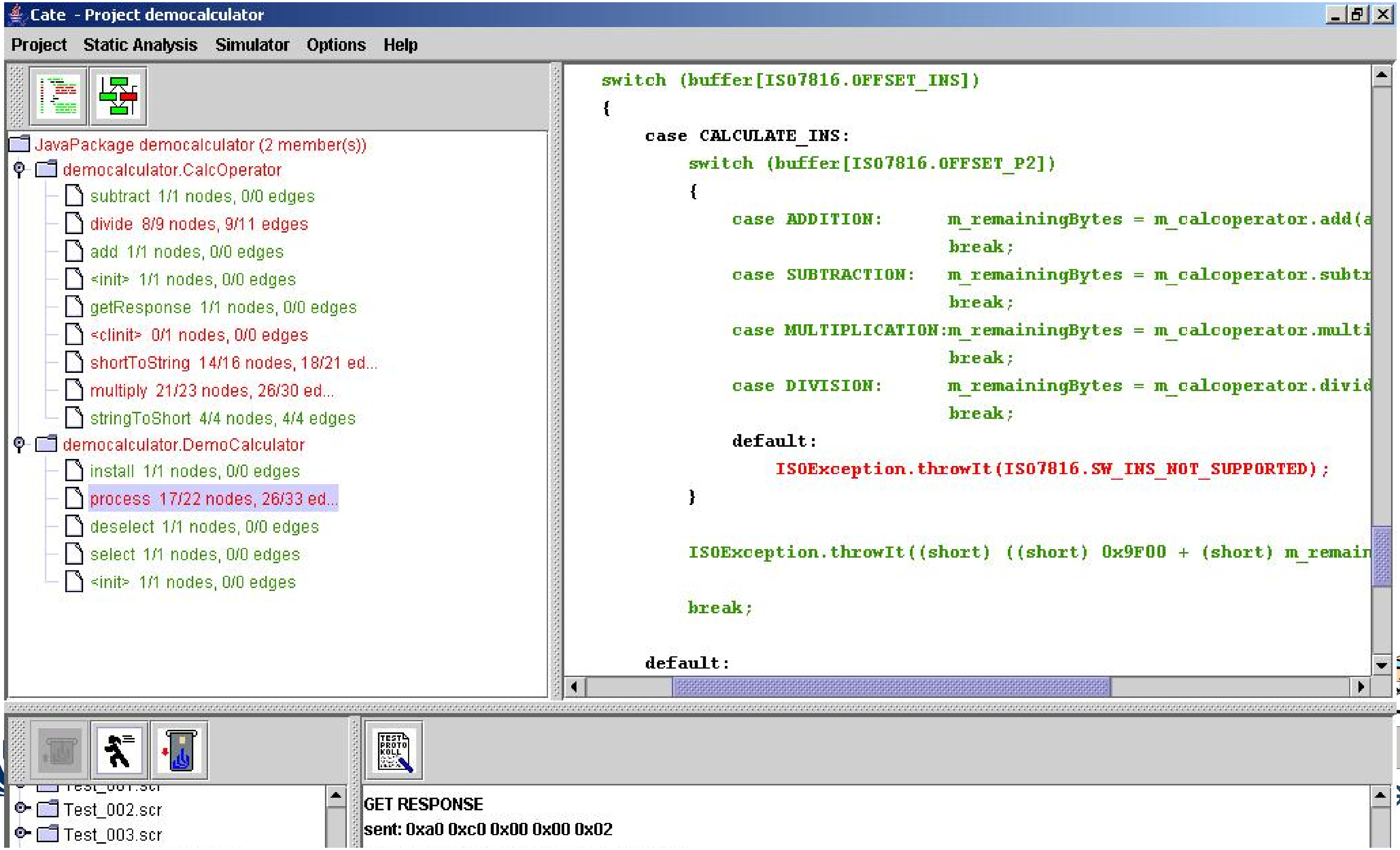
Code Coverage

Basic Block	**Executed**
B1	yes
B2	no
B3	yes
B4	no
B5	no
B6	yes

$$C_0 = 3/6 = 50\ \%$$

Dynamic Test Coverage Analysis

Results of dynamic analysis presented by Cate



- JavaPackage democalculator (2 member(s))
  - democalculator.CalcOperator
    - subtract 1/1 nodes, 0/0 edges
    - divide 8/9 nodes, 9/11 edges
    - add 1/1 nodes, 0/0 edges
    - <init> 1/1 nodes, 0/0 edges
    - getResponse 1/1 nodes, 0/0 edges
    - <clinit> 0/1 nodes, 0/0 edges
    - shortToString 14/16 nodes, 18/21 ed...
    - multiply 21/23 nodes, 26/30 ed...
    - stringToShort 4/4 nodes, 4/4 edges
  - democalculator.DemoCalculator
    - install 1/1 nodes, 0/0 edges
    - process 17/22 nodes, 26/33 ed...
    - deselect 1/1 nodes, 0/0 edges
    - select 1/1 nodes, 0/0 edges
    - <init> 1/1 nodes, 0/0 edges

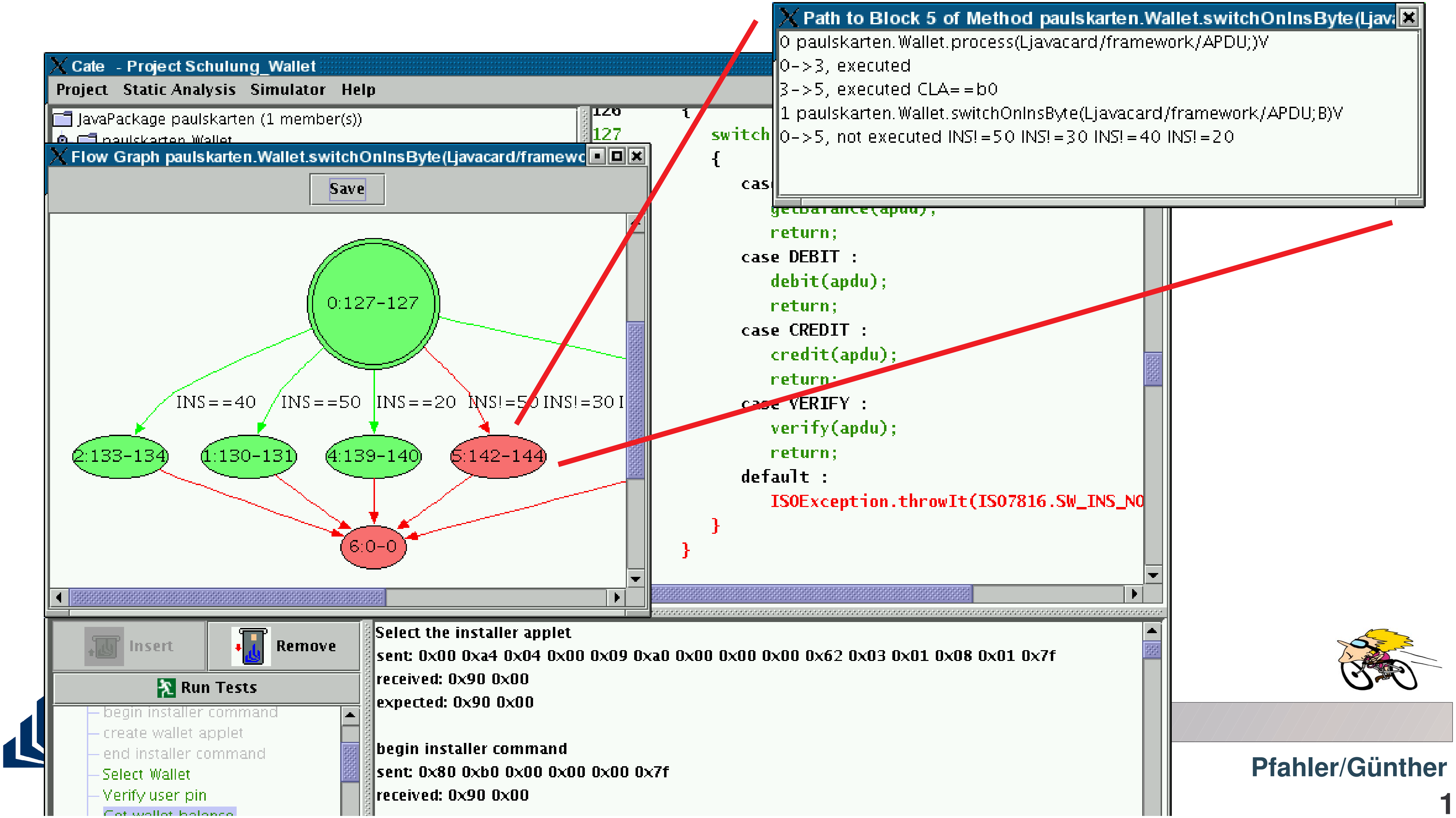
```
switch (buffer[ISO7816.OFFSET_INS])
{
    case CALCULATE_INS:
        switch (buffer[ISO7816.OFFSET_P2])
        {
            case ADDITION:      m_remainingBytes = m_calcoperator.add(a
                                break;
            case SUBTRACTION:   m_remainingBytes = m_calcoperator.subtr
                                break;
            case MULTIPLICATION:m_remainingBytes = m_calcoperator.multi
                                break;
            case DIVISION:      m_remainingBytes = m_calcoperator.divid
                                break;
            default:
                ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
        }

        ISOException.throwIt((short) ((short) 0x9F00 + (short) m_remain

        break;

    default:
```

- Test_001.scr
- Test_002.scr
- Test_003.scr

```
GET RESPONSE
sent: 0xa0 0xc0 0x00 0x00 0x02
```

Combining the results of static and dynamic analyzes

Support for the construction of new test cases

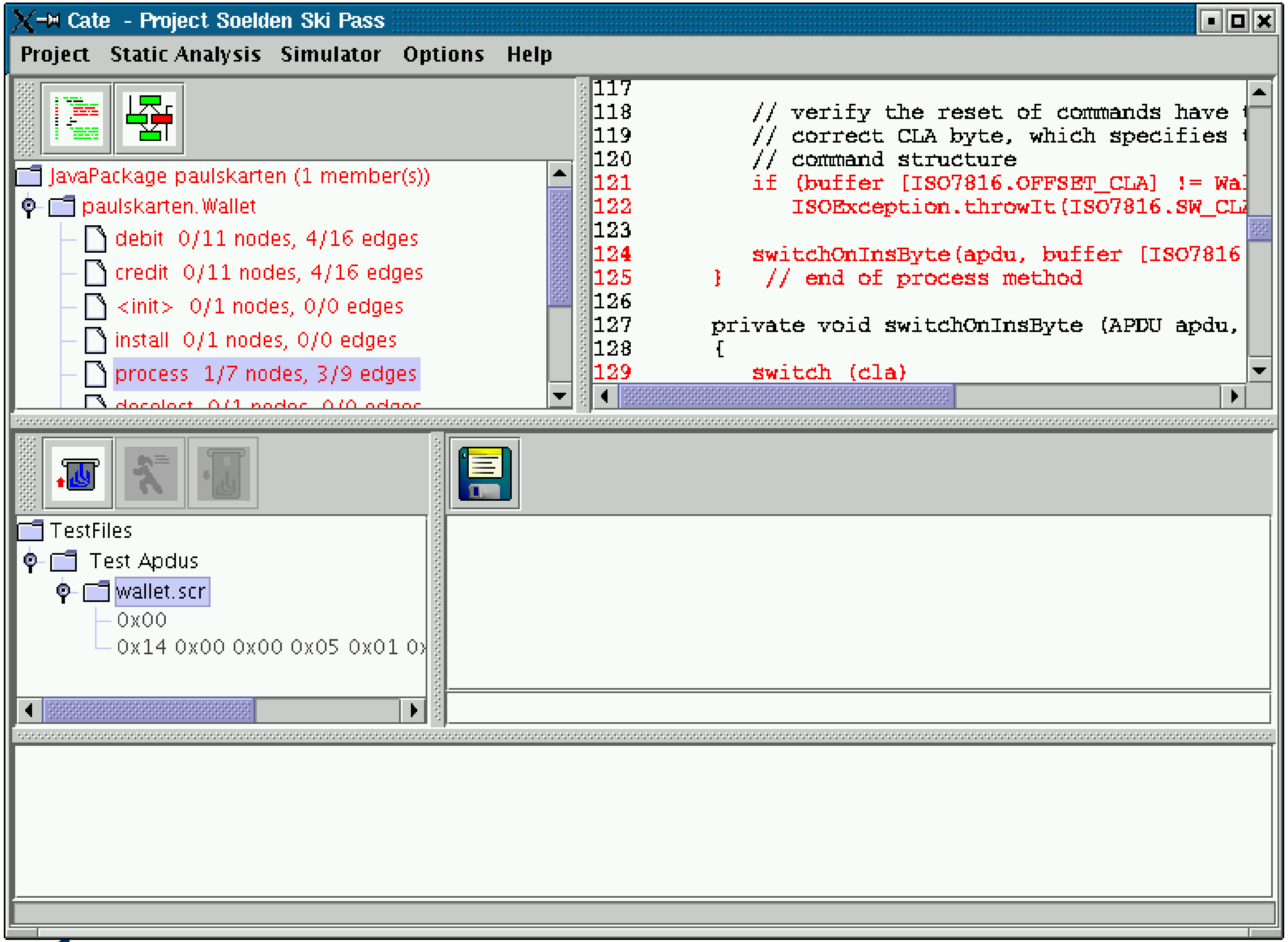


```
0 paulskarten.Wallet.process(Ljavacard/framework/APDU;)V
0->3, executed
3->5, executed CLA==b0
1 paulskarten.Wallet.switchOnInsByte(Ljavacard/framework/APDU;B)V
0->5, not executed INS!=50 INS!=30 INS!=40 INS!=20
```

```
126    {
127      switch
         {
           cas
              getBalance(apdu);
              return;
           case DEBIT :
              debit(apdu);
              return;
           case CREDIT :
              credit(apdu);
              return;
           case VERIFY :
              verify(apdu);
              return;
           default :
              ISOException.throwIt(ISO7816.SW_INS_NO
         }
       }
```

```
Select the installer applet
sent: 0x00 0xa4 0x04 0x00 0x09 0xa0 0x00 0x00 0x00 0x62 0x03 0x01 0x08 0x01 0x7f
received: 0x90 0x00
expected: 0x90 0x00

begin installer command
sent: 0x80 0xb0 0x00 0x00 0x00 0x7f
received: 0x90 0x00
```

Cate System Overview



- JavaPackage paulskarten (1 member(s))
  - paulskarten.Wallet
    - debit 0/11 nodes, 4/16 edges
    - credit 0/11 nodes, 4/16 edges
    - <init> 0/1 nodes, 0/0 edges
    - install 0/1 nodes, 0/0 edges
    - process 1/7 nodes, 3/9 edges
    - deselect 0/1 nodes, 0/0 edges

```

        // verify the reset of commands have t
        // correct CLA byte, which specifies t
        // command structure
        if (buffer [ISO7816.OFFSET_CLA] != Wal
            ISOException.throwIt(ISO7816.SW_CLA

        switchOnInsByte(apdu, buffer [ISO7816
    }   // end of process method

    private void switchOnInsByte (APDU apdu,
    {
        switch (cla)
```

- TestFiles
  - Test Apdus
    - wallet.scr
      - 0x00
      - 0x14 0x00 0x00 0x05 0x01 0x

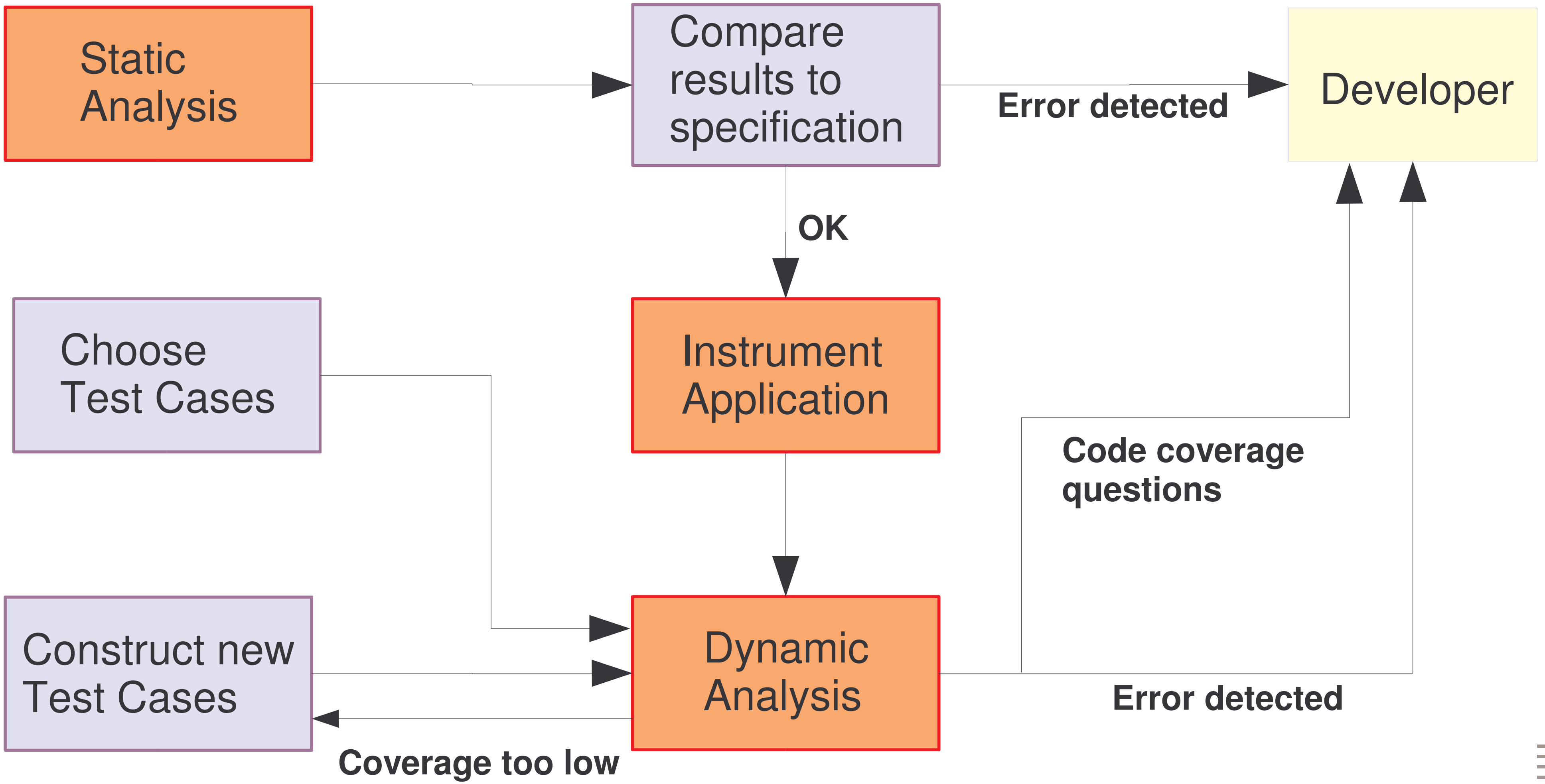
Static Analysis

- Project managment
- Source browser
- Control flow analysis
- CFG display
- Command/response

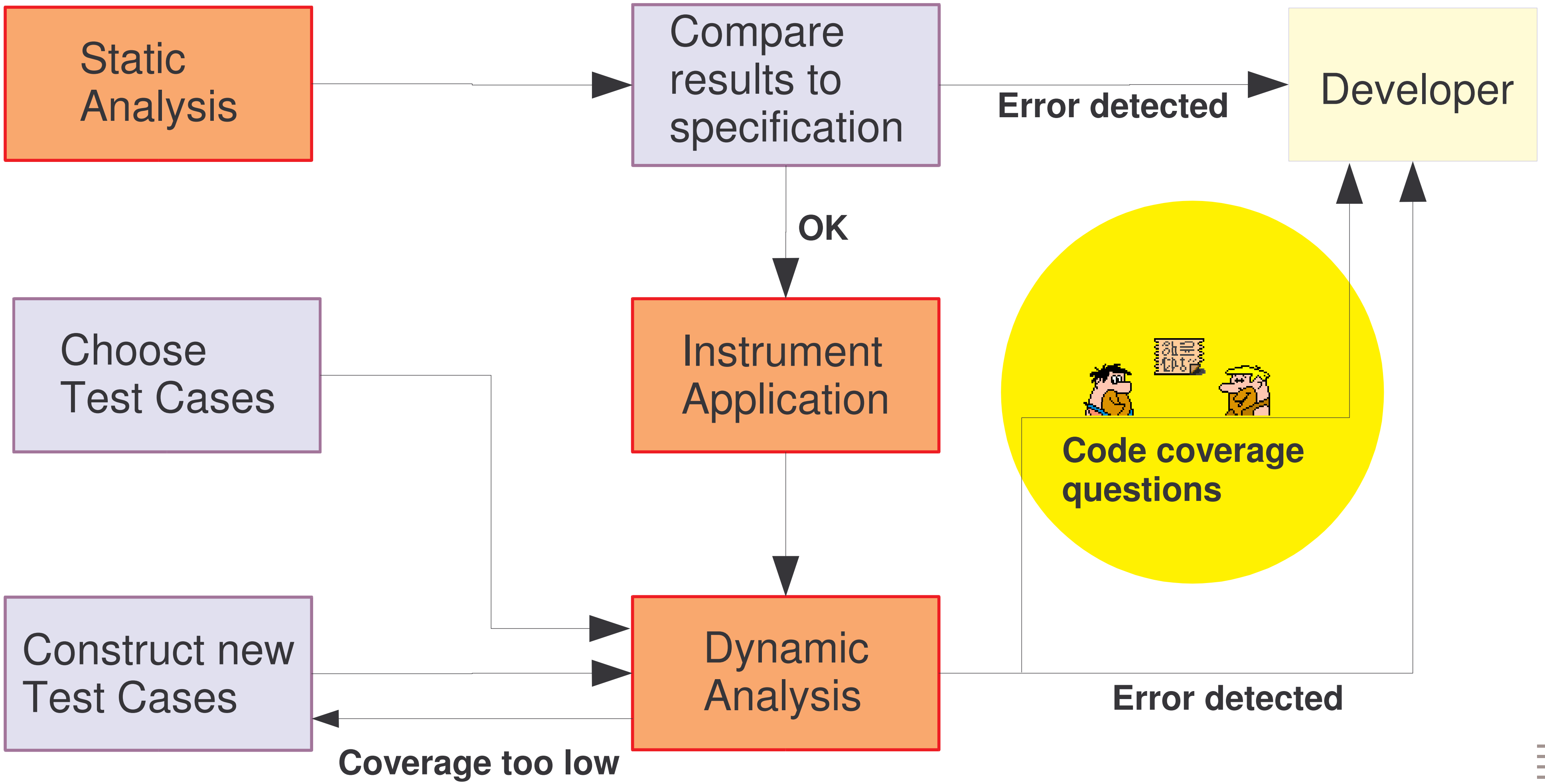
Dynamic Analysis

- Test browser
- Simulator control
- Test execution
- Test evaluation
- Coverage analysis

Applying the Cate System



Applying the Cate System



Static Analysis

Compare results to specification

Error detected

Developer

OK

Choose Test Cases

Instrument Application

Code coverage questions

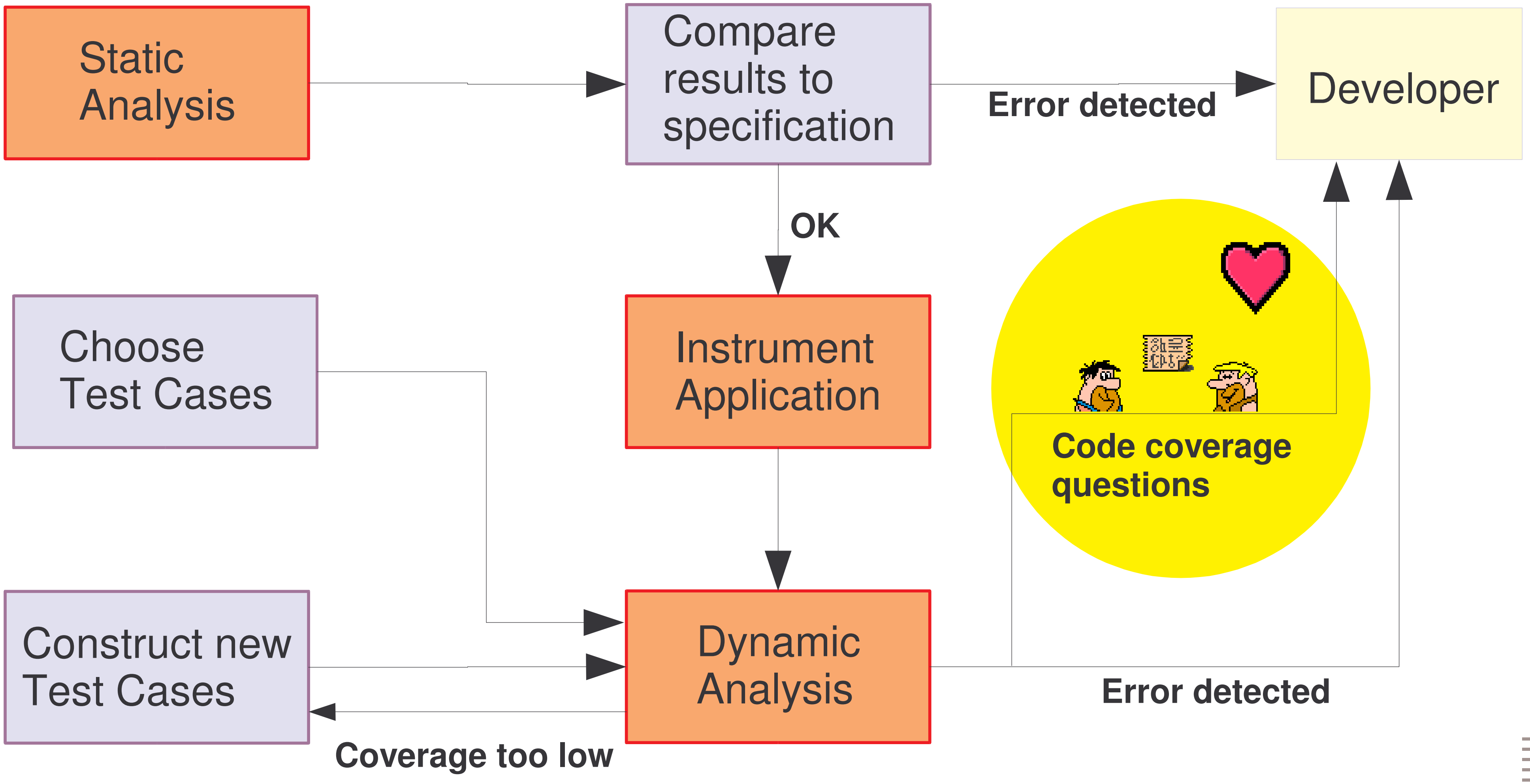
Construct new Test Cases

Dynamic Analysis

Error detected

Coverage too low

Applying the Cate System


Static Analysis

Compare results to specification

Error detected

Developer

OK

Choose Test Cases

Instrument Application

Code coverage questions

Construct new Test Cases

Dynamic Analysis

Error detected

Coverage too low

Summary

Cate: A System for Analysis and Test of Java Card Applications

Smart card basics:	Master/Slave, Java Card
Static Analysis:	Command-Response behavior
Dynamic Analysis:	Test coverage
The Cate System:	Practical experience