Replaying and Isolating Failing Multi-Object Interactions

Martin Burger • Andreas Zeller
Saarland University
• e-mail client written in Java
• 100,200 LOC
• ~ 1,600 Java classes
• 17 developers
• Actively developed from February 2001 to August 2007
Addressbook import failed! No contacts were added to your folder.

OK
Debugging 101

1. reproduce the original failure
   - manually by using GUI
   - test driver that reproduce faulty behavior
Debugging 101

1. reproduce the original failure
   - manually by using GUI
   - test driver that reproduce faulty behavior

2. fix the actual defect
   - focus on relevant behavior
   - simplify faulty behavior
Faulty Component Behavior

application

component

set of classes
Faulty Component Behavior

crosses border

interaction
- method calls
- field accesses
- exception flow
- ...

application

component
set of classes
Faulty Component Behavior

- method calls
- field accesses
- exception flow
- ...

application

crosses border

interaction

component

set of classes

failure
Reproducing

component
Reproducing

1. interaction

component
Reproducing

1. interaction

2. … component computes in the same way
Reproducing

1. interaction

2. … component computes in the same way

3. … failure reproduced
Simplifying
Simplifying many interactions
Simplifying

many interactions

replaying causes original failure
Simplifying minimal set of interactions by replaying causes original failure.
Simplifying

delta debugging
binary search

minimal set of interactions

replaying causes original failure
Implementation: JINSI

- debugging tool to reproduce and simplify failing components in Java programs
- reproduce capture and replay of interactions at component level
- simplify delta debugging minimizes interactions

Isolating Relevant Component Interactions with JINSI.
Orso + Joshi + Burger + Zeller
WODA 2006
Original Import Run

Columba

address book importer (GUI)
Capture

Columba

JINSI

instrumentation

address book importer (GUI)
Capture

address book importer (GUI)

“incoming call”
- method name
- arguments

written to event log

Columba

JINSI instrumentation
Replay

Columba

address book importer (GUI)
Replay

JINSI replaces Columba

address book importer (GUI)
Replay

JINSI replaces Columba

address book importer (GUI)

read from event log
Replay

address book importer (GUI)

JINSI replaces Columba

original failure reproduced

read from event log
Delta Debugging

JINSI uses delta debugging

address book importer (GUI)

event log
Delta Debugging

AddressbookImporter()

address book importer (GUI)

JINSI uses delta debugging

event log
Delta Debugging

AddressbookImporter()

init(file)

address book importer (GUI)

JINSI uses delta debugging

event log
JINSI uses delta debugging

AddressbookImporter()

init(file)

wizardFinished()
Delta Debugging

Problem
- already minimal
- too high level

Solution
- much finer grained

JINSI uses delta debugging

AddressbookImporter()
init(file)
wizardFinished()
Inside the Component

black box
(JINSI 2006)

course grained view
Object Interactions

finer grained view
Object Interactions

object

finer grained view
Object Interactions

finer grained view

object
Object Interactions

finer grained view
Object Interactions

Border crossing interactions are captured by JINSI 2006.
Object Interactions

all object interactions are captured by JINSI 2008
Object Trace

all object interactions are captured by JINSI 2008

for all objects
Importing the Addresses
Object Trace

ContactModel()
setSortString()
setFormattedName()
setNickName()
setFamilyName()

addressbook.
model
(19 classes)

49,849 more
Importing the Addresses
Object Trace

ContactModel()
  setSortString()
  setFormattedName()
  setNickName()
  setFamilyName()

addressbook.

49,849 more

delta debugging:
> 2 hours

JINSI
Object Slice

- dynamic backward slice: subset that may have influenced a specific object
- focuses on objects, not on statements
- objects are a natural abstraction
Object Interactions
Object Slice

at first, include suspicious object

suspicious object
Object Slice

include interacting objects
include objects that interact with those already in slice
Object Slice

stop when reached largest connected graph
Object Slice

simplified set of failure-inducing interactions within the component
Object Slice

simplified set of failure-inducing interactions within the component
Importing Addresses

Object Slice

ContactModel()

setGivenName()

setFamilyName()

setSortString()

getSortString()

getGivenName()

getPreferredEmail()

addressbook.model

(19 classes)
Delta Debugging

apply delta debugging to object slice
Delta Debugging

original failure

minimized set of failure-inducing interactions within the component
Importing Addresses
Delta Debugging

ContactModel() → addressbook. model (19 classes) → getPreferredEmail()
Slicing + Delta Debugging
Incoming Interactions

49,855
original run

> 2 hours
delta debugging

2
minimized
Slicing + Delta Debugging

Incoming Interactions

49,855
original run

object slice

7
simplified

> 2 hours
delta debugging

2
minimized
Slicing + Delta Debugging
Incoming Interactions

49,855
original run

> 2 hours
delta debugging

object slice
7
simplified

> 2 hours
delta debugging
2
minimized
Slicing + Delta Debugging
Incoming Interactions

49,855 original run

< 1 min object slice

> 2 hours delta debugging

< 1 min simplified

7 delta debugging

2 minimized

2 minimized
Slicing + Delta Debugging

Incoming Interactions

49,855 original run

< 1 min object slice

7 simplified

< 1 min delta debugging

> 2 hours delta debugging

2 minimized

2 minimized
public String getPreferredEmail() {  
  Iterator it = getEmailIterator();

  // get first item
  IEmailModel model = (IEmailModel) it.next();

  // backwards compatibility -> its not possible
  // anymore to create a model without email
  if (model == null)  
    return null;

  return model.getAddress();
}
public String getPreferredEmail() {
    Iterator it = getEmailIterator();

    // get first item
    IEmailModel model = (IEmailModel) it.next();

    // backwards compatibility -> its not possible
    // anymore to create a model without email
    if (model == null) {
        return null;
    }

    return model.getAddress();
}
public String getPreferredEmail() {  
    Iterator it = getEmailIterator();

    // get first item
    IEmailModel model = null;
    if (it.hasNext())
        model = (IEmailModel) it.next();

    // anymore to create a model without email
    if (model == null)
        return null;

    return model.getAddress();
}
Related Work

- **Test Factoring** David Saff et al.
- **Selective Capture-Replay** Shrinivas Joshi et al.
  JINSI is based on these methods

- **Efficient Test Case Minimization** Andreas Leitner et al.
  method calls in random tests; static slicing

- **ReCrash** Shay Artzi et al.
  - ReCrash focuses on stack trace
  - JINSI focuses on object interactions
Future Work

application

failing but not defective component
Future Work

Application

Failing but not defective component
Future Work

When observing application: too many interactions failing but not defective component
Future Work

application

failing but not defective component
Future Work

Application

Failing but not defective component
Future Work

Cause-Effect-Chain

final state
e.g. contact model
Future Work

Cause-Effect-Chain

intermediate states final state
e.g. contact model
Future Work

Cause-Effect-Chain

input
e.g. keystrokes in GUI

intermediate states

final state
e.g. contact model
Future Work

Capture Component Level

JINSI captures component interaction

event log
Future Work

Replay Component Level

JINSI replays component interaction...

event log
Future Work

Replay Component Level

JINSI replays component interaction...

event log
Future Work

Replay Component + Capture Objects

JINSI replays component interaction...

for all objects

... and captures object interactions

event log

object trace
Challenges

• capture constructors
  - super call
  - constructor call as argument

• who is calling
  - caller registry to get caller of method
Debugging 101

1. reproduce the original failure
   - manually by using GUI
   - test driver that reproduce faulty behavior

2. fix the actual defect
   - focus on relevant behavior
   - simplify faulty behavior
Debugging 101

1. reproduce the original failure
   - manually by using GUI
   - test driver that reproduce faulty behavior

2. fix the actual defect
   - focus on relevant behavior
   - simplify faulty behavior
Debugging 101

1. reproduce the original failure - manually by using GUI
   - test driver that reproduce faulty behavior

2. fix the actual defect - focus on relevant behavior
   - simplify faulty behavior
Contributions
Debugging 101

1. reproduce the original failure
   - manually by using GUI
   - test driver that reproduce faulty behavior
2. fix the actual defect
   - focus on relevant behavior
   - simplify faulty behavior

Contributions
Debugging 101

1. reproduce the original failure
   - manually by using GUI
   - test driver that reproduce faulty behavior

2. fix the actual defect
   - focus on relevant behavior
   - simplify faulty behavior

Object Trace

all object interactions are captured by JINSI 2008

contributions
Debugging 101

1. reproduce the original failure
   - manually by using GUI
   - test driver that reproduce faulty behavior

2. fix the actual defect
   - focus on relevant behavior
   - simplify faulty behavior

Object Trace

all object interactions are captured by JINSI 2008

Object Slice

simplified set of failure-inducing interactions within the component
**Debugging 101**

1. reproduce the original failure
   - manually by using GUI
   - test driver that reproduce faulty behavior

2. fix the actual defect
   - focus on relevant behavior
   - simplify faulty behavior

**Object Trace**

- all object interactions are captured by JINSI 2008
- event log object trace

**Object Slice**

- simplified set of failure-inducing interactions within the component

**Slicing + Delta Debugging**

- incoming interactions
- 49,855 original run
- > 2 hours delta debugging
- < 1 min object slice
- < 1 min delta debugging
- minimized