Replaying and Isolating Failing Multi-Object Interactions

Martin Burger • Andreas Zeller
Saarland University

Columba

File Edit View Utilities Help

Add Contact Add Group Properties Remove

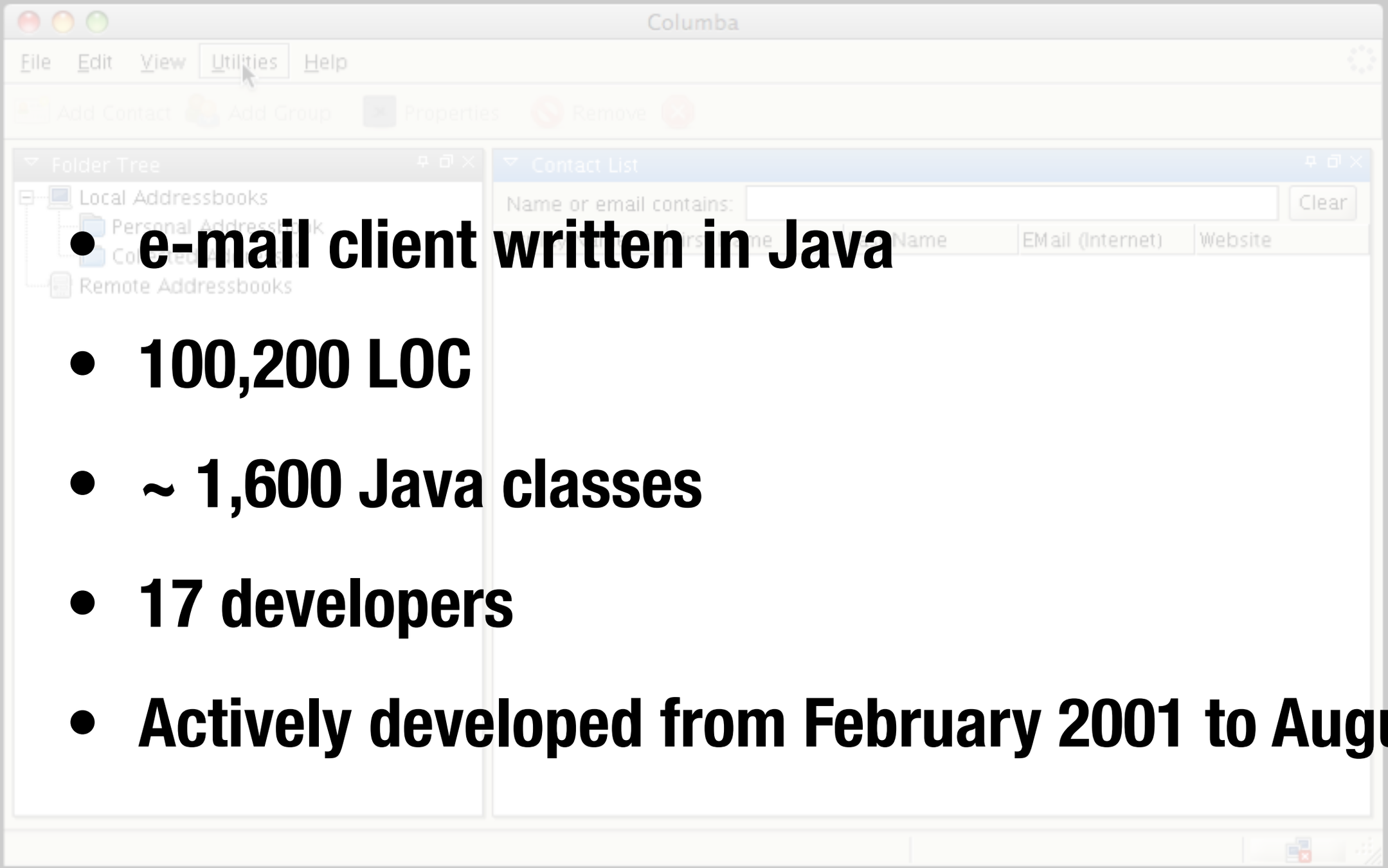
Folder Tree

- Local Addressbooks
  - Personal Addressbook
  - Collected Addresses
- Remote Addressbooks

Contact List

Name or email contains: Clear

Display Name	First Name	Last Name	EMail (Internet)	Website



- e-mail client written in Java
- 100,200 LOC
- ~ 1,600 Java classes
- 17 developers
- Actively developed from February 2001 to August 2007

Columba

File Edit View Utilities Help

Add Contact Add Group Properties Remove

Folder Tree

- Local Addressbooks
  - Personal Addressbook
  - Collected Addresses
- Remote Addressbooks

Contact List

Name or email contains: Clear

Display Name	First Name	Last Name	EMail (Internet)	Website

Columba

File Edit View Utilities Help

Add Contact Add Group Properties Remove

Folder Tree

- Local Addressbooks
  - Personal Addressbook
  - Collected Addresses
- Remote Addressbooks

Contact List

Name or email contains: Clear

Display Name	First Name	Last Name	EMail (Internet)	Website

Addressbook import failed! No contacts were added to your folder.

OK

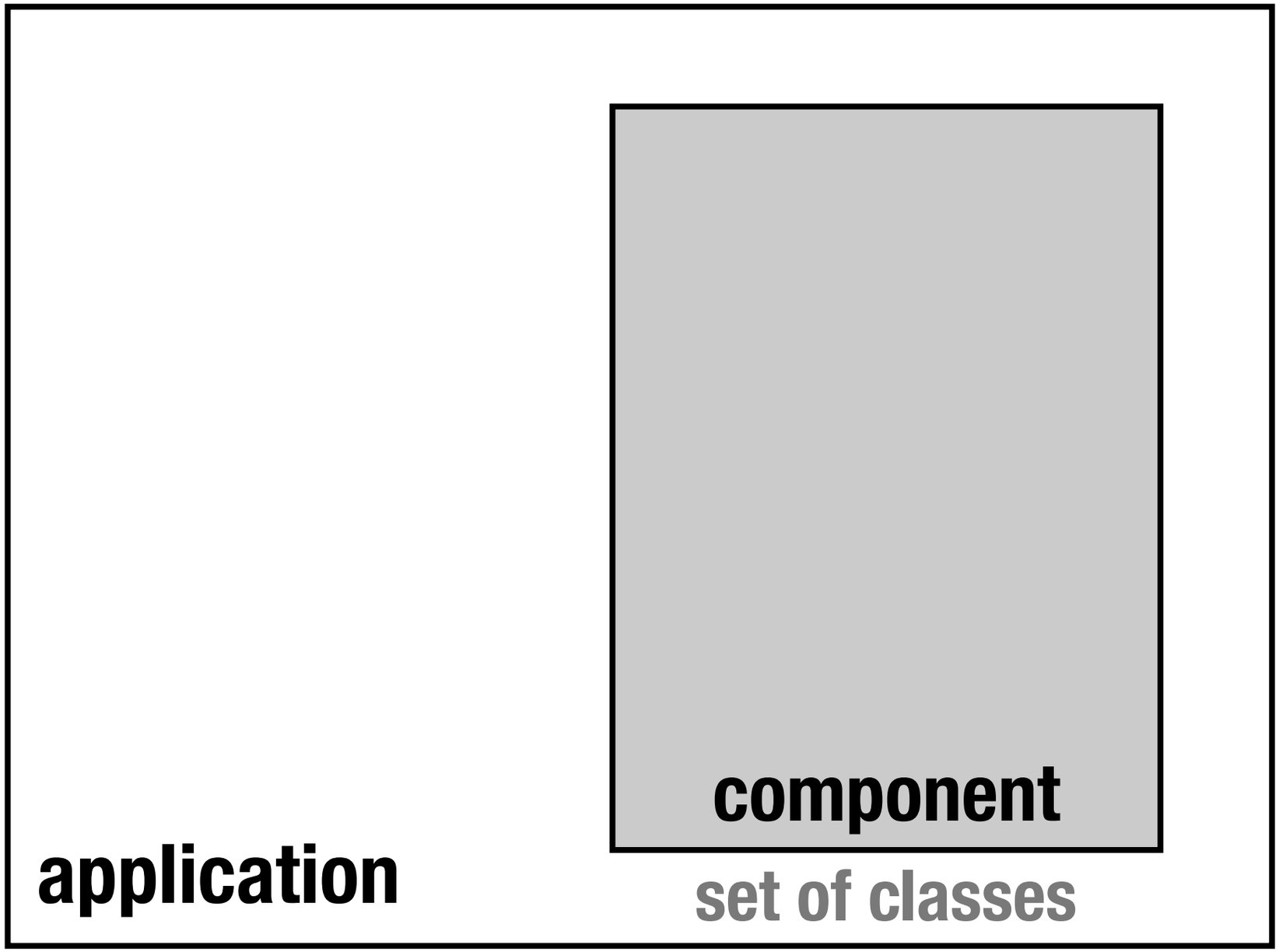
Debugging 101

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

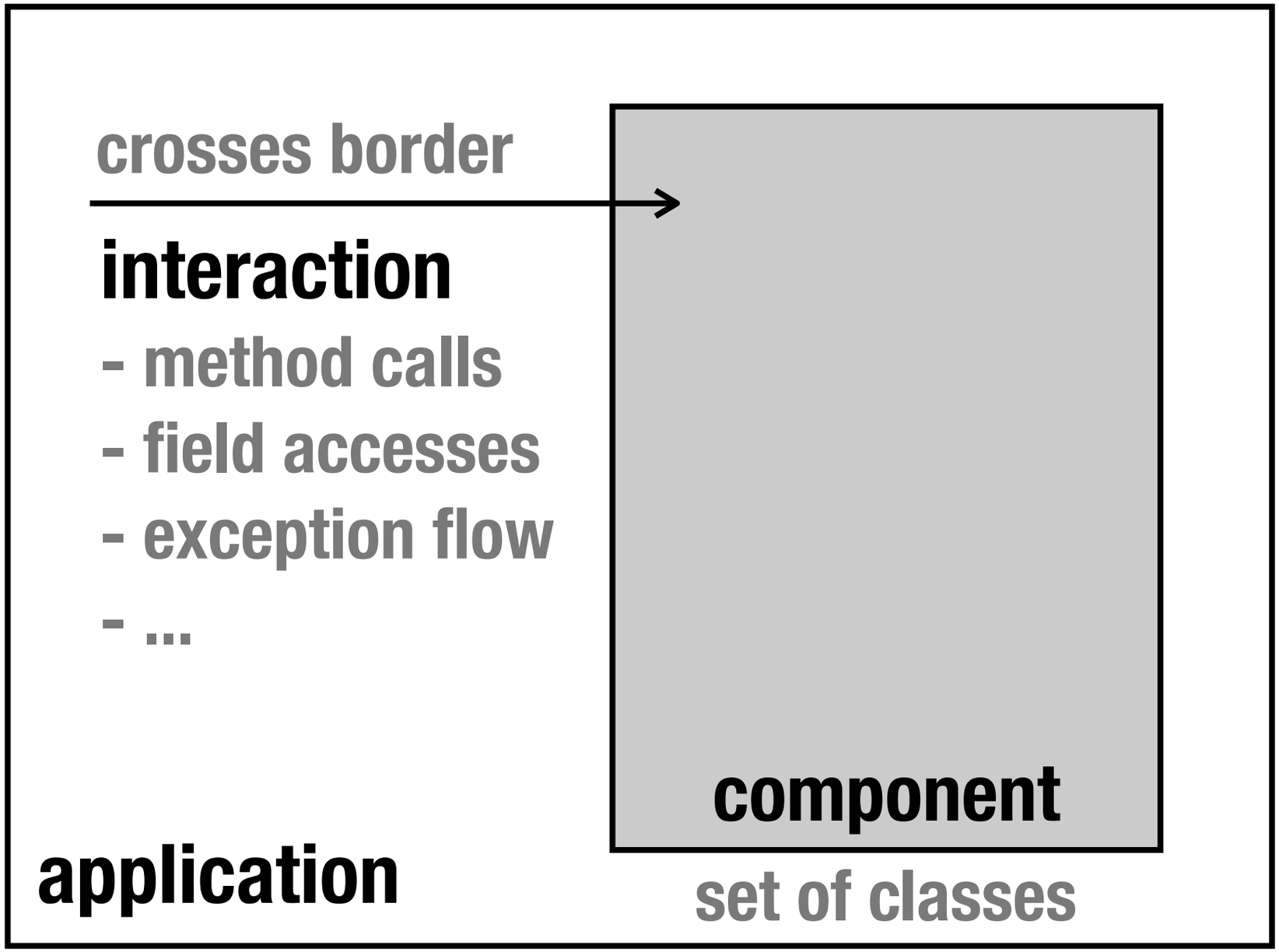
Debugging 101

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

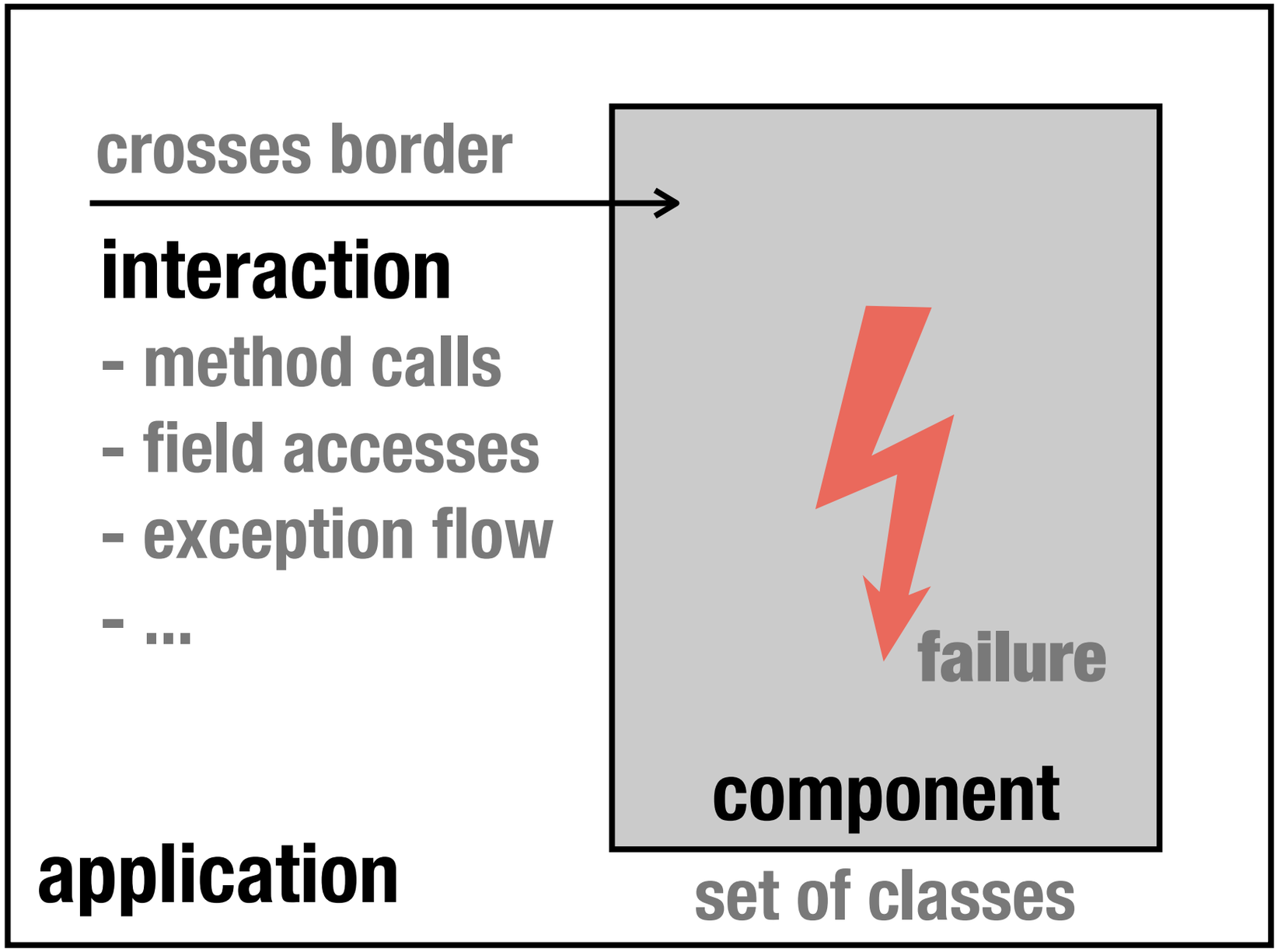
Faulty Component Behavior



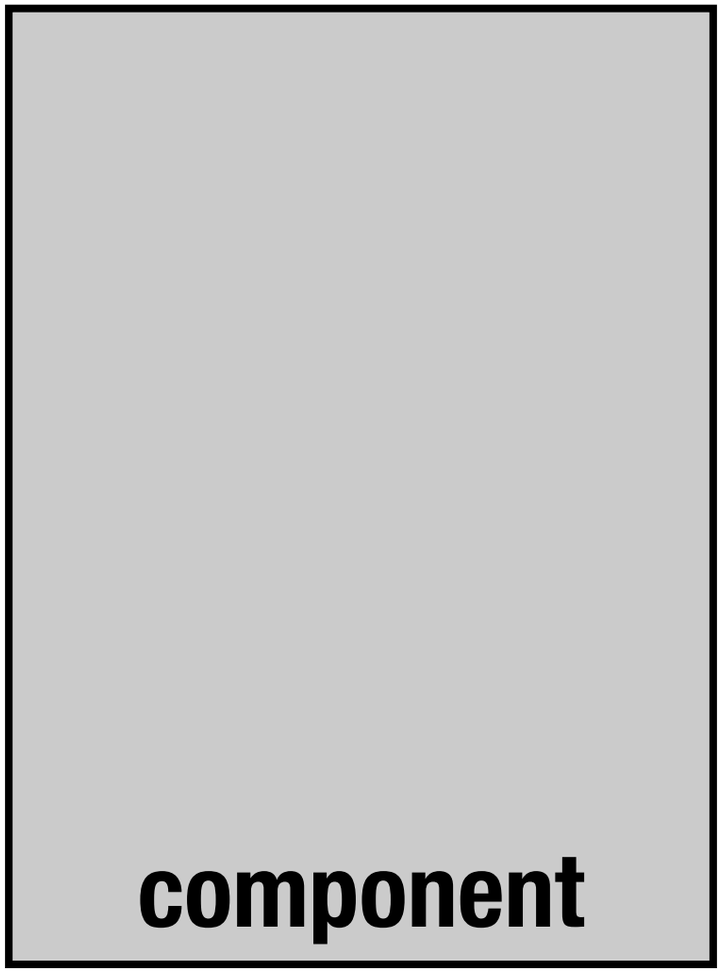
Faulty Component Behavior



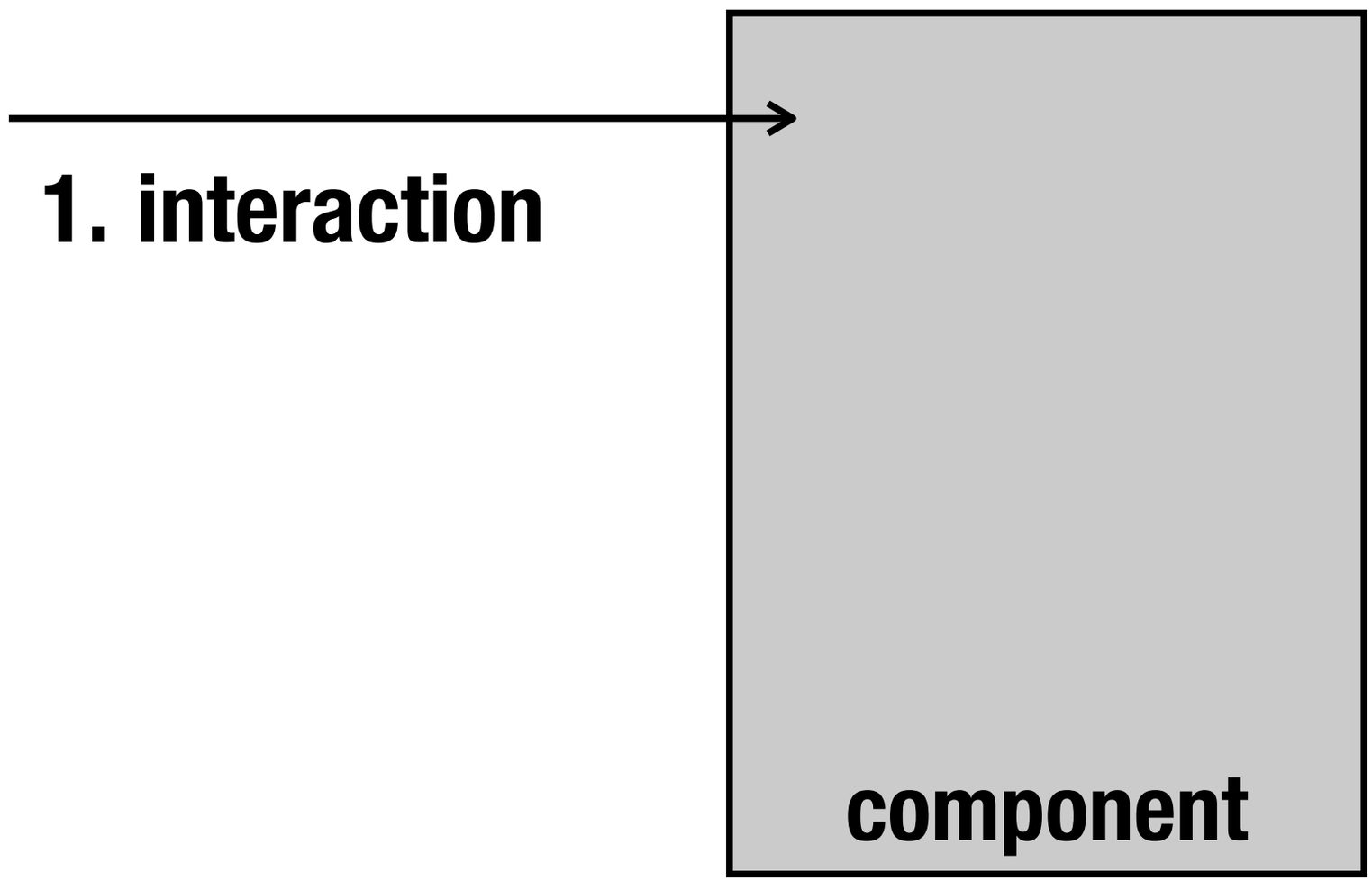
Faulty Component Behavior



Reproducing



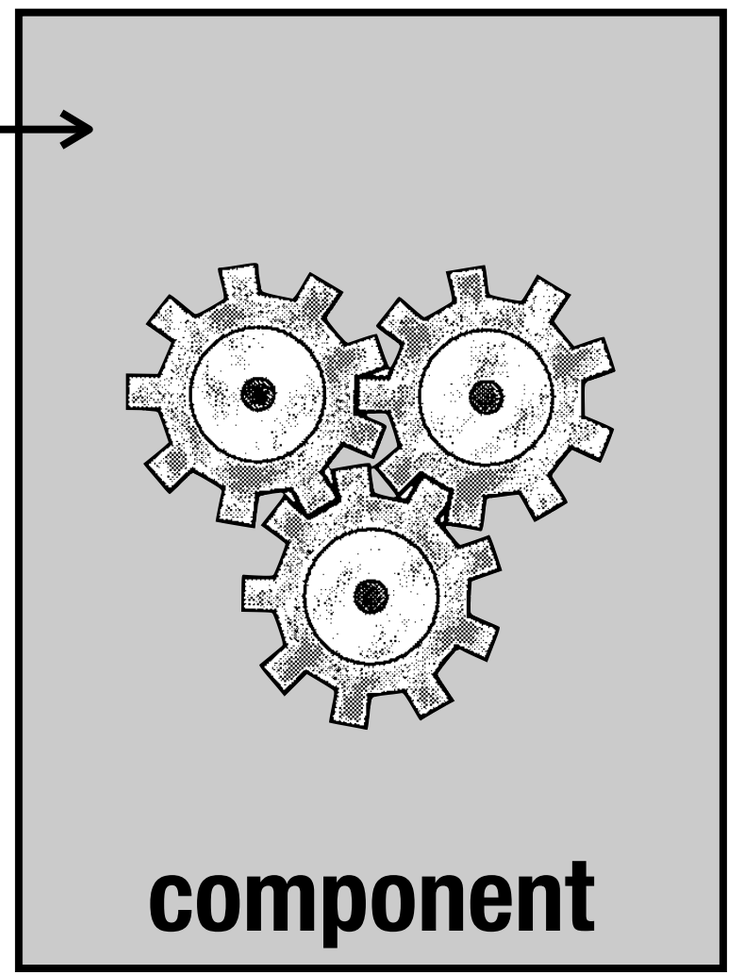
Reproducing



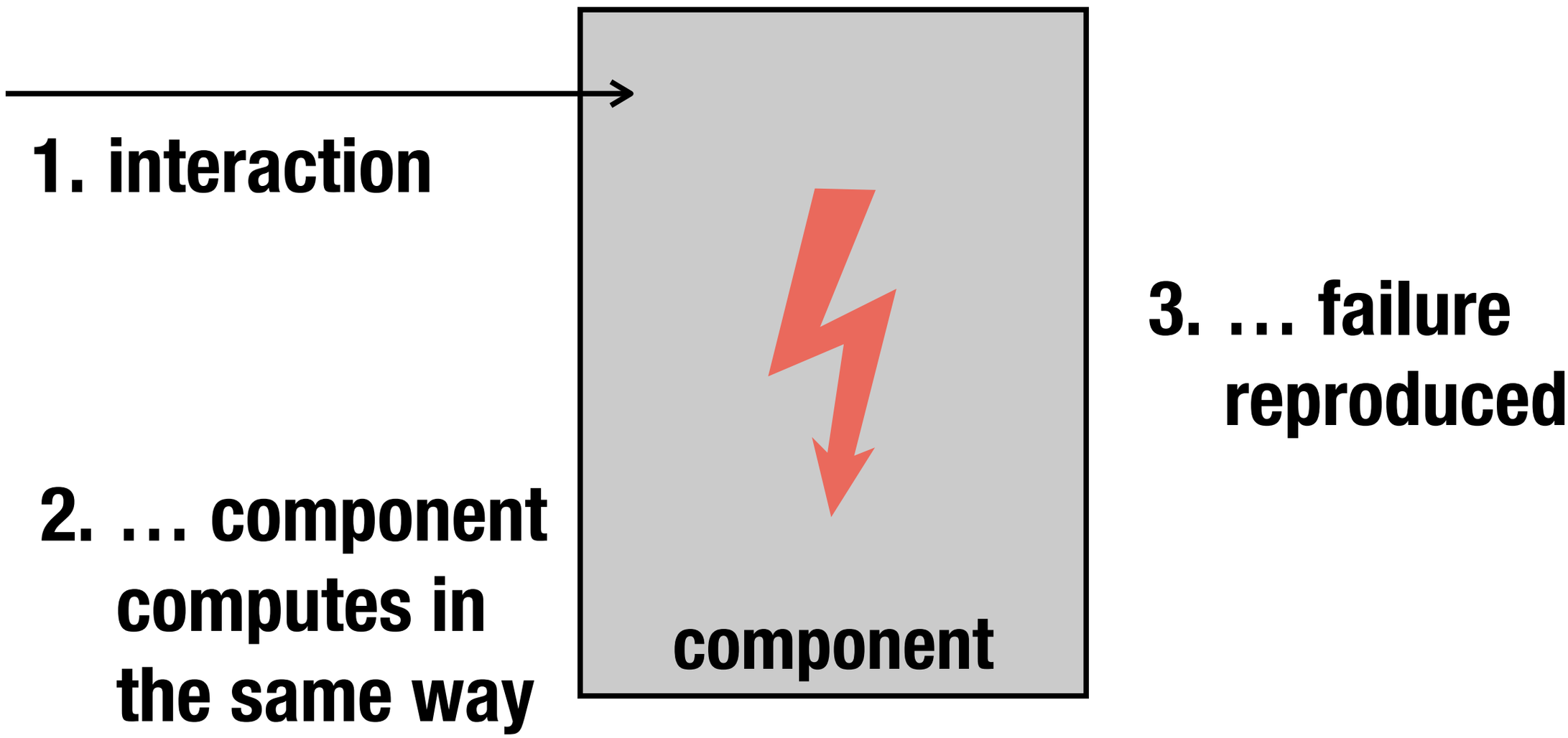
Reproducing

1. interaction

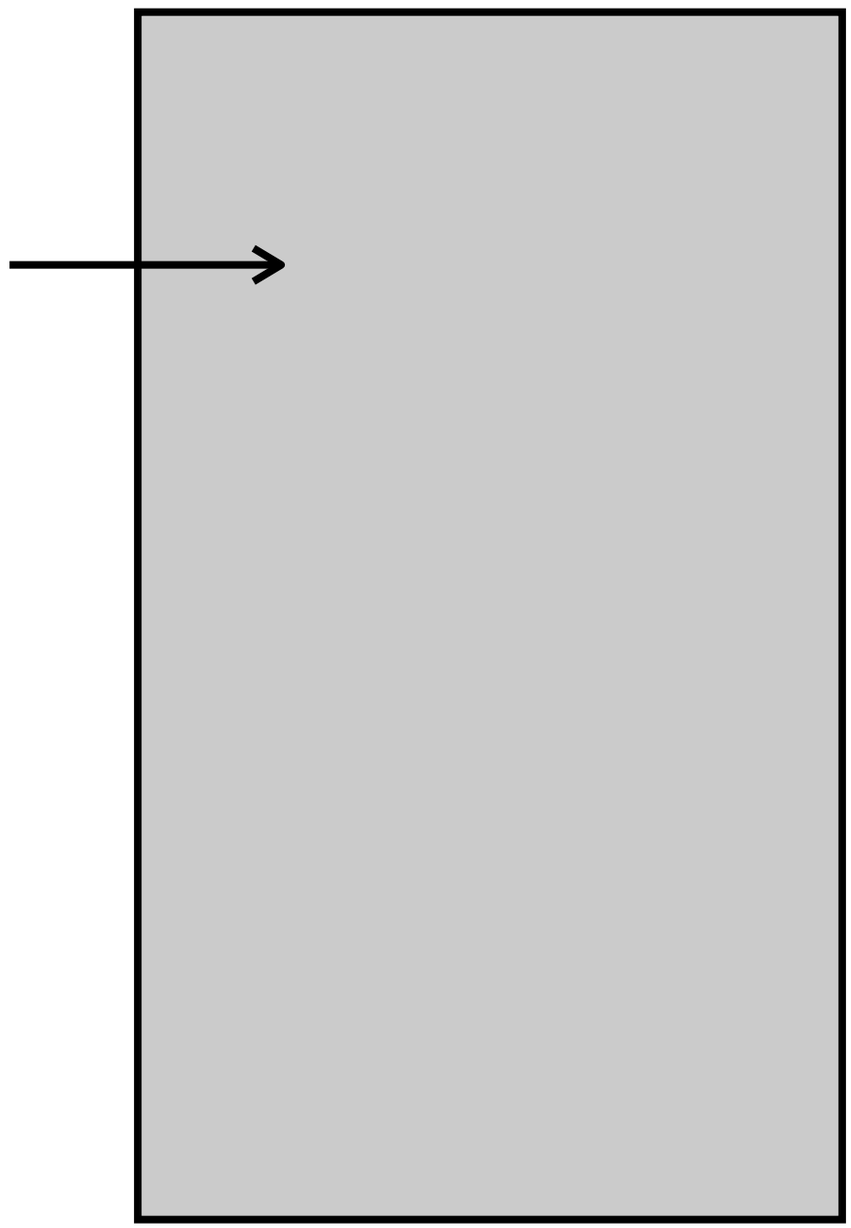
2. … component computes in the same way



Reproducing

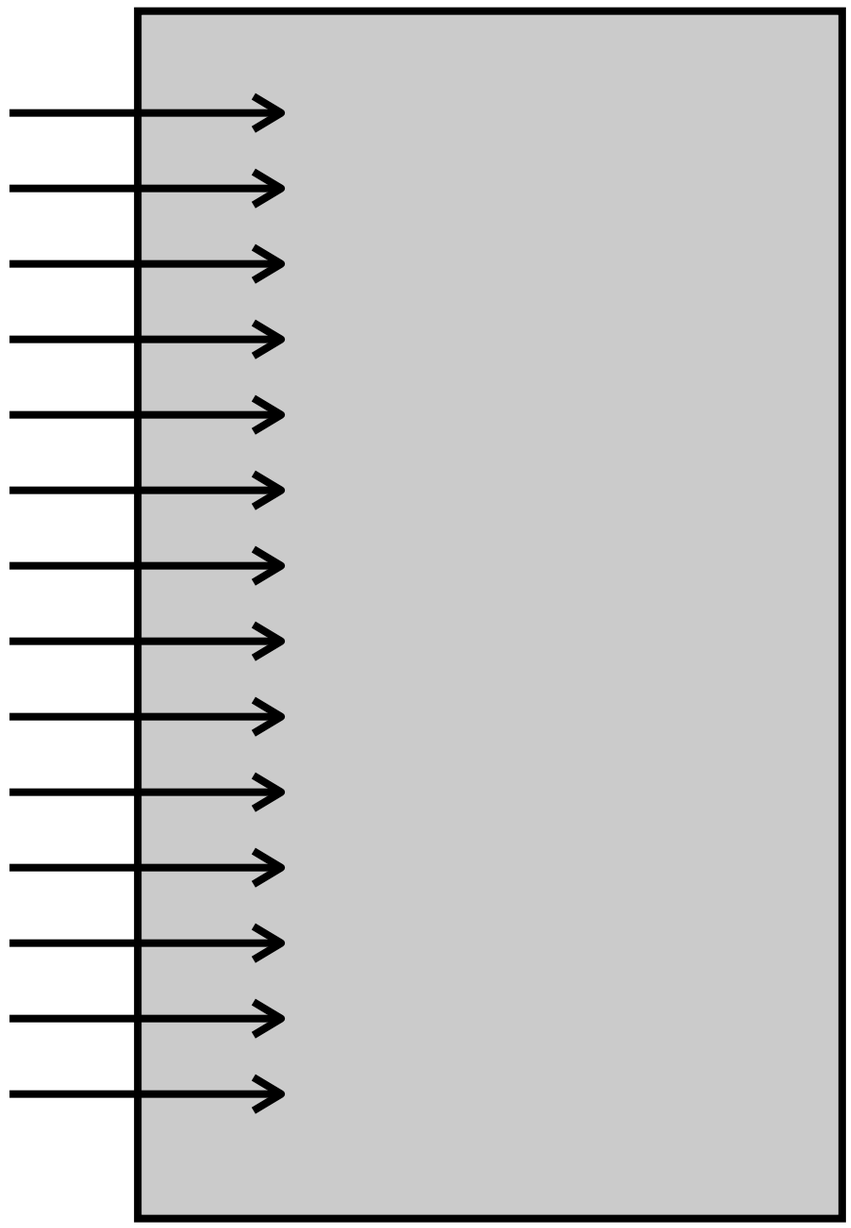


Simplifying



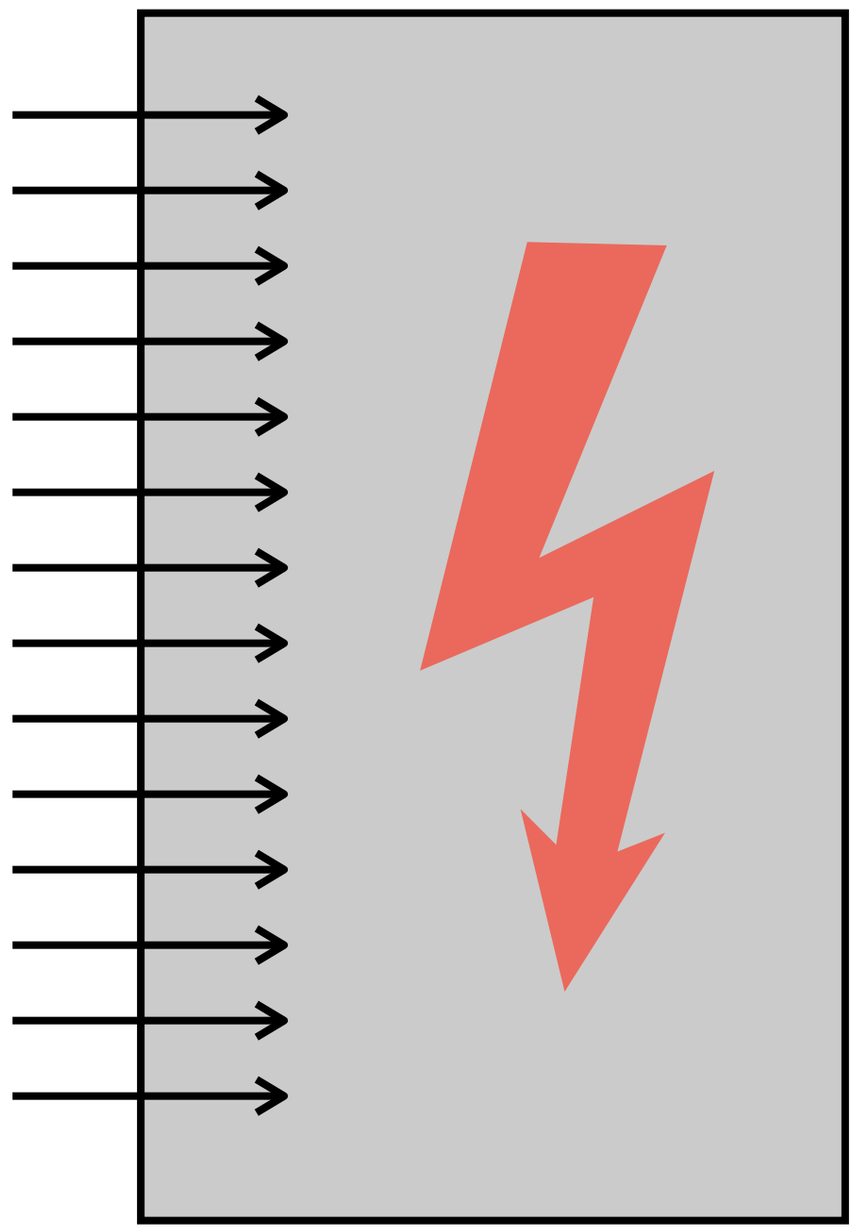
Simplifying

many
interactions



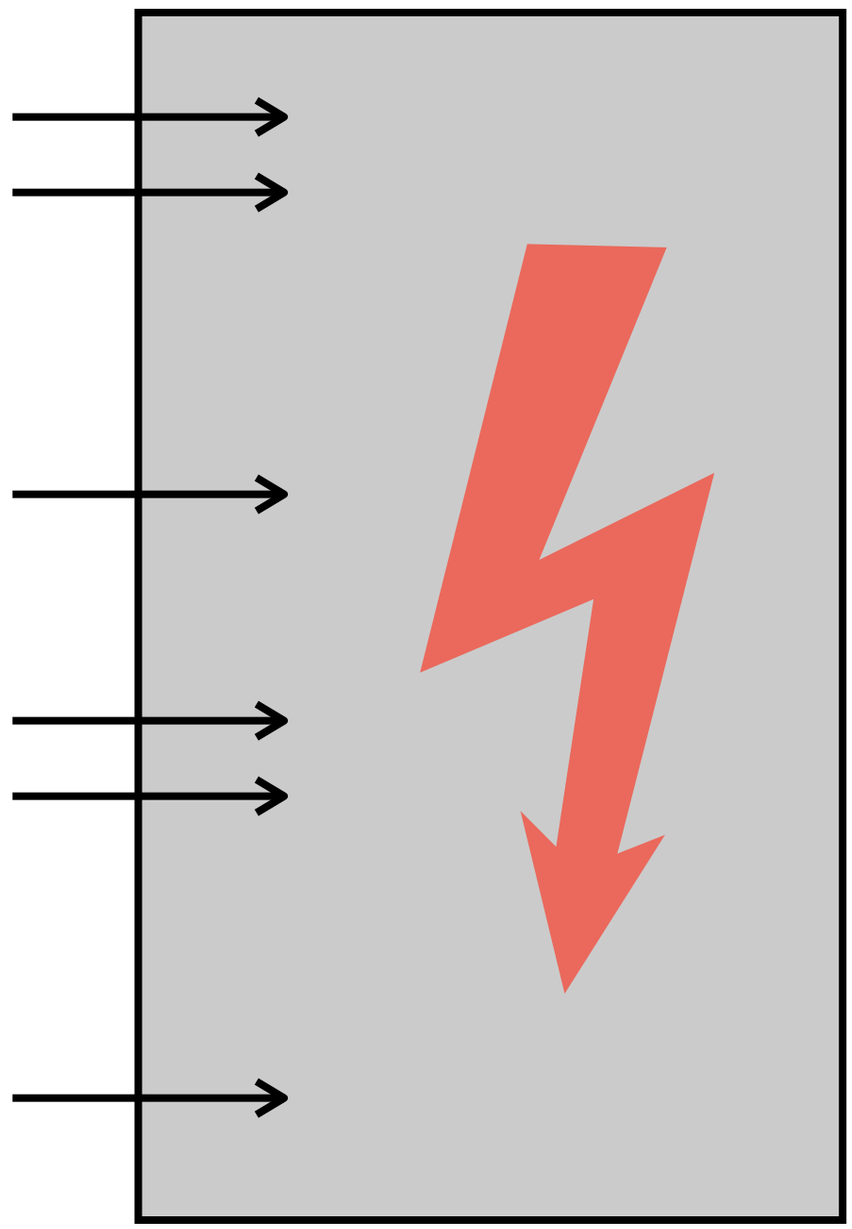
Simplifying

many interactions



replaying causes original failure

Simplifying

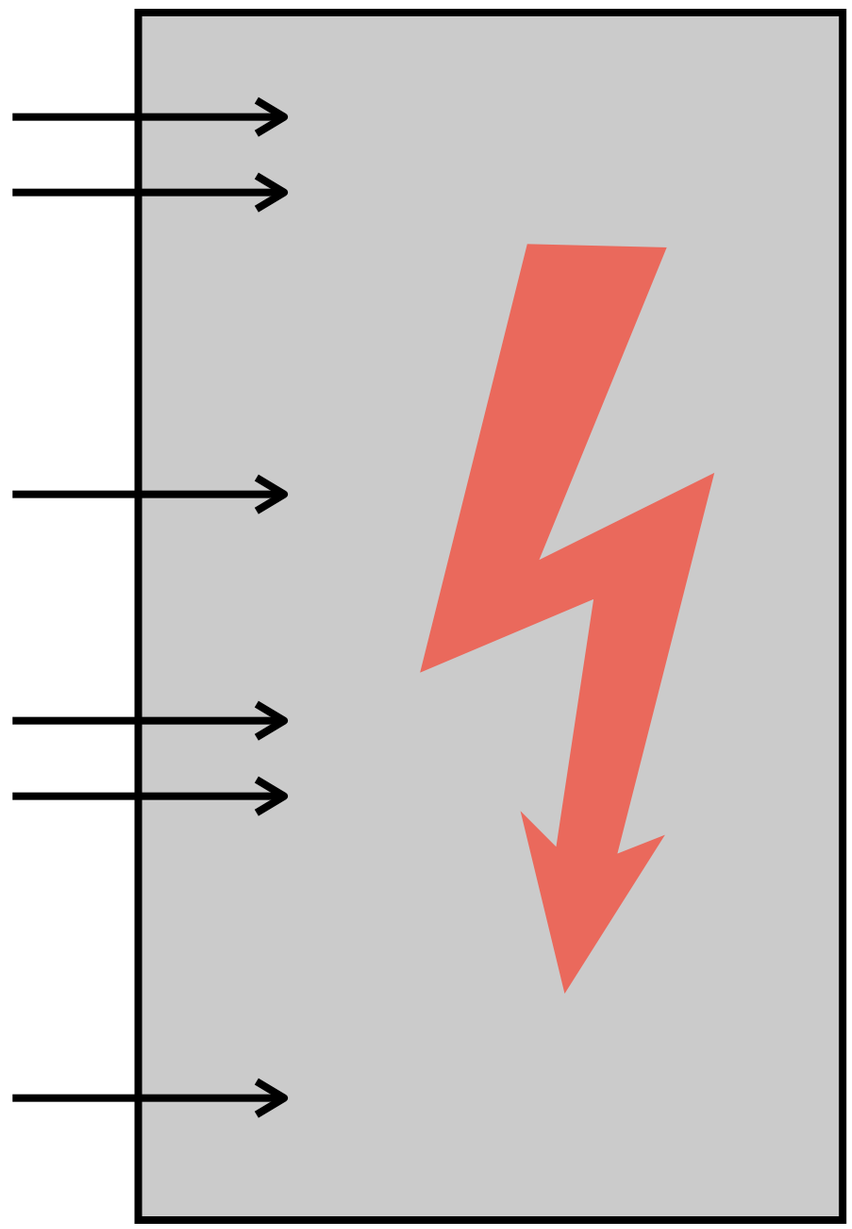


minimal set of interactions

replaying causes original failure

Simplifying

delta debugging
binary search



minimal set of interactions

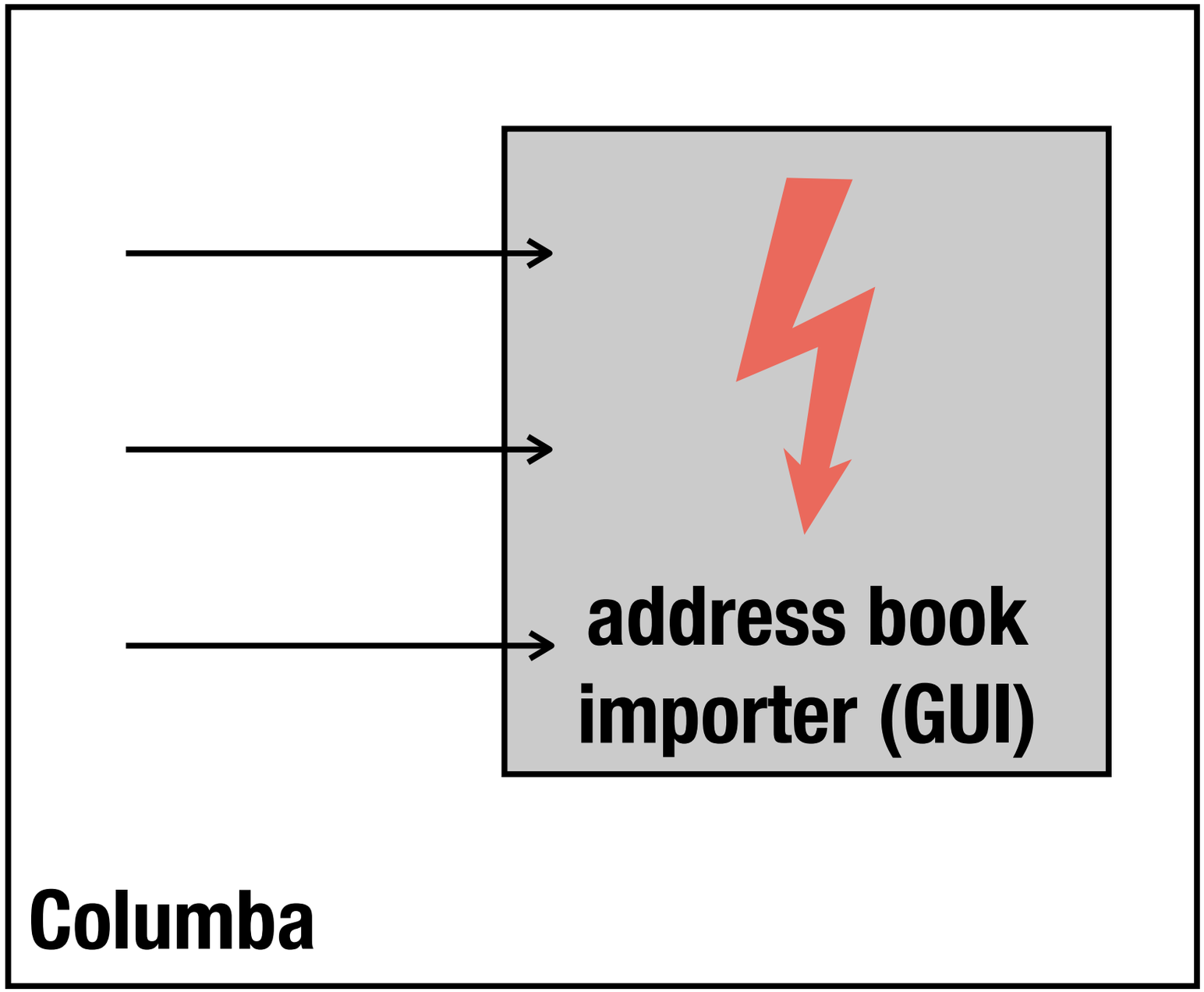
replaying causes original failure

Implementation: JINSI

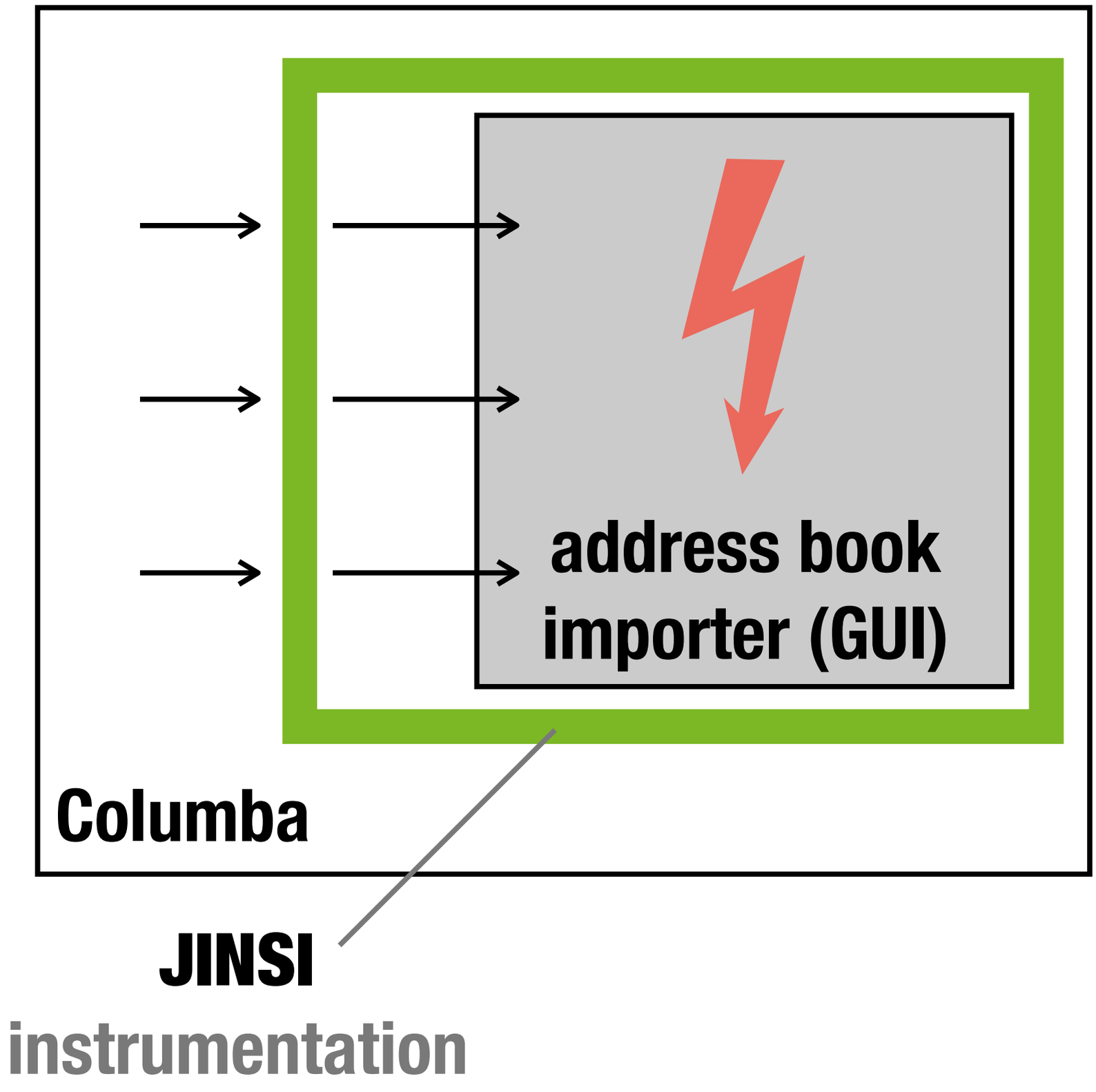
- debugging tool to reproduce and simplify failing components in Java programs
- **reproduce**
  capture and replay of interactions at component level
- **simplify**
  delta debugging minimizes interactions

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

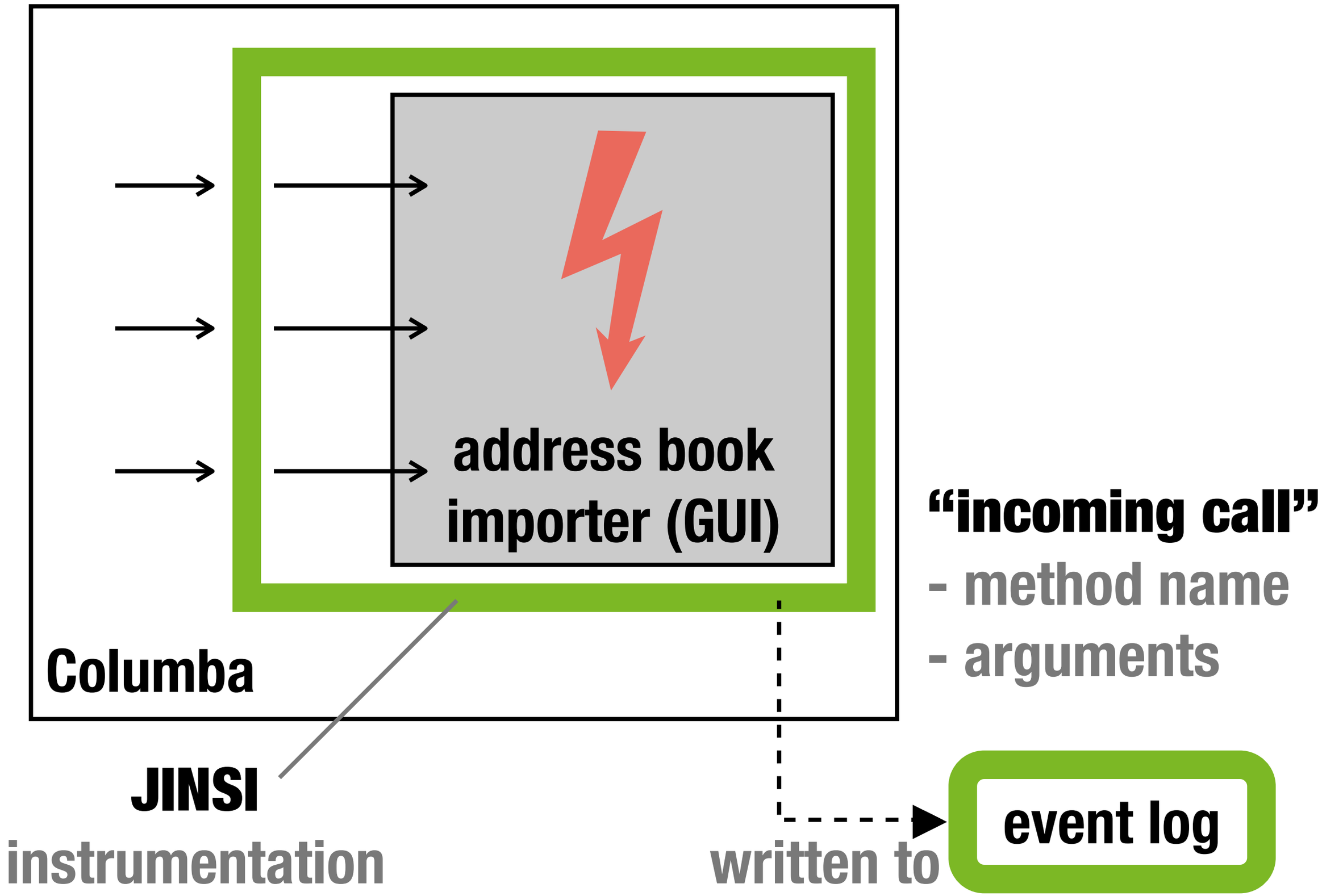
Original Import Run



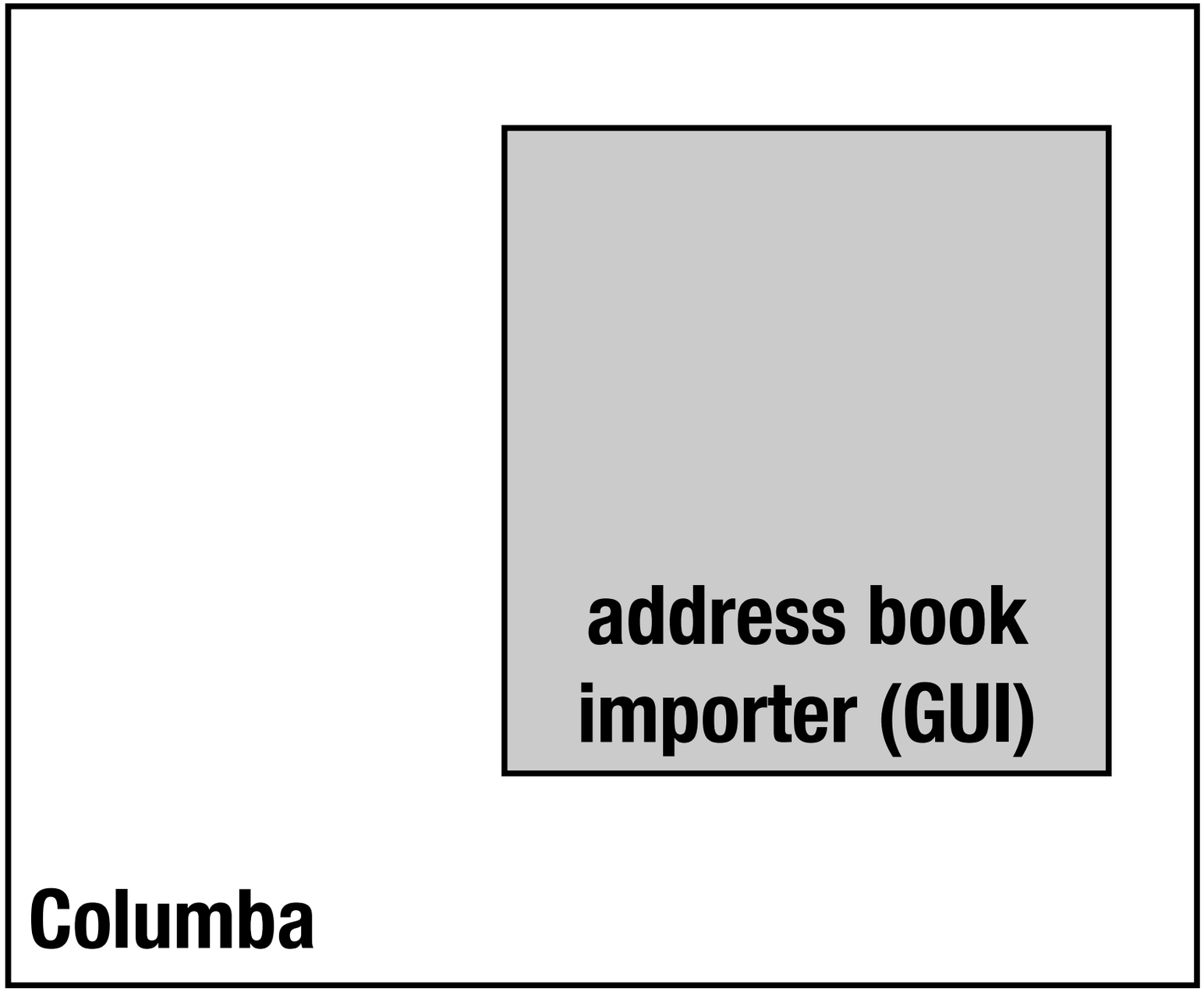
Capture



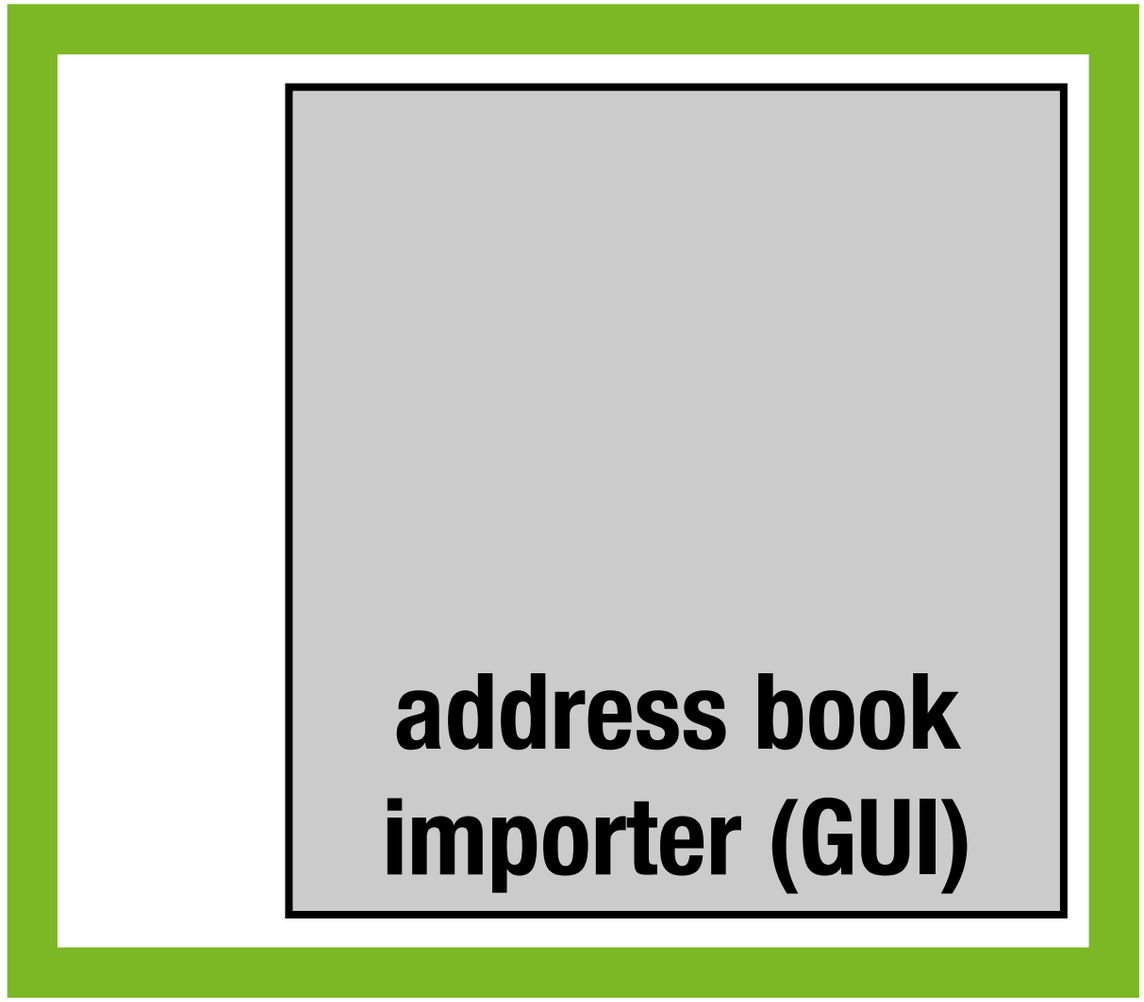
Capture


address book
importer (GUI)
Columba
"incoming call"
- method name
- arguments
JINSI
instrumentation
written to
event log

Replay



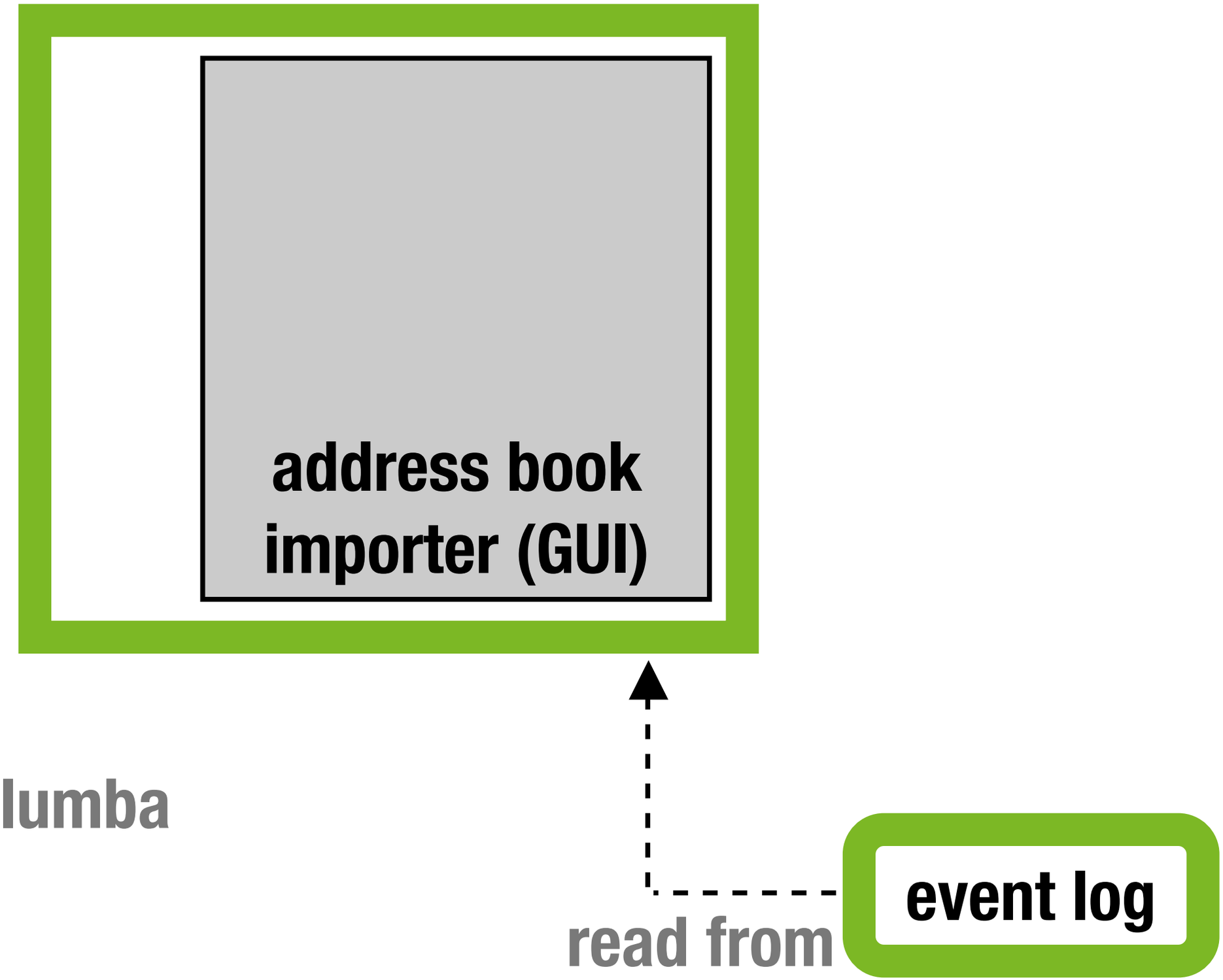
Replay



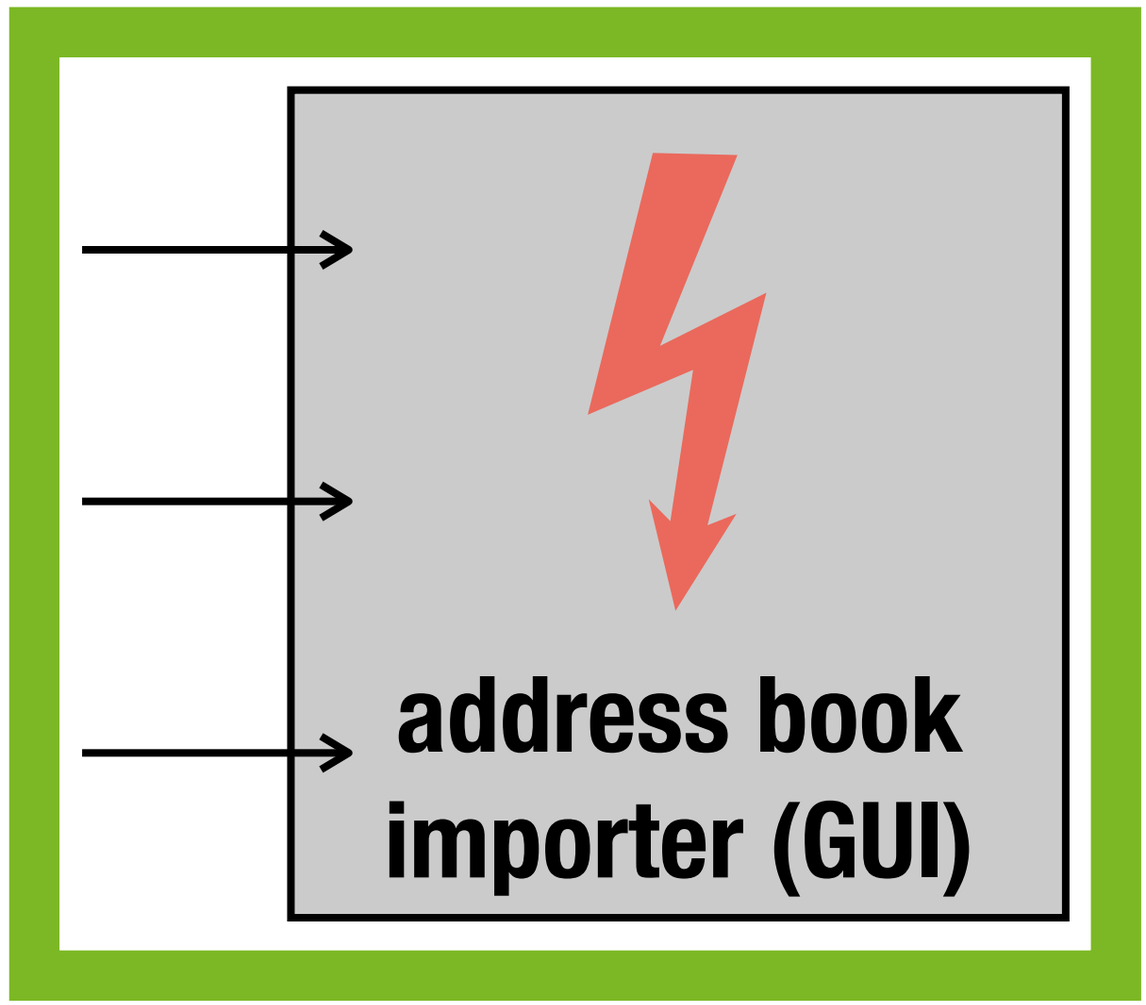
JINSI

replaces Columba

Replay


address book
importer (GUI)
JINSI
replaces Columba
read from
event log

Replay



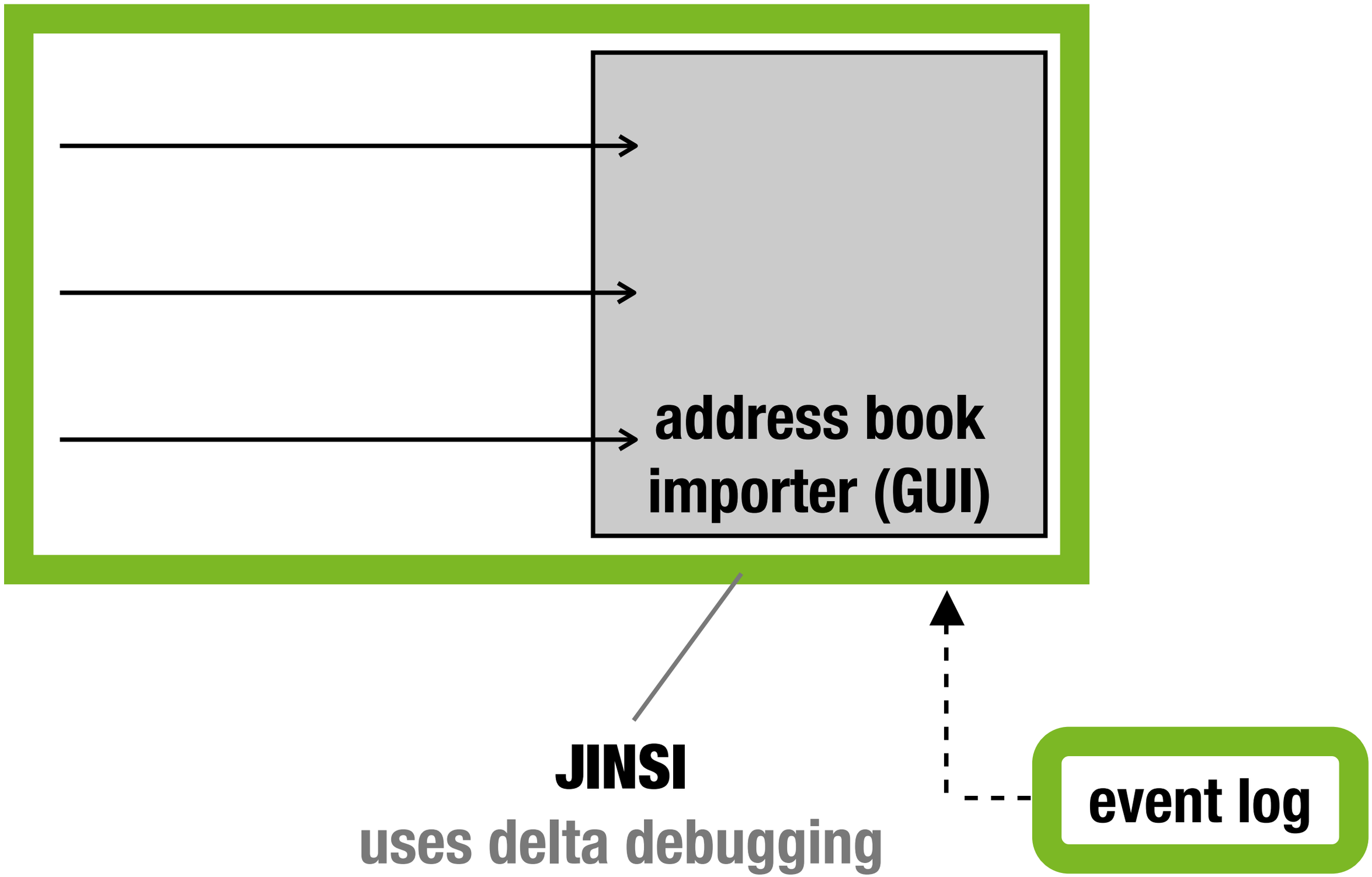
original failure reproduced

JINSI
replaces Columba

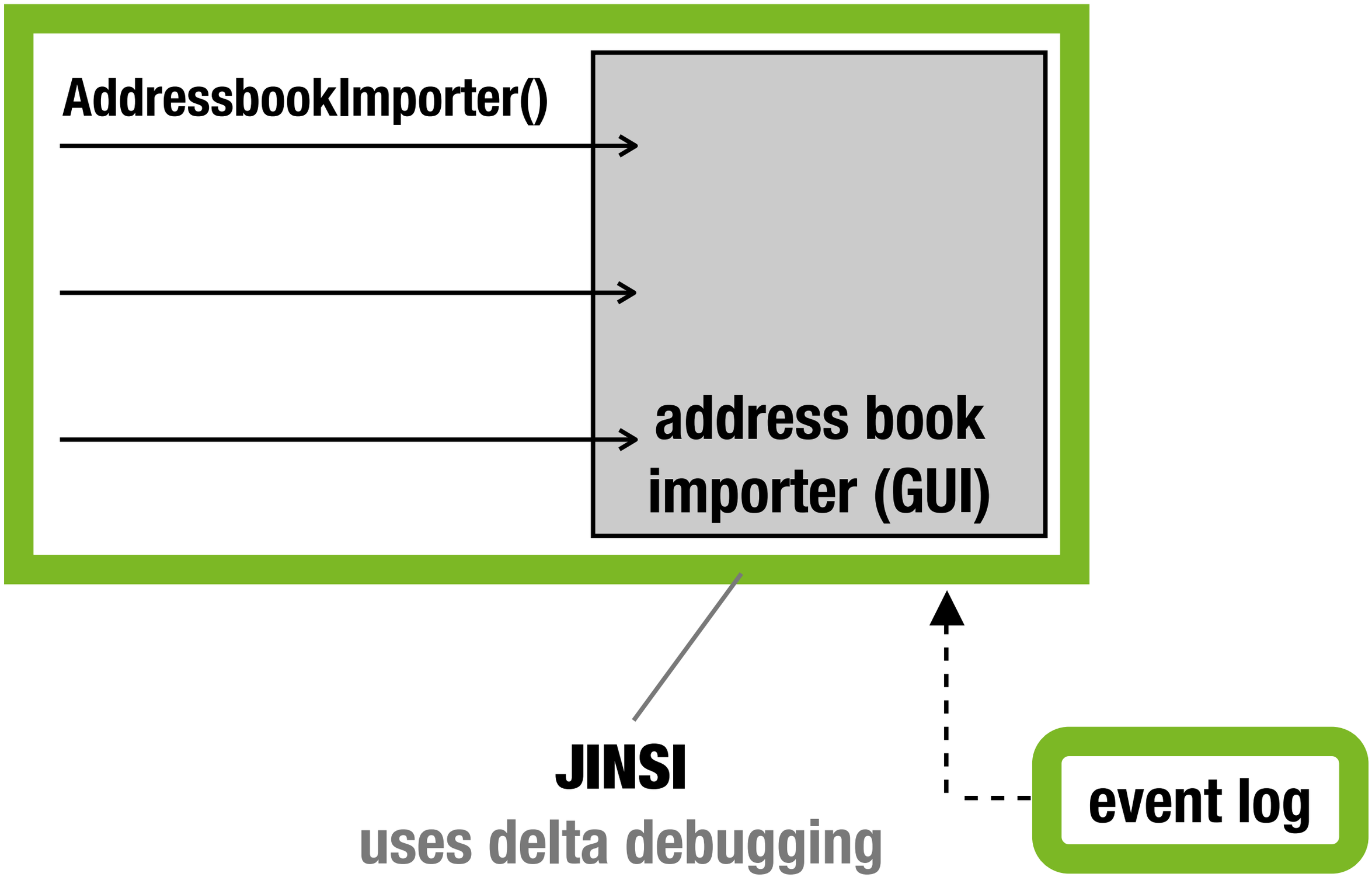
read from

event log

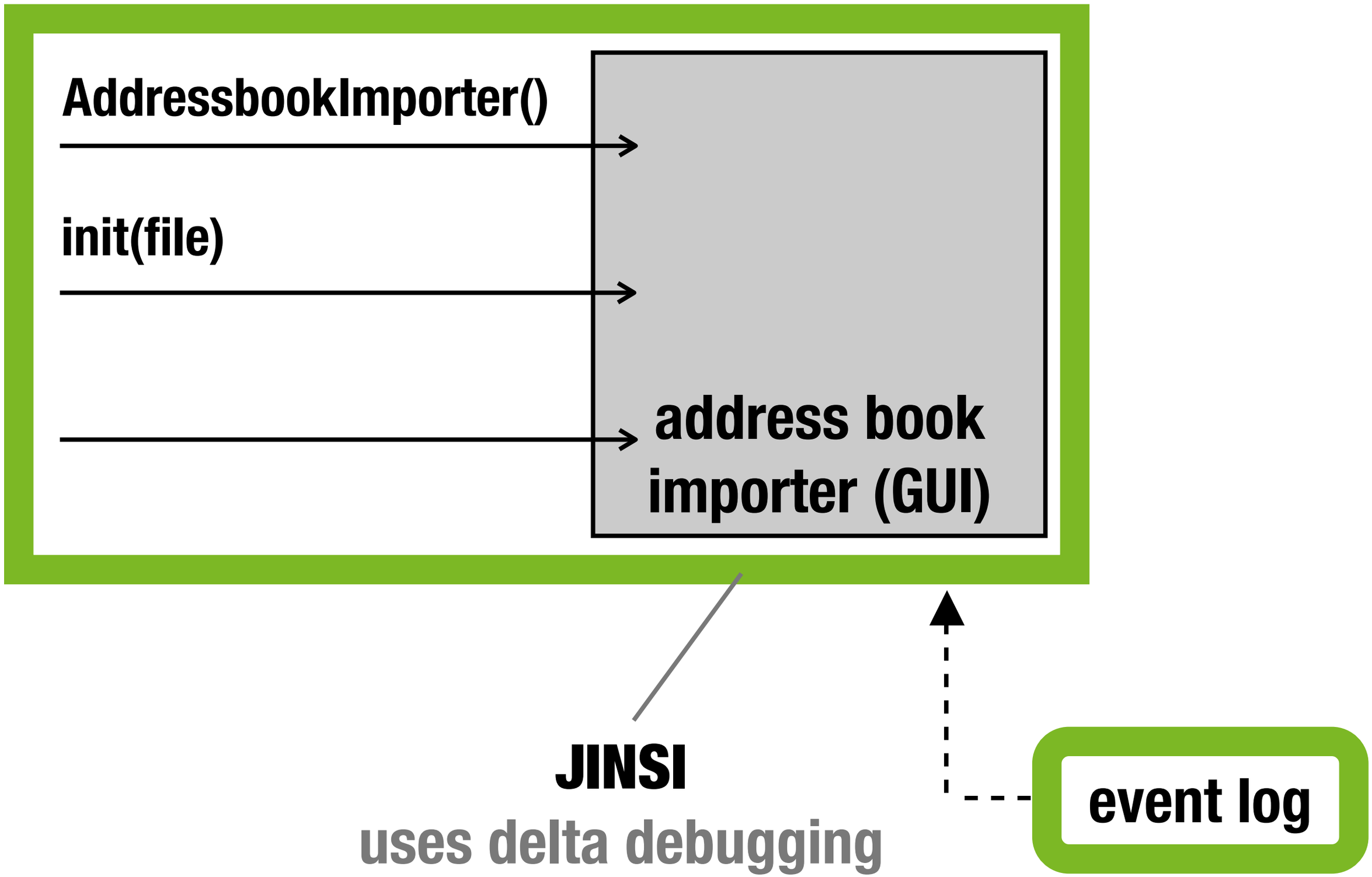
Delta Debugging



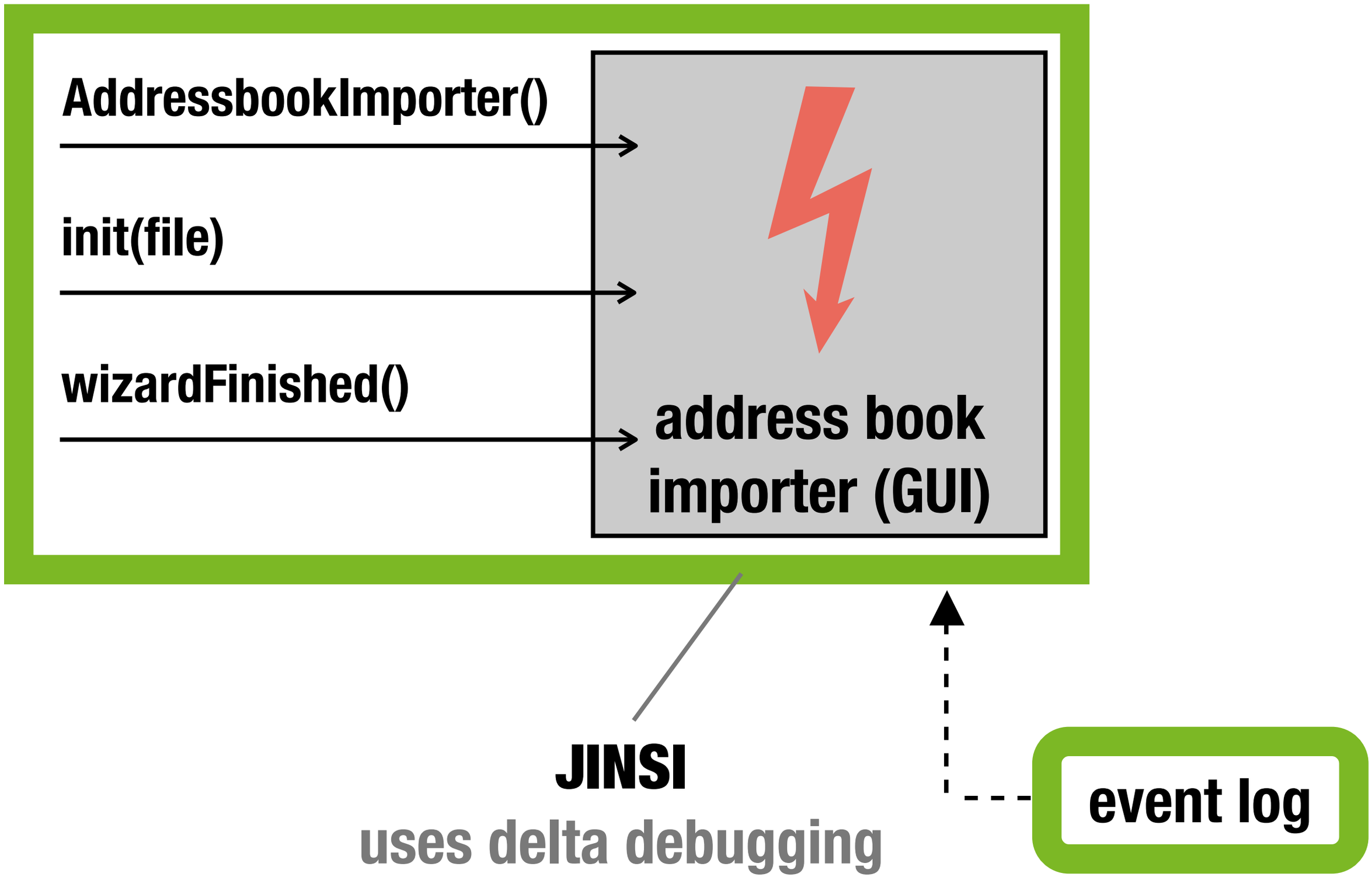
Delta Debugging



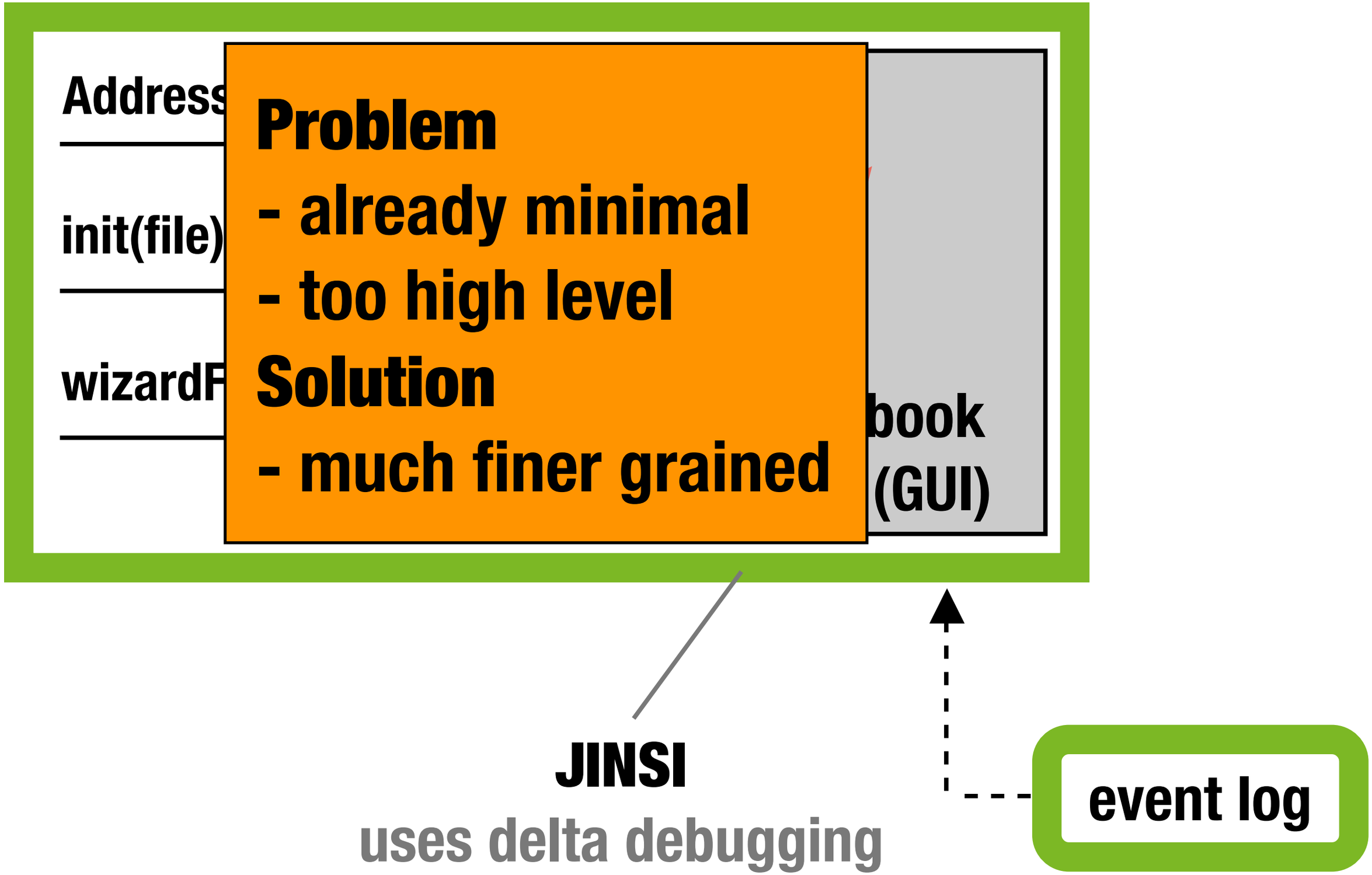
Delta Debugging



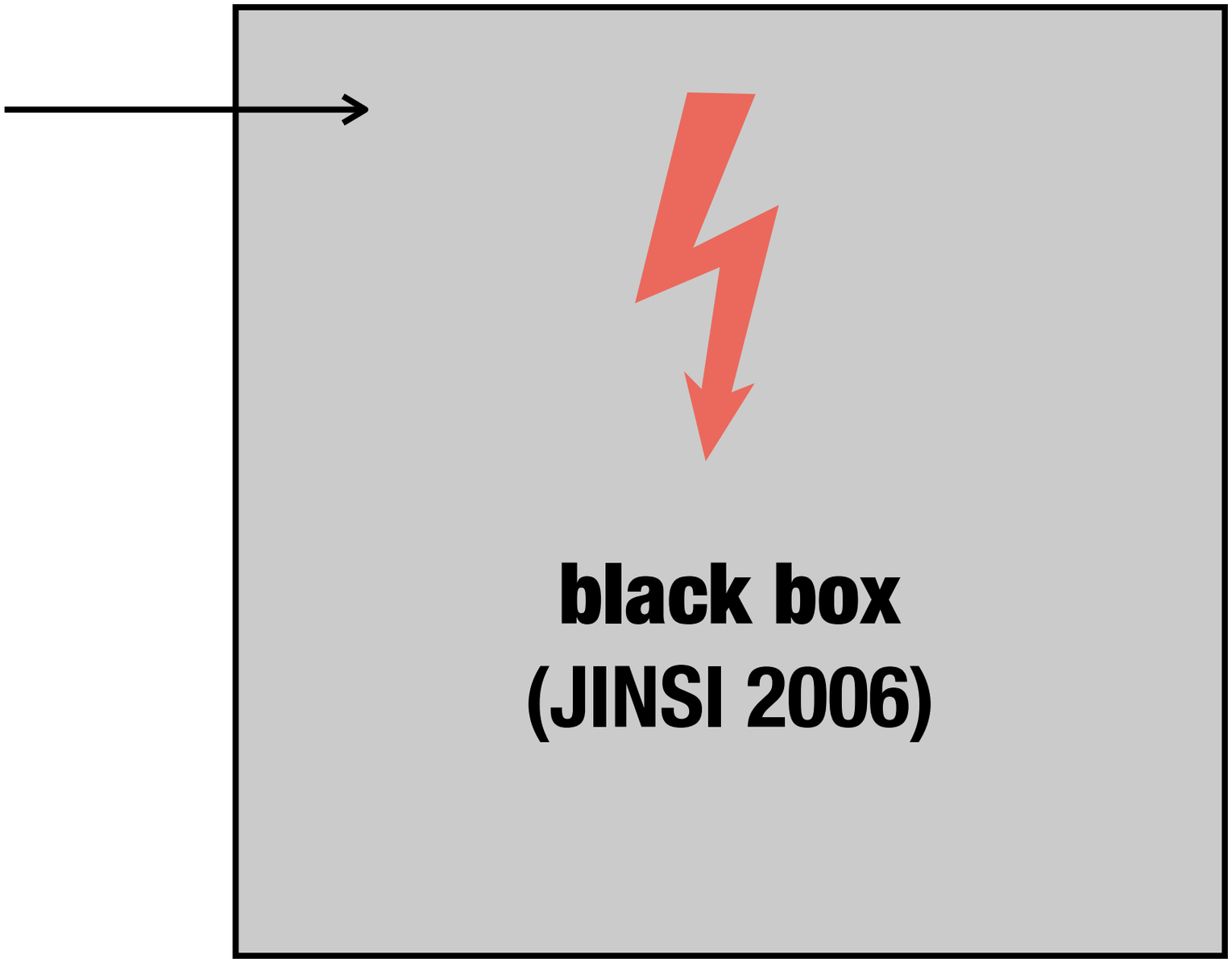
Delta Debugging



Delta Debugging

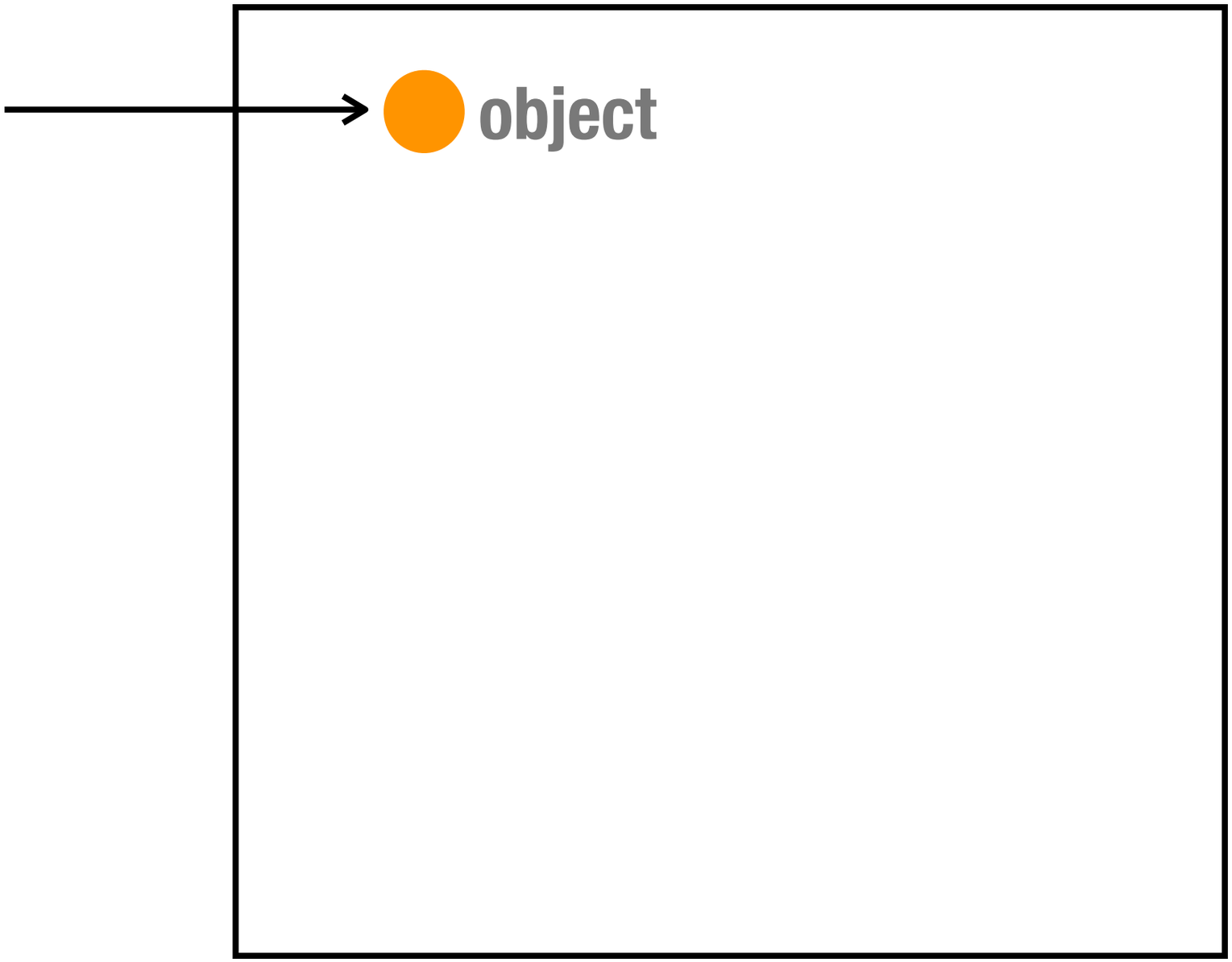


Inside the Component



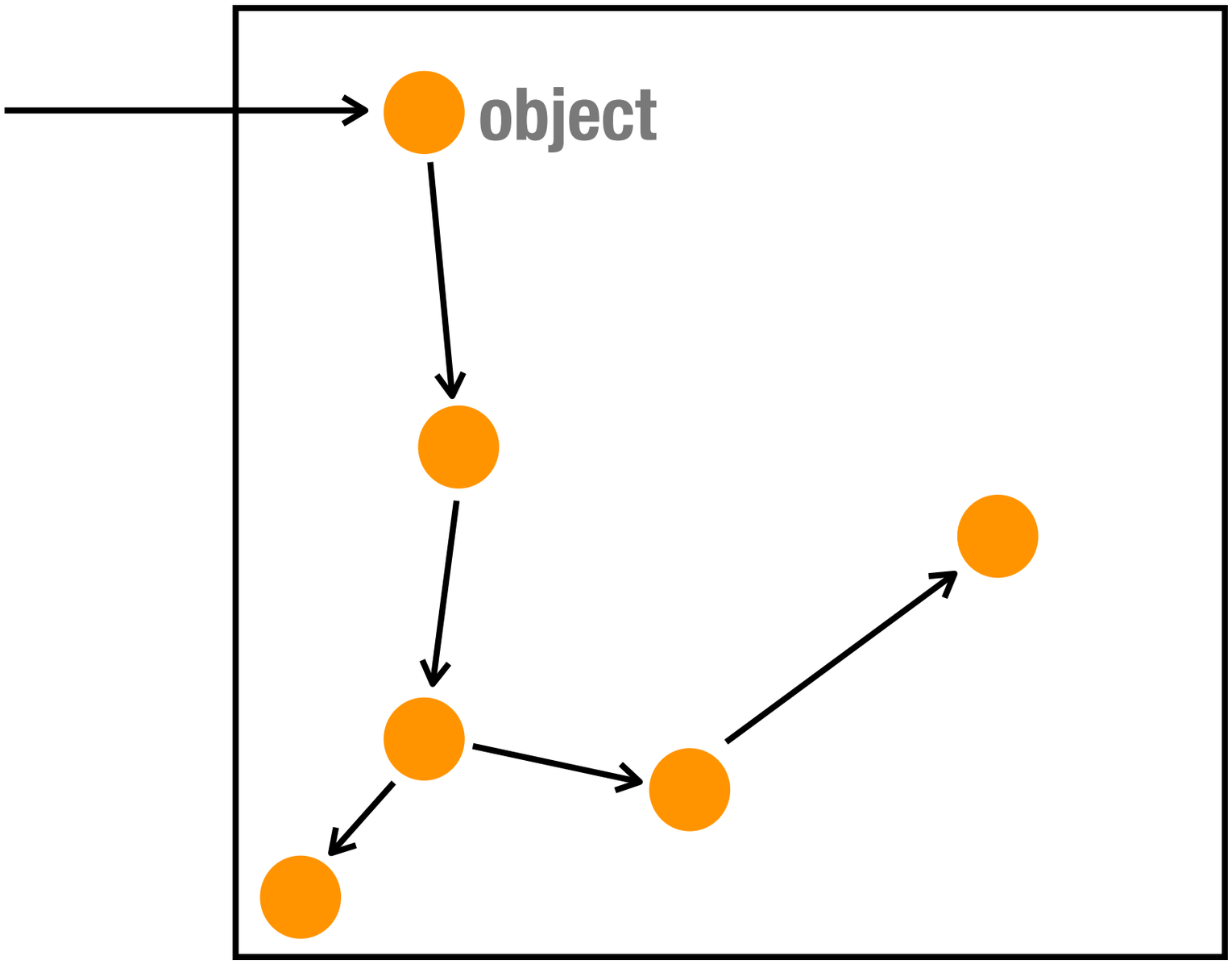
coarse grained view

Object Interactions



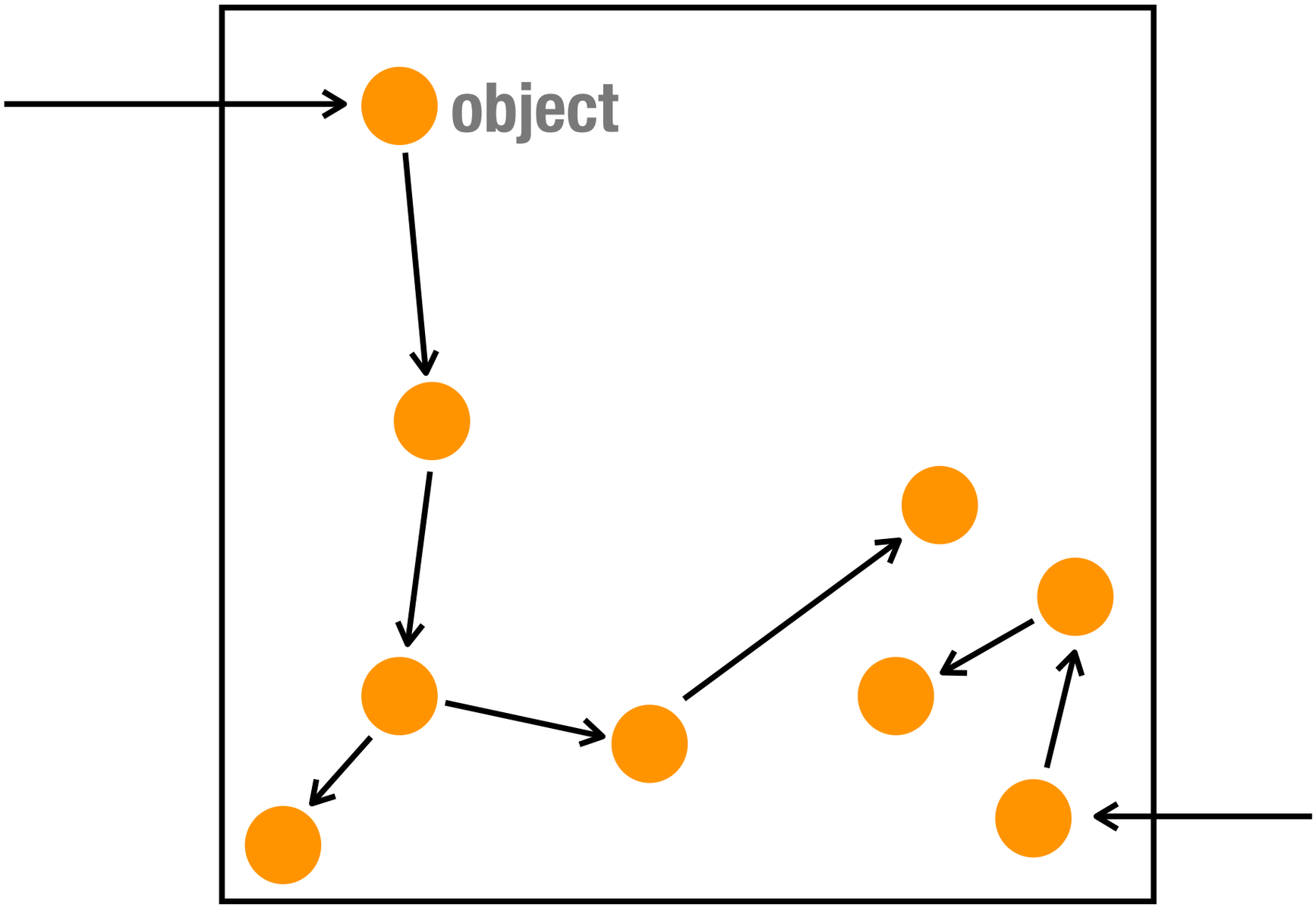
finer grained view

Object Interactions



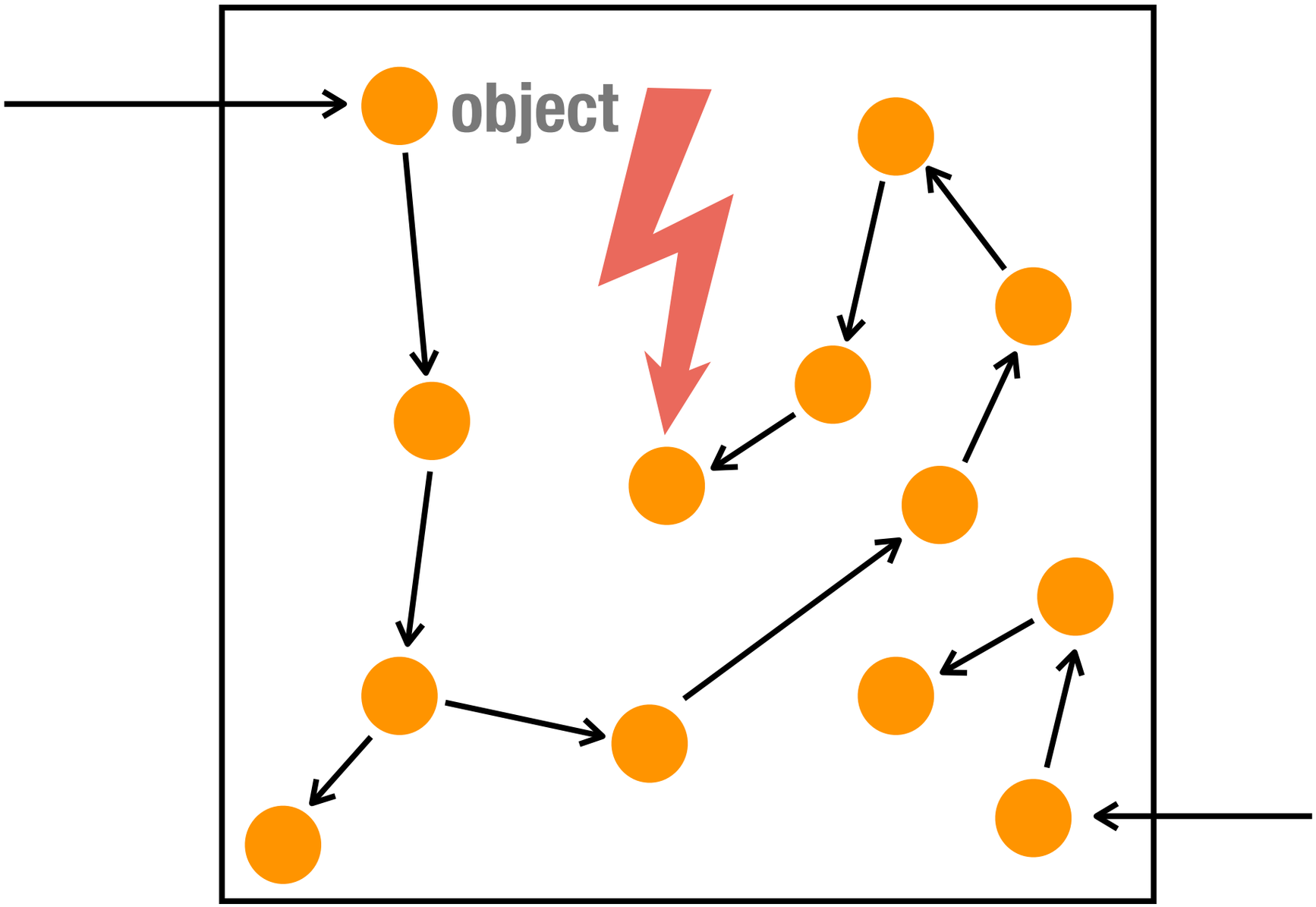
finer grained view

Object Interactions



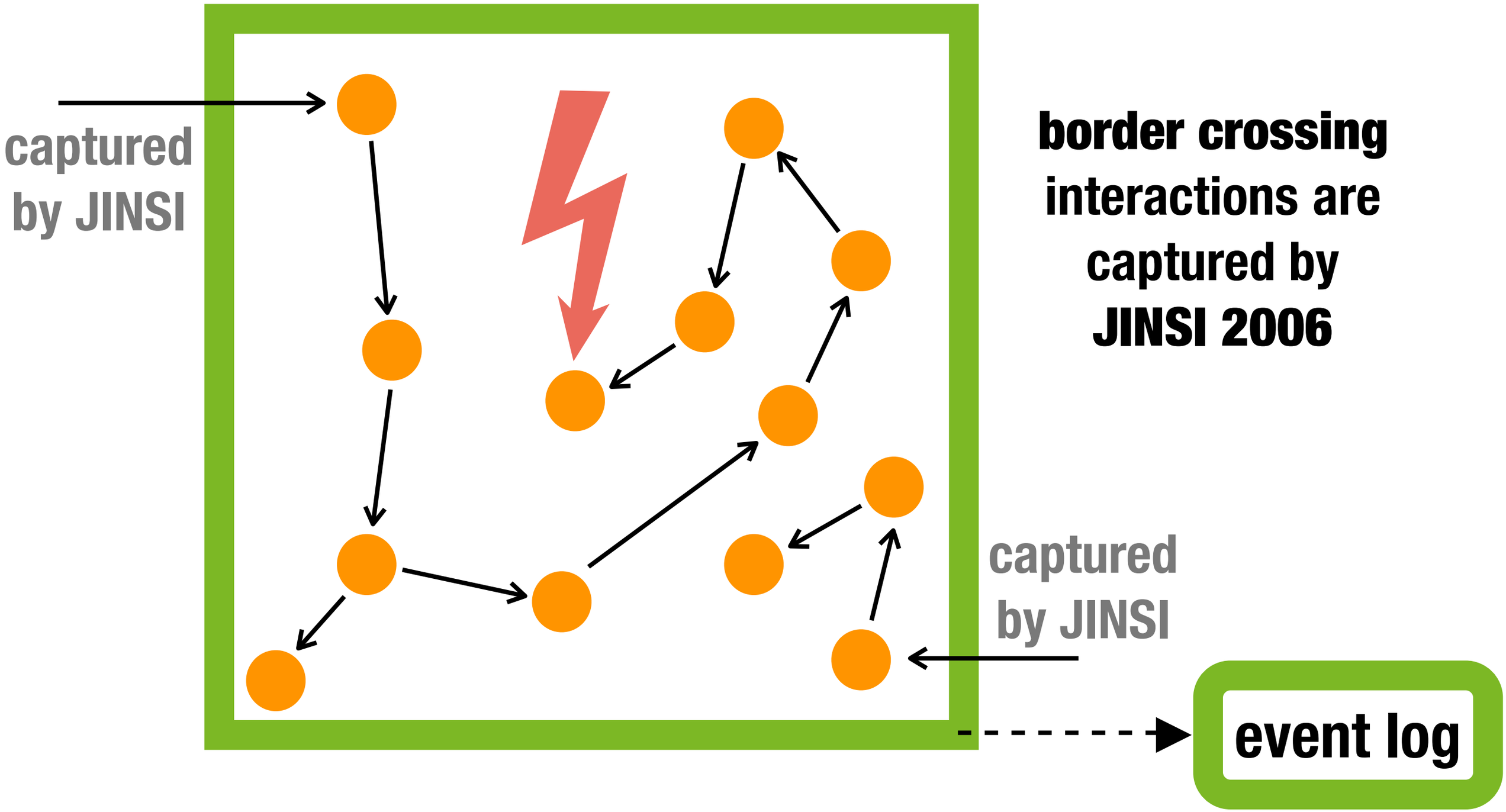
finer grained view

Object Interactions



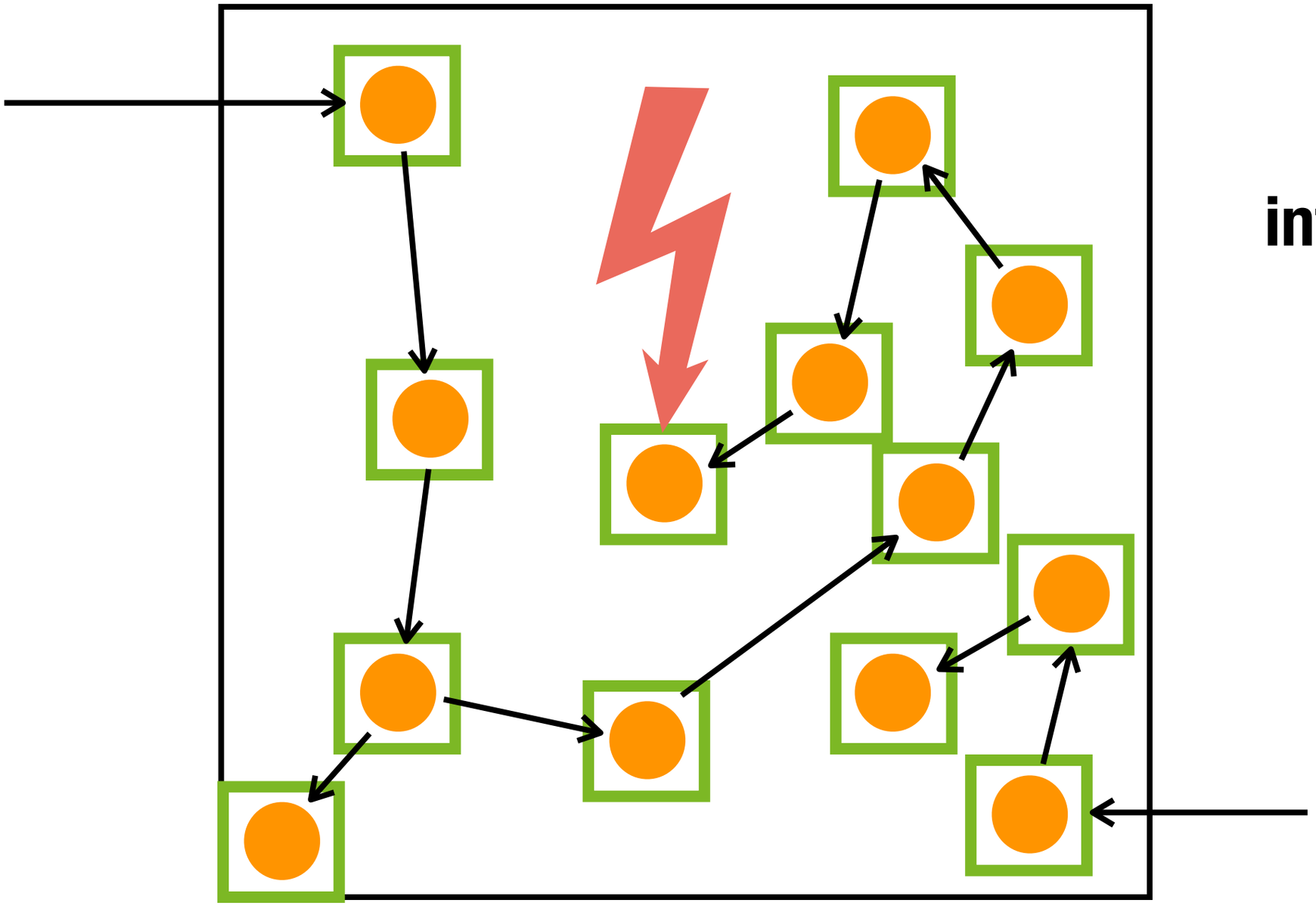
finer grained view

Object Interactions



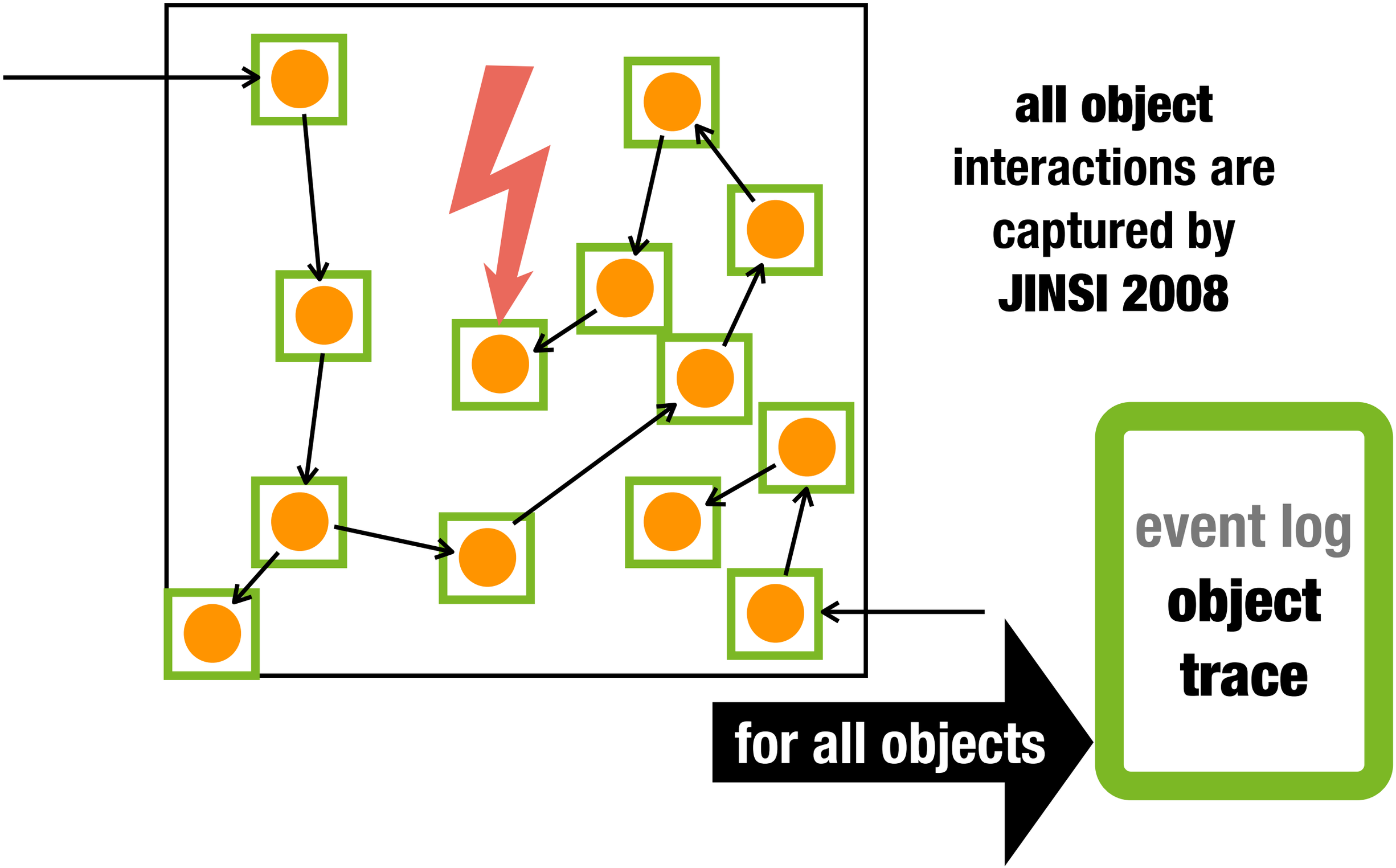
border crossing interactions are captured by JINSI 2006

Object Interactions



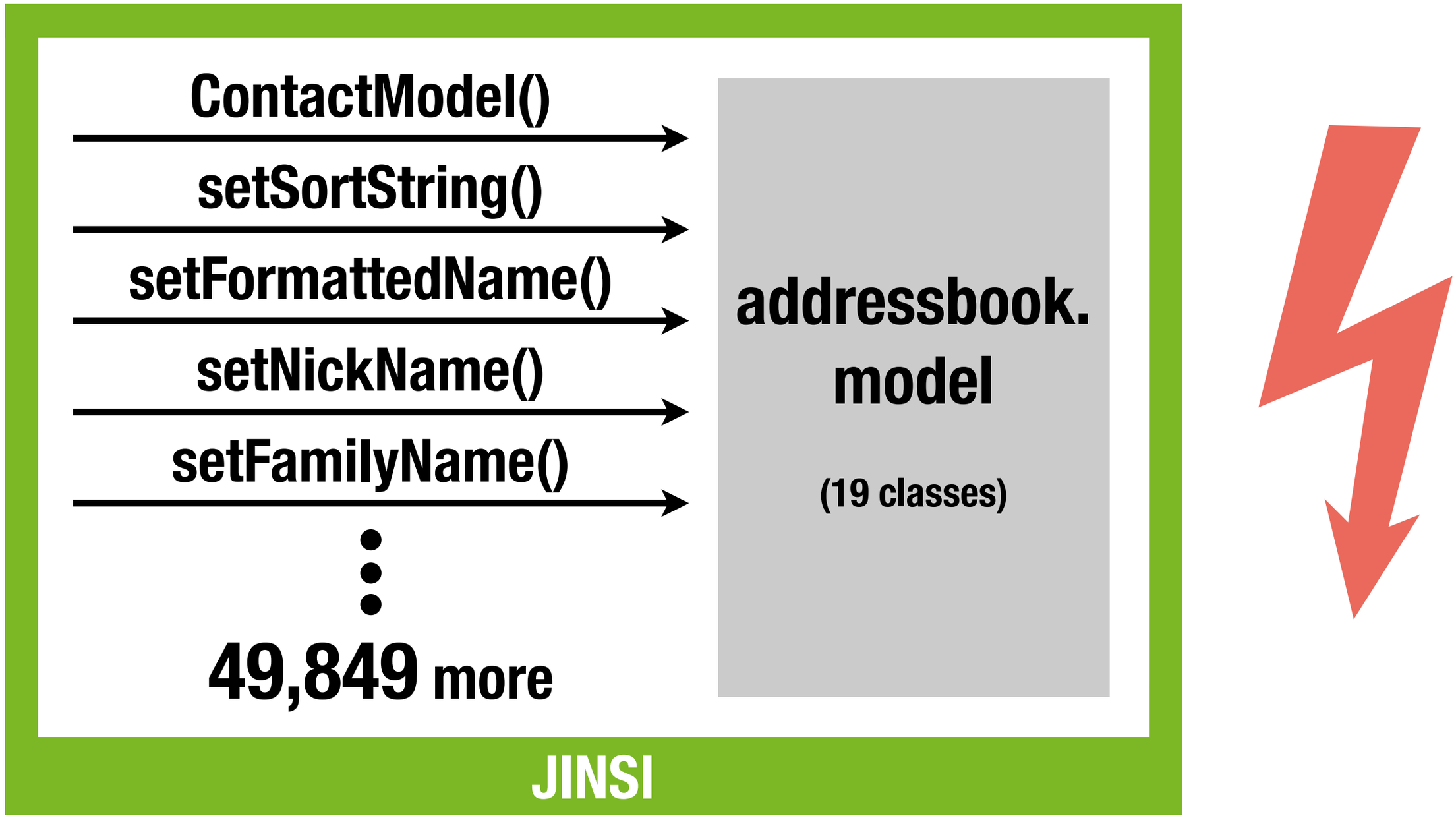
all object interactions are captured by JINSI 2008

Object Trace



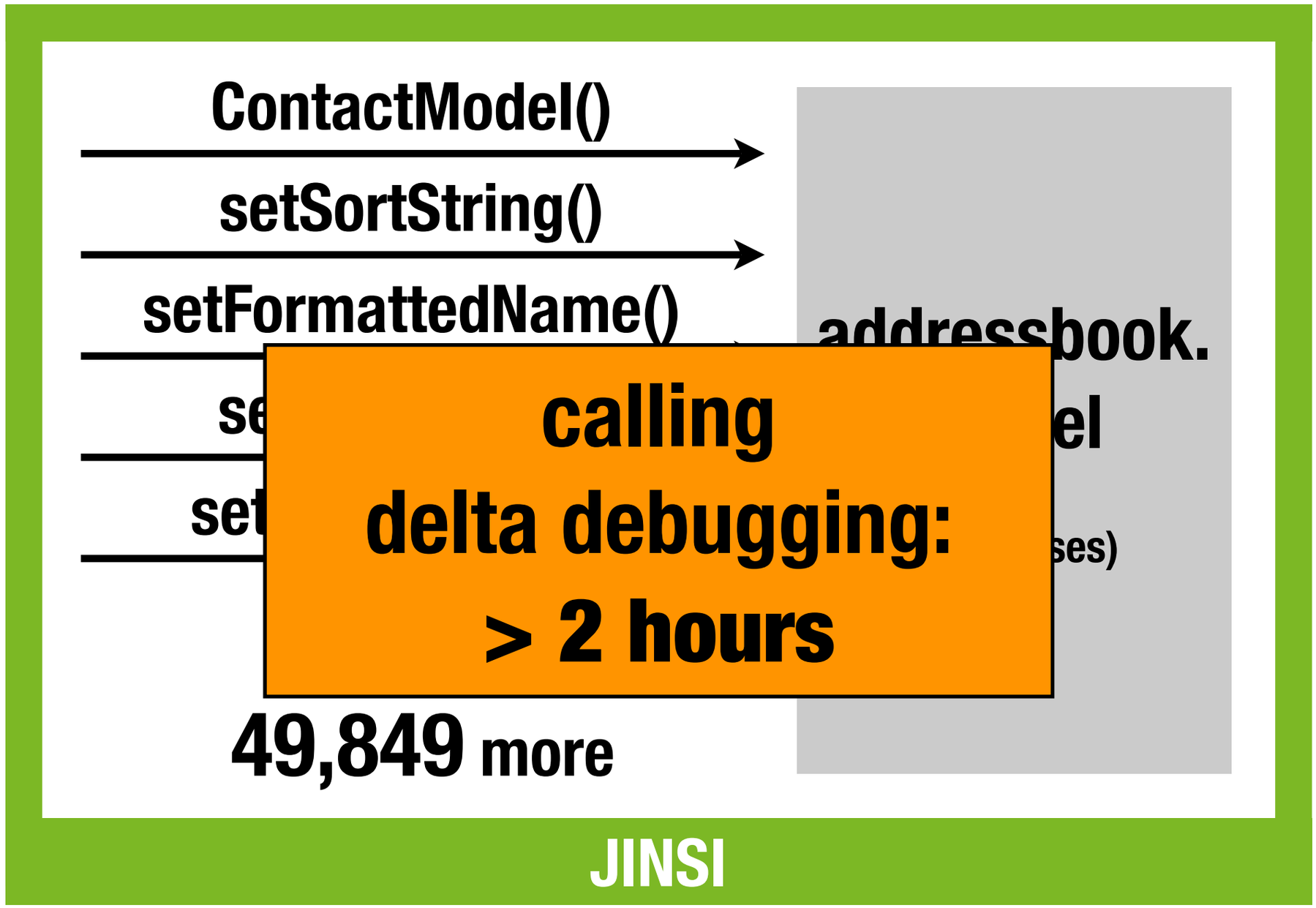
Importing the Addresses

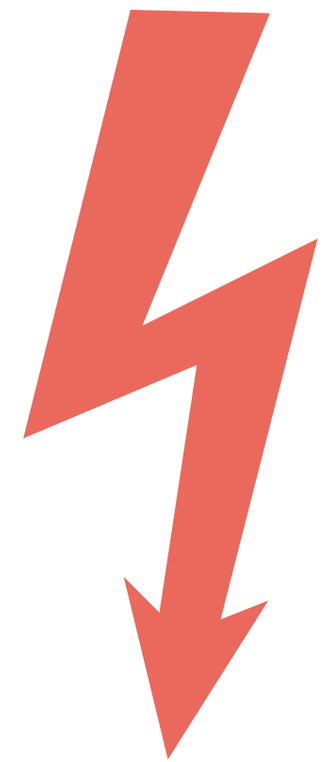
Object Trace



Importing the Addresses

Object Trace

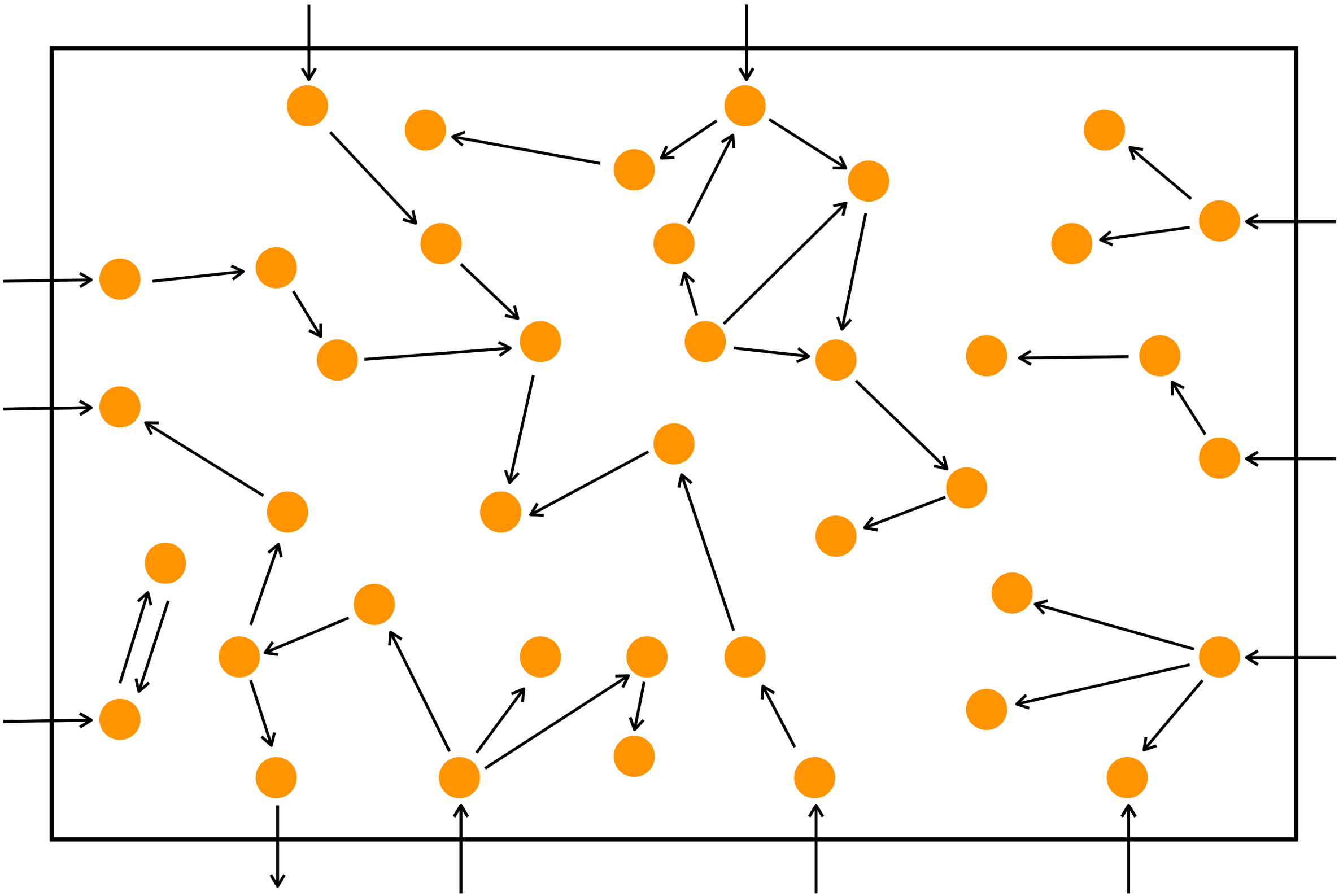




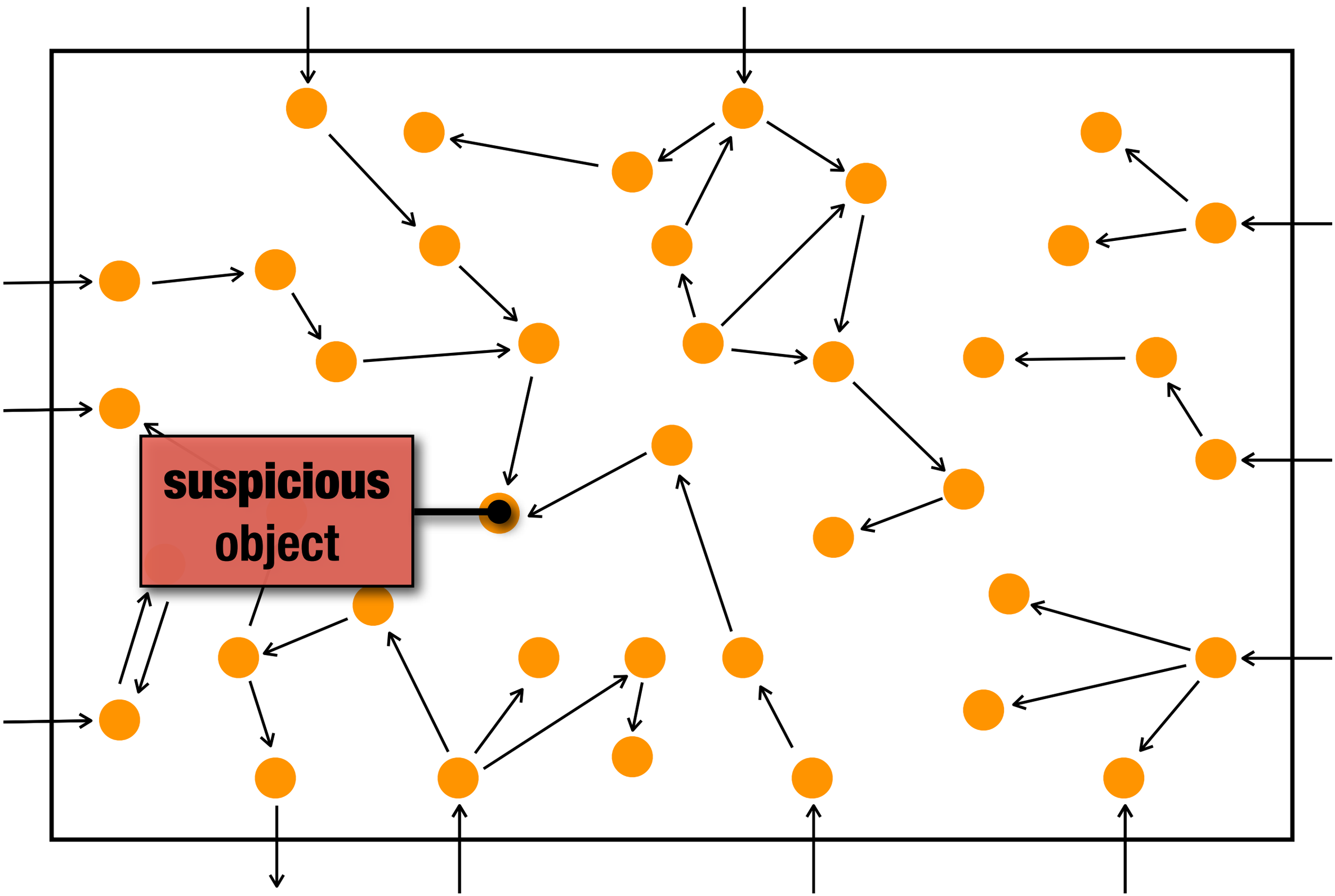
Object Slice

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

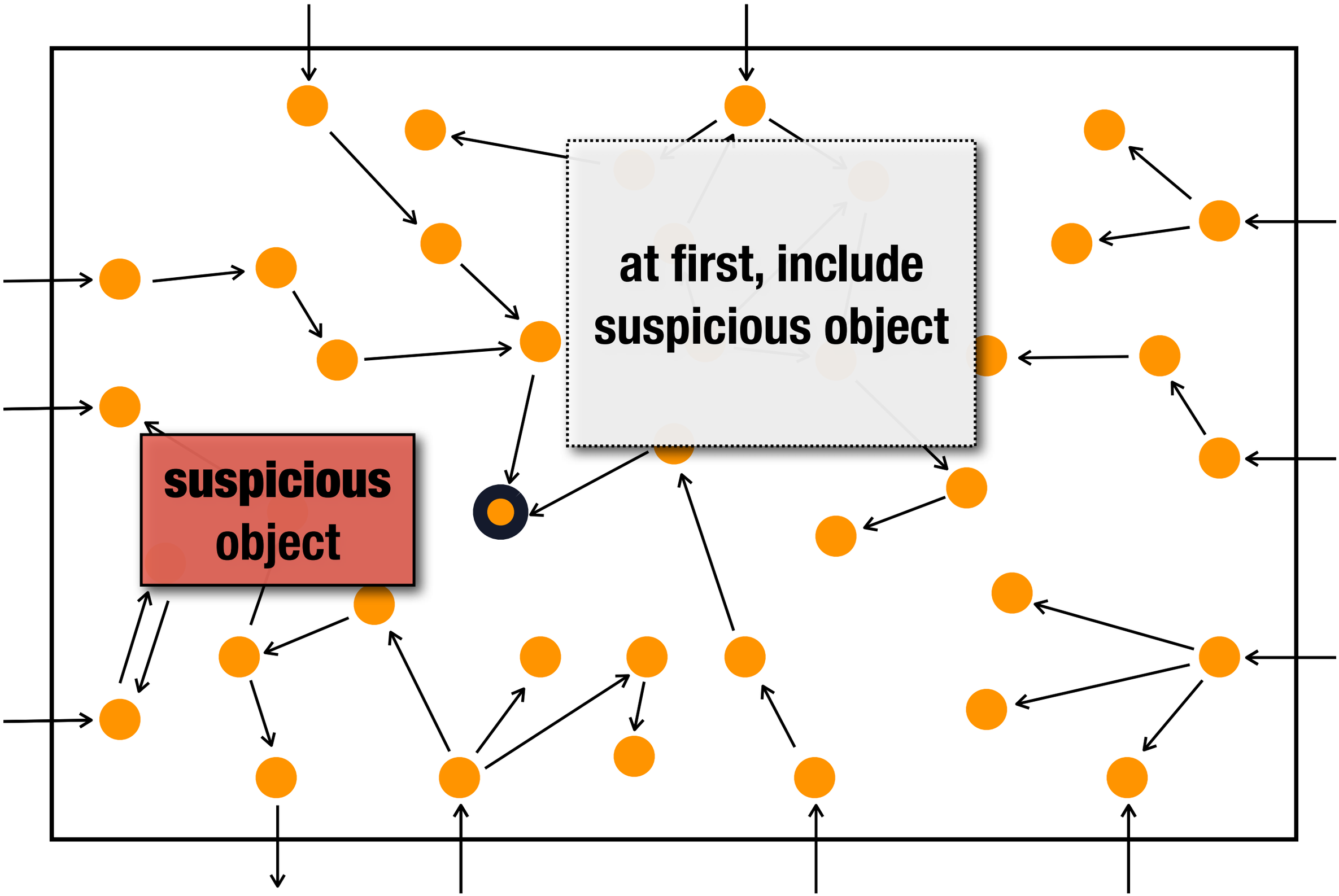
Object Interactions



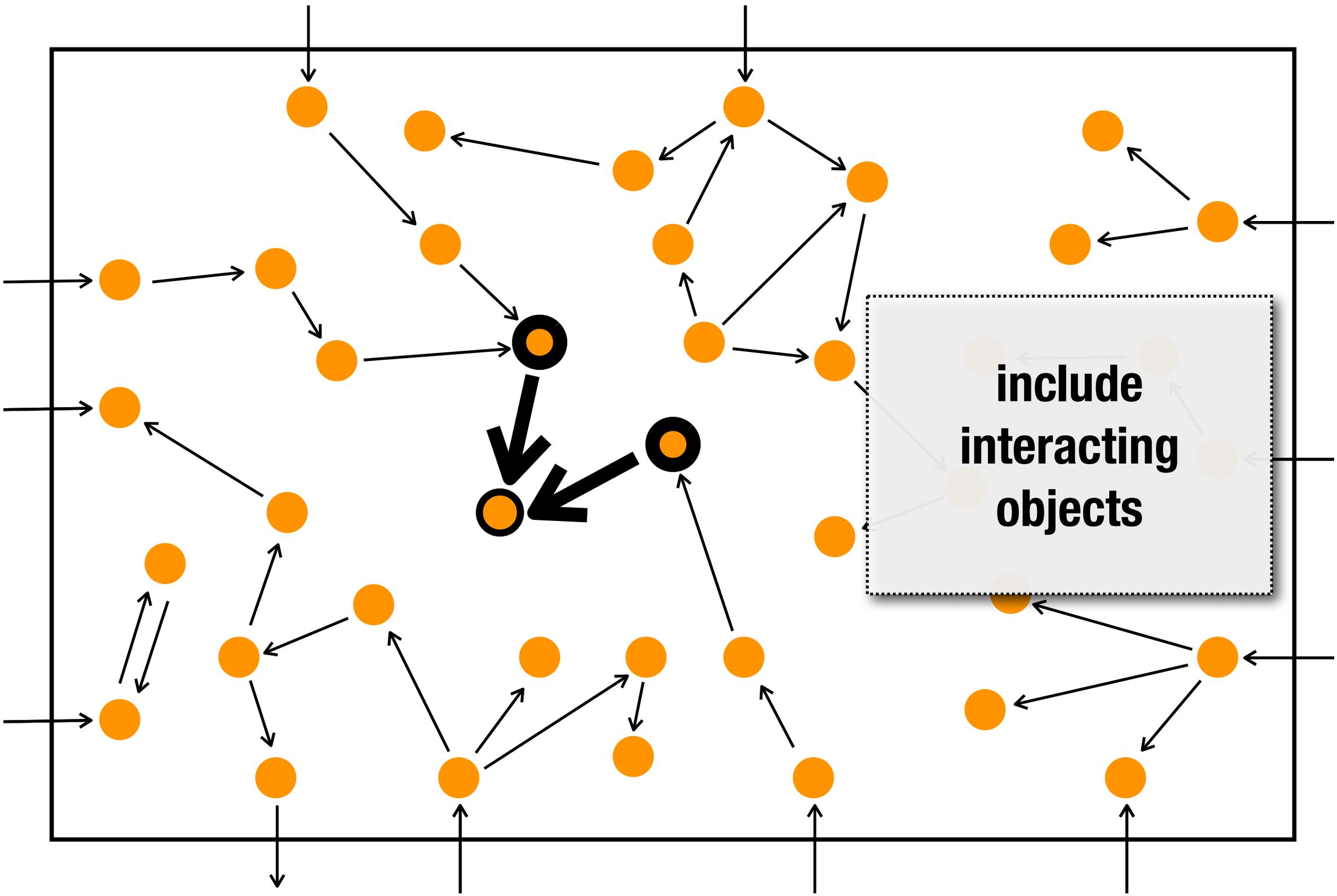
Object Interactions



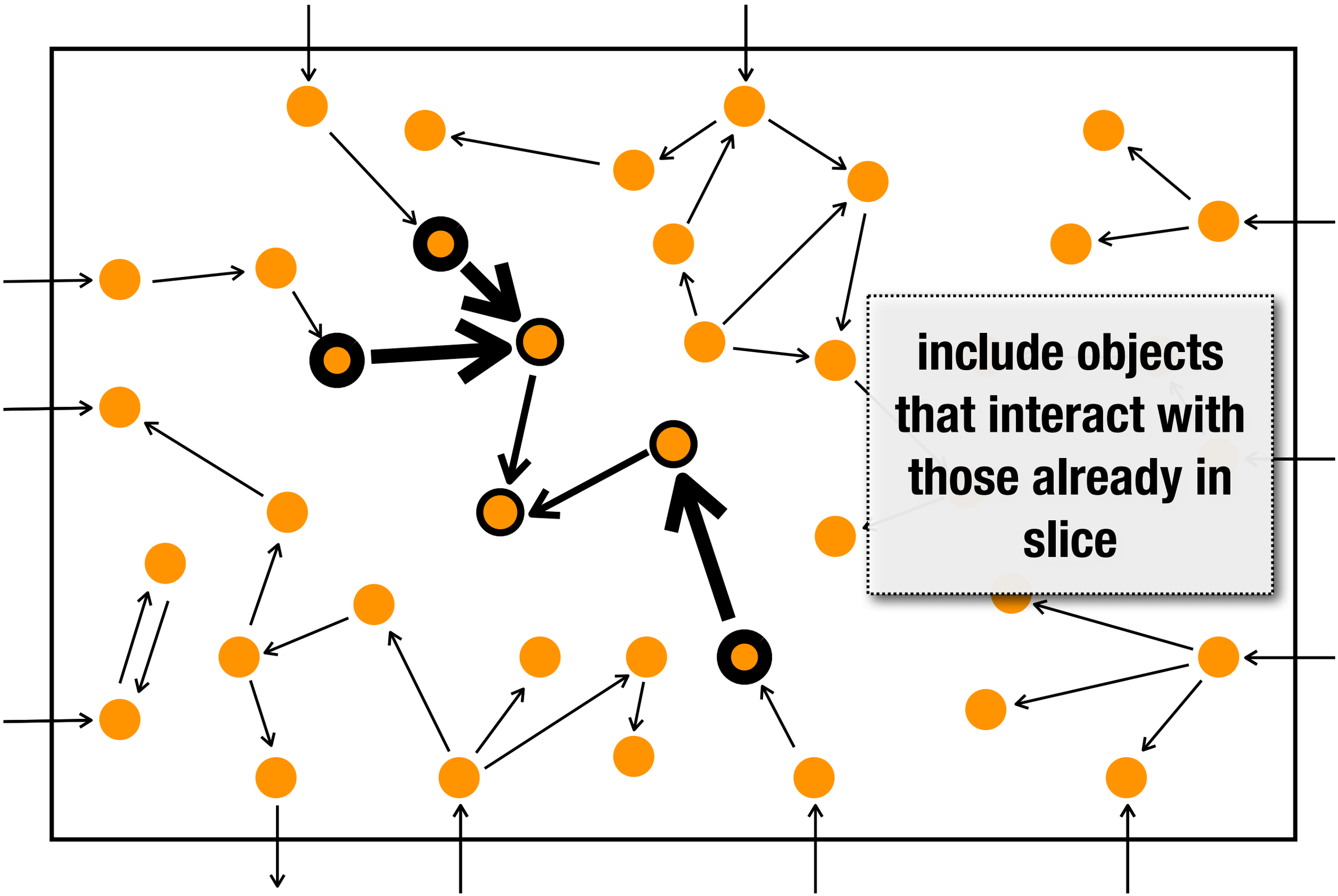
Object Slice



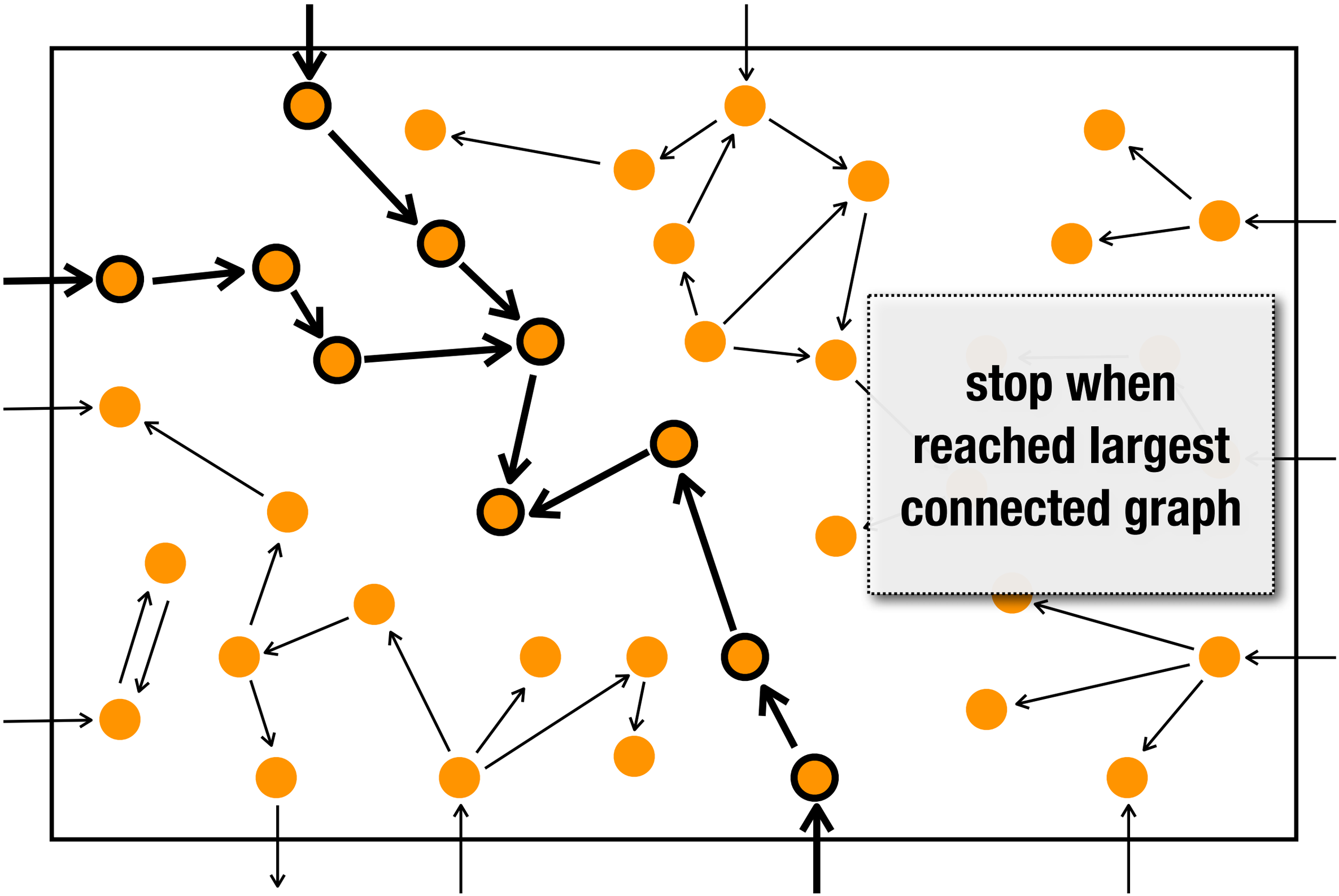
Object Slice



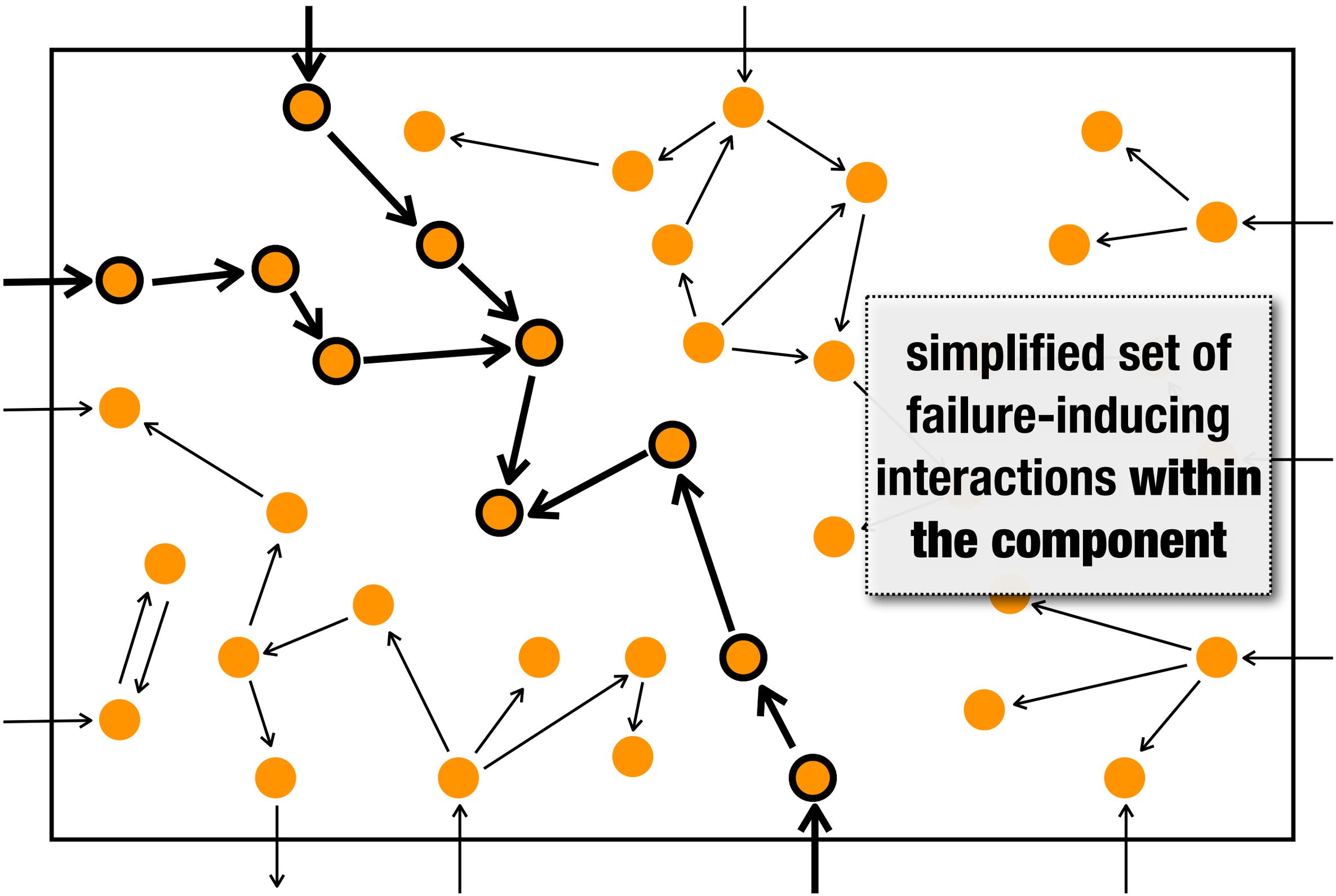
Object Slice



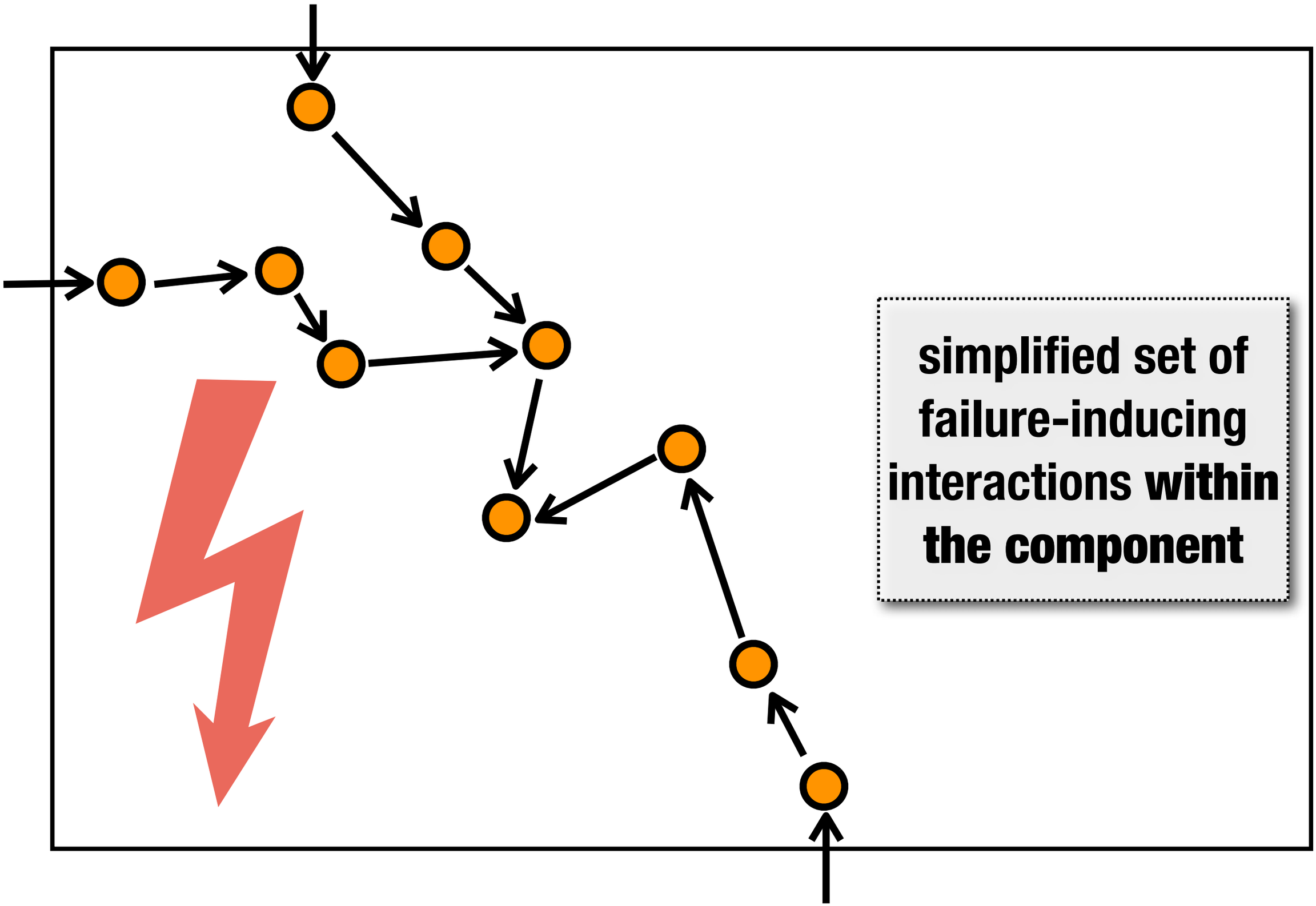
Object Slice



Object Slice

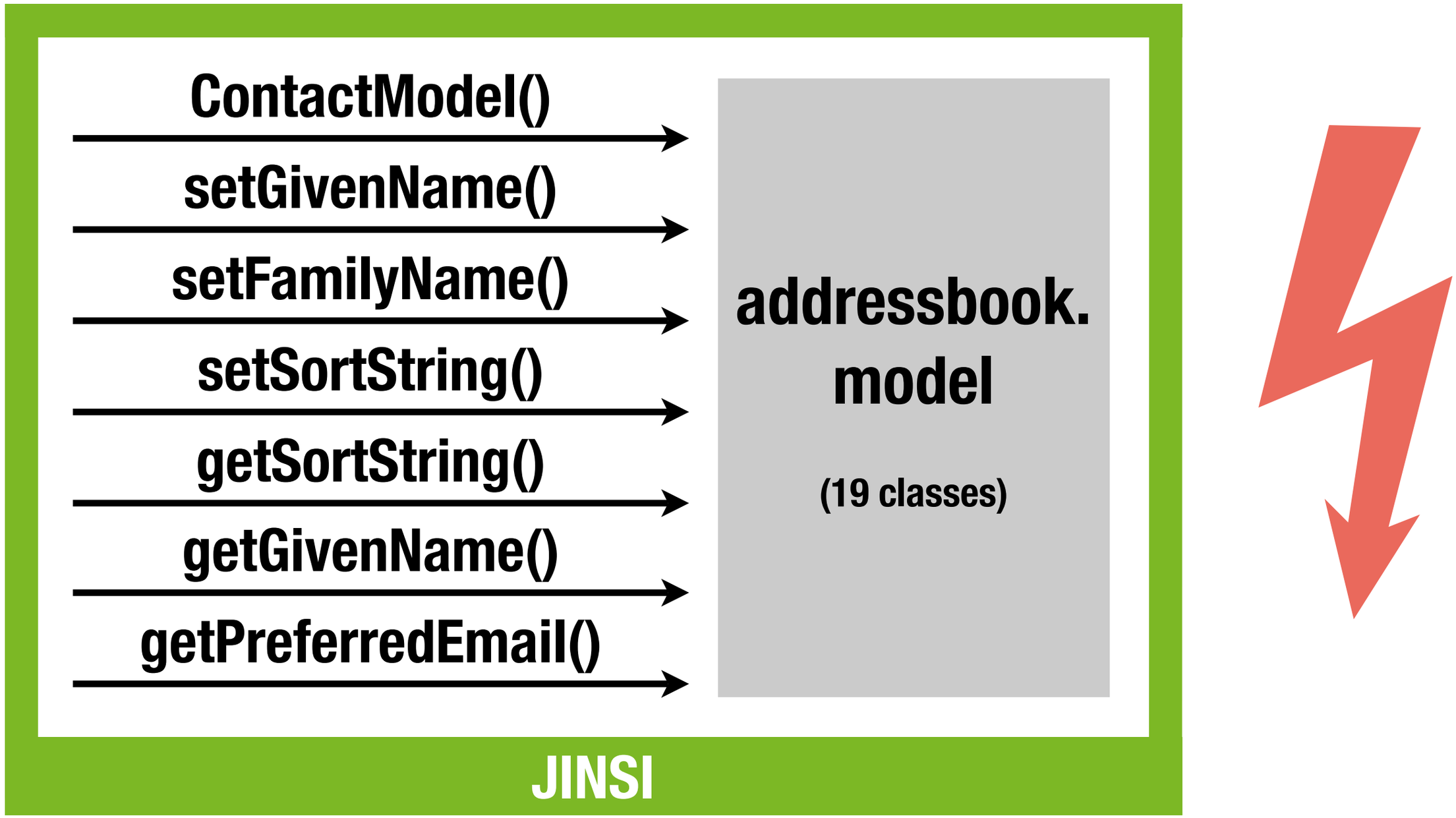


Object Slice

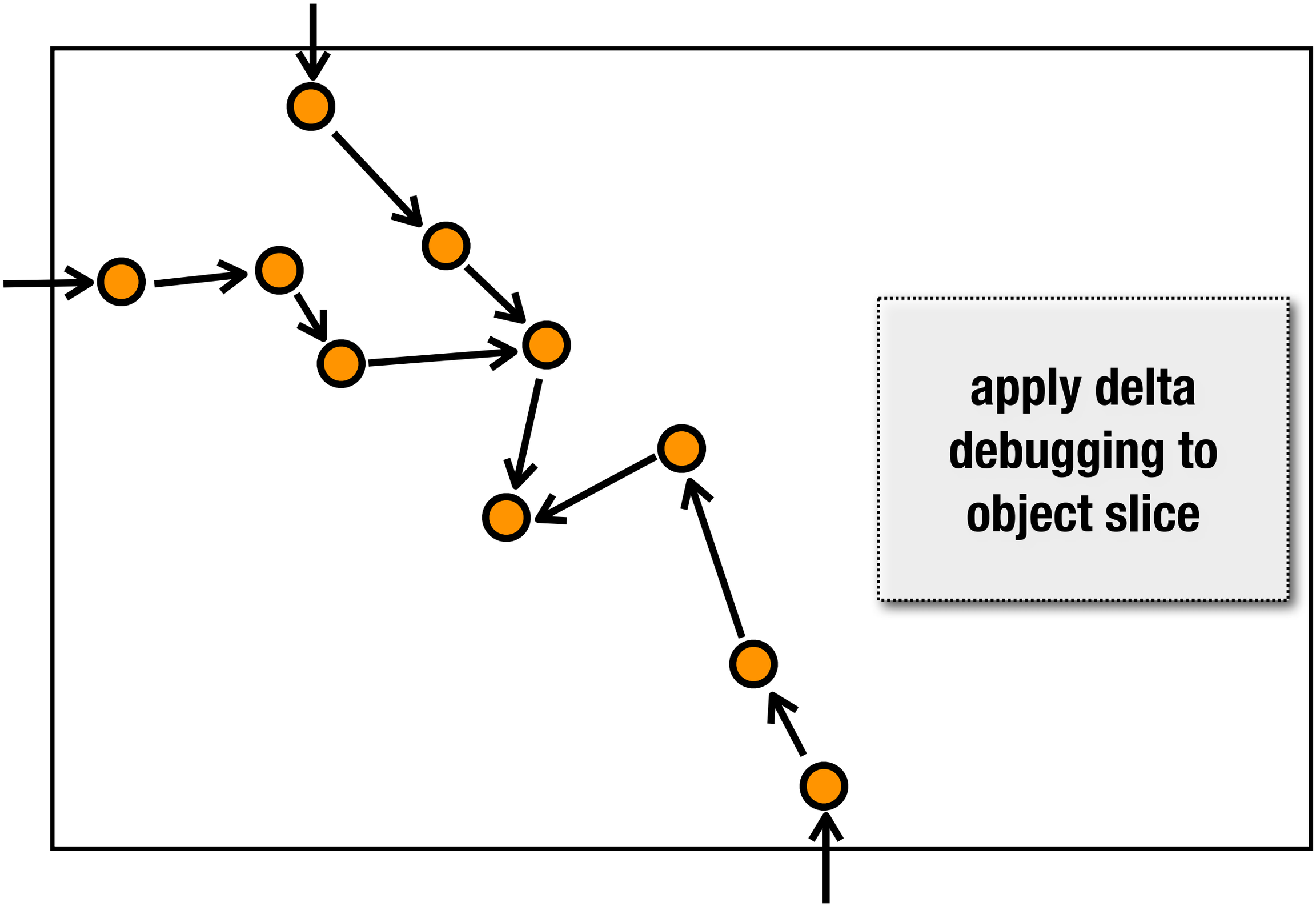


Importing Addresses

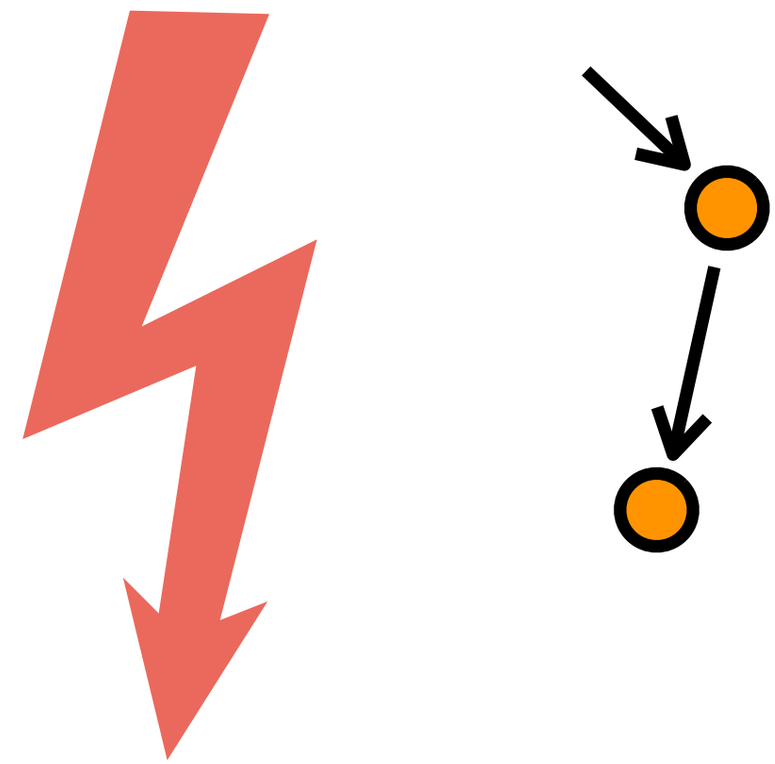
Object Slice



Delta Debugging



Delta Debugging

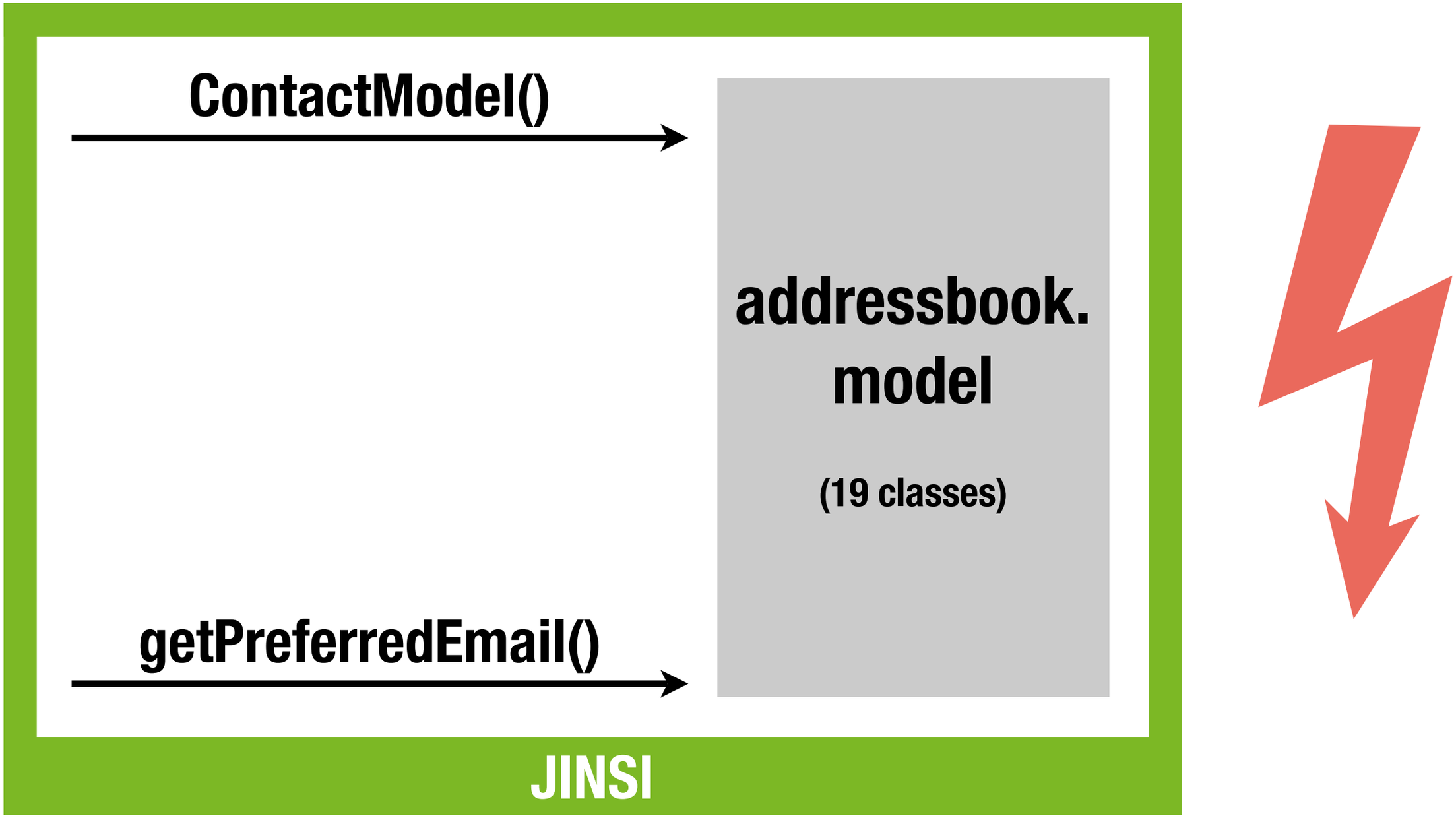


original failure

minimized set of failure-inducing interactions **within the component**

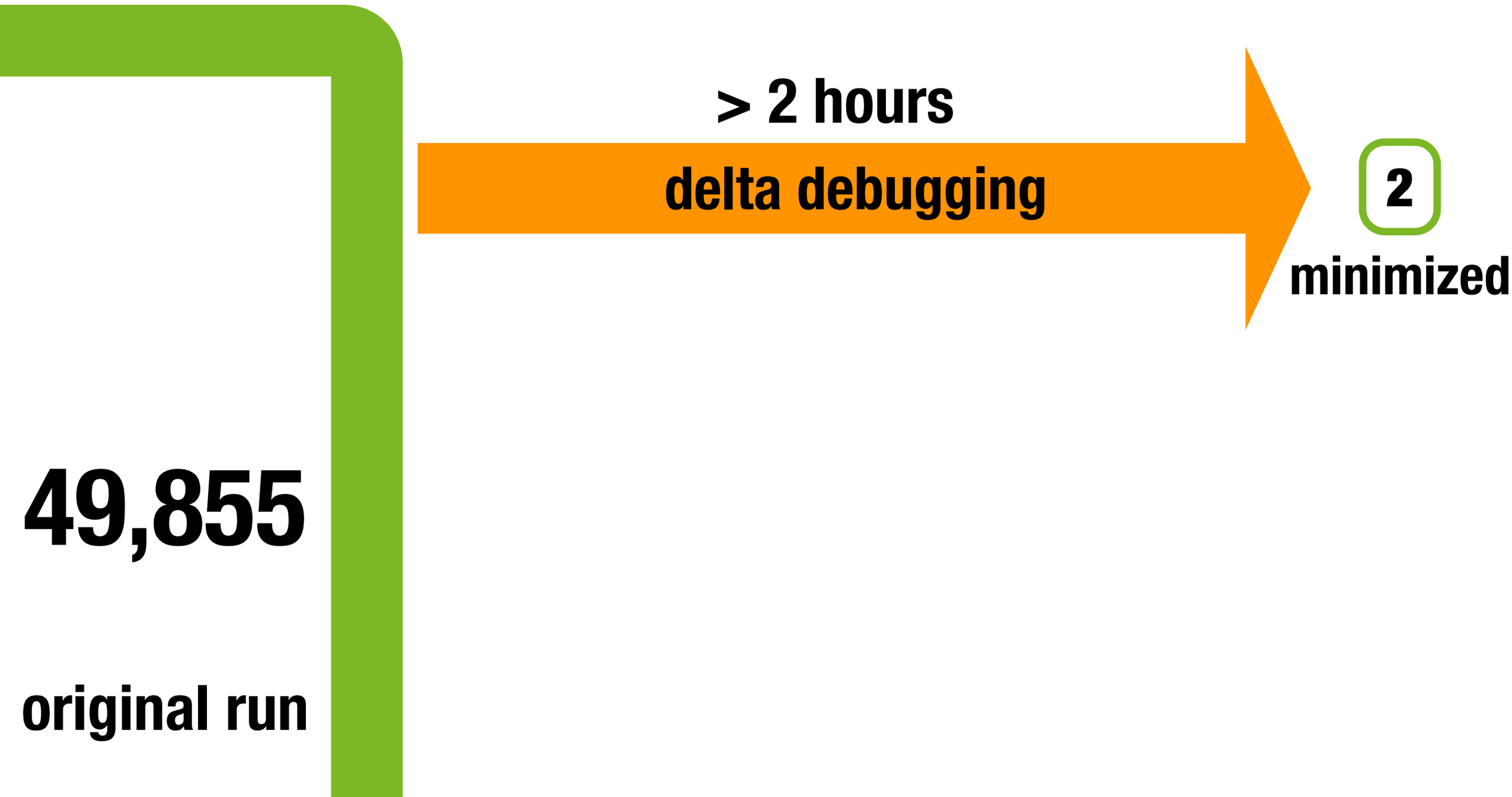
Importing Addresses

Delta Debugging



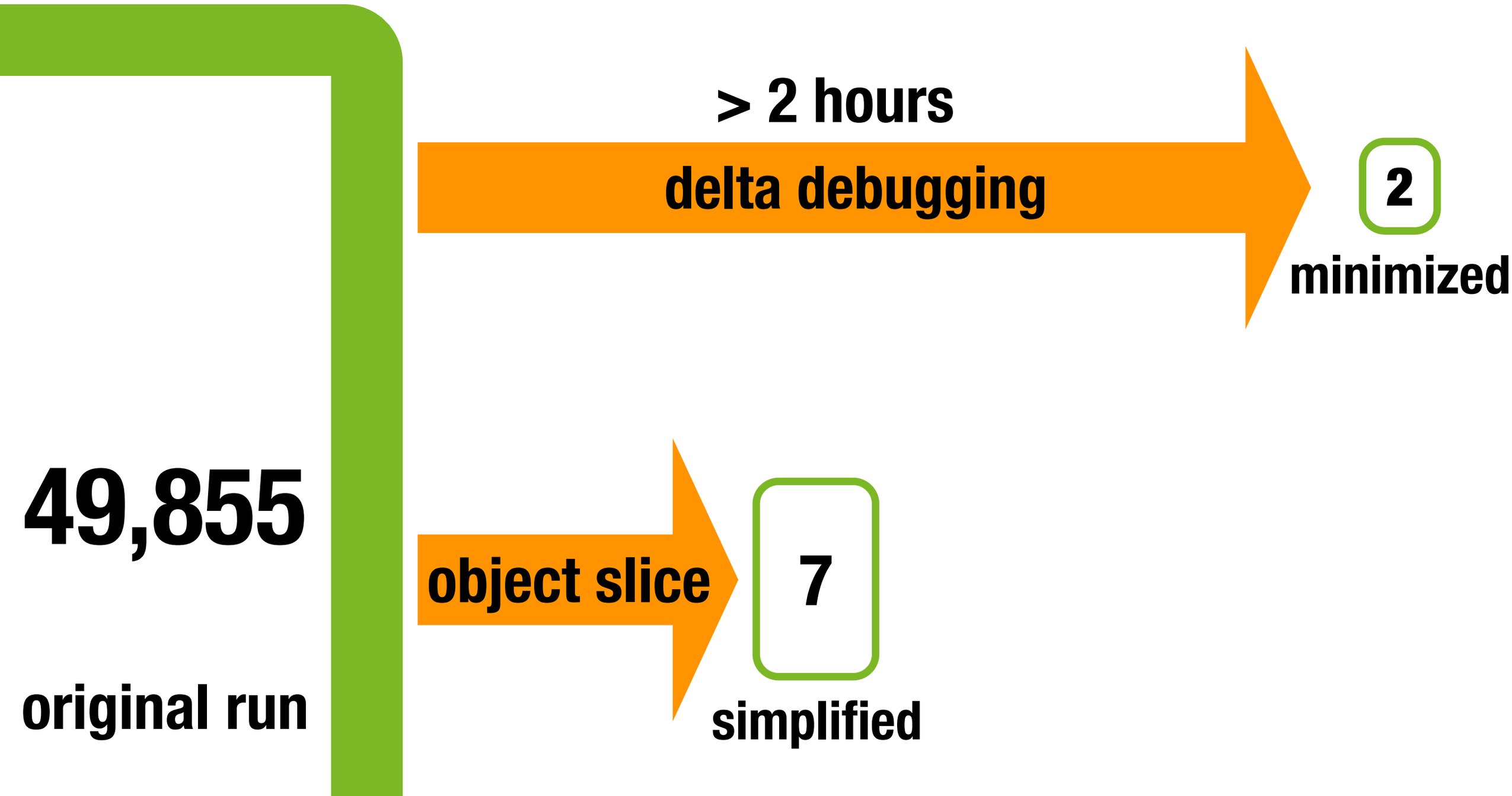
Slicing + Delta Debugging

Incoming Interactions



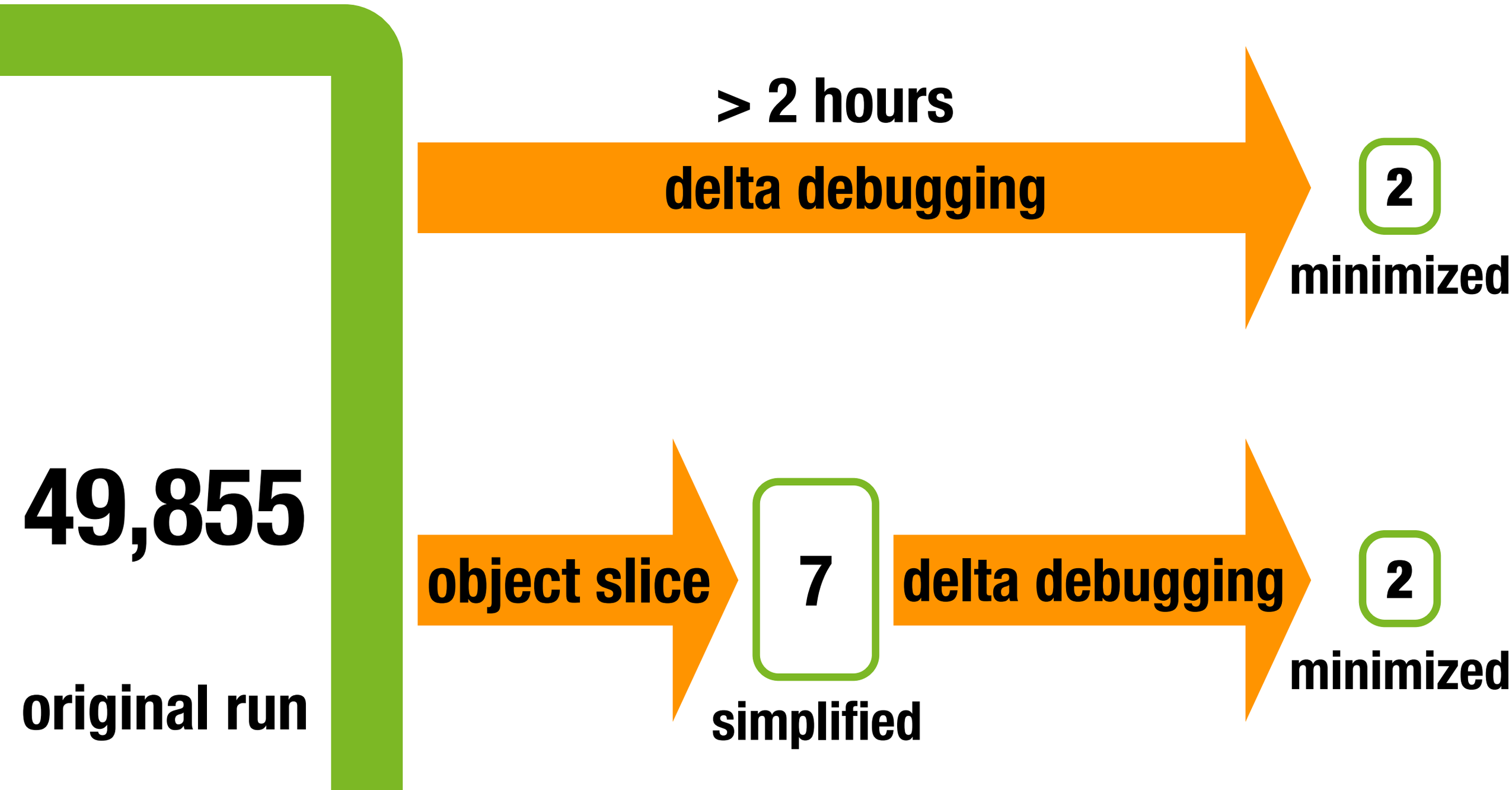
Slicing + Delta Debugging

Incoming Interactions



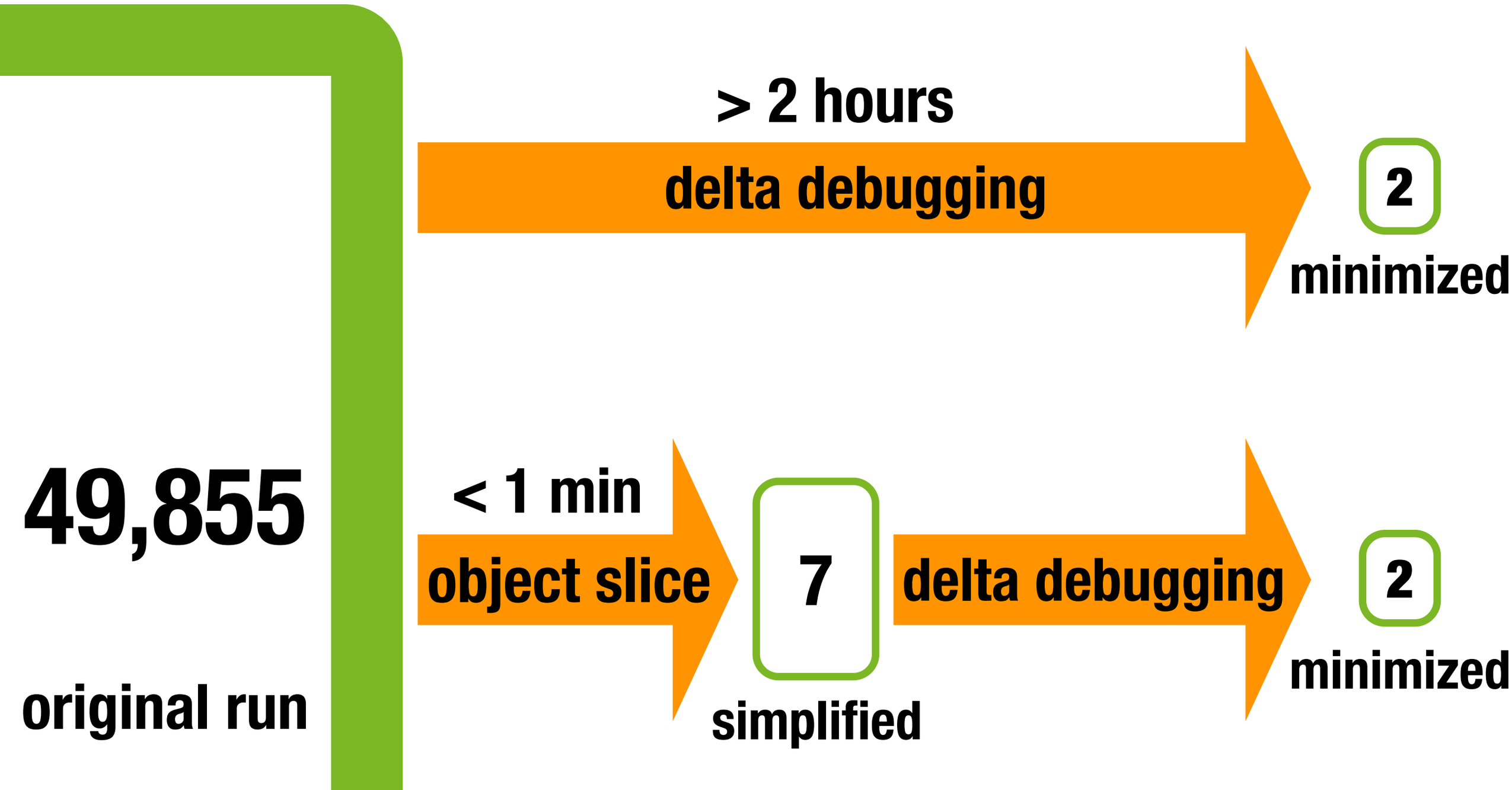
Slicing + Delta Debugging

Incoming Interactions



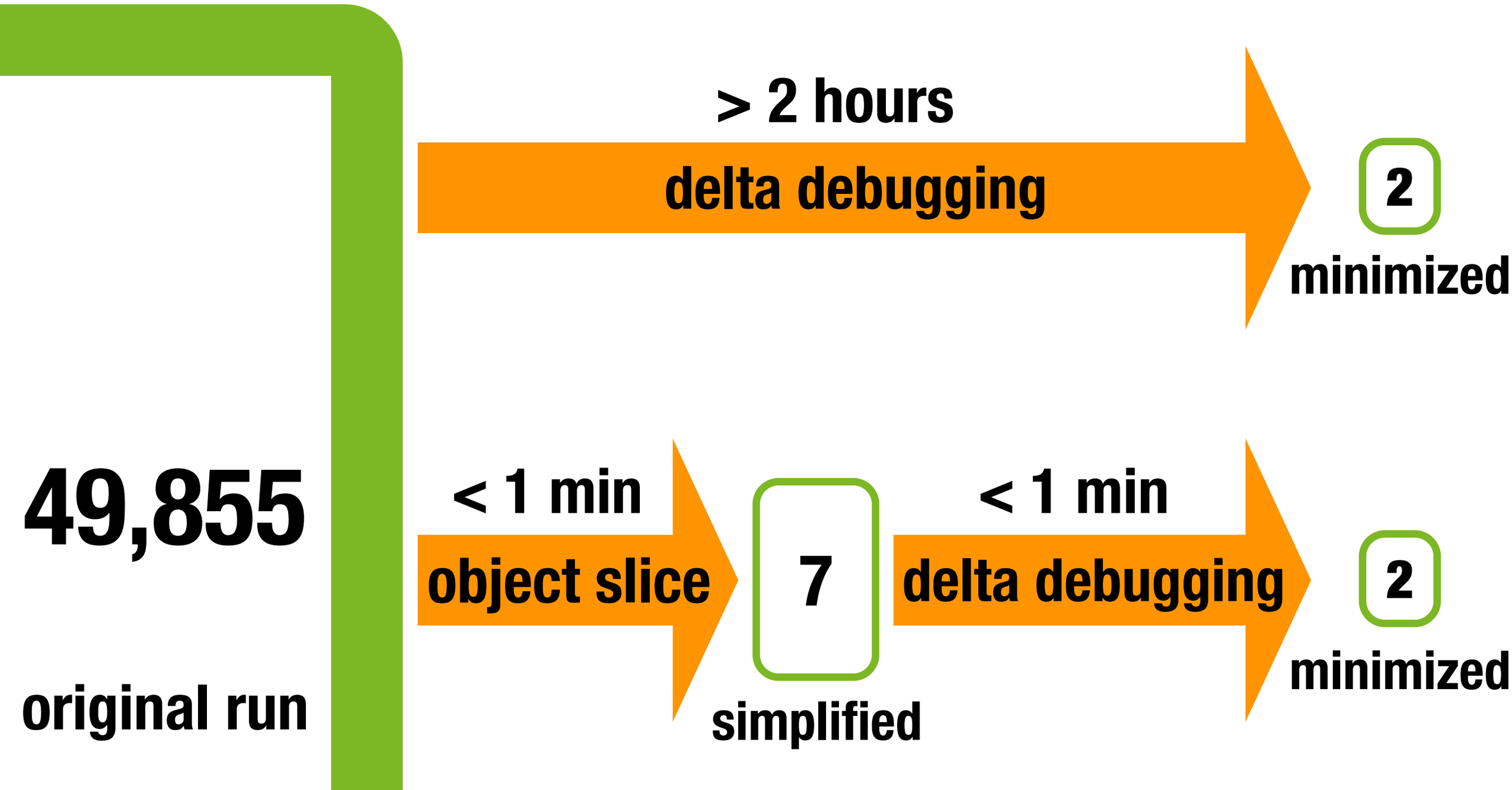
Slicing + Delta Debugging

Incoming Interactions



Slicing + Delta Debugging

Incoming Interactions



Method from Minimal Trace

```
public String getPreferredEmail() {
  Iterator it = getEmailIterator();

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


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

  return model.getAddress();
}
```

Method from Minimal Trace

```
public String getPreferredEmail() {
  Iterator it = getEmailIterator();

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


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

  return model.getAddress();
}
```

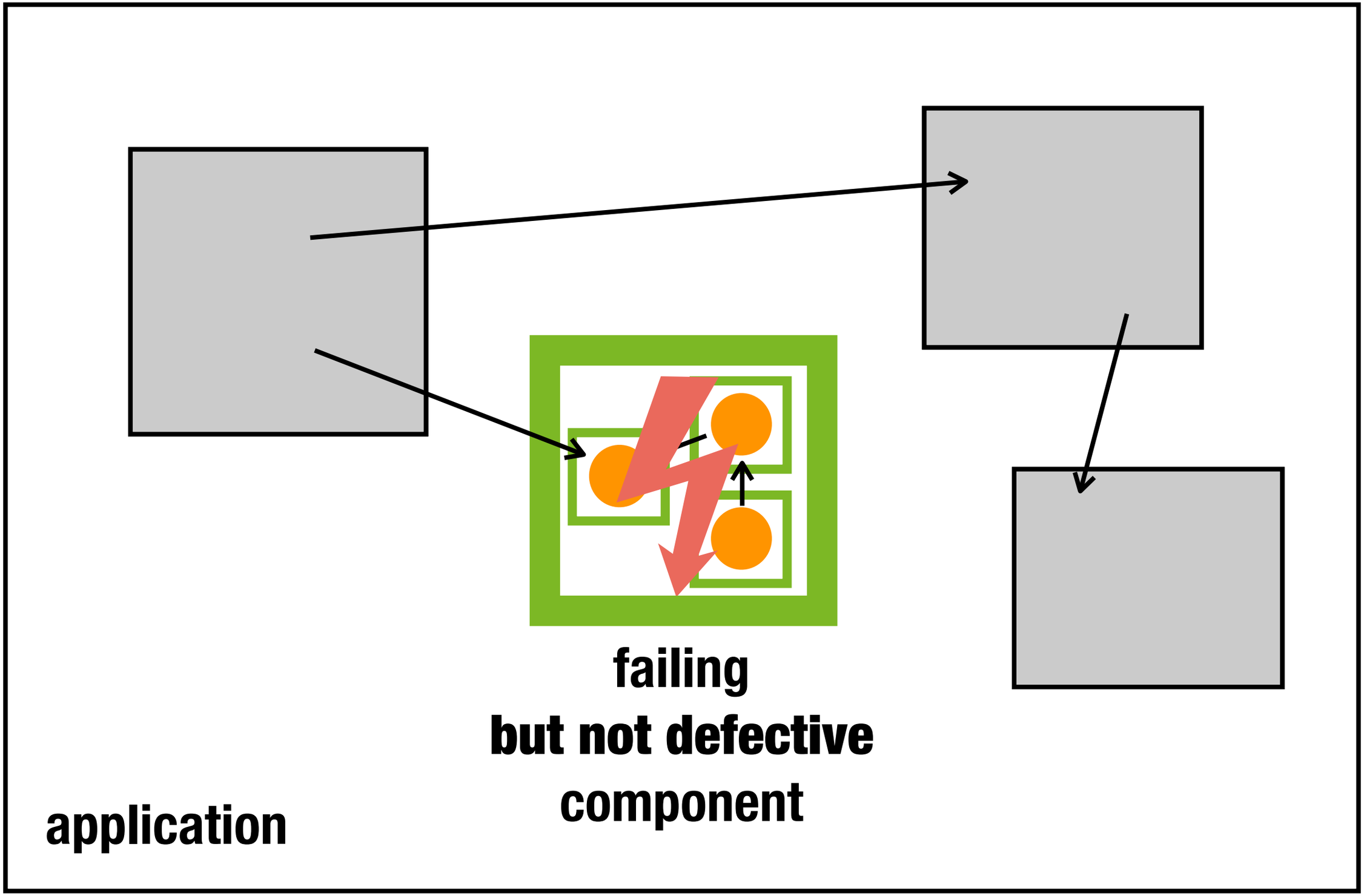
Method from Minimal Trace

```
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();
}
```

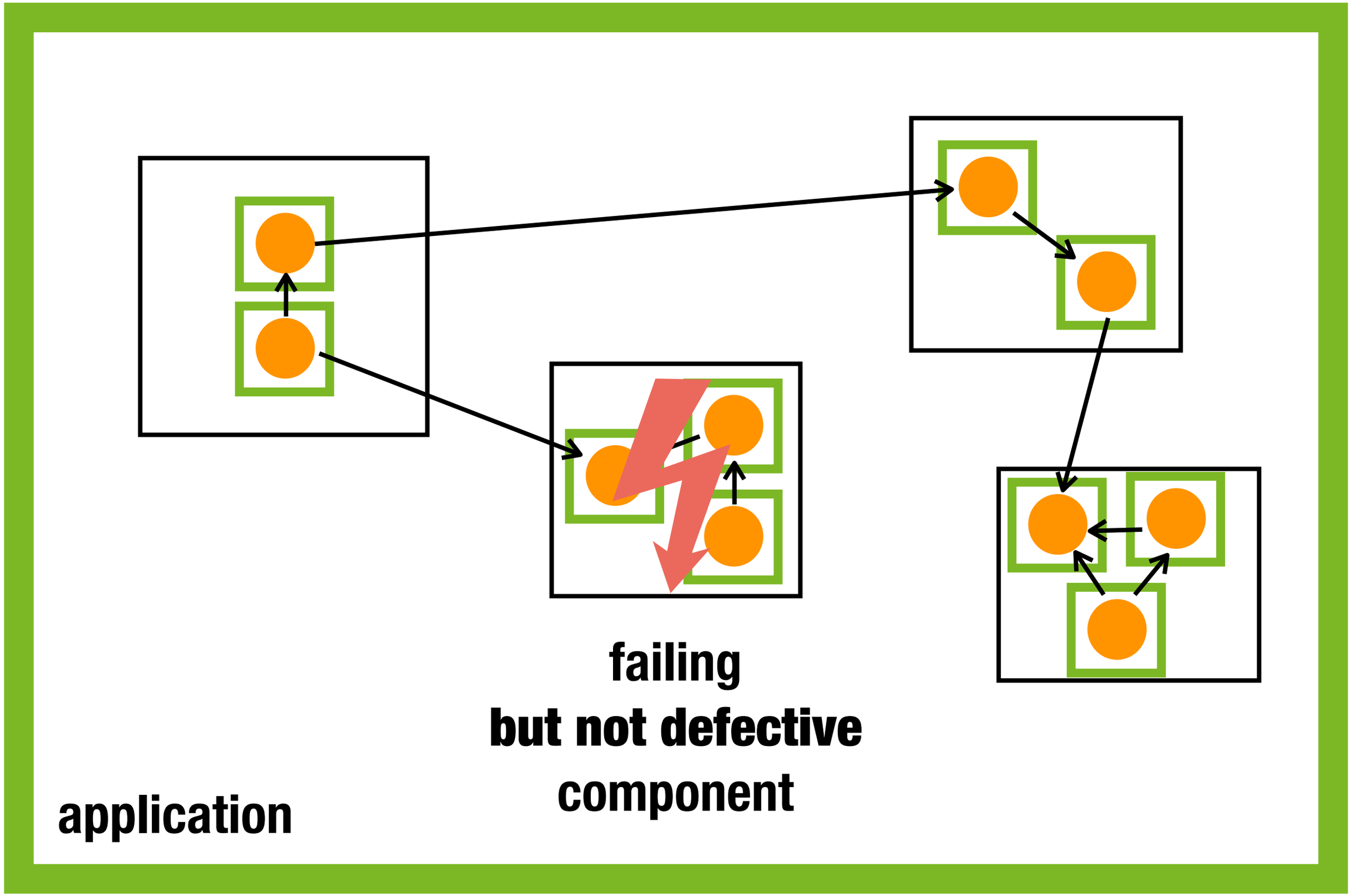
Related Work

- **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

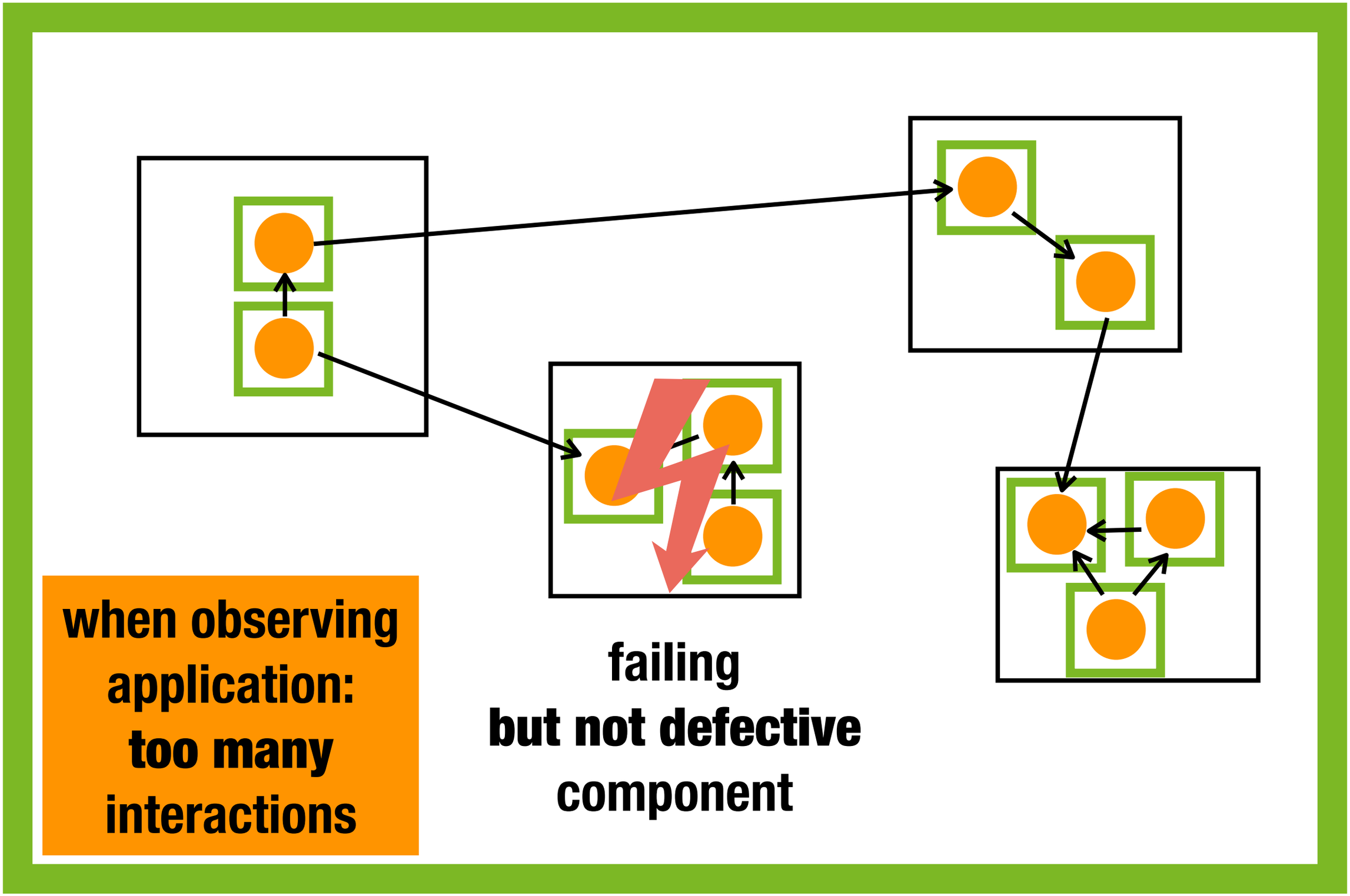
Future Work



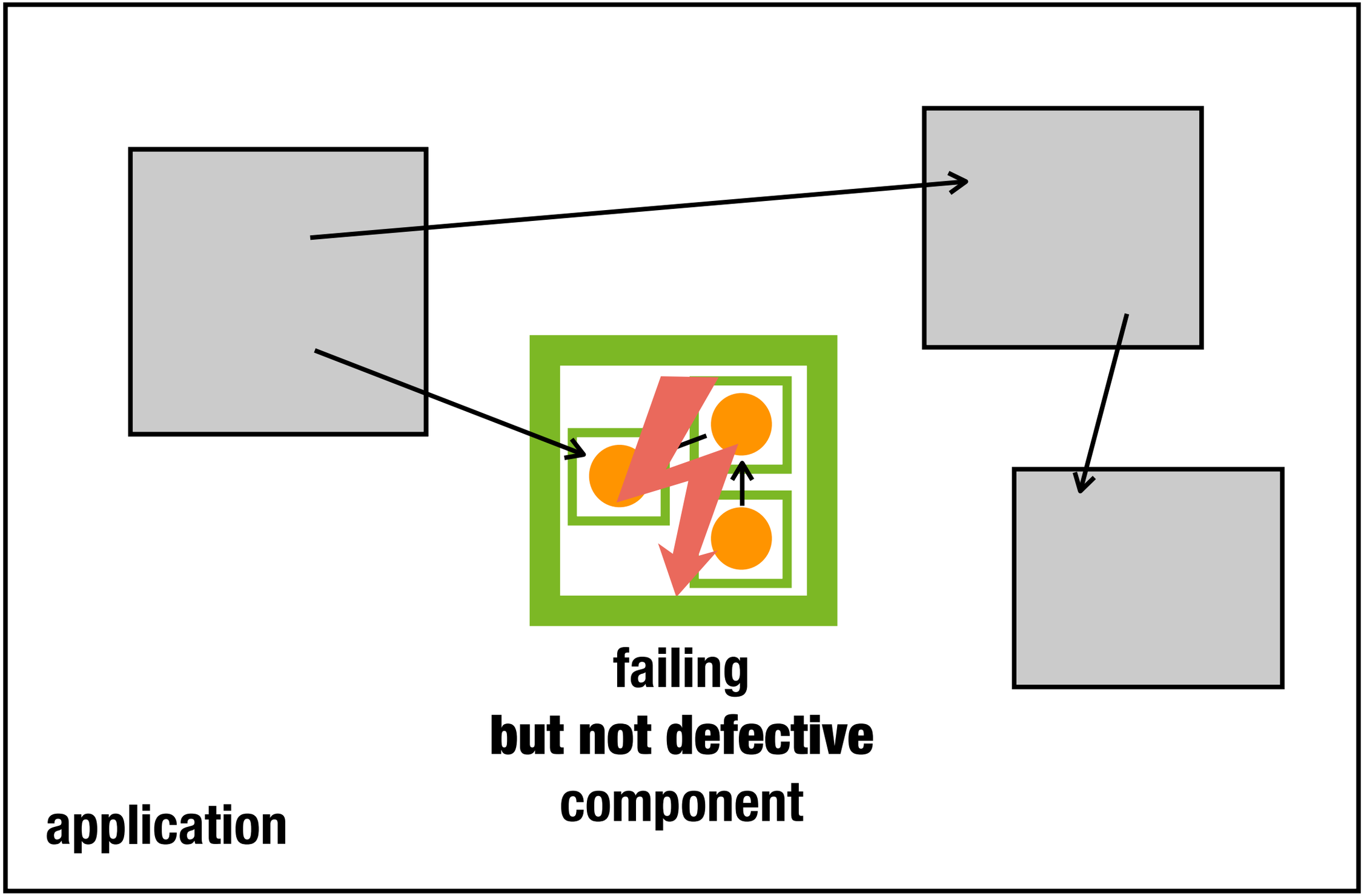
Future Work



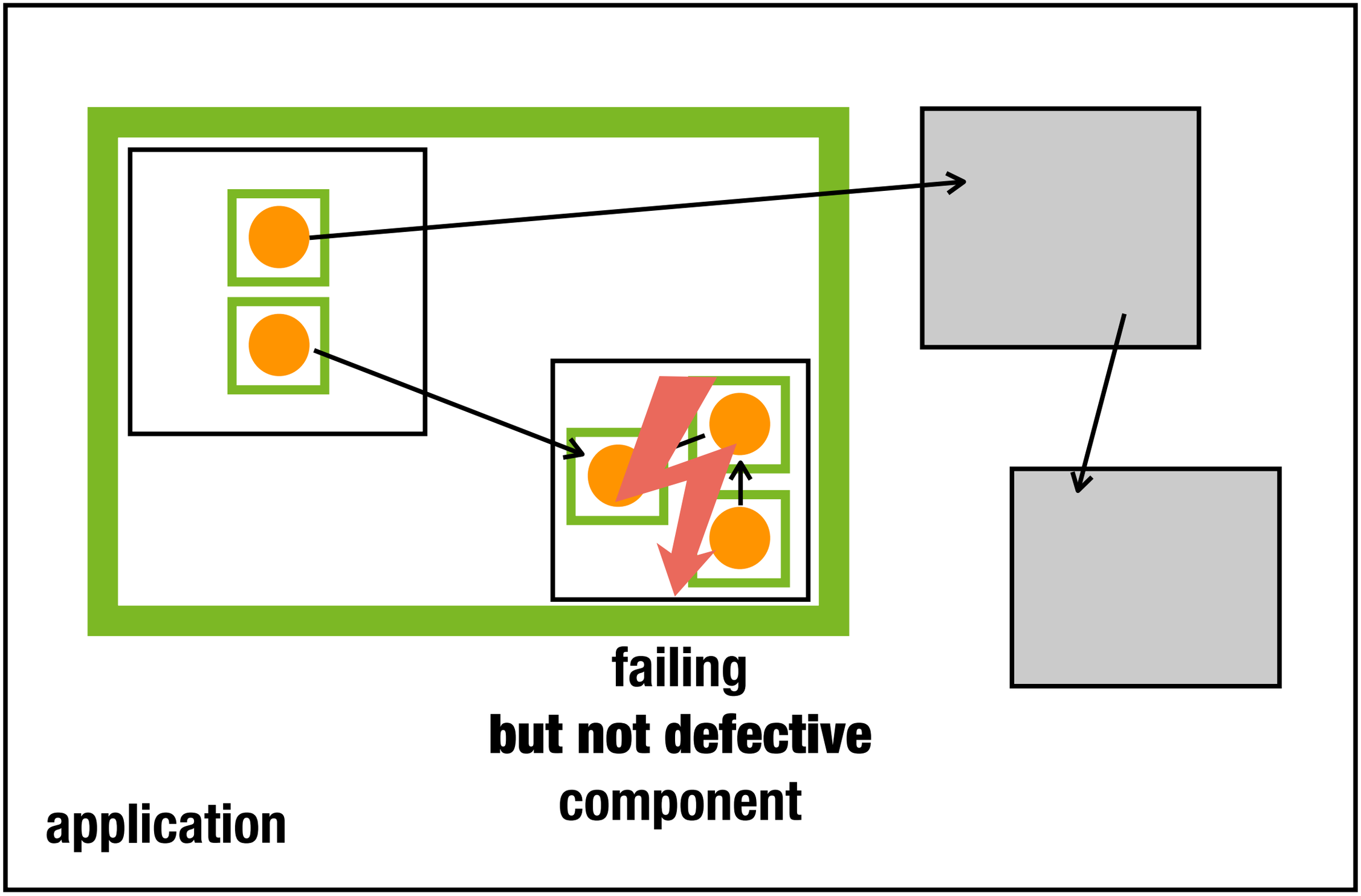
Future Work


when observing application: too many interactions
failing
but not defective
component

Future Work

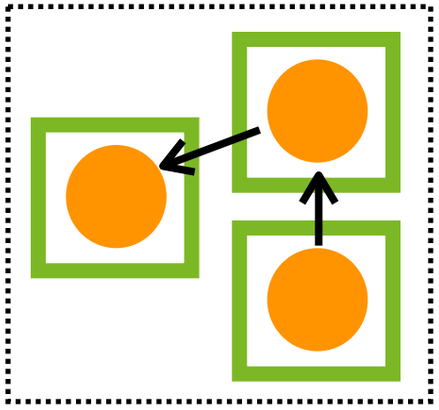


Future Work



Future Work

Cause-Effect-Chain

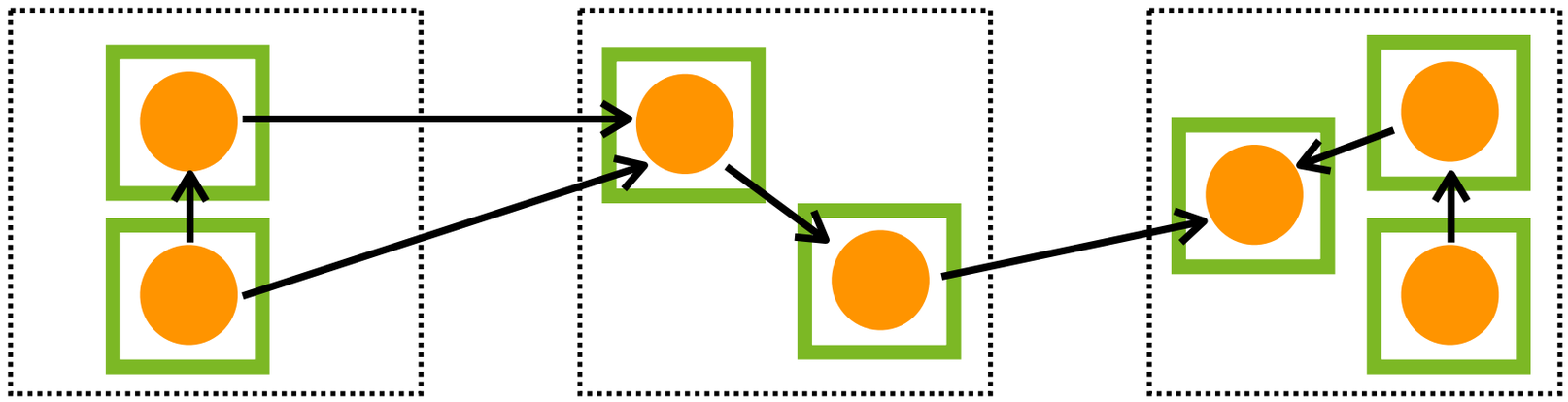


final state

e.g. contact model

Future Work

Cause-Effect-Chain

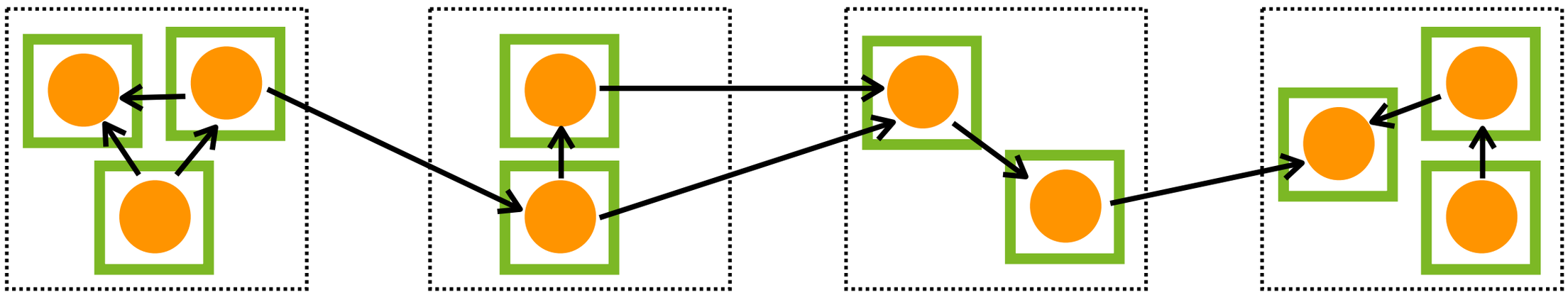


intermediate states **final state**

e.g. contact model

Future Work

Cause-Effect-Chain



input

e.g. keystrokes in GUI

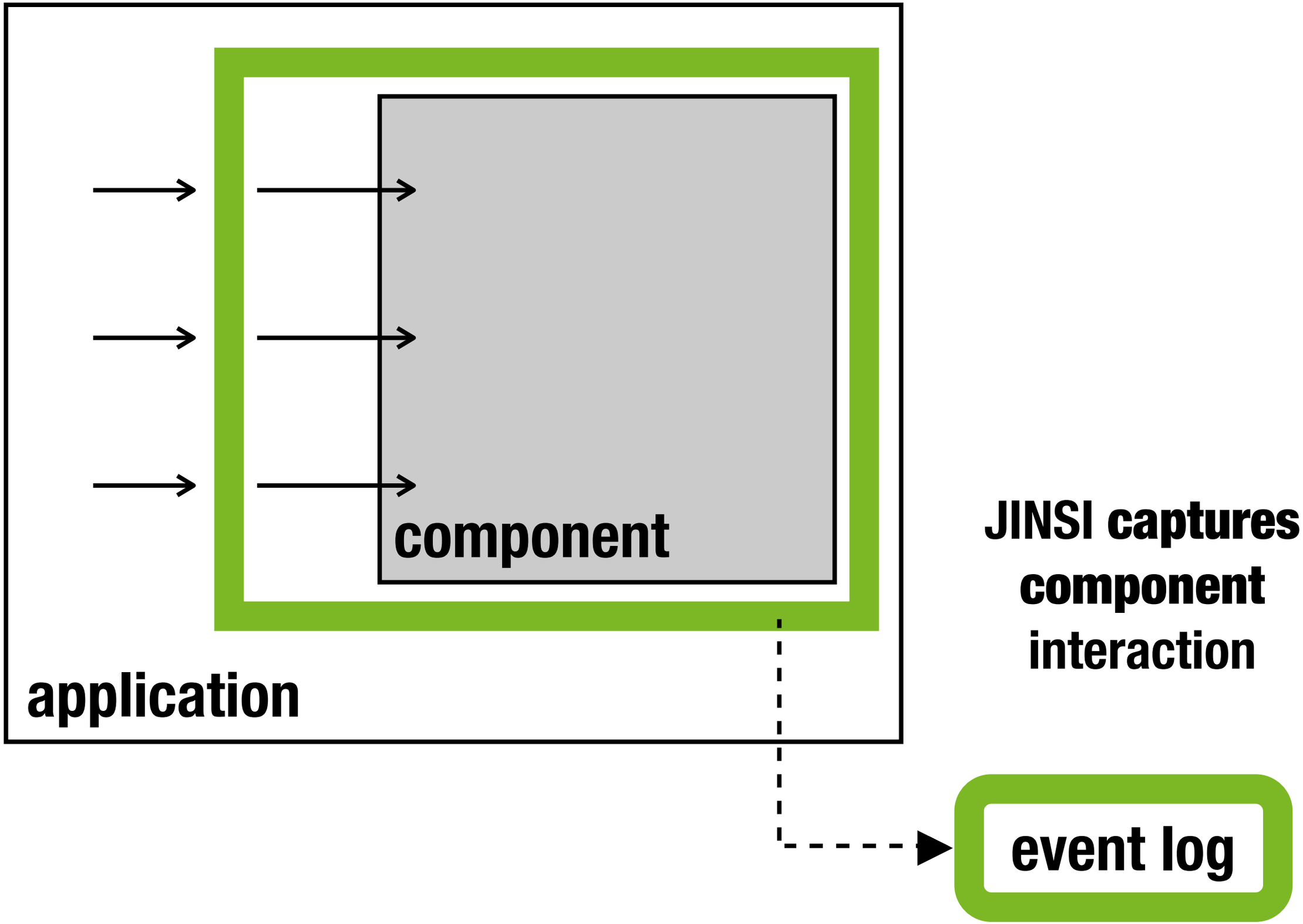
intermediate states

final state

e.g. contact model

Future Work

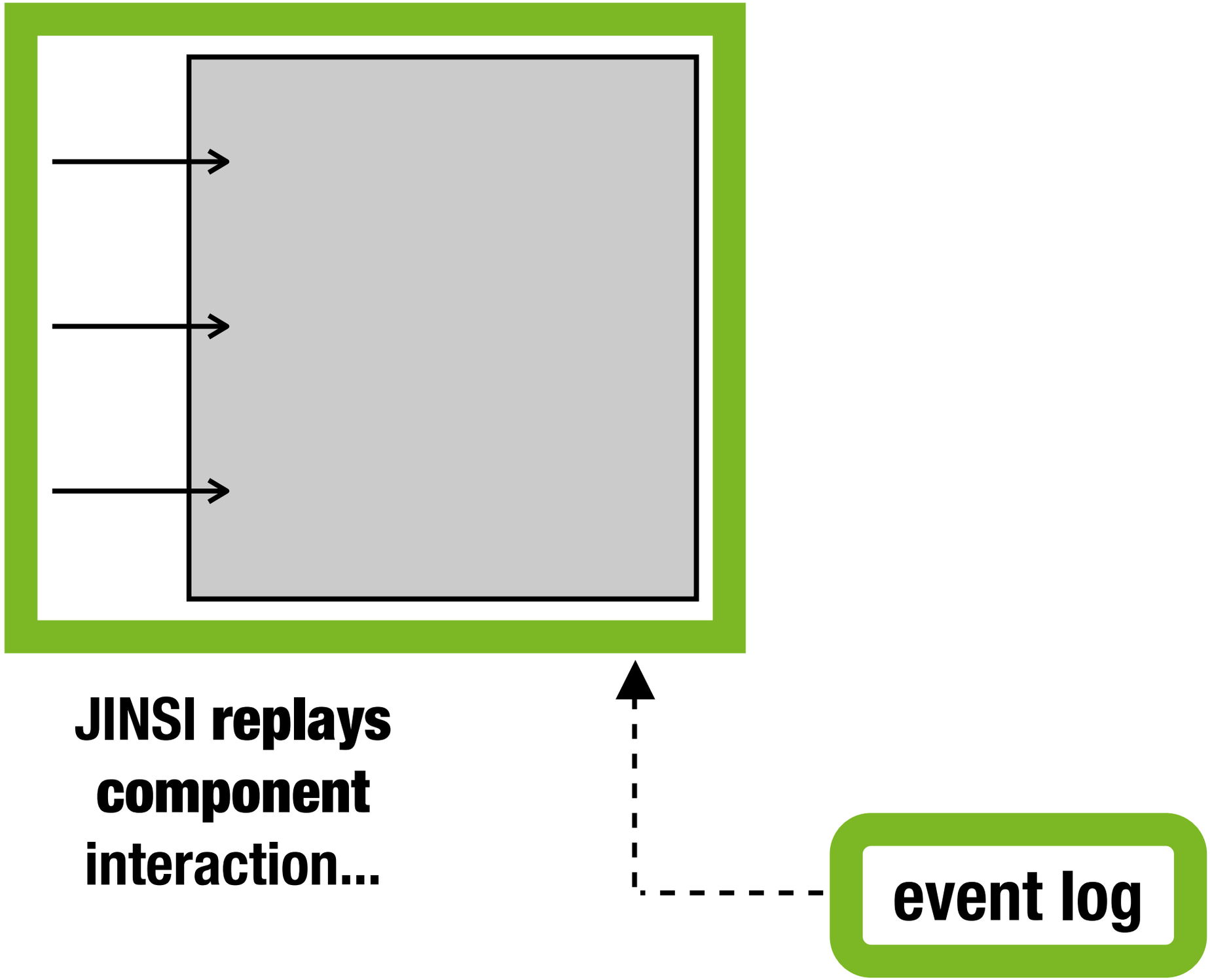
Capture Component Level



JINSI captures component interaction

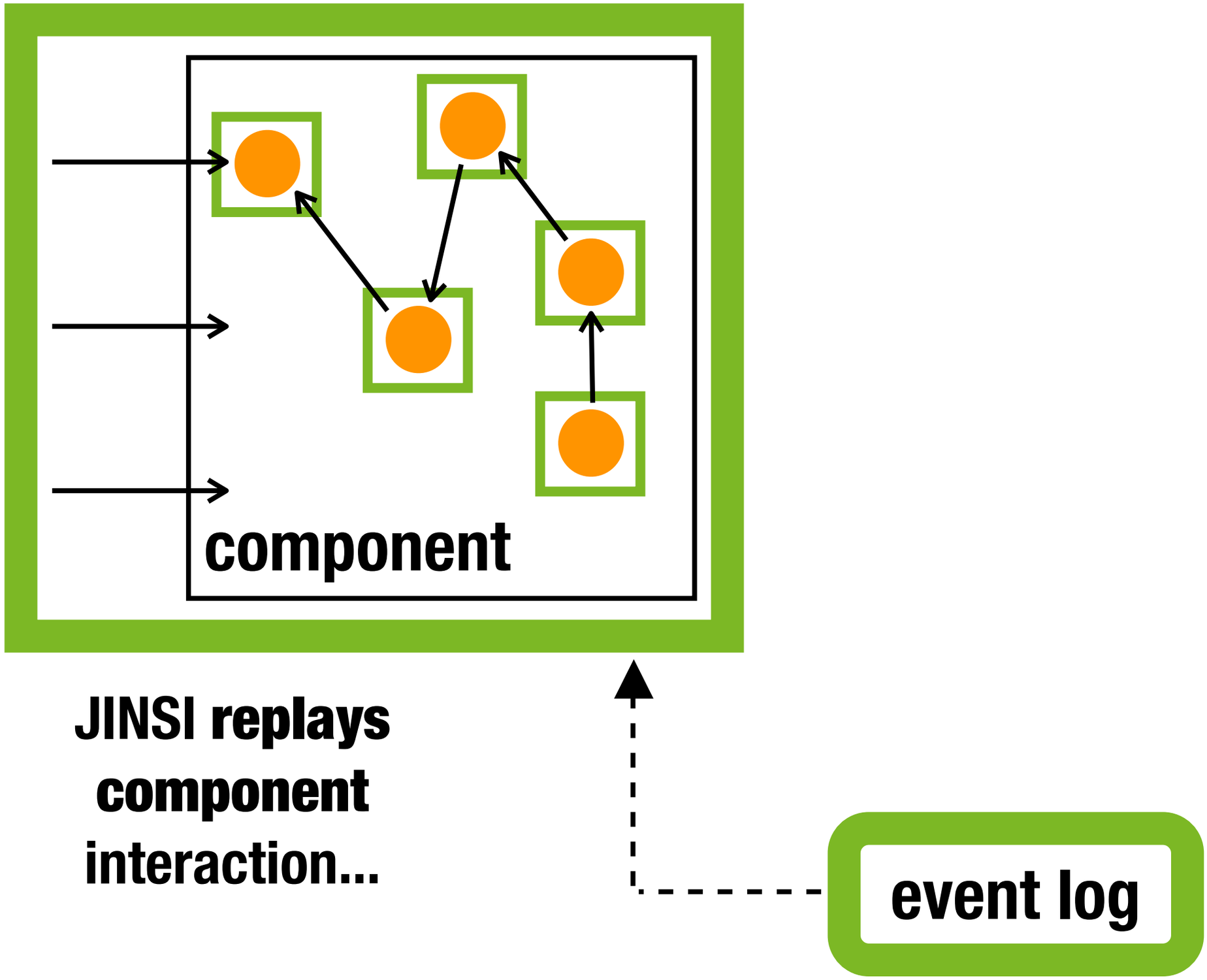
Future Work

Replay Component Level



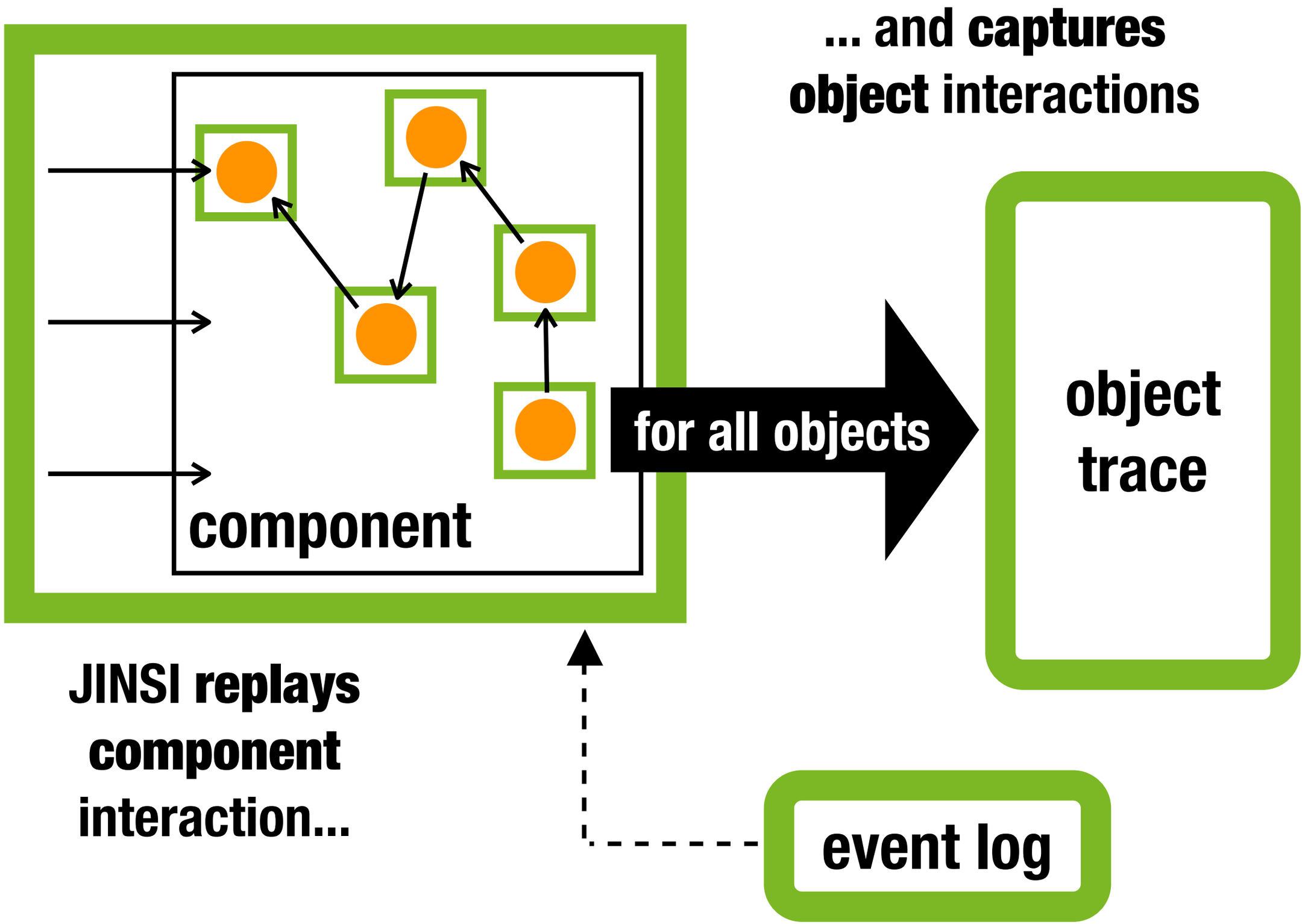
Future Work

Replay Component Level



Future Work

Replay Component + Capture Objects



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

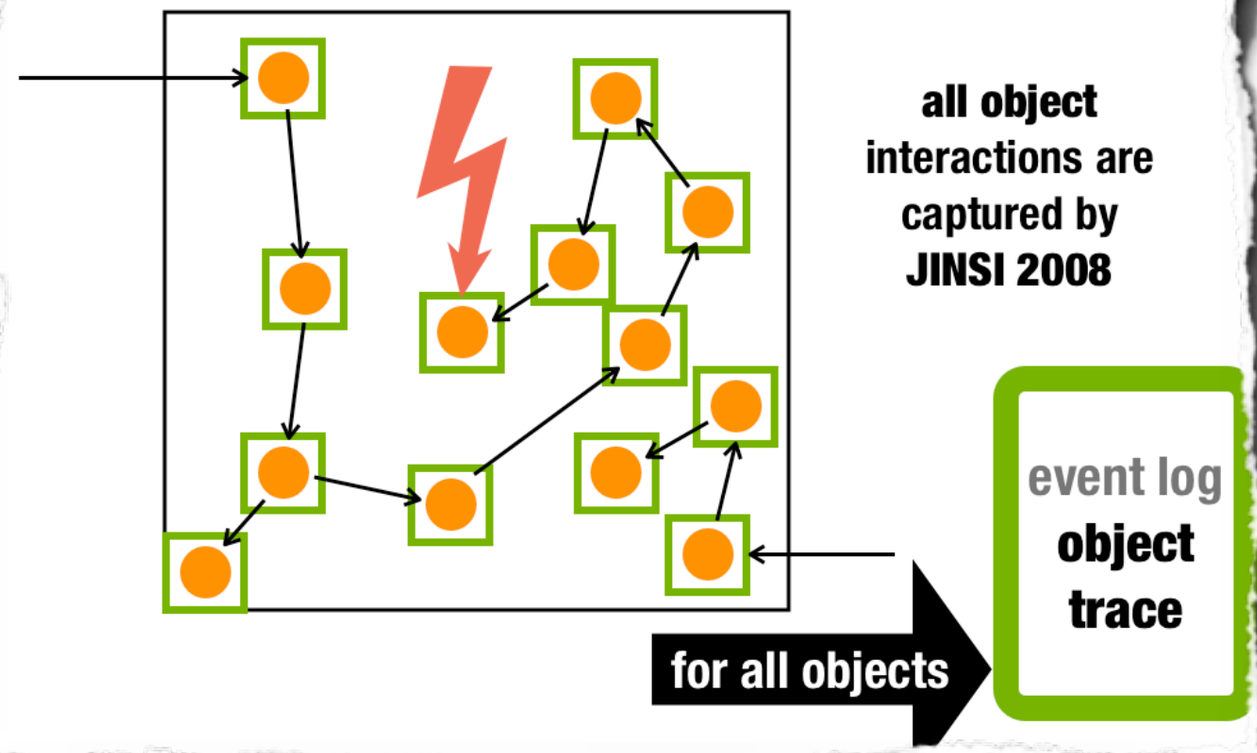
1. **reproduce the original failure** ✔
   - manually by using GUI
   - test driver that reproduce faulty behavior
2. **fix the actual defect** ✔
   - focus on relevant behavior
   - simplify faulty behavior

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

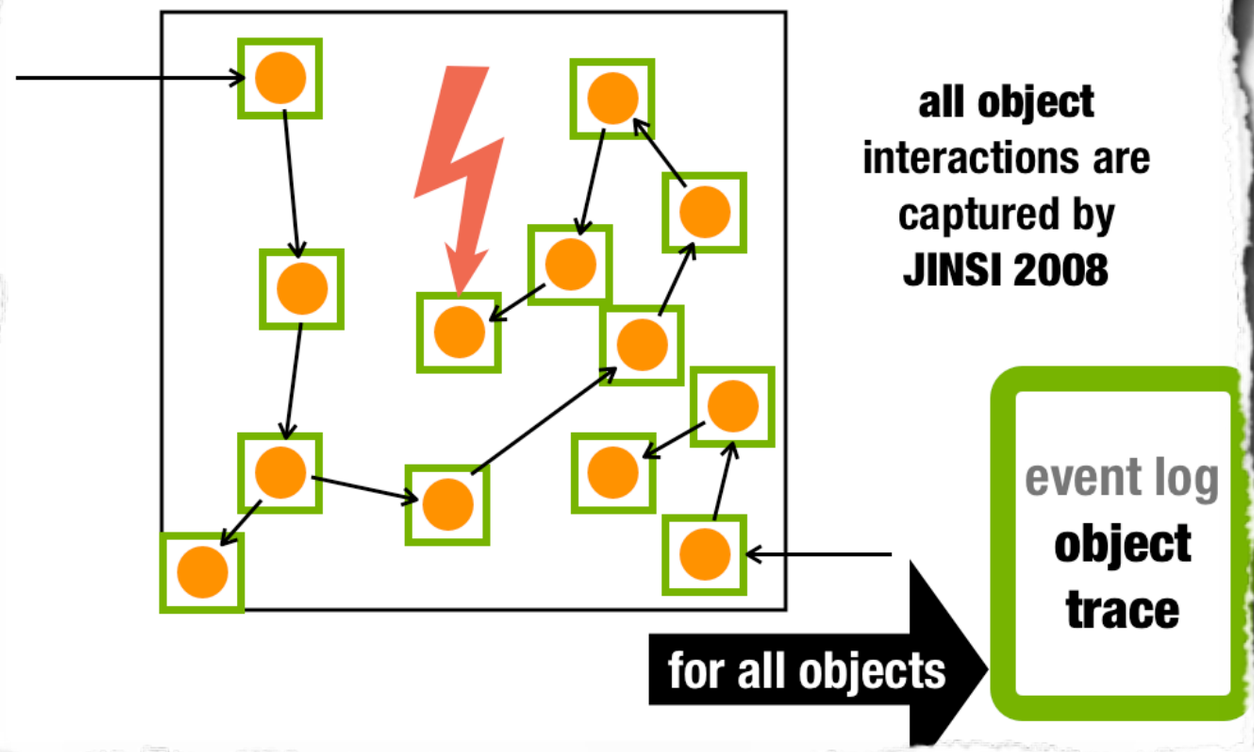


Contributions

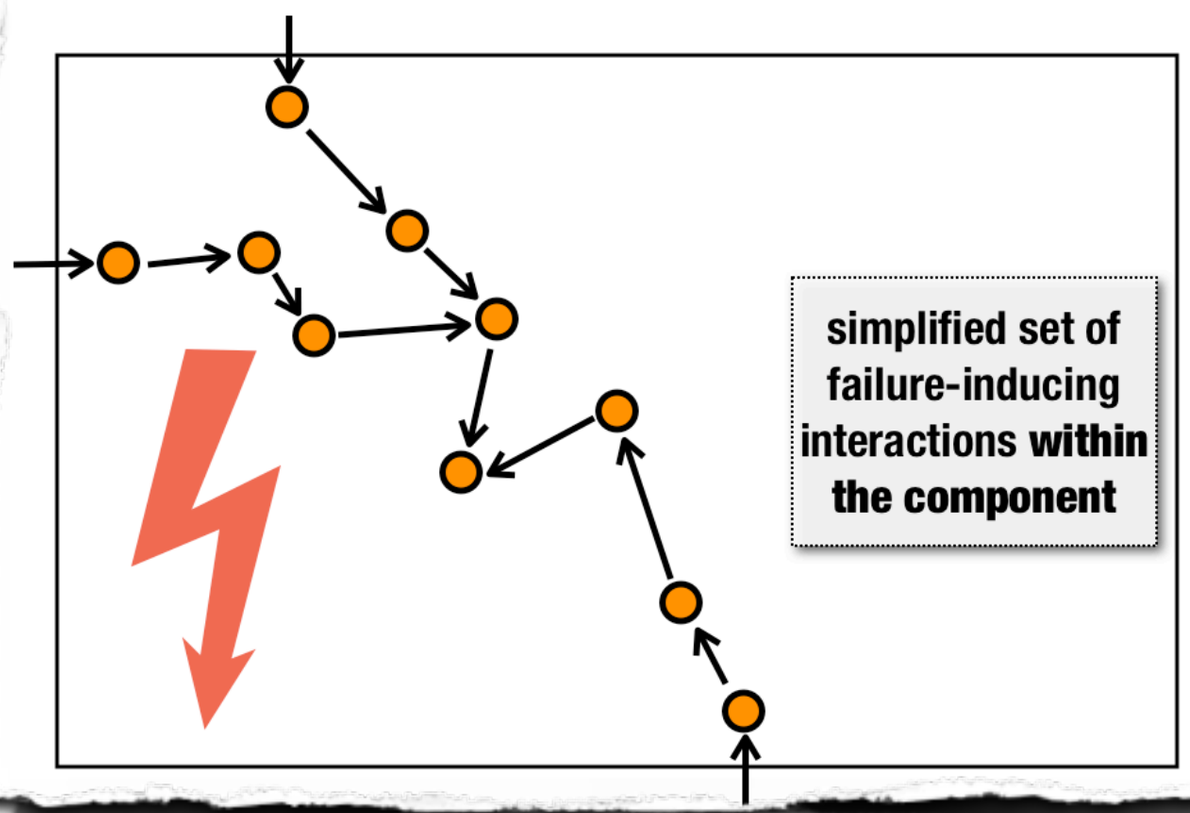
Debugging 101

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



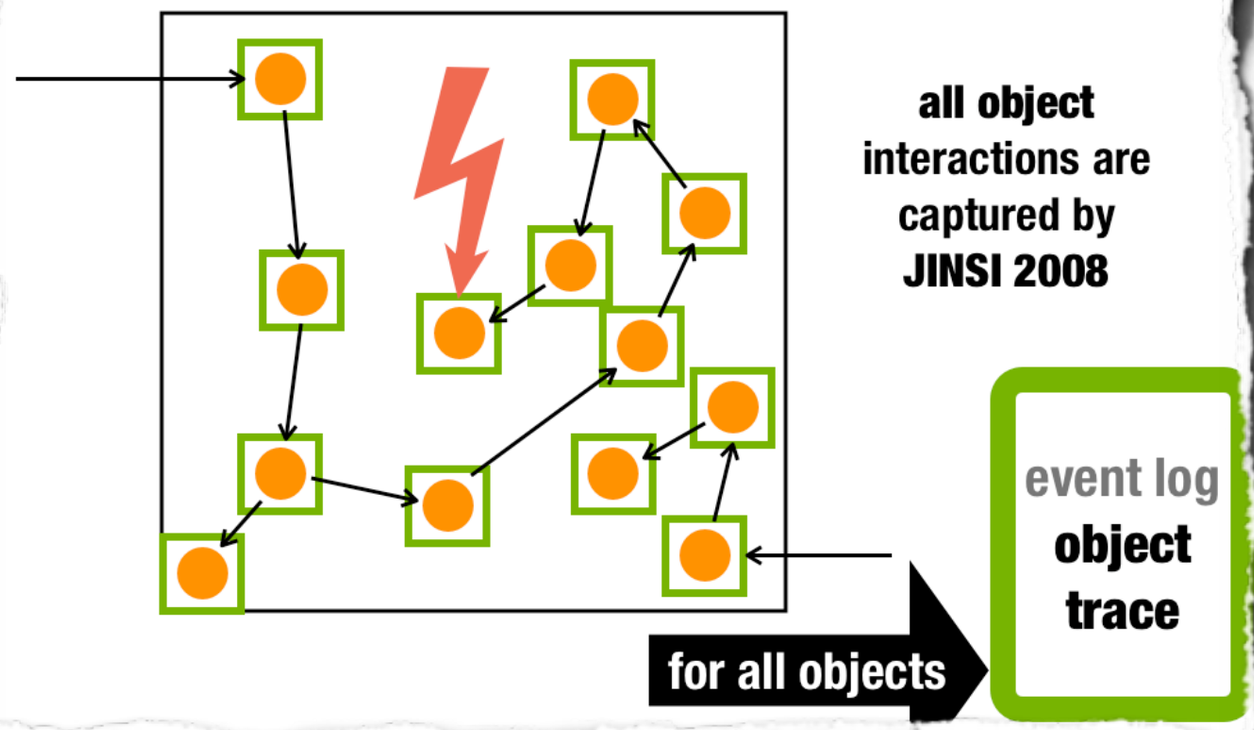
Object Slice



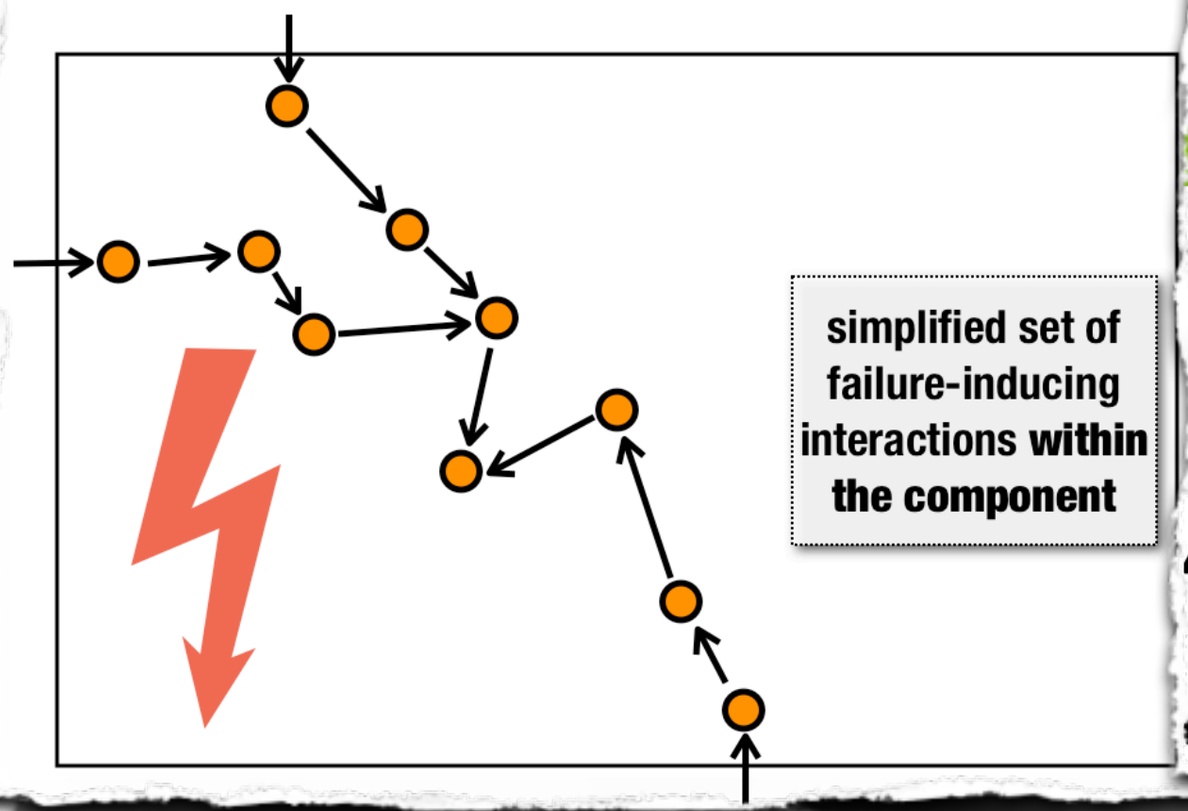
Debugging 101

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



Object Slice



Slicing + Delta Debugging

Incoming Interactions