Testing and Maintenance of Graphical User Interfaces

Valéria Lelli

PhD Thesis Defense, 19th November 2015

INSA/INRIA, Rennes – France

Jury

Lydie du BOUSQUET (Rapporteur)	Francois-Xavier DORMOY (Examinateur)
Philippe PALANQUE (Rapporteur)	Benoit BAUDRY (Directeur)
Pascale SÉBILLOT (Examinateur)	Arnaud BLOUIN (Co-encadrant)

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

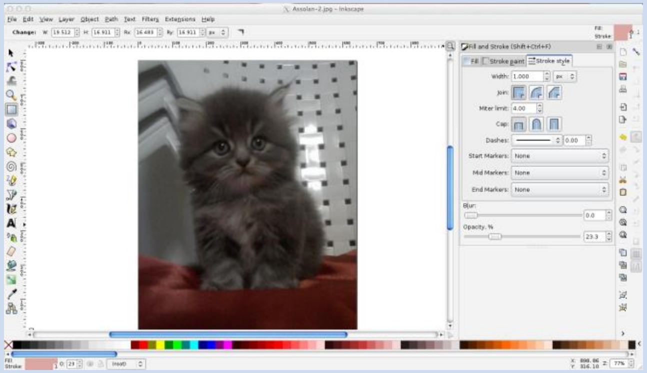
Inria INVENTORS FOR THE DIGITAL WORLD

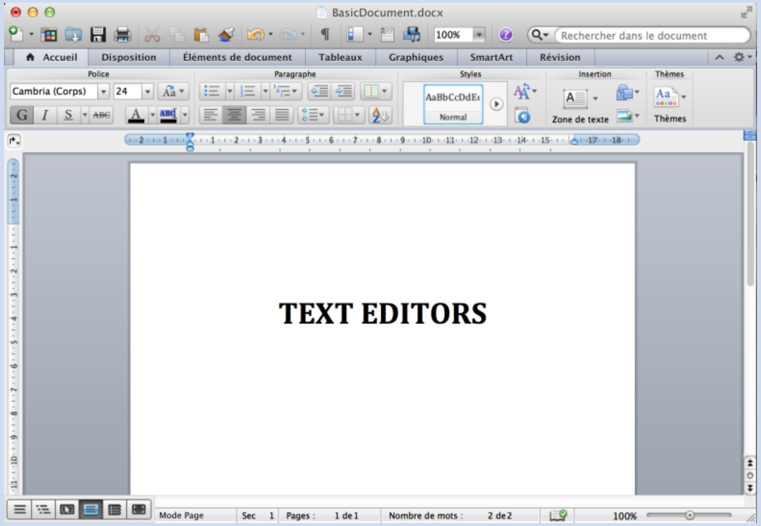
DiverSE Diversity-Centric Software Engineering

Connexion Contrôle-commande numérique pour le nucléaire

Context

- **Graphical user interfaces (GUIs)**
  - Designed for being controlled by the users
  - Composed of graphical interactive widgets



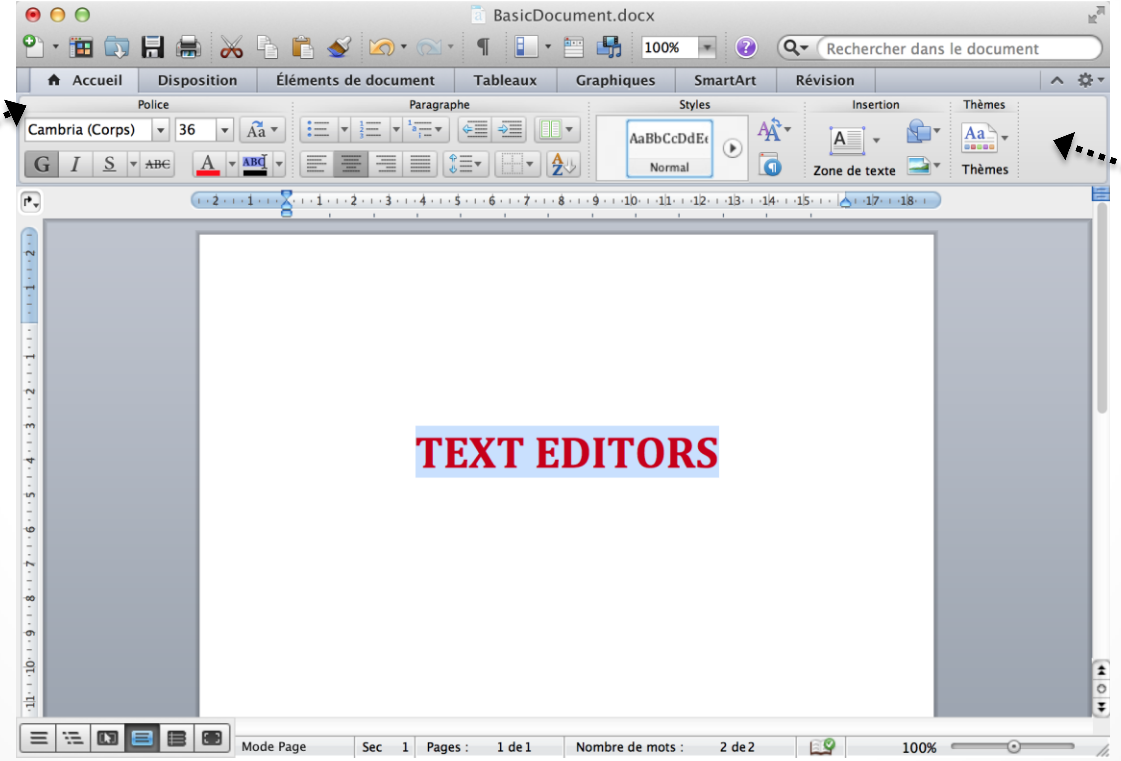

TEXT EDITORS

Set up Firefox Sync to access bookmarks, history and tabs from your other devices »

Validation of GUIs

- GUI designers concern the design and qualitative assessment of GUIs

Arrangement of widgets

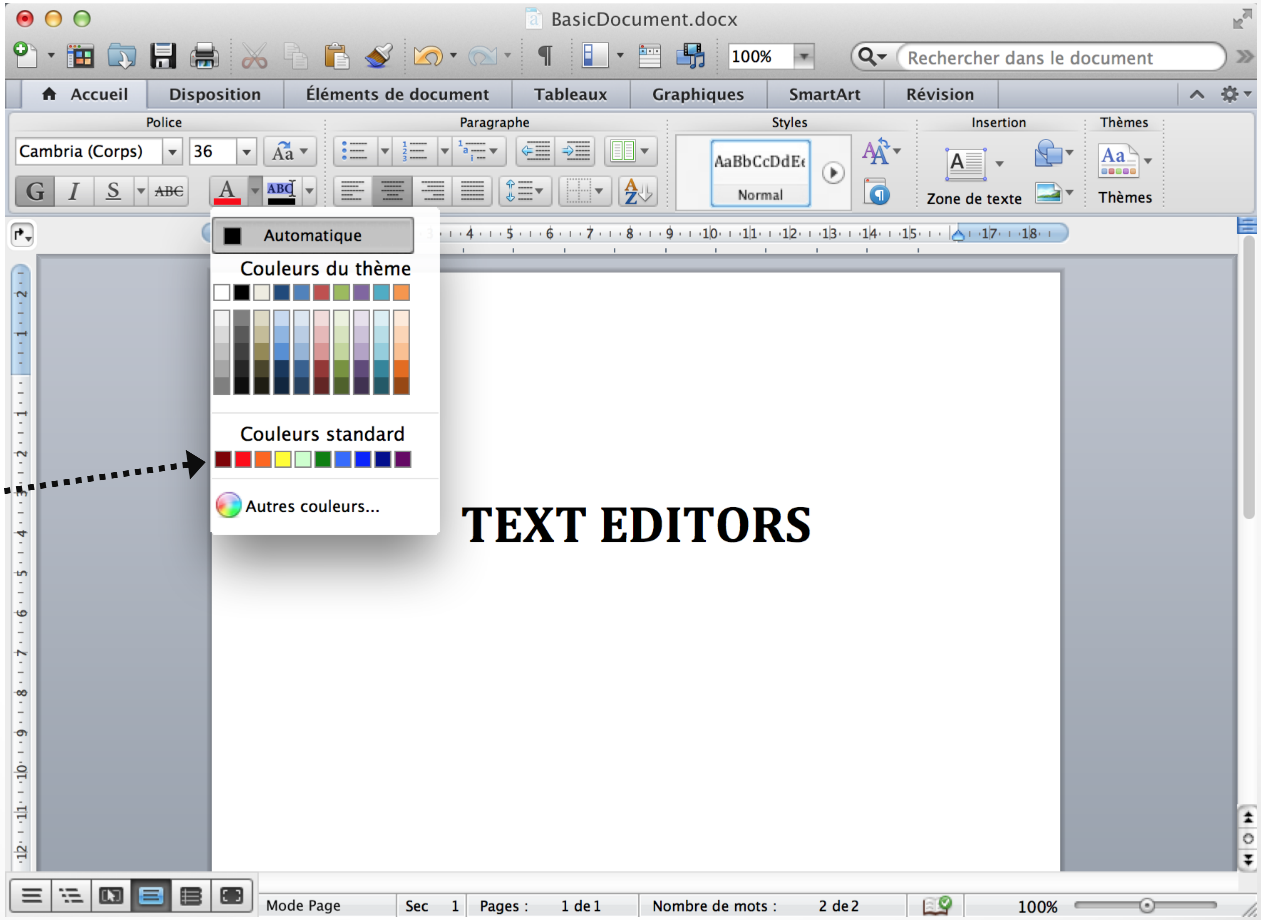


TEXT EDITORS

Using suitable colors

Validation of GUIs

- Software engineers ensure that
  - GUIs react correctly to user interactions



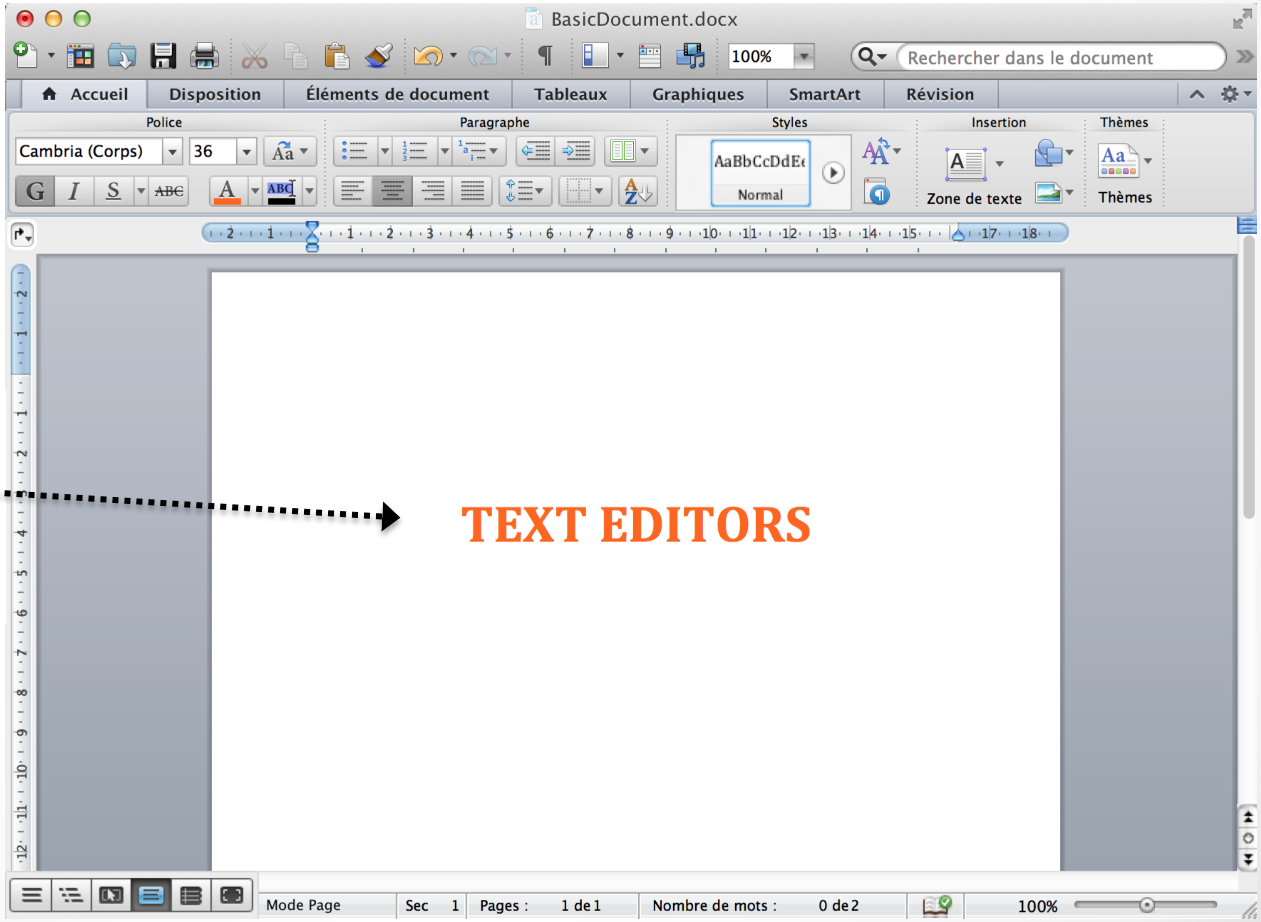
TEXT EDITORS

Pressing on a button

Validation of GUIs

- Software engineers ensure that
  - GUIs react correctly to user interactions

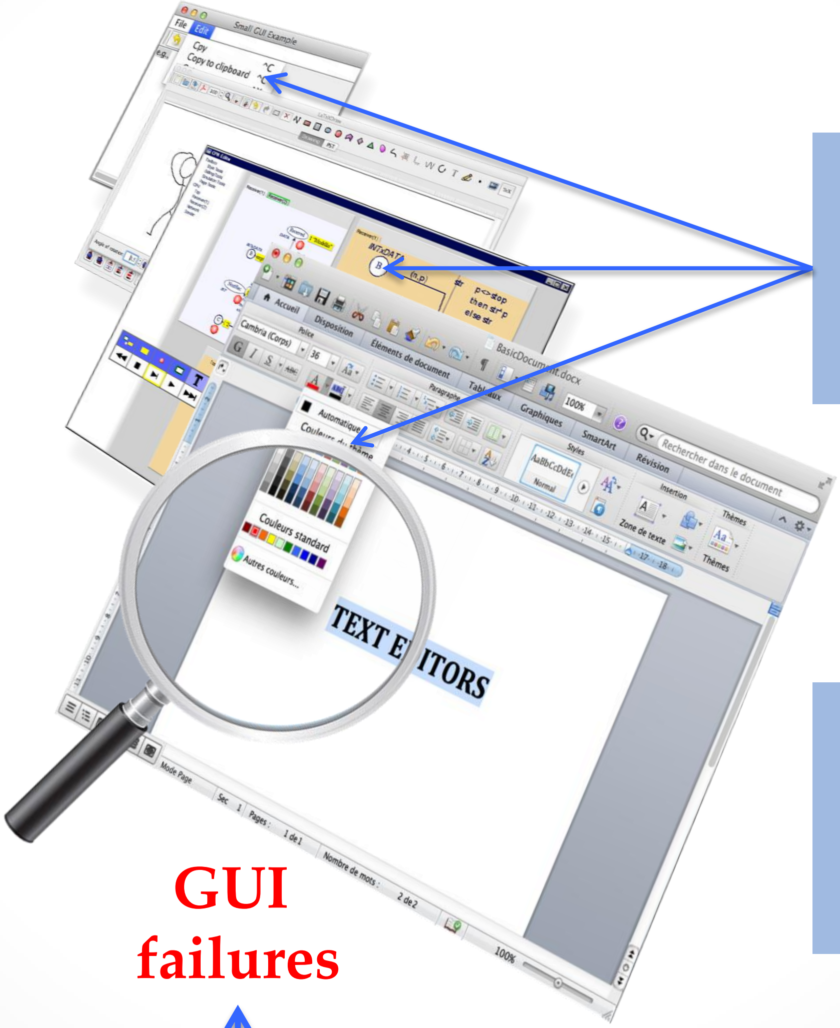
produces the expected action



TEXT EDITORS

GUI testing

GUI testing tools



Numerous and different kinds of widgets

GUI faults are multiple and diverse

GUI failures

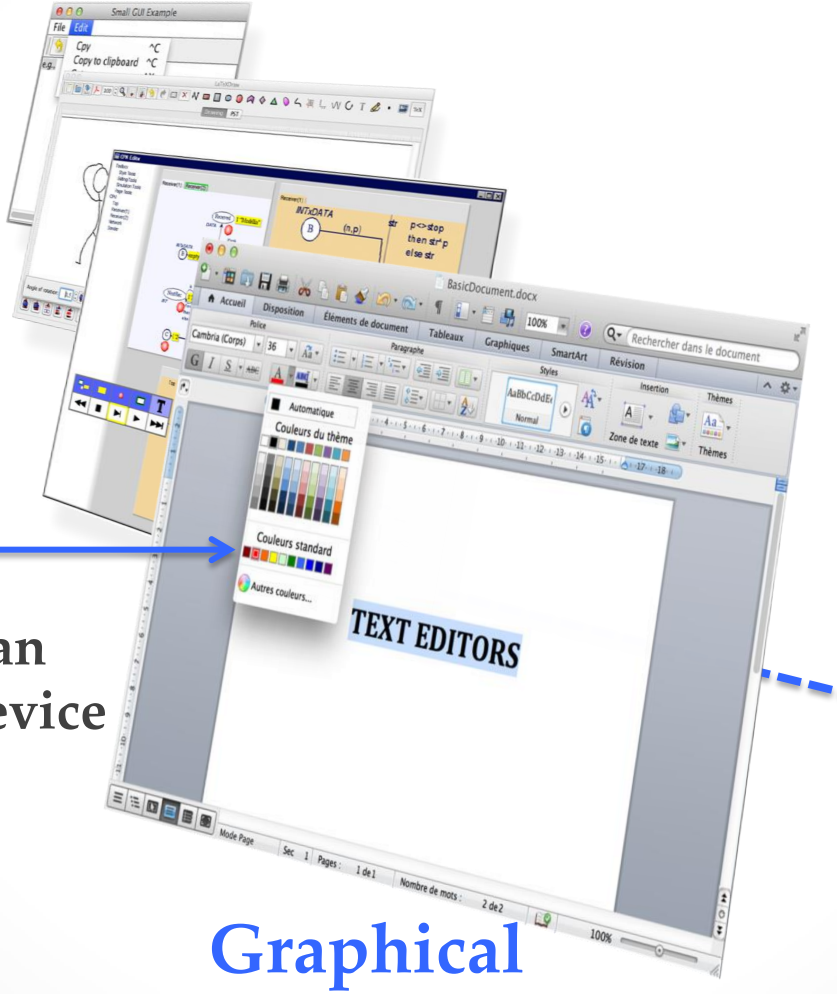
GUI testing techniques

- Capture and Replay tools
  - Recording user interactions to be replayed
- Monkey tools
  - Sending random events such as mouse events
- Functional GUI testing tools
  - Pre-defined libraries to write test cases
- Event-flow graphs
  - Based on the sequence of events to automatically generate test cases
  - ✗ GUI failures from the recent GUI developments

Graphical User Interfaces

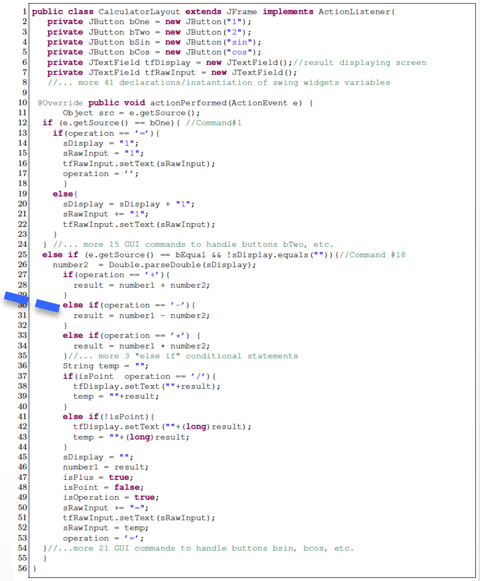
Users

Human input device



Graphical elements

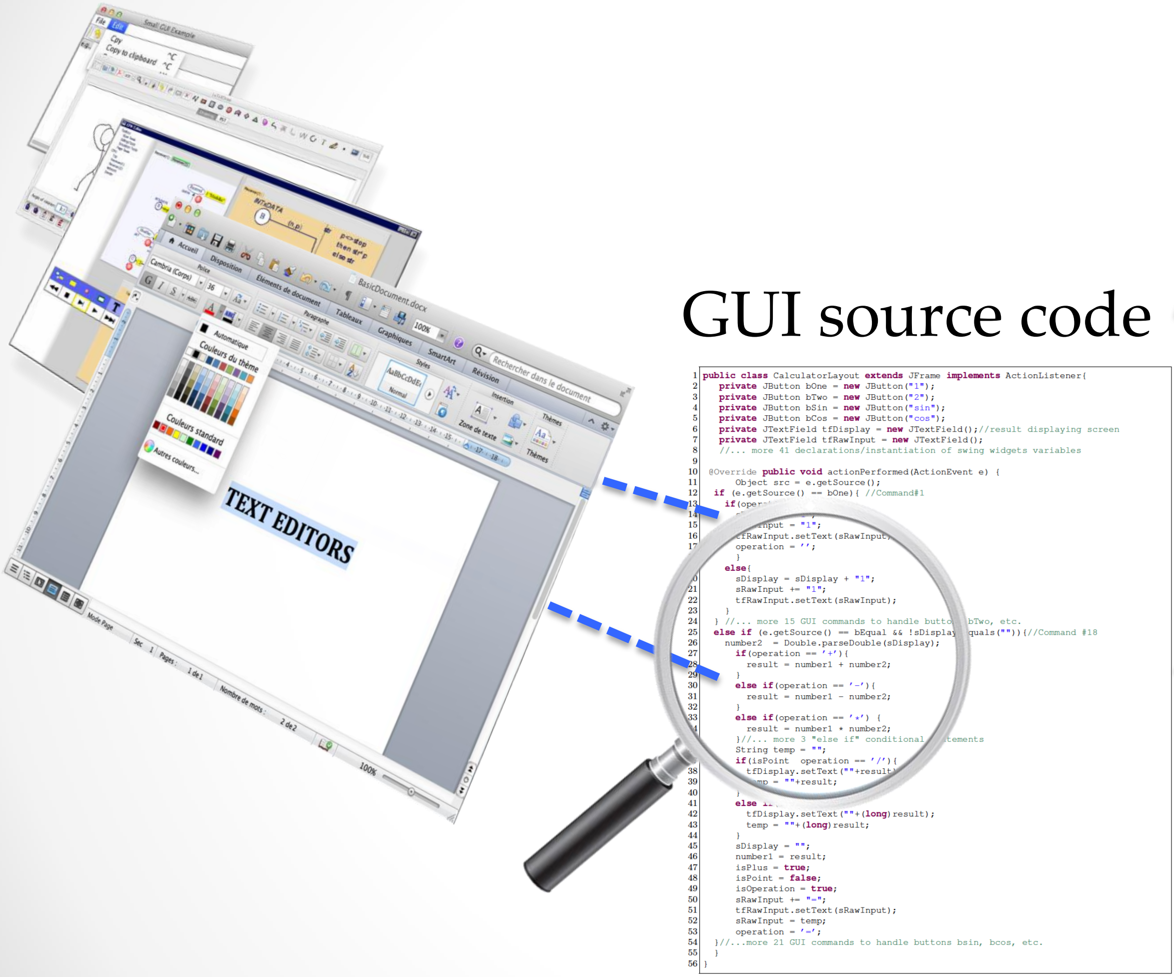
GUI source code



```
public class CalculatorLayout extends JFrame implements ActionListener{
   private JButton bOne = new JButton("1");
   private JButton bTwo = new JButton("2");
   private JButton bSin = new JButton("sin");
   private JButton bCos = new JButton("cos");
   private JTextField tfDisplay = new JTextField();//result displaying screen
   private JTextField tfRawInput = new JTextField();
   //... more 41 declarations/instantiation of swing widgets variables

 @Override public void actionPerformed(ActionEvent e) {
      Object src = e.getSource();
  if (e.getSource() == bOne){ //Command#1
    if(operation == ’=’){
      sDisplay = "1";
      sRawInput = "1";
      tfRawInput.setText(sRawInput);
      operation = ’’;
      }
    else{
      sDisplay = sDisplay + "1";
      sRawInput += "1";
      tfRawInput.setText(sRawInput);
    }
  } //... more 15 GUI commands to handle buttons bTwo, etc.
  else if (e.getSource() == bEqual && !sDisplay.equals("")){//Command #18
    number2  = Double.parseDouble(sDisplay);
      if(operation == ’+’){
        result = number1 + number2;
      }
      else if(operation == ’-’){
        result = number1 - number2;
      }
      else if(operation == ’*’) {
        result = number1 * number2;
      }//... more 3 "else if" conditional statements
      String temp = "";
      if(isPoint  operation == ’/’){
        tfDisplay.setText(""+result);
        temp = ""+result;
      }
      else if(!isPoint){
        tfDisplay.setText(""+(long)result);
        temp = ""+(long)result;
      }
      sDisplay = "";
      number1 = result;
      isPlus = true;
      isPoint = false;
      isOperation = true;
      sRawInput += "=";
      tfRawInput.setText(sRawInput);
      sRawInput = temp;
      operation = ’=’;
  }//...more 21 GUI commands to handle buttons bsin, bcos, etc.
  }
}
```

Validation of GUIs

GUI source code



```
public class CalculatorLayout extends JFrame implements ActionListener{
   private JButton bOne = new JButton("1");
   private JButton bTwo = new JButton("2");
   private JButton bSin = new JButton("sin");
   private JButton bCos = new JButton("cos");
   private JTextField tfDisplay = new JTextField();//result displaying screen
   private JTextField tfRawInput = new JTextField();
   //... more 41 declarations/instantiation of swing widgets variables

 @Override public void actionPerformed(ActionEvent e) {
      Object src = e.getSource();
  if (e.getSource() == bOne){ //Command#1
    if(oper
      s
      nput = "1";
      tfRawInput.setText(sRawInput)
      operation = '';
      }
    else{
      sDisplay = sDisplay + "1";
      sRawInput += "1";
      tfRawInput.setText(sRawInput);
    }
  } //... more 15 GUI commands to handle butto  bTwo, etc.
  else if (e.getSource() == bEqual && !sDisplay quals("")){//Command #18
    number2  = Double.parseDouble(sDisplay);
      if(operation == '+'){
        result = number1 + number2;
      }
      else if(operation == '-'){
        result = number1 - number2;
      }
      else if(operation == '*') {
        result = number1 * number2;
      }//... more 3 "else if" conditional  tements
      String temp = "";
      if(isPoint  operation == '/'){
        tfDisplay.setText(""+result)
        mp = ""+result;
      }
      else i
        tfDisplay.setText(""+(long)result);
        temp = ""+(long)result;
      }
      sDisplay = "";
      number1 = result;
      isPlus = true;
      isPoint = false;
      isOperation = true;
      sRawInput += "=";
      tfRawInput.setText(sRawInput);
      sRawInput = temp;
      operation = '=';
  }//...more 21 GUI commands to handle buttons bsin, bcos, etc.
  }
}
```

60% of the total software

Few works focus on GUI code analysis

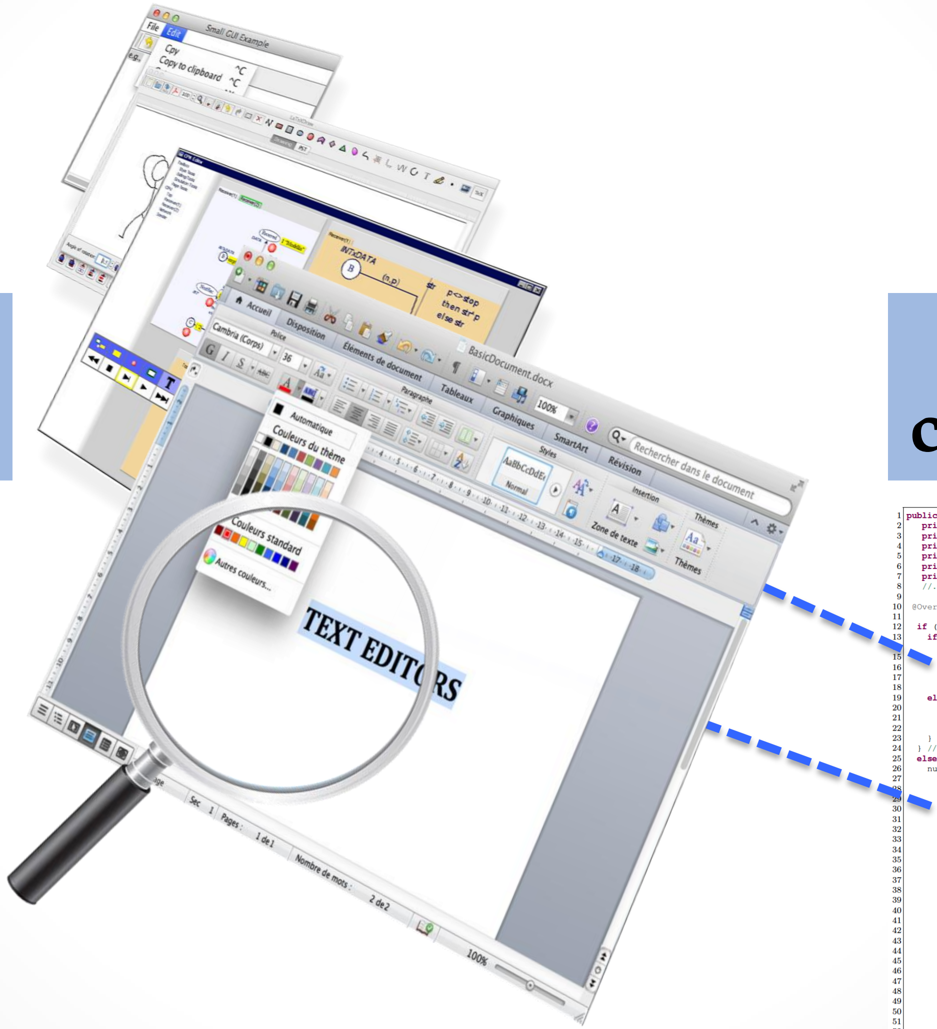
GUI code analysis

- GUI design smells
  - Bad coding practices that degrade GUI source code

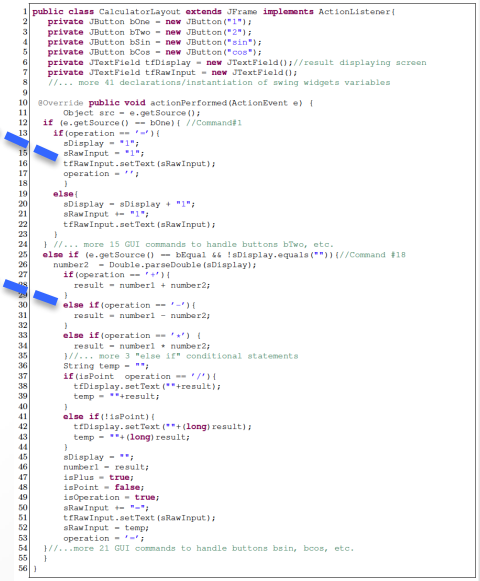
- Bug finder tools
  - ✗ FindBugs and PMD do not focus on detect problems that affect the GUI source code
  - ✗ Absence of GUI metrics/rules to detect GUI design smells

Contributions

GUI fault model



GUI source code analysis



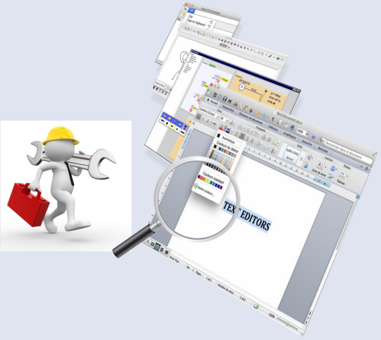
```
public class CalculatorLayout extends JFrame implements ActionListener{
   private JButton bOne = new JButton("1");
   private JButton bTwo = new JButton("2");
   private JButton bSin = new JButton("sin");
   private JButton bCos = new JButton("cos");
   private JTextField tfDisplay = new JTextField();//result displaying screen
   private JTextField tfRawInput = new JTextField();
   //... more 41 declarations/instantiation of swing widgets variables

 @Override public void actionPerformed(ActionEvent e) {
      Object src = e.getSource();
  if (e.getSource() == bOne){ //Command#1
    if(operation == '='){
      sDisplay = "1";
      sRawInput = "1";
      tfRawInput.setText(sRawInput);
      operation = '';
      }
    else{
      sDisplay = sDisplay + "1";
      sRawInput += "1";
      tfRawInput.setText(sRawInput);
    }
  } //... more 15 GUI commands to handle buttons bTwo, etc.
  else if (e.getSource() == bEqual && !sDisplay.equals("")){//Command #18
    number2  = Double.parseDouble(sDisplay);
      if(operation == '+'){
        result = number1 + number2;
      }
      else if(operation == '-'){
        result = number1 - number2;
      }
      else if(operation == '*') {
        result = number1 * number2;
      }//... more 3 "else if" conditional statements
      String temp = "";
      if(isPoint  operation == '/'){
        tfDisplay.setText(""+result);
        temp = ""+result;
      }
      else if(!isPoint){
        tfDisplay.setText(""+(long)result);
        temp = ""+(long)result;
      }
      sDisplay = "";
      number1 = result;
      isPlus = true;
      isPoint = false;
      isOperation = true;
      sRawInput += "=";
      tfRawInput.setText(sRawInput);
      sRawInput = temp;
      operation = '=';
  }//...more 21 GUI commands to handle buttons bsin, bcos, etc.
  }
}
```

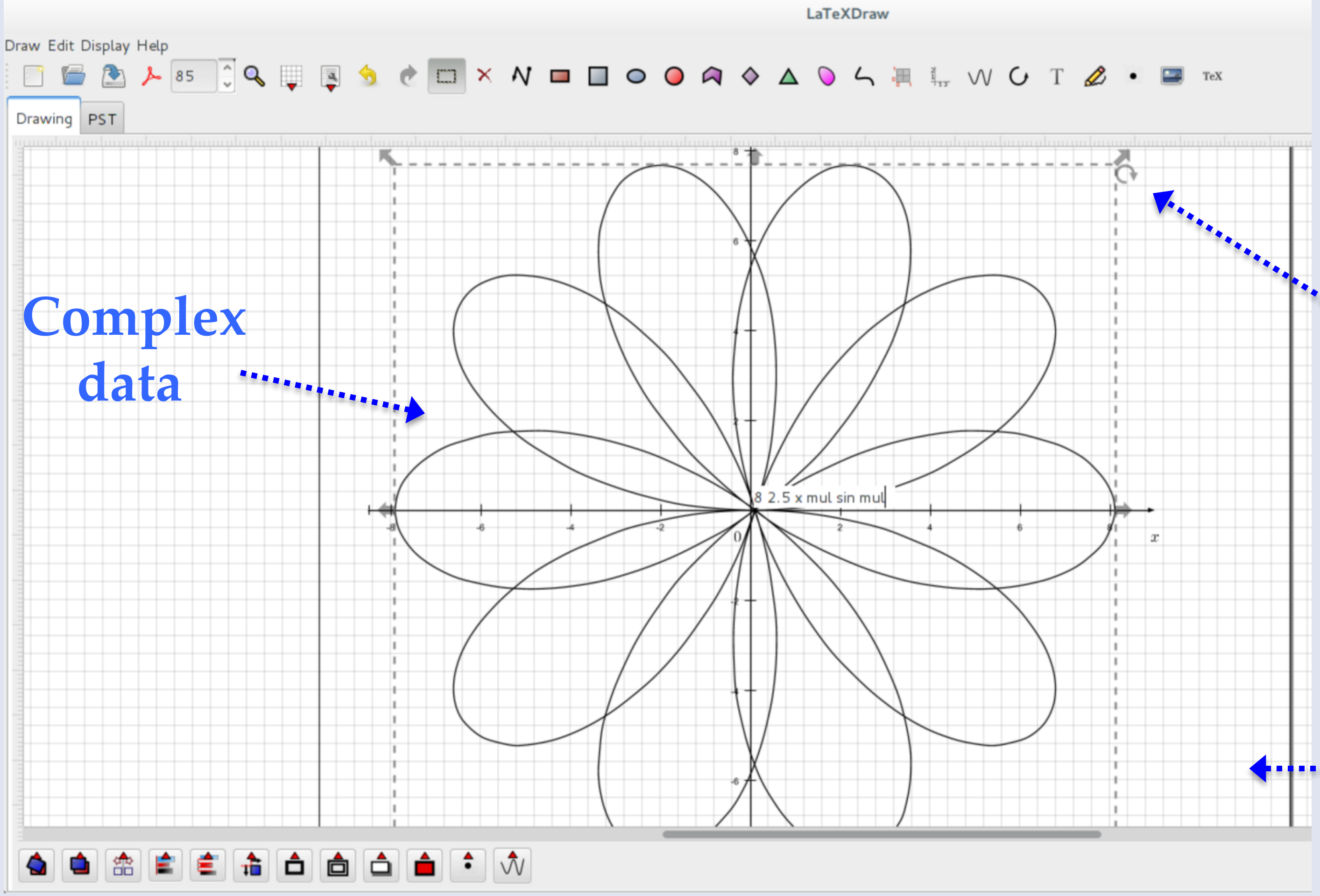
GUI Fault Model

GUI Design

- Recent developments of GUIs involve more advanced user interactions
- Current GUI testing tools focus on finding bugs in classical GUIs
- How the characteristics of recent developments of GUIs impact on GUI testing?

Post-WIMP GUIs





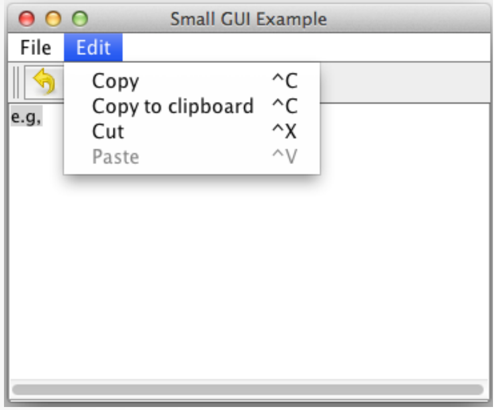
Complex data

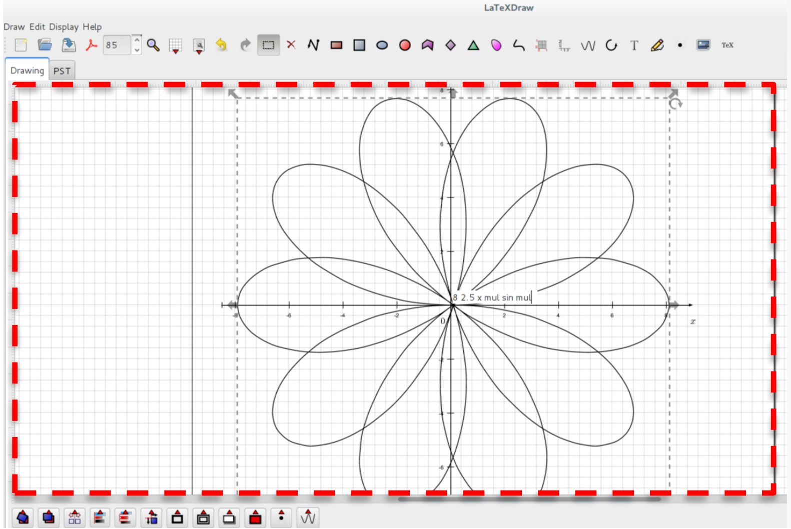
Complex interactions

Ad hoc widgets

- *Ad hoc* widgets such as drawing areas
- Complex interactions: multi-touch, etc.

WIMP vs. post-WIMP GUIs





Event-based GUIs

- Standard widgets
- Mono-event interactions

Interaction-based GUIs

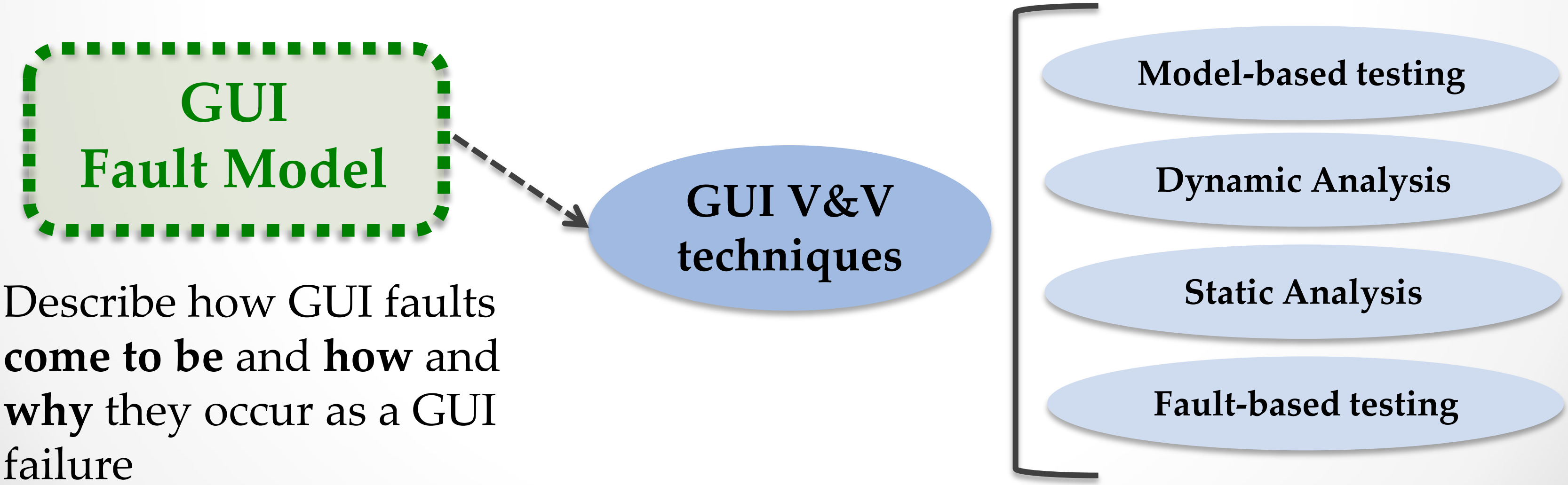
- *Ad hoc* widgets
- Multi-event interactions

✔ New problems of GUI faults

✗ Current GUI testing tools

GUI Fault Model

- Objectives
  - Baseline to evaluate the effectiveness of GUI testing techniques
  - Developing GUI testing techniques



GUI Fault Model

GUI V&V techniques

Model-based testing

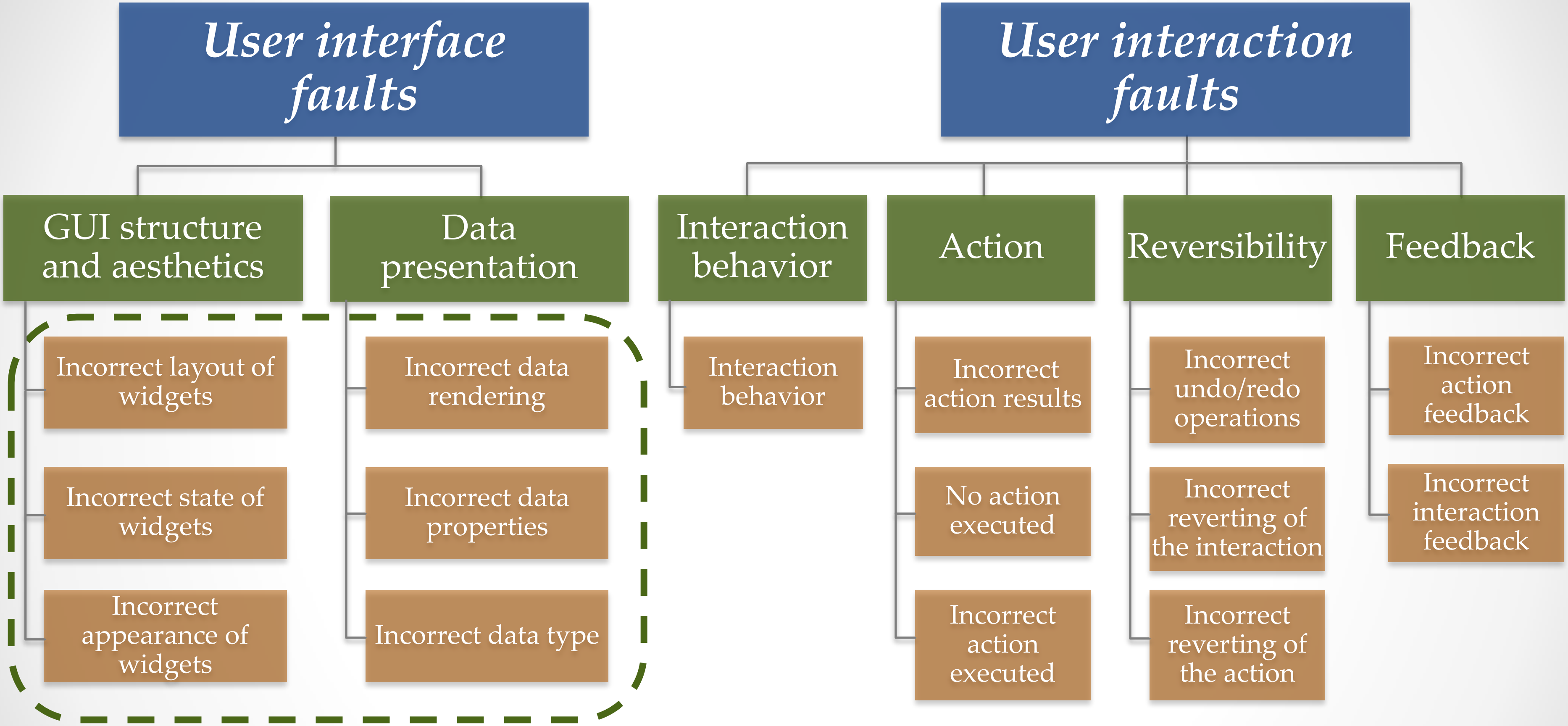
Dynamic Analysis

Static Analysis

Fault-based testing

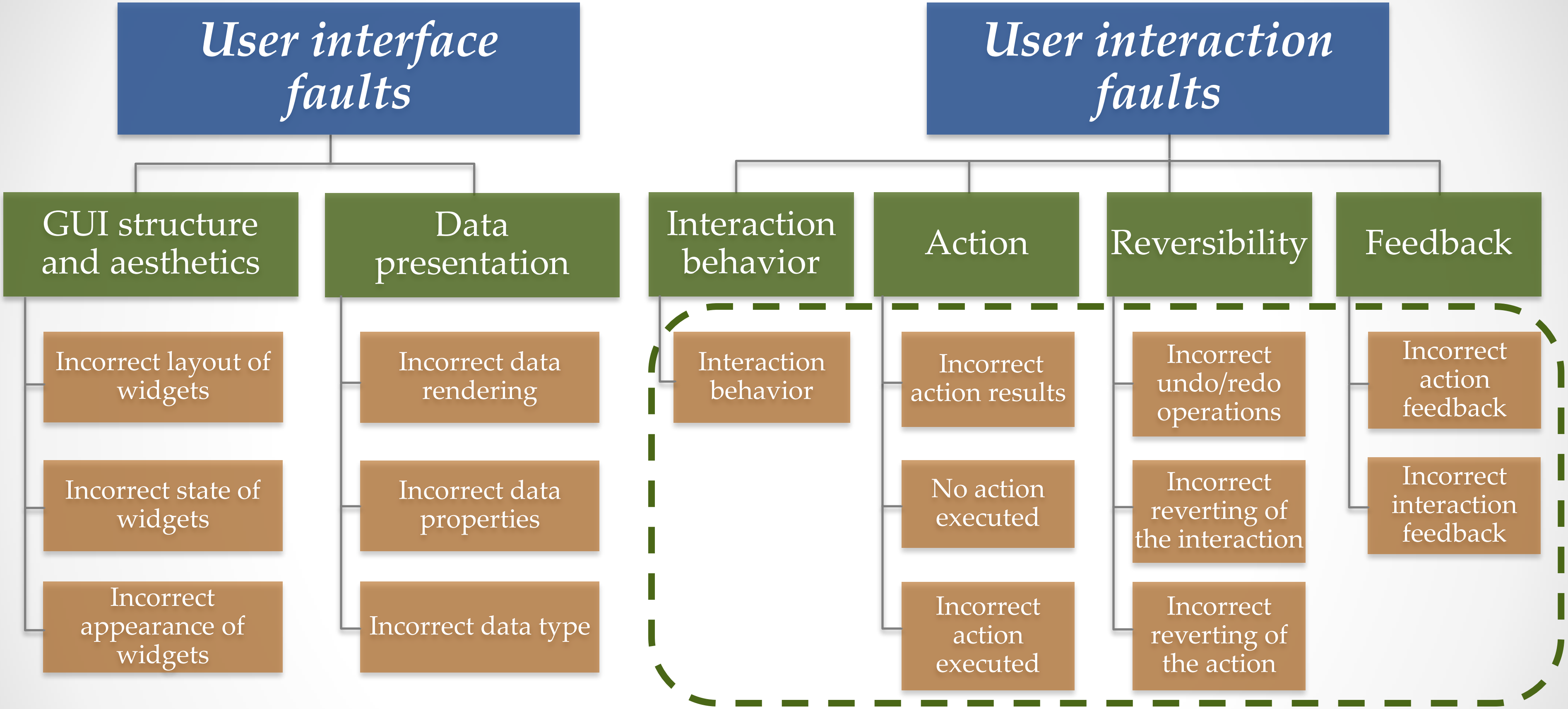
Describe how GUI faults **come to be** and **how** and **why** they occur as a GUI failure

GUI Fault Model


User interface faults
GUI structure and aesthetics
Incorrect layout of widgets
Incorrect state of widgets
Incorrect appearance of widgets
Data presentation
Incorrect data rendering
Incorrect data properties
Incorrect data type
User interaction faults
Interaction behavior
Interaction behavior
Action
Incorrect action results
No action executed
Incorrect action executed
Reversibility
Incorrect undo/redo operations
Incorrect reverting of the interaction
Incorrect reverting of the action
Feedback
Incorrect action feedback
Incorrect interaction feedback

- Structure and behavior of the graphical components

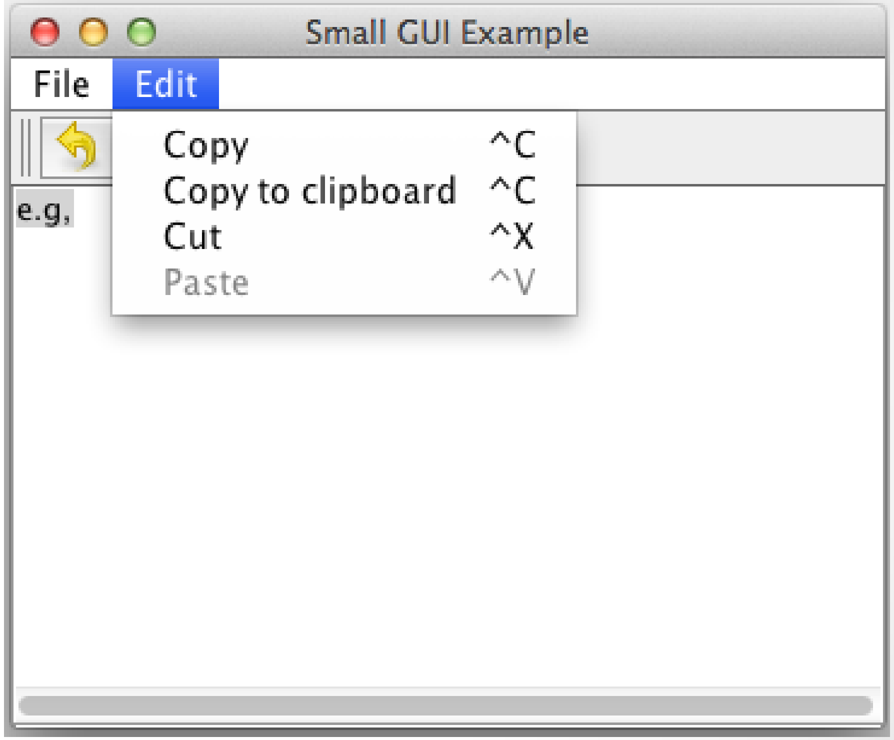
GUI Fault Model


User interface faults
GUI structure and aesthetics
Incorrect layout of widgets
Incorrect state of widgets
Incorrect appearance of widgets
Data presentation
Incorrect data rendering
Incorrect data properties
Incorrect data type
User interaction faults
Interaction behavior
Interaction behavior
Action
Incorrect action results
No action executed
Incorrect action executed
Reversibility
Incorrect undo/redo operations
Incorrect reverting of the interaction
Incorrect reverting of the action
Feedback
Incorrect action feedback
Incorrect interaction feedback

- The interaction process when a user interacts with a GUI

Concrete examples of user interface faults

```
…
//Set widget properties
widget.setVisible(true);
widget.setAlignment(5);
…
```

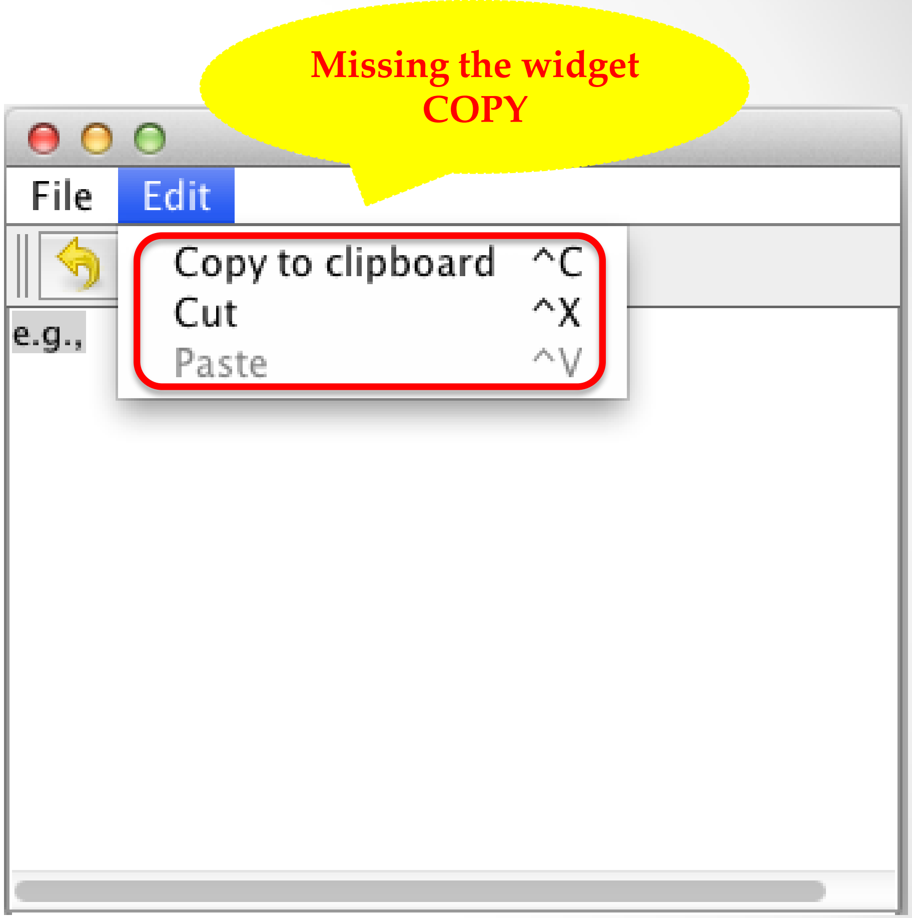


- *E.g.* of GUI Fault: incorrect vs. correct lines of GUI code
- *E.g.* of GUI failure: a widget is not visible

Concrete examples of user interface faults

Incorrect state of widgets fault

```
…
//Set widget properties
copyMenu.setVisible(false);
widget.setAlignment(5);
…
```



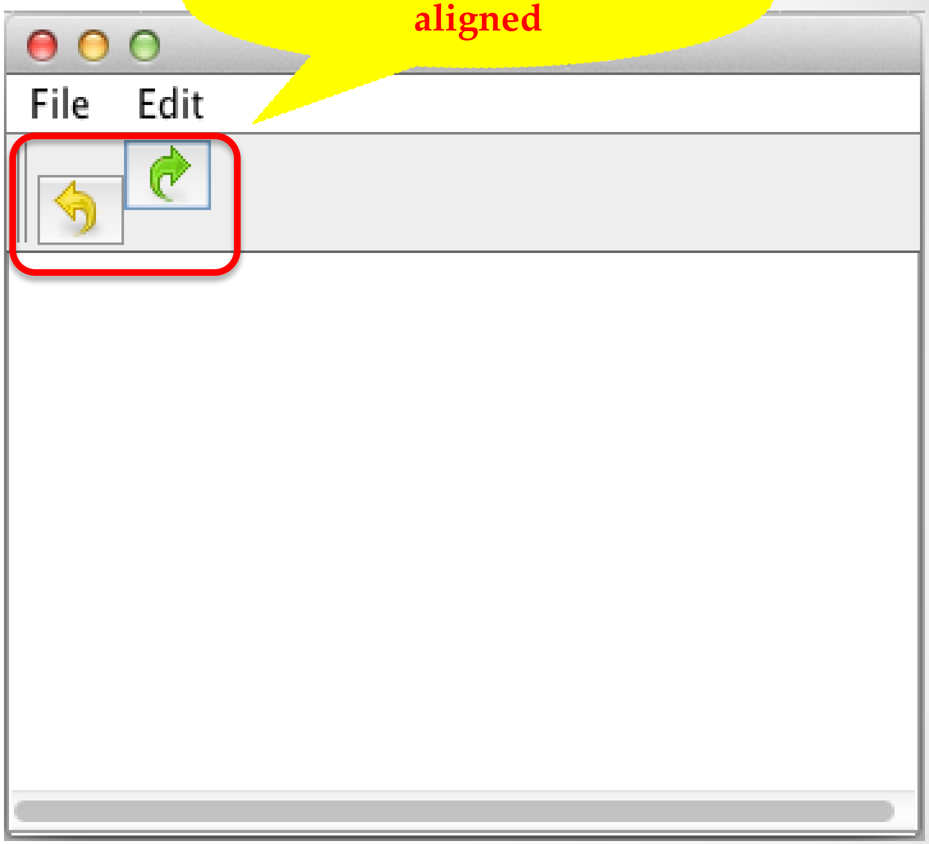
Missing the widget COPY

- *E.g.* of GUI Fault: incorrect vs. correct lines of GUI code
- *E.g.* of GUI failure: a widget is not visible

Concrete examples of user interface faults

```
…
//Set widget properties
copyMenu.setVisible(false);
btnRedo.setAlignmentY(10);
…
```

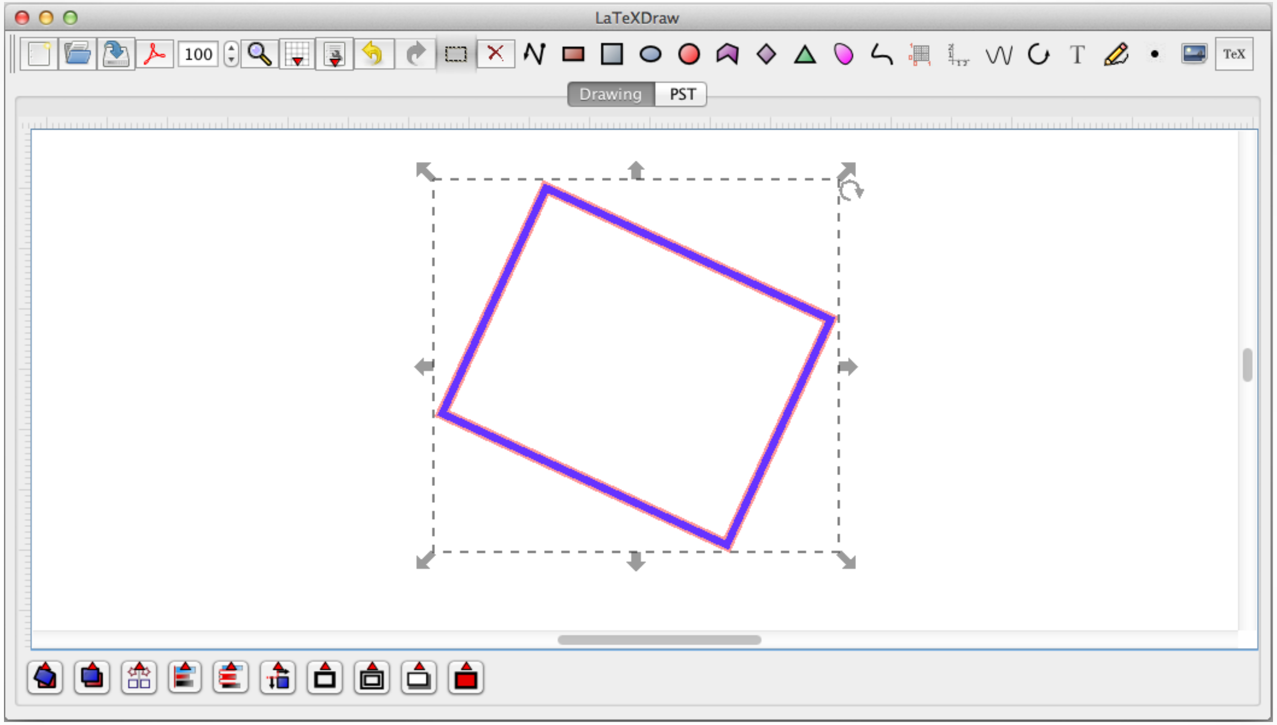
Incorrect layout of widgets fault



Widgets are not aligned

- *E.g.* of GUI Fault: incorrect vs. correct lines of GUI code
- *E.g.* of GUI failure: a widget is not visible

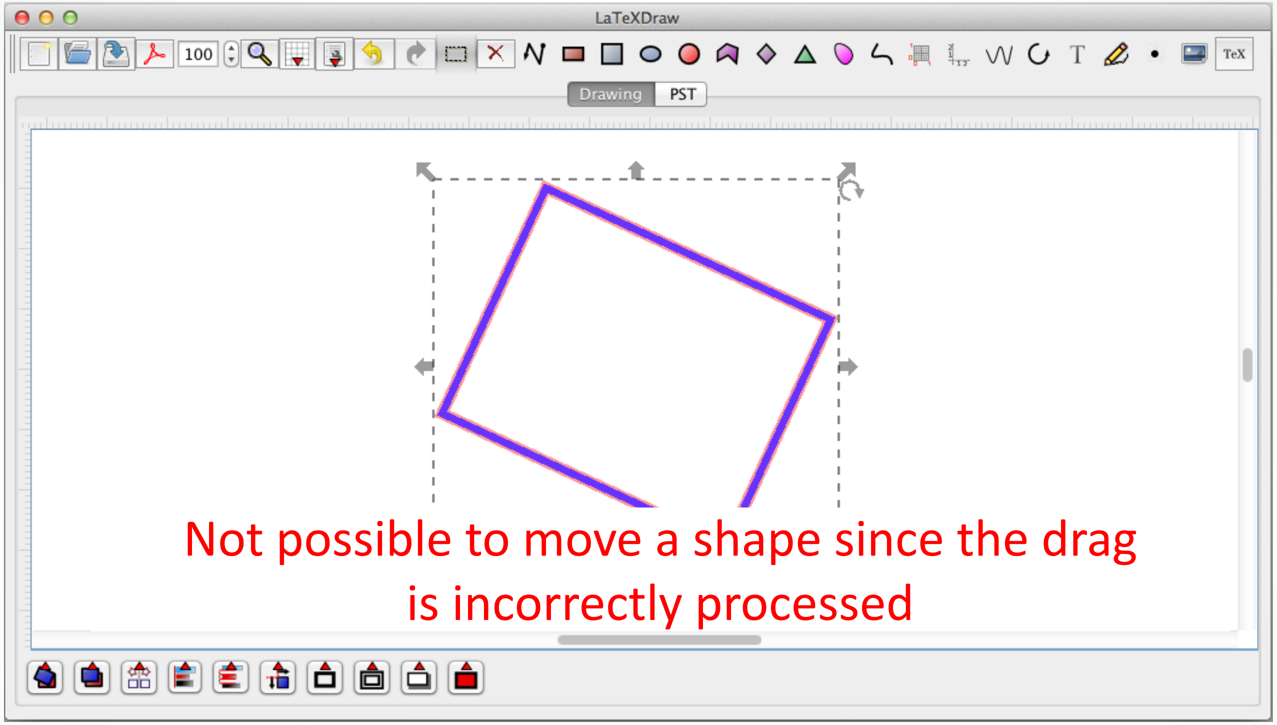
Concrete examples of user interaction faults



Interaction behavior

1. figures.firstElement().onDragged(formerPt, newPt);

Concrete examples of user interaction faults



Not possible to move a shape since the drag is incorrectly processed

Interaction behavior

```
figures.firstElement().onDragged(newPt, formerPt);
```

Fault model assessment

RQ1: Is the **GUI fault model** relevant against **real GUI failures**?

RQ2: Are **GUI testing tools** able to detect the **failures classified** in our fault model?

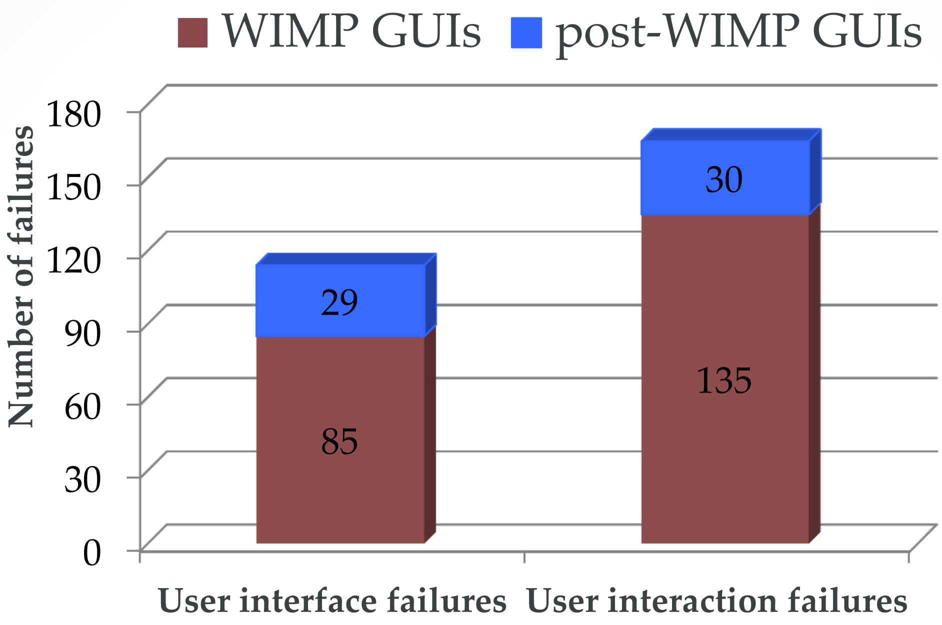
Experiment (RQ1): relevance

- **GUI bug reports of 5 open-source software systems**
  - Sweet Home 3D
  - File-roller
  - JabRef
  - Inkscape
  - Firefox Android

Several kinds of widgets, interactions and platforms

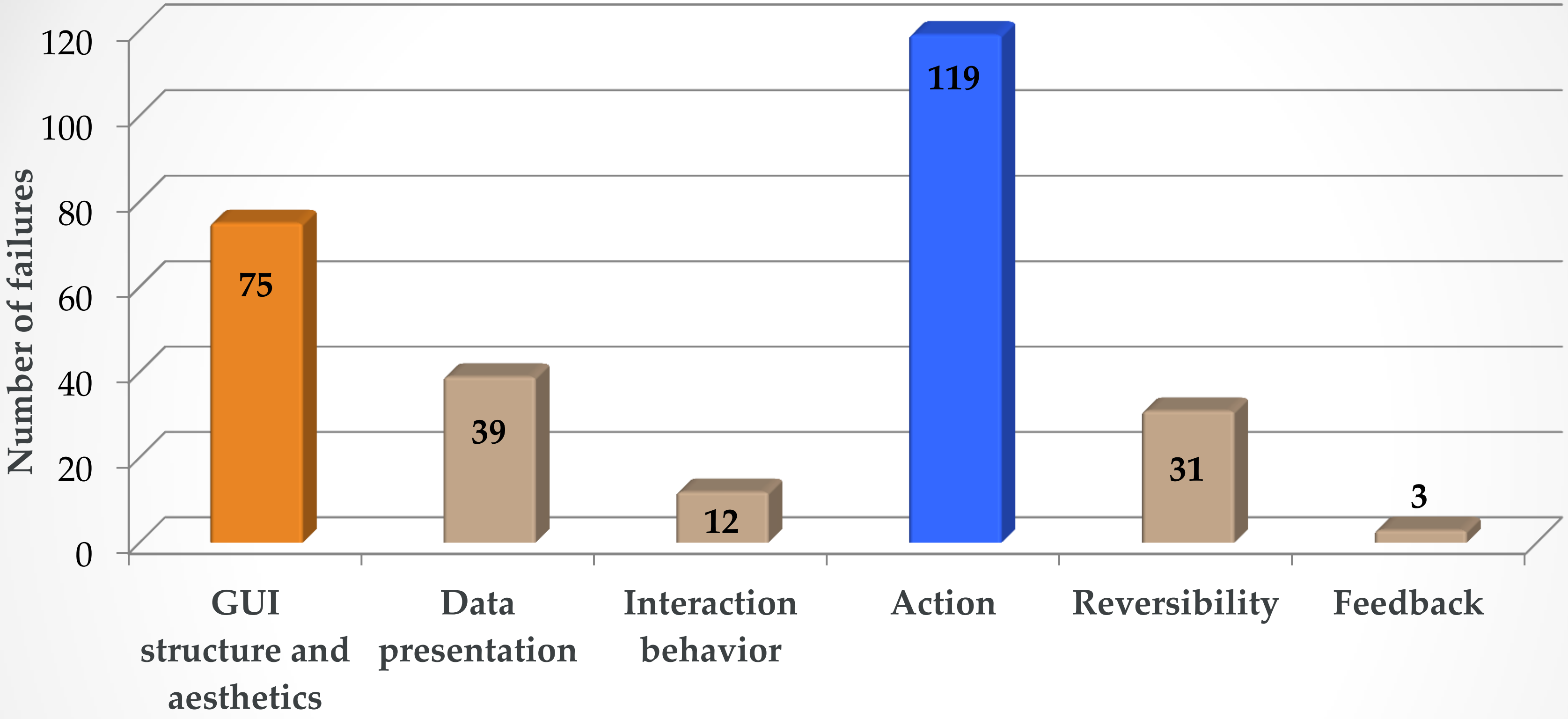
- **Manual analysis of the real GUI bug reports**
  - Source forge, bugzilla, etc.
  - Root cause: description, patches, comments, or stack traces

Experiment Results



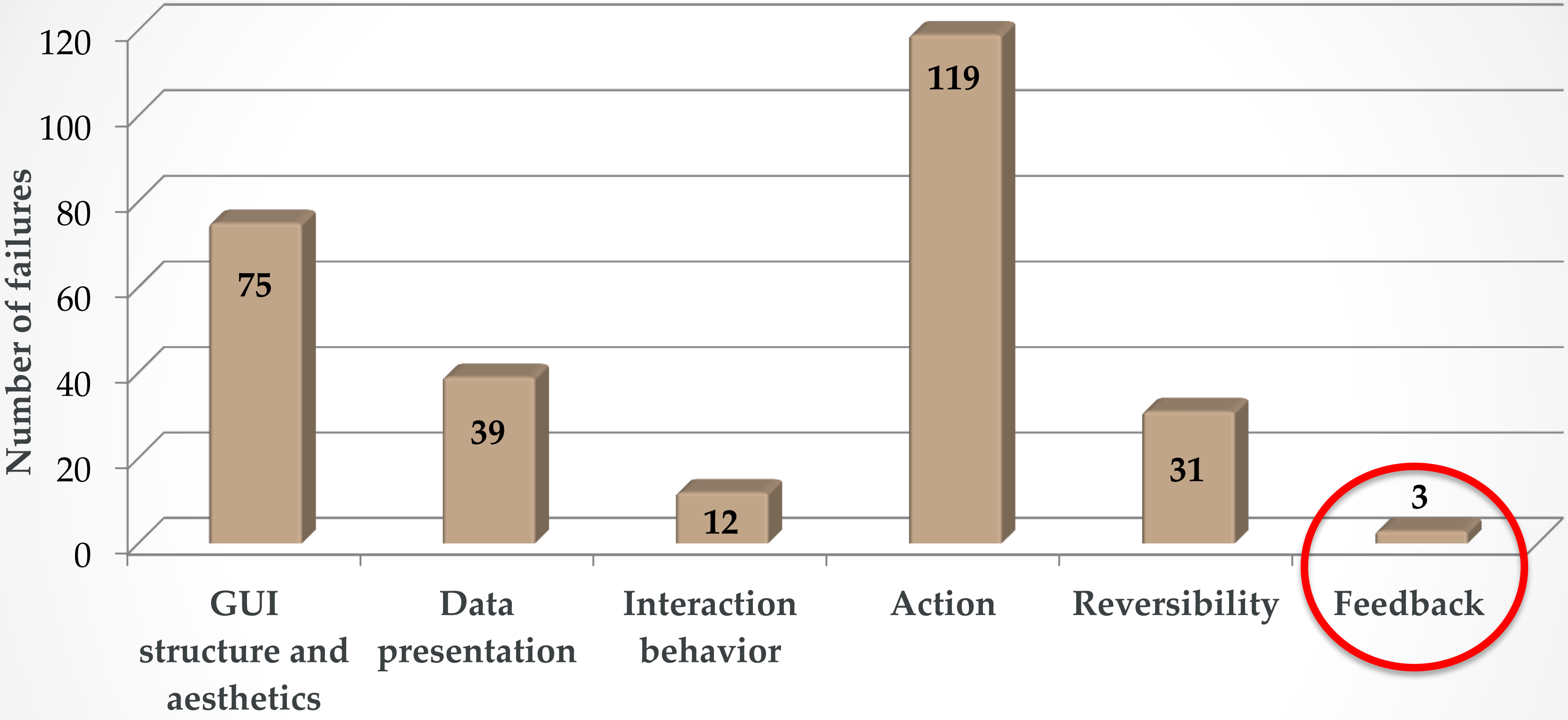
- All GUI failures (279) were classified into the fault model
- User interface (41%) and user interaction (59%)
- Post-WIMP
  - ✓ 25% (user interface)
  - ✓ 18% (user interaction)

Experiment Results



- 43% into **Action** and
- 27% into **GUI structure and aesthetics**

Experiment Results



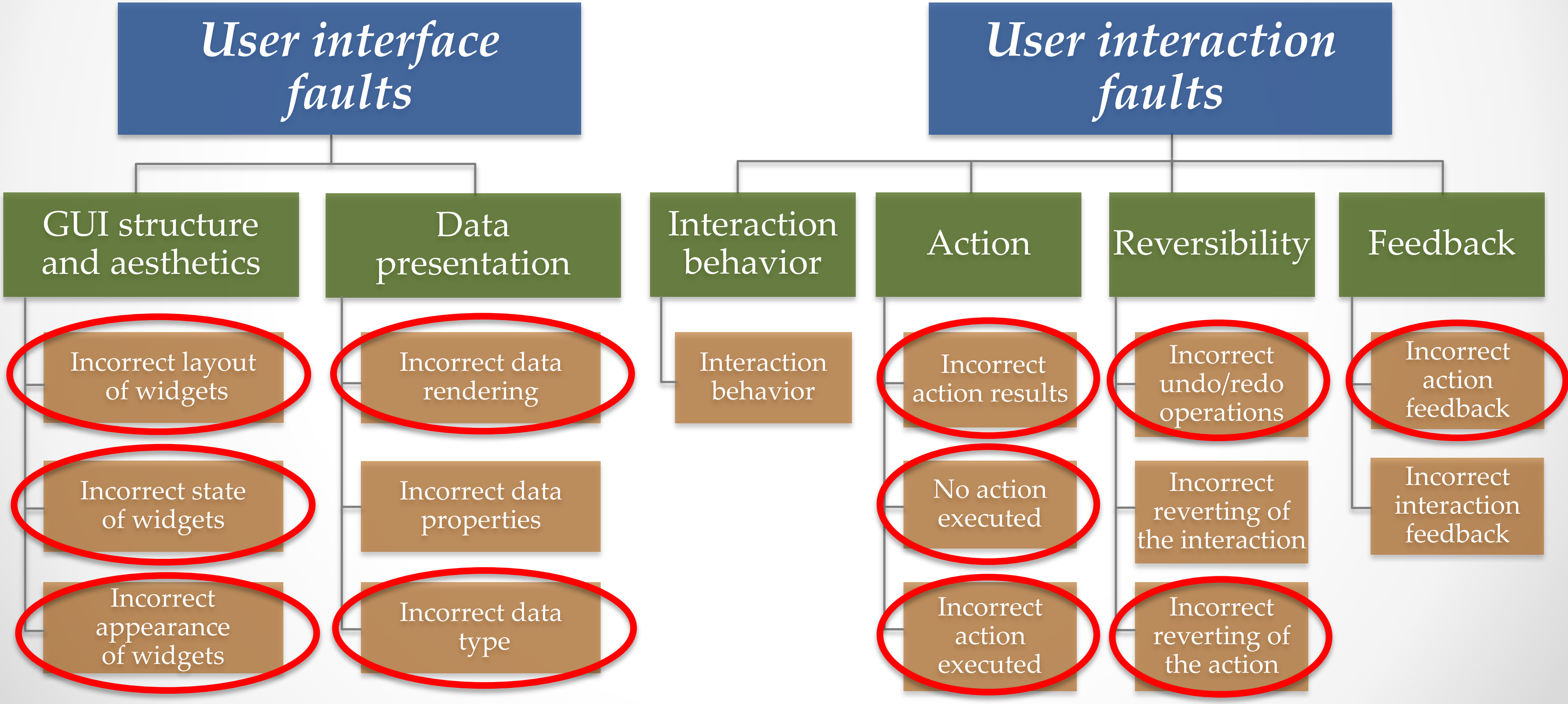
- 1% of GUI failures classified into **Feedback**
- Several "failures" were considered by developers as improvements

RQ1: Is the GUI fault model relevant against real GUI failures?

- ✓ All GUI related faults of 5 large scale case studies can be classified
  - ✓ 279 GUI bug reports
- ✓ All the 6 fault categories are covered
- ✓ Faults concern WIMP and post-WIMP GUIs
  - ✓ *Ad hoc* widgets (59 faults)

Experiment (RQ2): GUI testing tools

- **JabRef**: selected 11 out of 15 GUI faults



User interface faults

- GUI structure and aesthetics
  - [✓ Incorrect layout of widgets]
  - [✓ Incorrect state of widgets]
  - [✓ Incorrect appearance of widgets]
- Data presentation
  - [✓ Incorrect data rendering]
  - Incorrect data properties
  - [✓ Incorrect data type]

User interaction faults

- Interaction behavior
  - Interaction behavior
- Action
  - [✓ Incorrect action results]
  - [✓ No action executed]
  - [✓ Incorrect action executed]
- Reversibility
  - [✓ Incorrect undo/redo operations]
  - Incorrect reverting of the interaction
  - [✓ Incorrect reverting of the action]
- Feedback
  - [✓ Incorrect action feedback]
  - Incorrect interaction feedback

Experiment (RQ2): GUI testing tools

- GUITAR[2]
  - Most popular academic tool in GUI testing
  - **Automated** test cases generation
  - Event-flow graph is built by reverse engineering
- Jubula[3]
  - **Partially** manual generation of test cases
  - Reuse pre-defined libraries to create manually test cases

[2] B. N. Nguyen, B. Robbins, I. Banerjee, A. Memon: GUITAR: an innovative tool for automated testing of GUI-driven software. Autom. Softw. Eng. 21(1): 65-105 (2014)

[3] http://www.eclipse.org/jubula

Experiment Results

- **GUITAR** detected
  - **3** out of 11 GUI faults
- **Jubula** detected
  - **9** out of 11 GUI faults

- GUI failures detected

  ✔ Properties of standard widgets

  ✔ Crashes

  ➔ Oracle for standard widgets

Experiment Results

- **GUITAR**
  - Missed 8 out of 11 GUI faults reported in JabRef

- GUITAR builds the event-flow graph by
  - Extracting the sequence of events behind standard widgets
  - Collecting the information in the properties of standard widgets as event logs

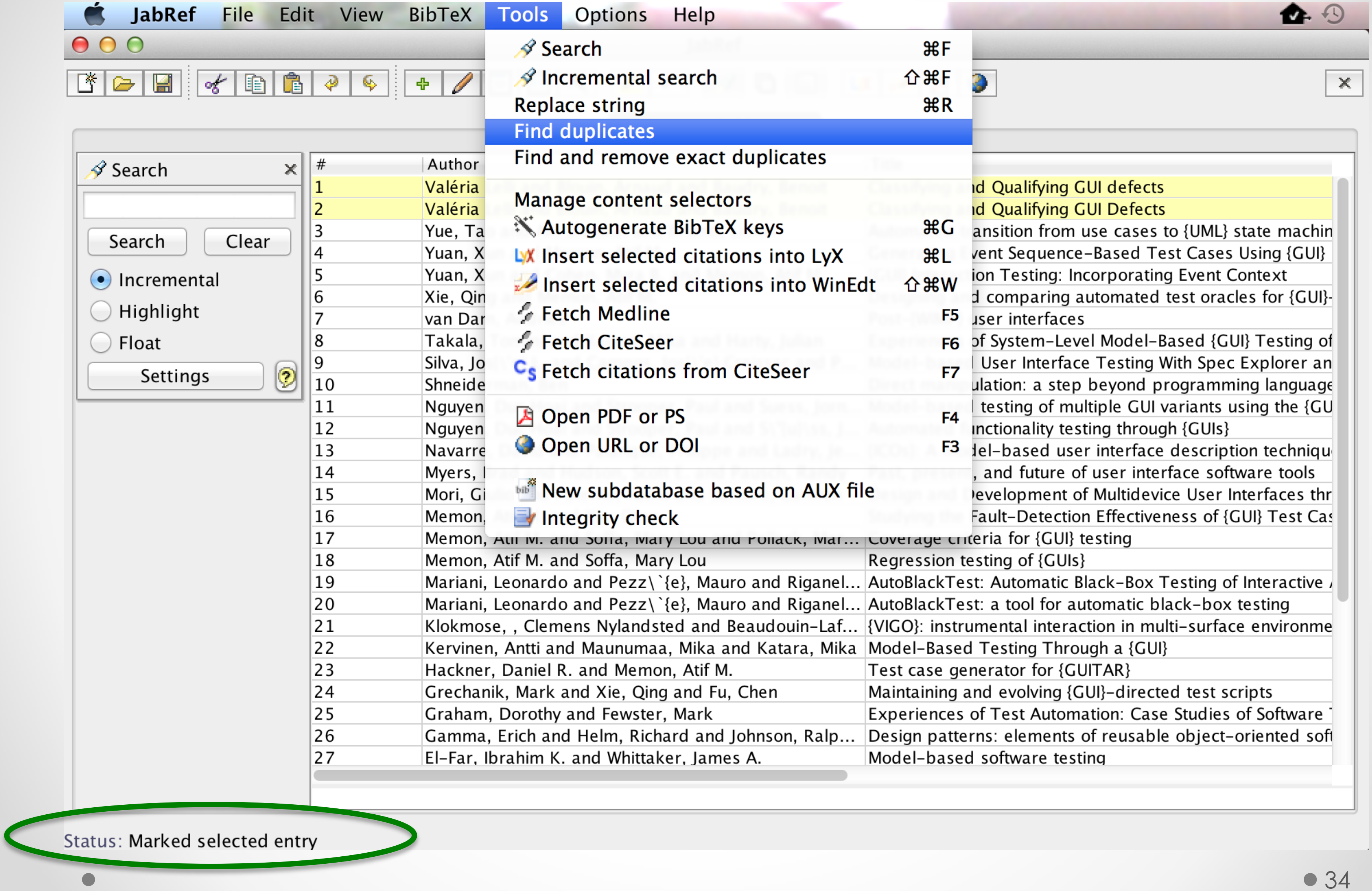
✗ User interface failures into properties of standard widgets

✗ Complex data in *ad hoc widgets*

✗ Events are both widgets and their underlying interactions

✗ *Ad hoc widgets* and their multi-event interactions

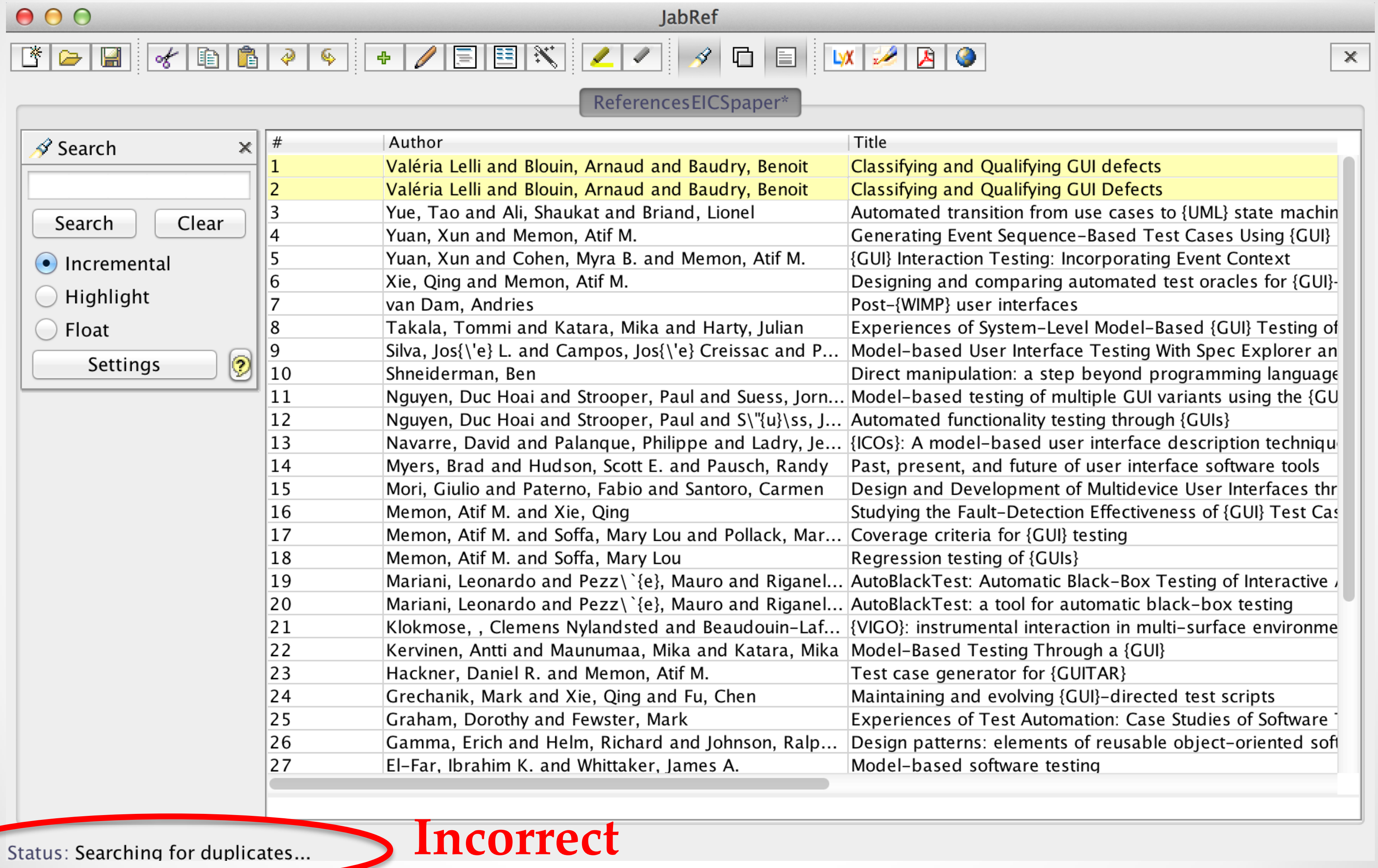
Real example of a GUI failure



#	Author	
1	Valéria	d Qualifying GUI defects
2	Valéria	d Qualifying GUI Defects
3	Yue, Ta	ansition from use cases to {UML} state machin
4	Yuan, X	vent Sequence-Based Test Cases Using {GUI}
5	Yuan, X	ion Testing: Incorporating Event Context
6	Xie, Qin	d comparing automated test oracles for {GUI}-
7	van Dar	user interfaces
8	Takala,	of System-Level Model-Based {GUI} Testing of
9	Silva, Jo	l User Interface Testing With Spec Explorer an
10	Shneide	ulation: a step beyond programming language
11	Nguyen	l testing of multiple GUI variants using the {GU
12	Nguyen	nctionality testing through {GUIs}
13	Navarre	del-based user interface description techniqu
14	Myers, I	, and future of user interface software tools
15	Mori, Gi	evelopment of Multidevice User Interfaces thr
16	Memon,	Fault-Detection Effectiveness of {GUI} Test Cas
17	Memon, Atif M. and Soffa, Mary Lou and Pollack, Mar...	Coverage criteria for {GUI} testing
18	Memon, Atif M. and Soffa, Mary Lou	Regression testing of {GUIs}
19	Mariani, Leonardo and Pezz\`{e}, Mauro and Riganel...	AutoBlackTest: Automatic Black-Box Testing of Interactive
20	Mariani, Leonardo and Pezz\`{e}, Mauro and Riganel...	AutoBlackTest: a tool for automatic black-box testing
21	Klokmose, , Clemens Nylandsted and Beaudouin-Laf...	{VIGO}: instrumental interaction in multi-surface environme
22	Kervinen, Antti and Maunumaa, Mika and Katara, Mika	Model-Based Testing Through a {GUI}
23	Hackner, Daniel R. and Memon, Atif M.	Test case generator for {GUITAR}
24	Grechanik, Mark and Xie, Qing and Fu, Chen	Maintaining and evolving {GUI}-directed test scripts
25	Graham, Dorothy and Fewster, Mark	Experiences of Test Automation: Case Studies of Software
26	Gamma, Erich and Helm, Richard and Johnson, Ralp...	Design patterns: elements of reusable object-oriented soft
27	El-Far, Ibrahim K. and Whittaker, James A.	Model-based software testing

Status: Marked selected entry

Real example of a GUI failure



#	Author	Title
1	Valéria Lelli and Blouin, Arnaud and Baudry, Benoit	Classifying and Qualifying GUI defects
2	Valéria Lelli and Blouin, Arnaud and Baudry, Benoit	Classifying and Qualifying GUI Defects
3	Yue, Tao and Ali, Shaukat and Briand, Lionel	Automated transition from use cases to {UML} state machin
4	Yuan, Xun and Memon, Atif M.	Generating Event Sequence-Based Test Cases Using {GUI}
5	Yuan, Xun and Cohen, Myra B. and Memon, Atif M.	{GUI} Interaction Testing: Incorporating Event Context
6	Xie, Qing and Memon, Atif M.	Designing and comparing automated test oracles for {GUI}-
7	van Dam, Andries	Post-{WIMP} user interfaces
8	Takala, Tommi and Katara, Mika and Harty, Julian	Experiences of System-Level Model-Based {GUI} Testing of
9	Silva, Jos{\'e} L. and Campos, Jos{\'e} Creissac and P...	Model-based User Interface Testing With Spec Explorer an
10	Shneiderman, Ben	Direct manipulation: a step beyond programming language
11	Nguyen, Duc Hoai and Strooper, Paul and Suess, Jorn...	Model-based testing of multiple GUI variants using the {GU
12	Nguyen, Duc Hoai and Strooper, Paul and S\"{u}\ss, J...	Automated functionality testing through {GUIs}
13	Navarre, David and Palanque, Philippe and Ladry, Je...	{ICOs}: A model-based user interface description techniqu
14	Myers, Brad and Hudson, Scott E. and Pausch, Randy	Past, present, and future of user interface software tools
15	Mori, Giulio and Paterno, Fabio and Santoro, Carmen	Design and Development of Multidevice User Interfaces thr
16	Memon, Atif M. and Xie, Qing	Studying the Fault-Detection Effectiveness of {GUI} Test Ca
17	Memon, Atif M. and Soffa, Mary Lou and Pollack, Mar...	Coverage criteria for {GUI} testing
18	Memon, Atif M. and Soffa, Mary Lou	Regression testing of {GUIs}
19	Mariani, Leonardo and Pezz\`{e}, Mauro and Riganel...	AutoBlackTest: Automatic Black-Box Testing of Interactive
20	Mariani, Leonardo and Pezz\`{e}, Mauro and Riganel...	AutoBlackTest: a tool for automatic black-box testing
21	Klokmose, , Clemens Nylandsted and Beaudouin-Laf...	{VIGO}: instrumental interaction in multi-surface environme
22	Kervinen, Antti and Maunumaa, Mika and Katara, Mika	Model-Based Testing Through a {GUI}
23	Hackner, Daniel R. and Memon, Atif M.	Test case generator for {GUITAR}
24	Grechanik, Mark and Xie, Qing and Fu, Chen	Maintaining and evolving {GUI}-directed test scripts
25	Graham, Dorothy and Fewster, Mark	Experiences of Test Automation: Case Studies of Software
26	Gamma, Erich and Helm, Richard and Johnson, Ralp...	Design patterns: elements of reusable object-oriented sof
27	El-Far, Ibrahim K. and Whittaker, James A.	Model-based software testing

Status: Searching for duplicates...

Incorrect Feedback

RQ2: Are GUI testing tools able to detect the classified failures?

✓ Most of GUI faults concern standard widgets

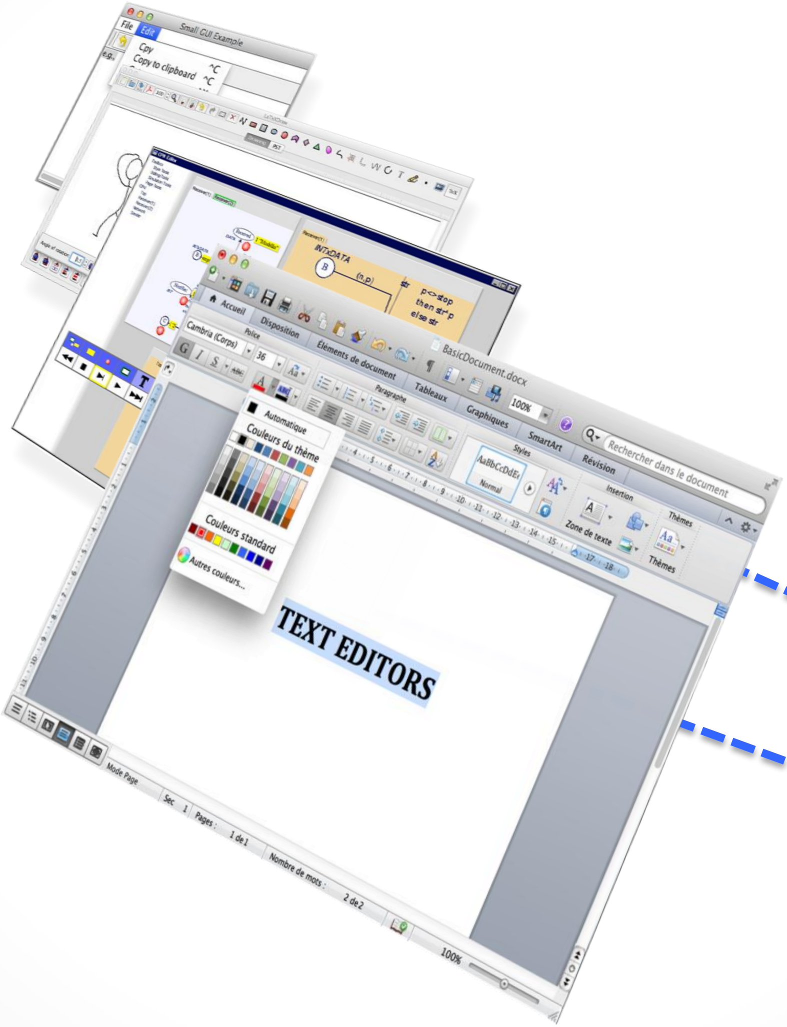
✗ Faults that concern the interactive features such as feedback and reversibility

Conclusion

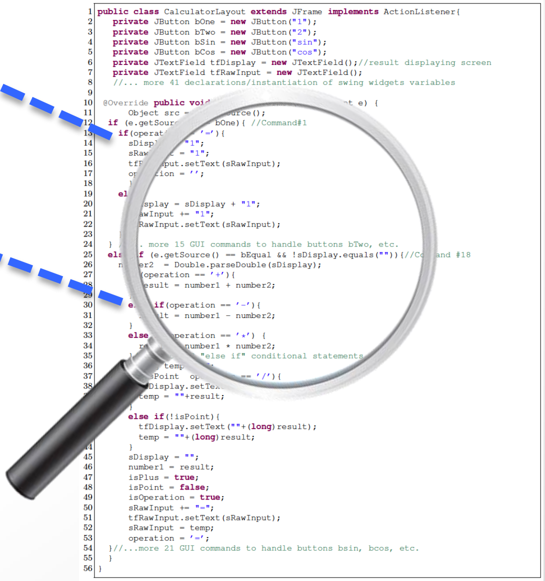
- **An empirical study of real GUI failures**
  - 279 GUI-related bug reports
- **Evaluation of two GUI testing tools against**
  - Real GUI failures into standard and *ad hoc* widgets
  - 65 GUI mutants derived from our fault model
    - 43 GUI mutants were not killed
- **A precise analysis of standard GUI testing frameworks**
  - Why GUI failures that stem from GUI faults described in our fault model were not detected?

Contributions

GUI fault model

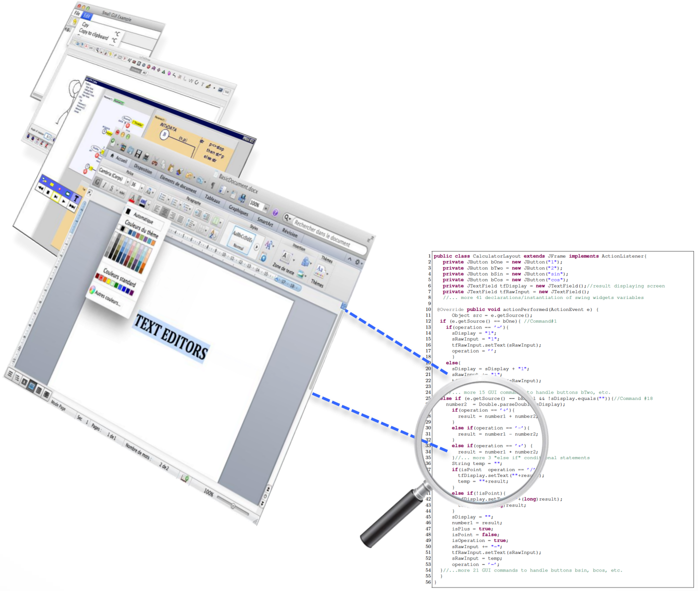


GUI source code analysis



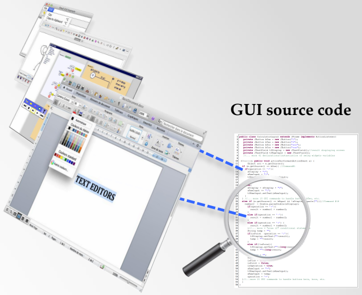
```
public class CalculatorLayout extends JFrame implements ActionListener{
   private JButton bOne = new JButton("1");
   private JButton bTwo = new JButton("2");
   private JButton bSin = new JButton("sin");
   private JButton bCos = new JButton("cos");
   private JTextField tfDisplay = new JTextField();//result displaying screen
   private JTextField tfRawInput = new JTextField();
   //... more 41 declarations/instantiation of swing widgets variables

 @Override public voi                          nt e) {
      Object src =       urce();
  if (e.getSour       bOne){ //Command#1
    if(operat       '=') {
      sDisp       "1";
      sRaw     t = "1";
      tfR     nput.setText(sRawInput);
      op     tion = '';
      }
    el
        isplay = sDisplay + "1";
        awInput += "1";
        RawInput.setText(sRawInput);
     }
  }     . more 15 GUI commands to handle buttons bTwo, etc.
  els  f (e.getSource() == bEqual && !sDisplay.equals("")){//Co   and #18
    n   er2  = Double.parseDouble(sDisplay);
        (operation == '+'){
        esult = number1 + number2;
      }
      e   if(operation == '-'){
          lt = number1 - number2;
      }
      else    peration == '*') {
        r       number1 * number2;
      }           "else if" conditional statements
          temp     ;
          sPoint  op        == '/'){
         Display.setTex
        temp = ""+result;
      }
      else if(!isPoint){
        tfDisplay.setText(""+(long)result);
        temp = ""+(long)result;
      }
      sDisplay = "";
      number1 = result;
      isPlus = true;
      isPoint = false;
      isOperation = true;
      sRawInput += "=";
      tfRawInput.setText(sRawInput);
      sRawInput = temp;
      operation = '=';
  }//...more 21 GUI commands to handle buttons bsin, bcos, etc.
  }
}
```

GUI Design Smells: The case of *Blob Listener*



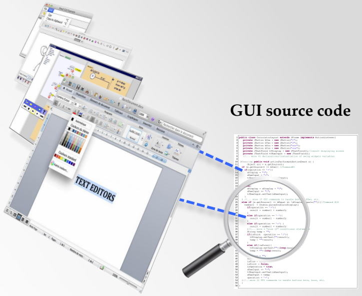
```
public class CalculatorLayout extends JFrame implements ActionListener{
   private JButton bOne = new JButton("1");
   private JButton bTwo = new JButton("2");
   private JButton bSin = new JButton("sin");
   private JButton bCos = new JButton("cos");
   private JTextField tfDisplay = new JTextField();//result displaying screen
   private JTextField tfRawInput = new JTextField();
   //... more 41 declarations/instantiation of swing widgets variables

 @Override public void actionPerformed(ActionEvent e) {
      Object src = e.getSource();
  if (e.getSource() == bOne){ //Command#1
    if(operation == '='){
      sDisplay = "1";
      sRawInput = "1";
      tfRawInput.setText(sRawInput);
      operation = '';
      }
    else{
      sDisplay = sDisplay + "1";
      sRawInput += "1";
      tfRawInput.setText(sRawInput);

   //... more 15 GUI commands to handle buttons bTwo, etc.
  else if (e.getSource() == bEqual && !sDisplay.equals("")){//Command #18
    number2  = Double.parseDouble(sDisplay);
      if(operation == '+'){
        result = number1 + number2;
      }
      else if(operation == '-'){
        result = number1 - number2;
      }
      else if(operation == '*') {
        result = number1 * number2;
      }//... more 3 "else if" conditional statements
      String temp = "";
      if(isPoint  operation == '/'
        tfDisplay.setText(""+result);
        temp = ""+result;
      }
      else if(!isPoint){
        tfDisplay.setText(""+(long)result);
        temp = ""+(long)result;
      }
      sDisplay = "";
      number1 = result;
      isPlus = true;
      isPoint = false;
      isOperation = true;
      sRawInput += "=";
      tfRawInput.setText(sRawInput);
      sRawInput = temp;
      operation = '=';
  }//...more 21 GUI commands to handle buttons bsin, bcos, etc.
  }
}
```

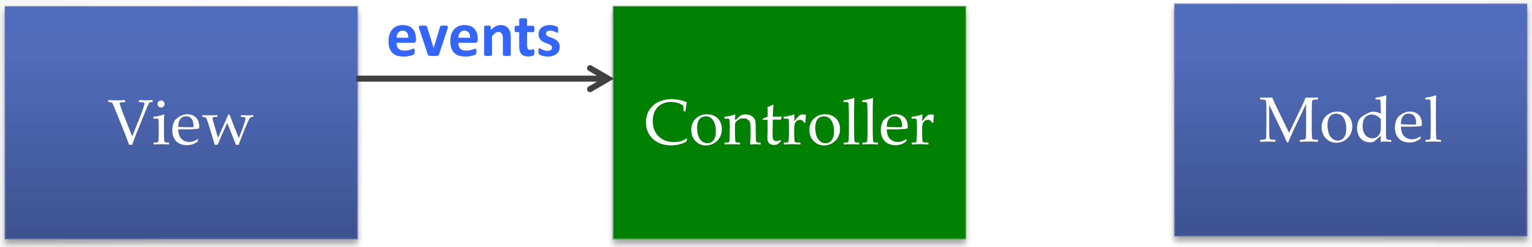
GUI code assurance quality



- Objectives
  - Identify and characterize design smells that degrade the GUI code quality
  - Develop a novel static analysis to detect GUI design smells

GUI implementations

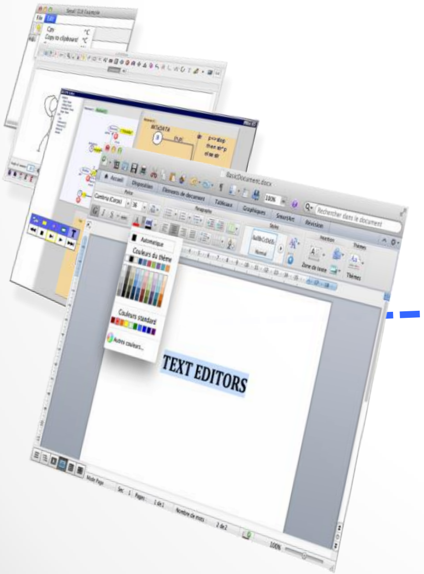


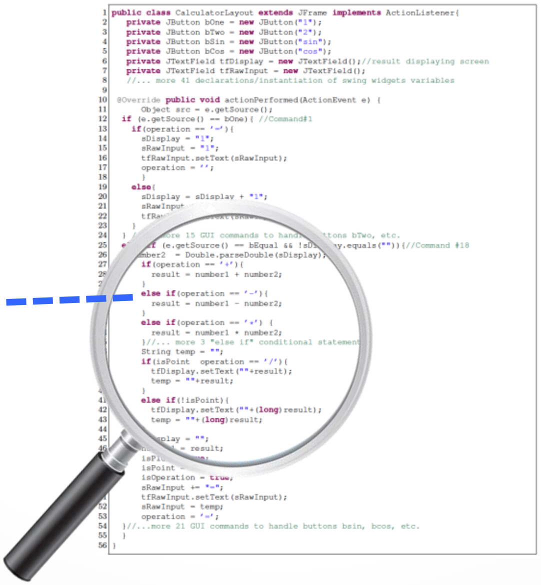


Represent of the GUI elements

Receive the events

Data model of an interactive system

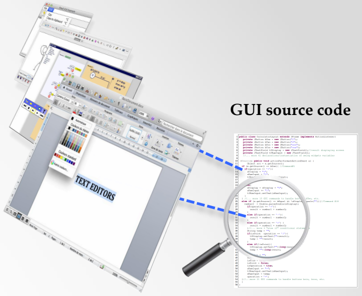




```
public class CalculatorLayout extends JFrame implements ActionListener{
   private JButton bOne = new JButton("1");
   private JButton bTwo = new JButton("2");
   private JButton bSin = new JButton("sin");
   private JButton bCos = new JButton("cos");
   private JTextField tfDisplay = new JTextField();//result displaying screen
   private JTextField tfRawInput = new JTextField();
   //... more 41 declarations/instantiation of swing widgets variables

 @Override public void actionPerformed(ActionEvent e) {
      Object src = e.getSource();
  if (e.getSource() == bOne){ //Command#1
    if(operation == '=')){
      sDisplay = "1";
      sRawInput = "1";
      tfRawInput.setText(sRawInput);
      operation = '';
      }
    else{
      sDisplay = sDisplay + "1";
      sRawInput
      tfRa
    }
  } /  more 15 GUI commands to hand  ttons bTwo, etc.
  e  if (e.getSource() == bEqual && !sDi  y.equals("")){//Command #18
   mber2  = Double.parseDouble(sDisplay);
      if(operation == '+'){
        result = number1 + number2;
      }
      else if(operation == '-'){
        result = number1 - number2;
      }
      else if(operation == '*') {
        result = number1 * number2;
      }//... more 3 "else if" conditional statement
      String temp = "";
      if(isPoint  operation == '/'){
        tfDisplay.setText(""+result);
        temp = ""+result;
      }
      else if(!isPoint){
        tfDisplay.setText(""+(long)result);
        temp = ""+(long)result;

       splay = "";
      n    1 = result;
      isPl
      isPoint
      isOperation = true;
      sRawInput += "=";
      tfRawInput.setText(sRawInput);
      sRawInput = temp;
      operation = '=';
  }//...more 21 GUI commands to handle buttons bsin, bcos, etc.
  }
}
```

Listener methods

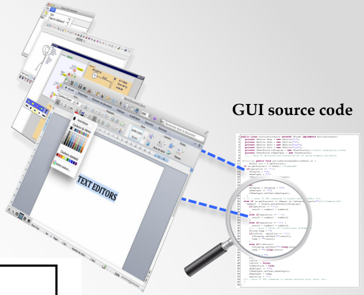
GUI implementations

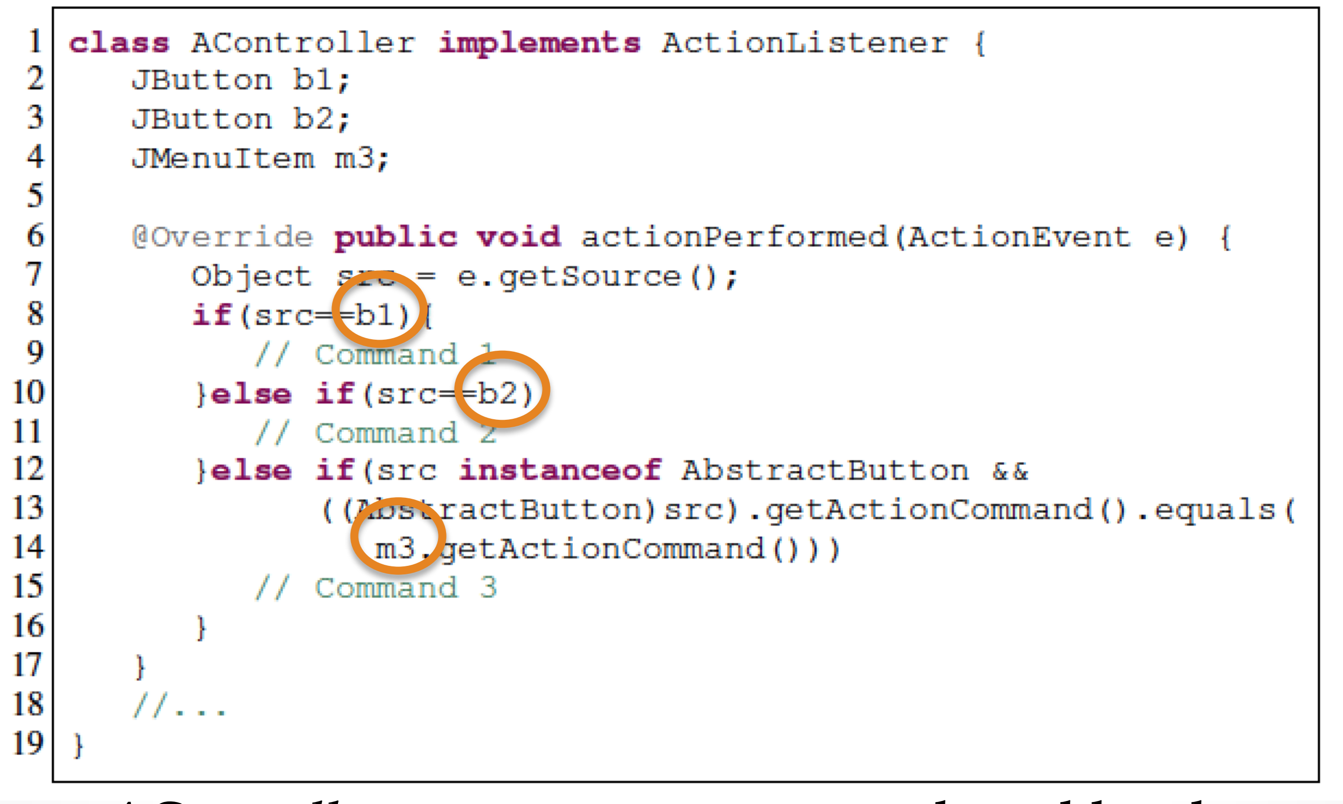


- Specific architectural design patterns
  - Organize the GUI components
  - Describe how the components interact with each other

- **Mode-View***
  - Model-View Controller (MVC)
  - Mode-View Presenter (MVP)
  - Model-View-ViewModel (MVVM), *etc.*

Java GUI controller



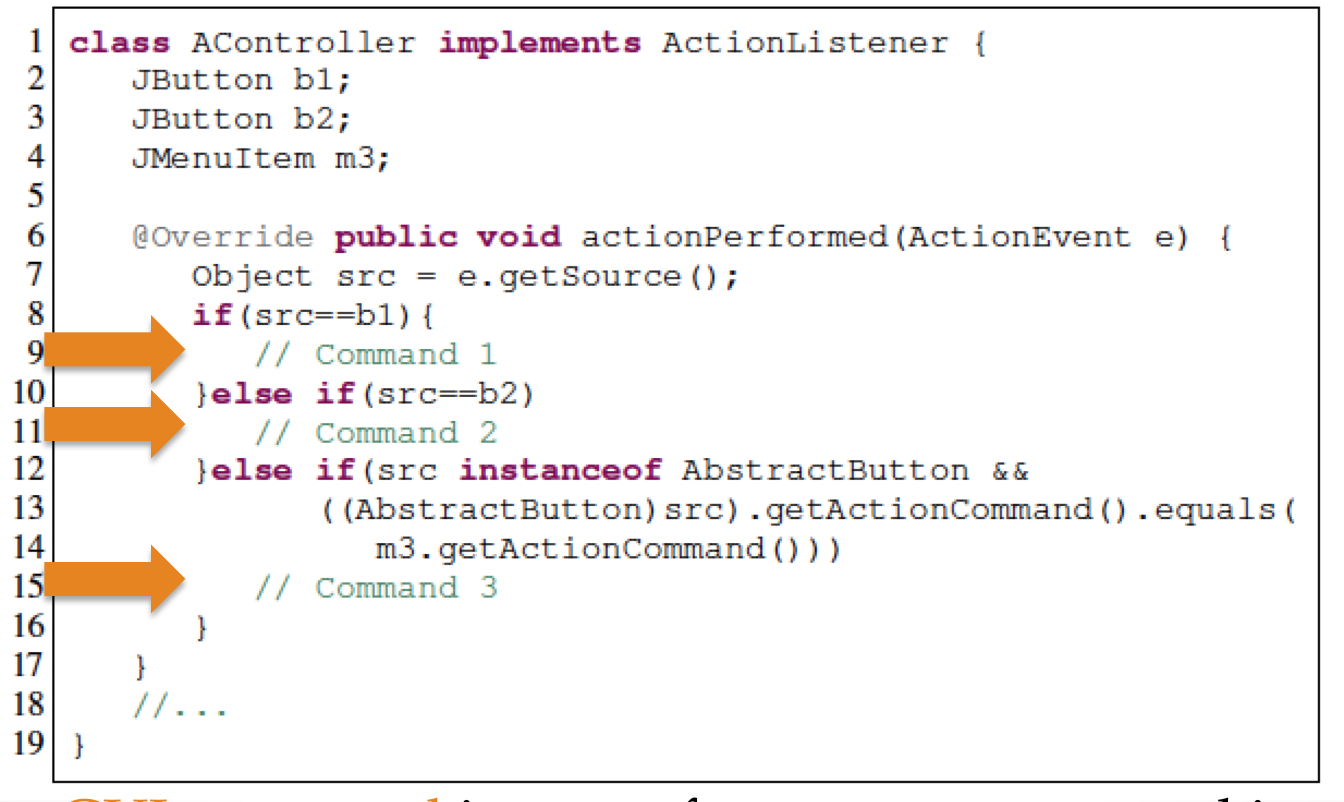


```java
class AController implements ActionListener {
   JButton b1;
   JButton b2;
   JMenuItem m3;

   @Override public void actionPerformed(ActionEvent e) {
      Object src = e.getSource();
      if(src==b1){
         // Command 1
      }else if(src==b2)
         // Command 2
      }else if(src instanceof AbstractButton &&
            ((AbstractButton)src).getActionCommand().equals(
               m3.getActionCommand()))
         // Command 3
      }
   }
   //...
}
```

- *AController* manages events produced by three widgets (b1, b2, and m3)

Blob listener

- A GUI listener produces **several** GUI commands



```java
class AController implements ActionListener {
   JButton b1;
   JButton b2;
   JMenuItem m3;

   @Override public void actionPerformed(ActionEvent e) {
      Object src = e.getSource();
      if(src==b1){
         // Command 1
      }else if(src==b2)
         // Command 2
      }else if(src instanceof AbstractButton &&
            ((AbstractButton)src).getActionCommand().equals(
               m3.getActionCommand()))
         // Command 3
      }
   }
   //...
}
```

GUI command is a set of statements executed in reaction of a user interaction

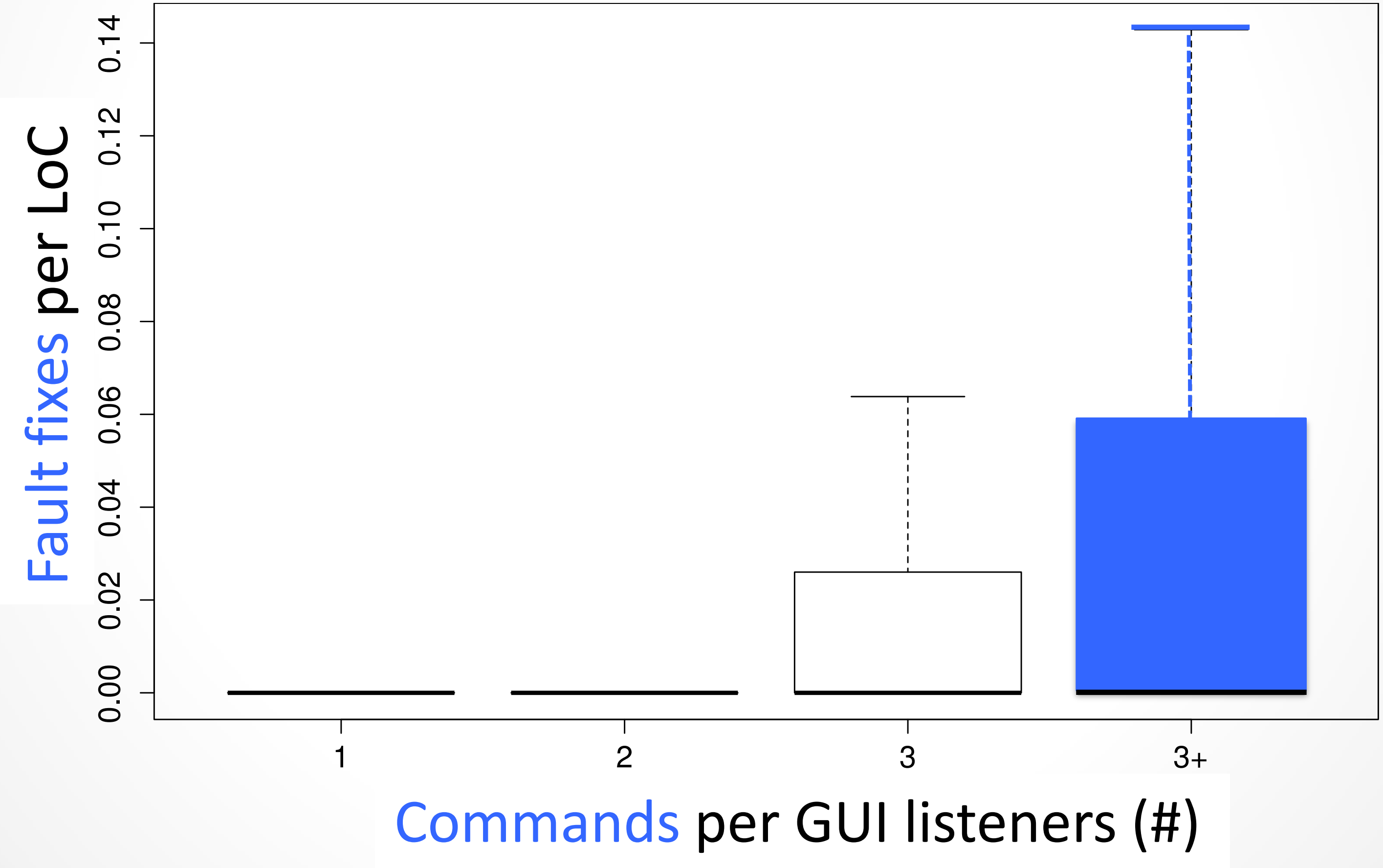
Empirical Study on GUI listeners

- **13 open-source software systems**
  - Github repository that use an issue-tracking system
  - Large Java systems
  - GUI size: 858 GUI listeners

- **Metrics**
  - Average commits
  - Average fault fixes
  - Number of commands

Results

- ✓ The number of commands per GUI listeners has a negative impact on fault-proneness of listeners code

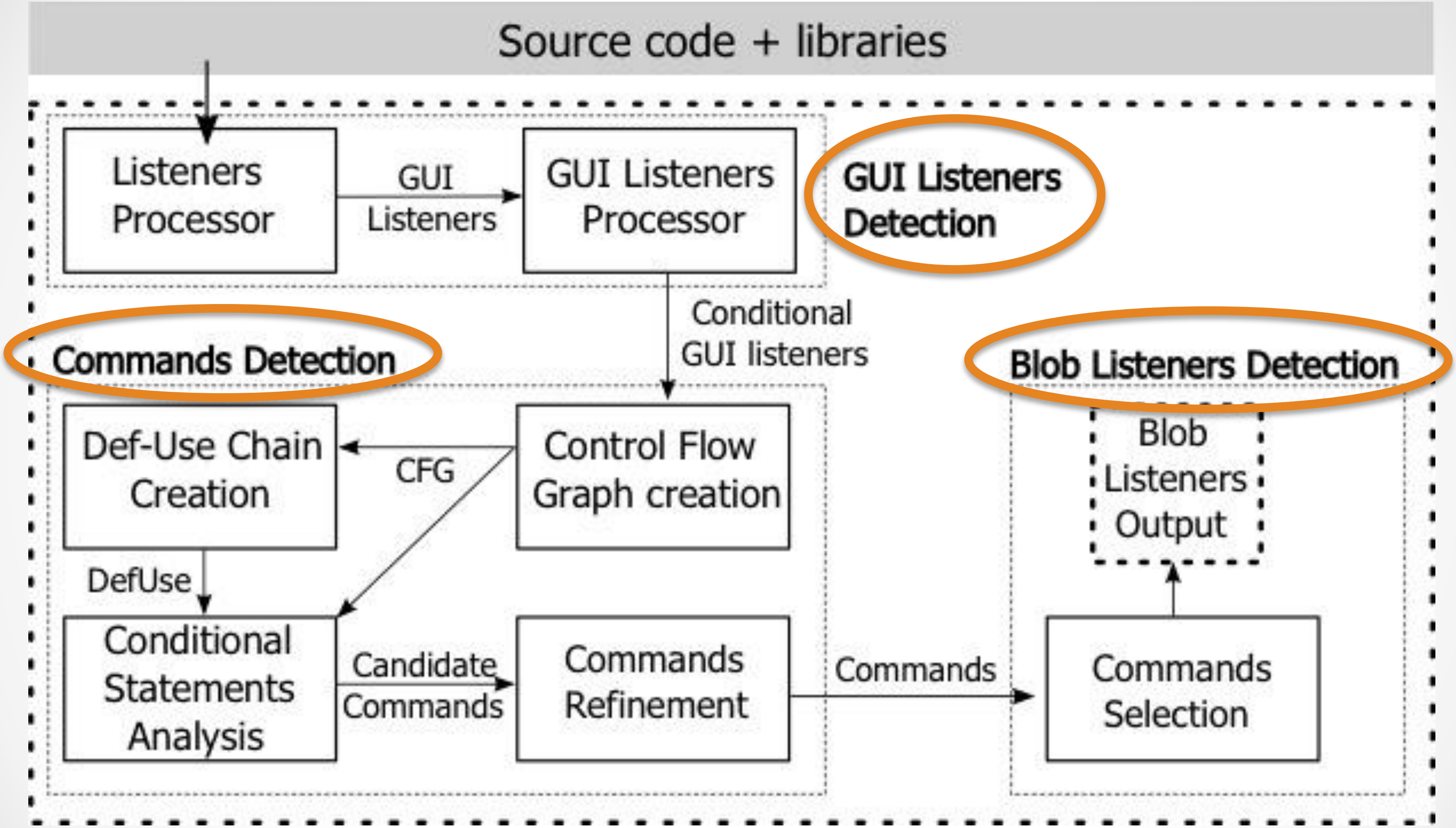


Results

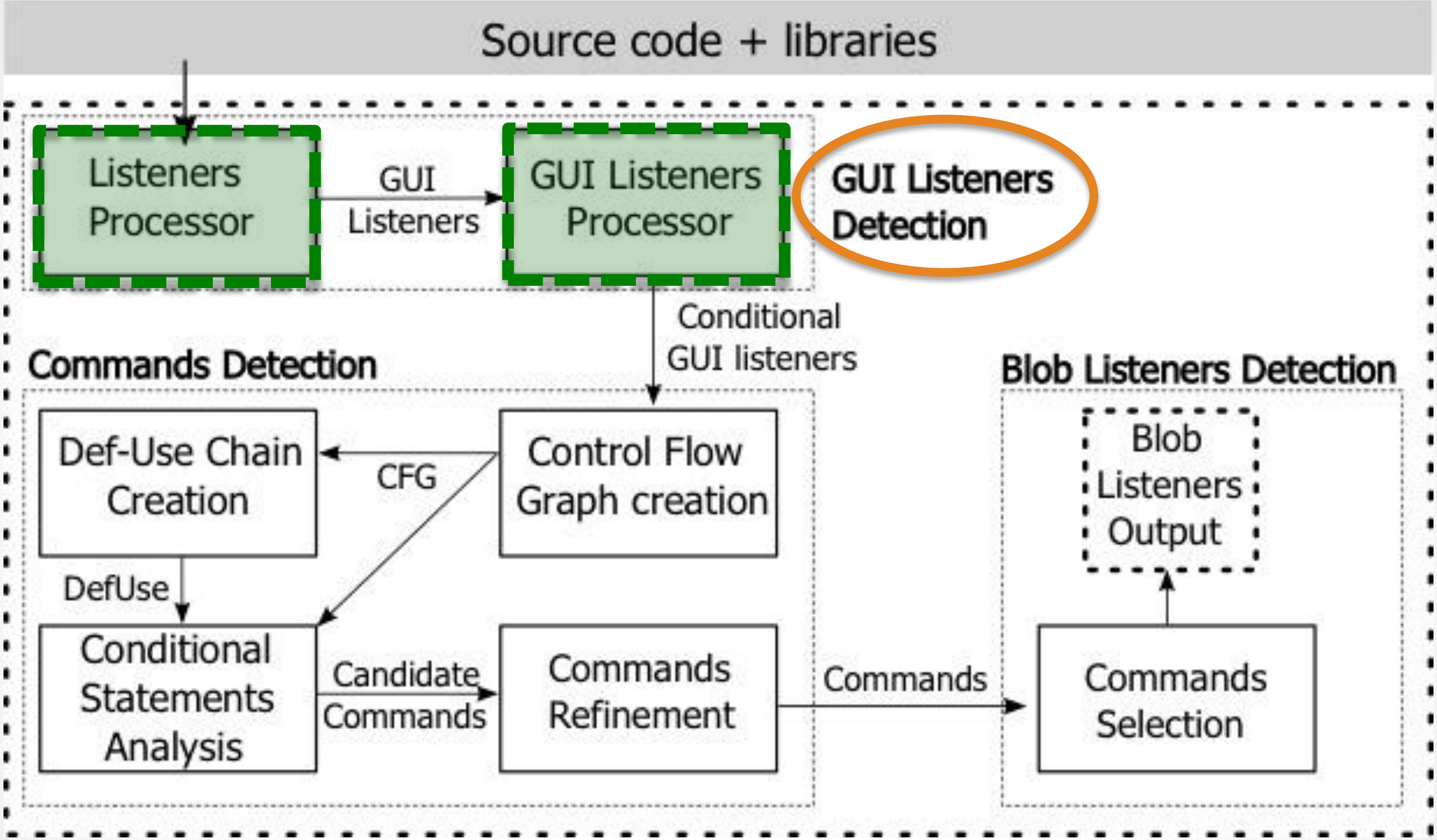
- ✓ Establish a threshold value to at least three commands per listener
  - ✓ 21% of the analyzed GUI listeners are *Blob listeners*

Blob Listener is a GUI listener that produces **more than two** GUI commands

Blob Listener detection

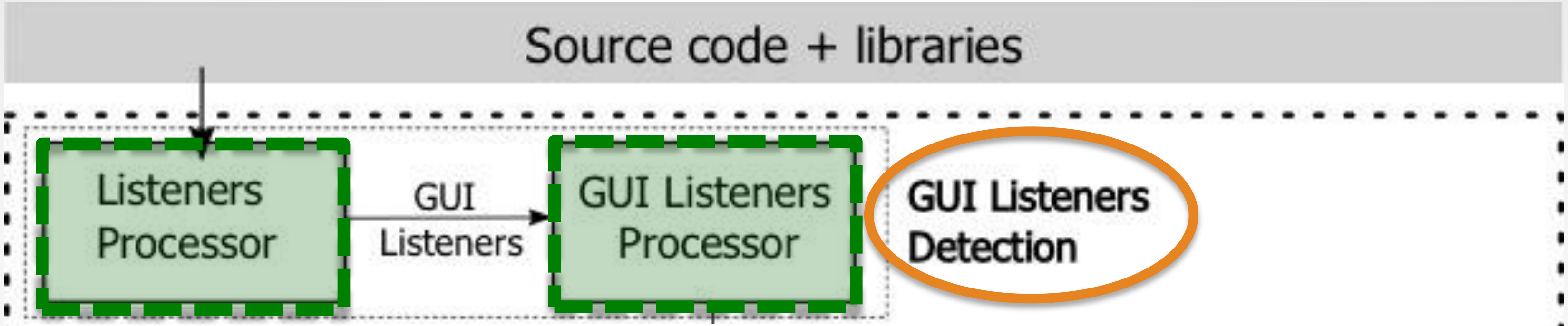


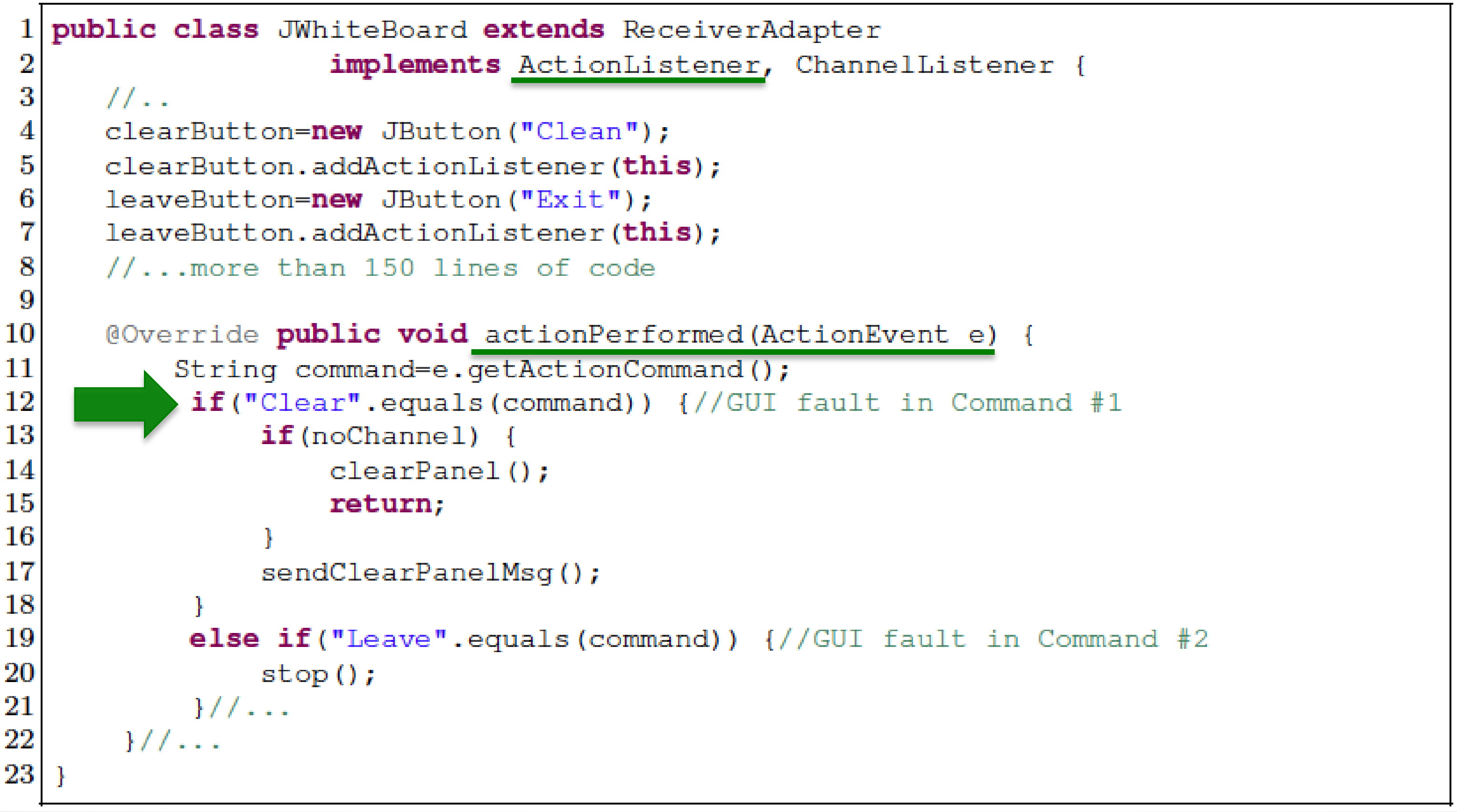
Blob Listener detection



- ✓ GUI listeners are analyzed to identify GUI listeners that have at least one conditional statement

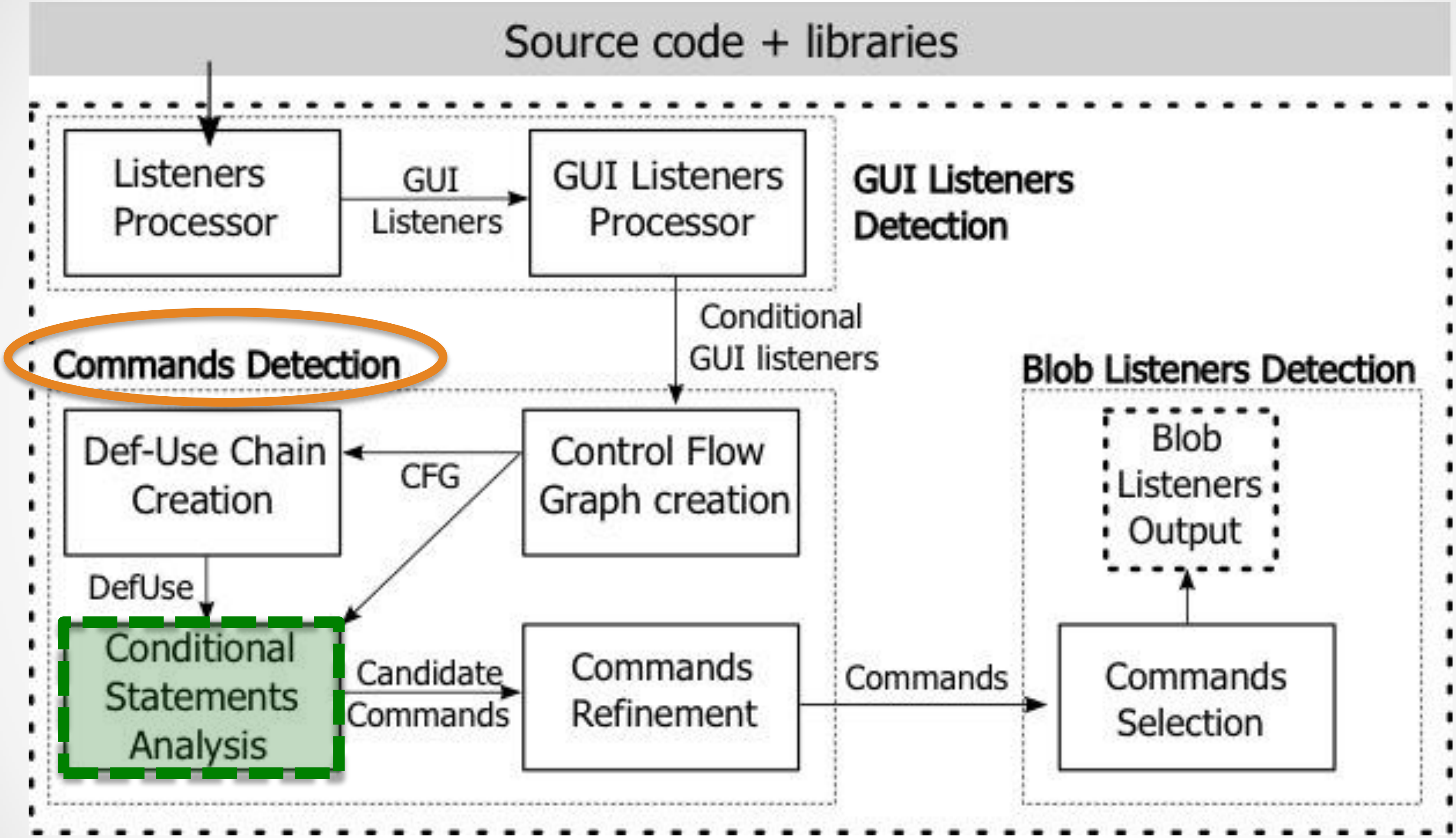
Blob Listener detection



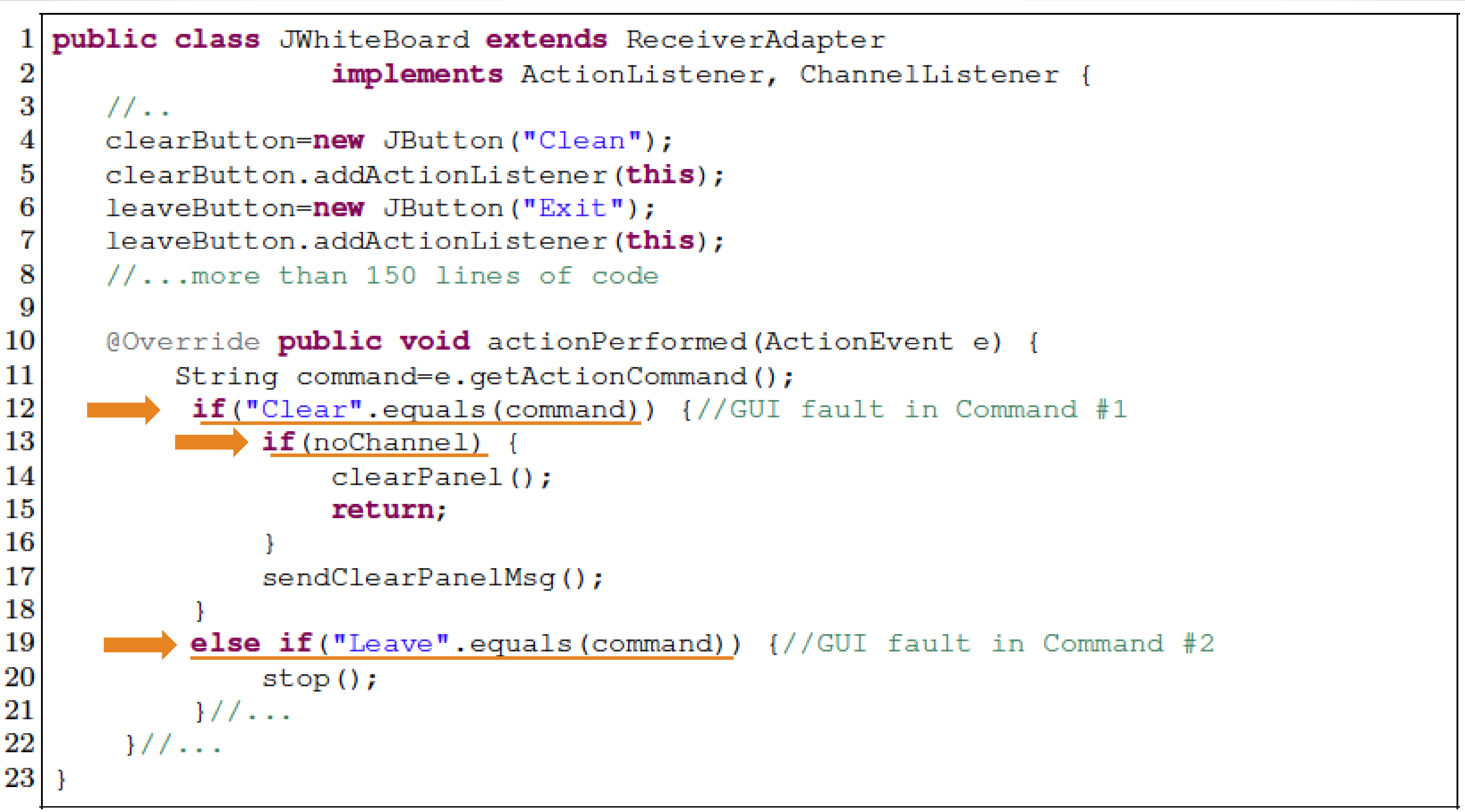


```
public class JWhiteBoard extends ReceiverAdapter
                implements ActionListener, ChannelListener {
   //..
   clearButton=new JButton("Clean");
   clearButton.addActionListener(this);
   leaveButton=new JButton("Exit");
   leaveButton.addActionListener(this);
   //...more than 150 lines of code

   @Override public void actionPerformed(ActionEvent e) {
       String command=e.getActionCommand();
        if("Clear".equals(command)) {//GUI fault in Command #1
            if(noChannel) {
                clearPanel();
                return;
            }
            sendClearPanelMsg();
        }
        else if("Leave".equals(command)) {//GUI fault in Command #2
            stop();
        }//...
    }//...
}
```

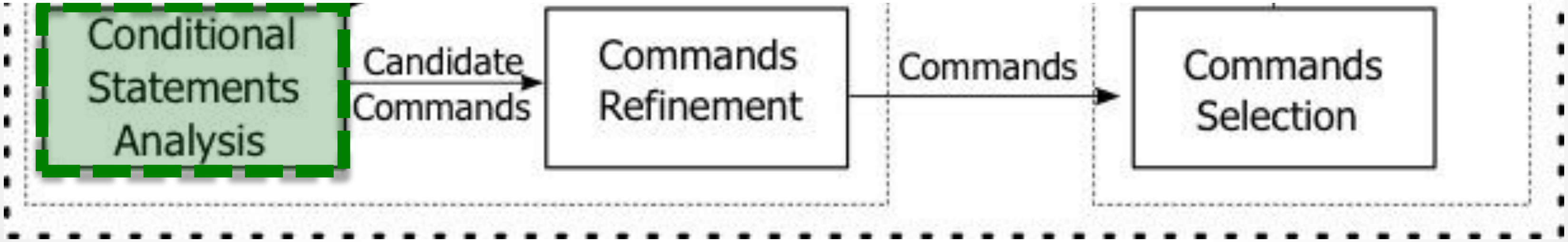
Blob Listener detection



✓ The conditionals are analyzed to detect any reference to a GUI event or widget



```
public class JWhiteBoard extends ReceiverAdapter
                implements ActionListener, ChannelListener {
   //..
   clearButton=new JButton("Clean");
   clearButton.addActionListener(this);
   leaveButton=new JButton("Exit");
   leaveButton.addActionListener(this);
   //...more than 150 lines of code

   @Override public void actionPerformed(ActionEvent e) {
       String command=e.getActionCommand();
        if("Clear".equals(command)) {//GUI fault in Command #1
            if(noChannel) {
                clearPanel();
                return;
            }
            sendClearPanelMsg();
        }
        else if("Leave".equals(command)) {//GUI fault in Command #2
            stop();
        }//...
    }//...
}
```



Conditional Statements Analysis

Candidate Commands

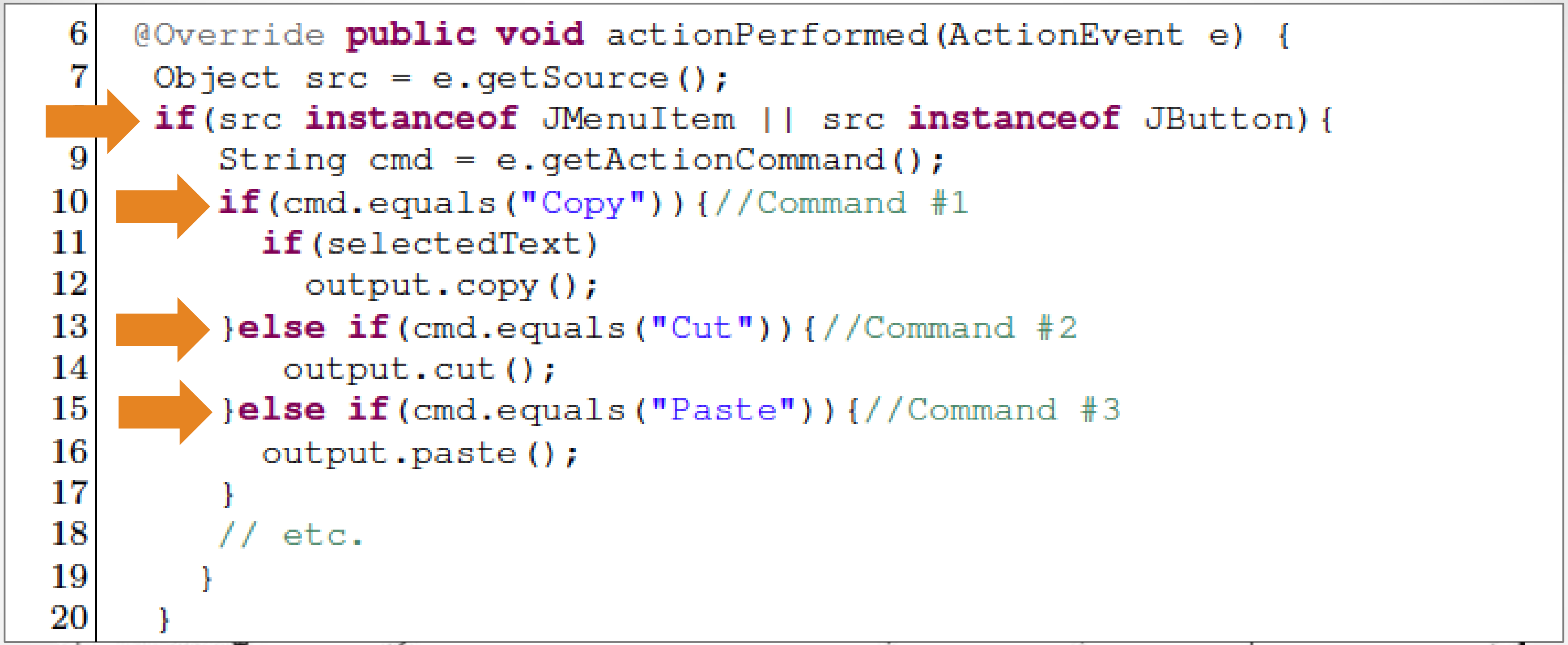
Commands Refinement

Commands

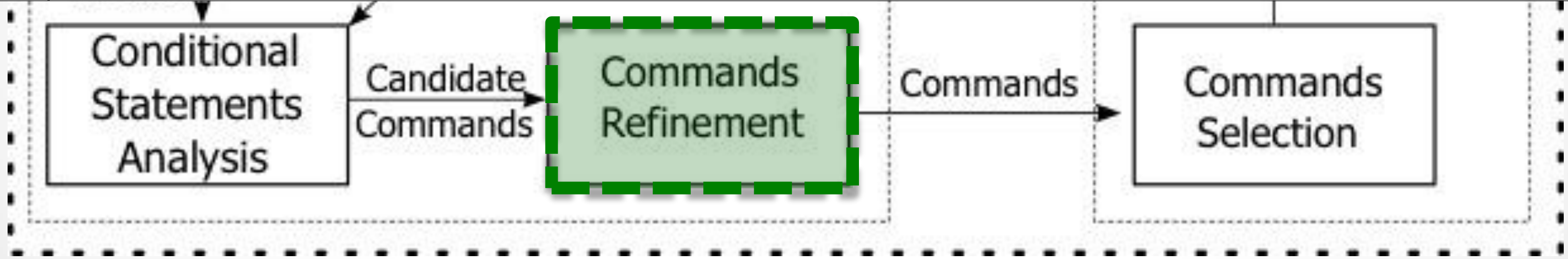
Commands Selection

✓ The conditionals are analyzed to detect any reference to a GUI event or widget

Blob Listener detection

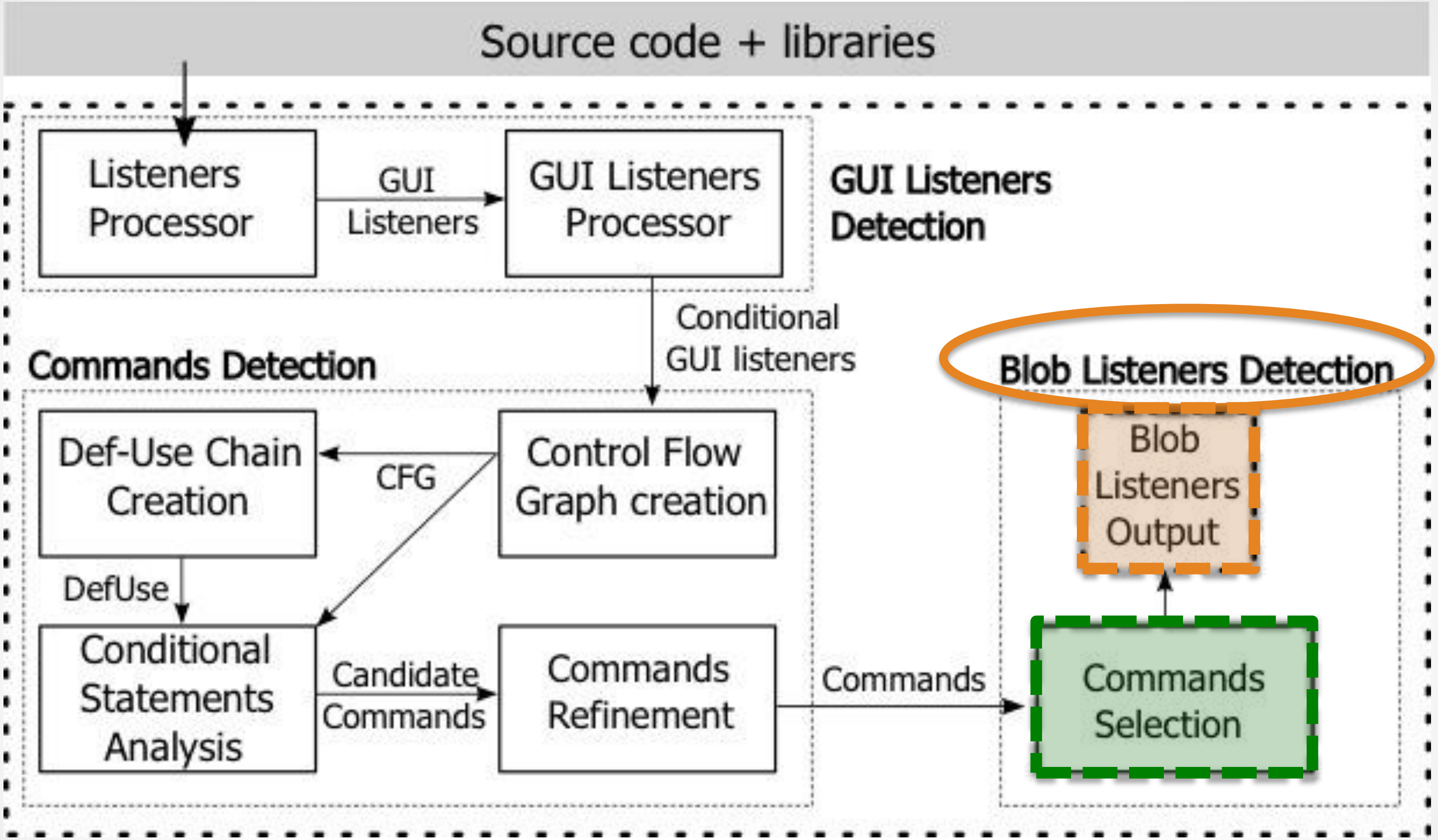


```
@Override public void actionPerformed(ActionEvent e) {
 Object src = e.getSource();
 if(src instanceof JMenuItem || src instanceof JButton){
    String cmd = e.getActionCommand();
    if(cmd.equals("Copy")){//Command #1
      if(selectedText)
        output.copy();
    }else if(cmd.equals("Cut")){//Command #2
       output.cut();
    }else if(cmd.equals("Paste")){//Command #3
      output.paste();
    }
    // etc.
   }
 }
```



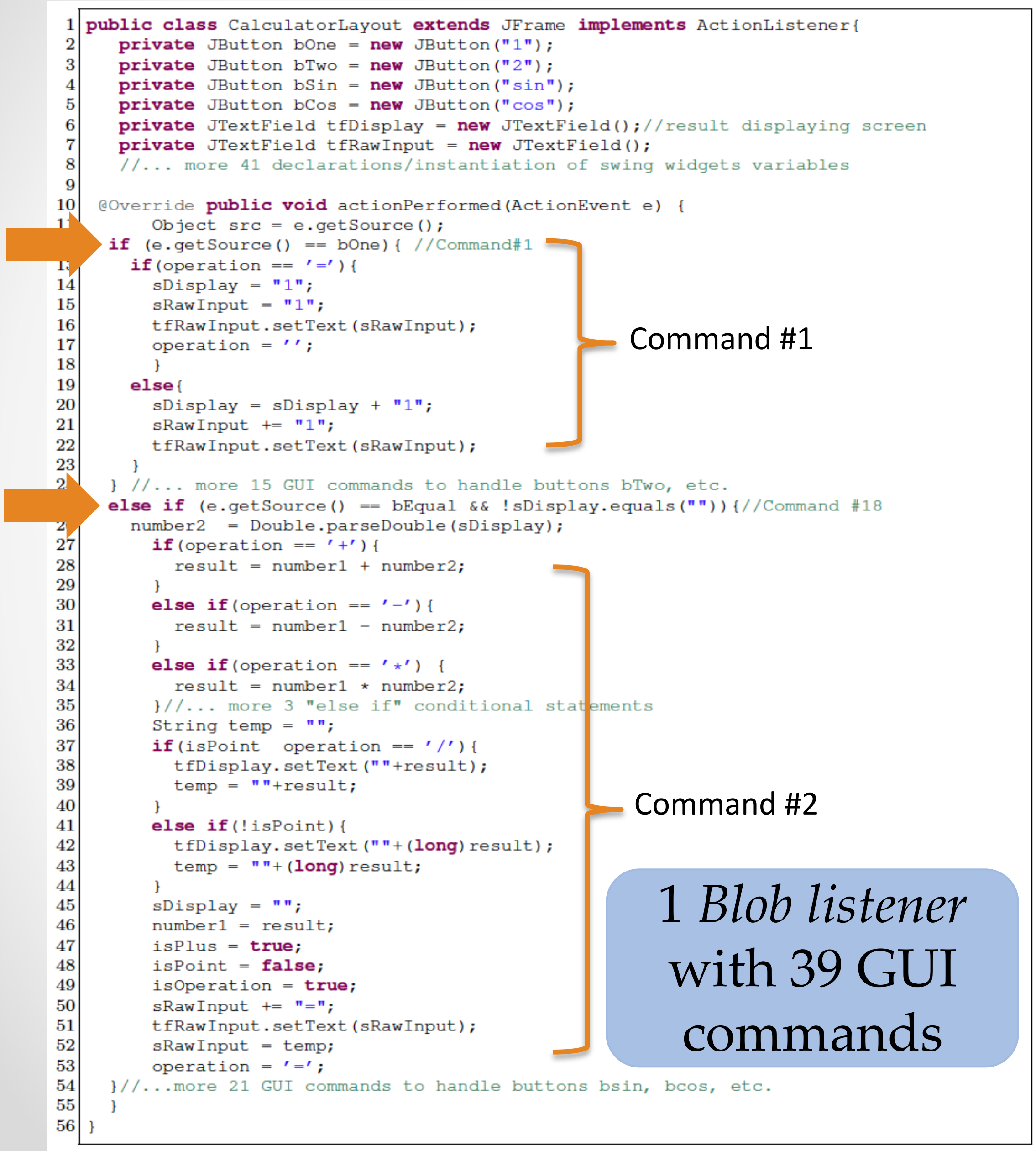
✓ The nested commands are removed

Blob Listener detection



✓ GUI listeners that contain more than two GUI commands are marked as Blob Listener

Blob Listener detection

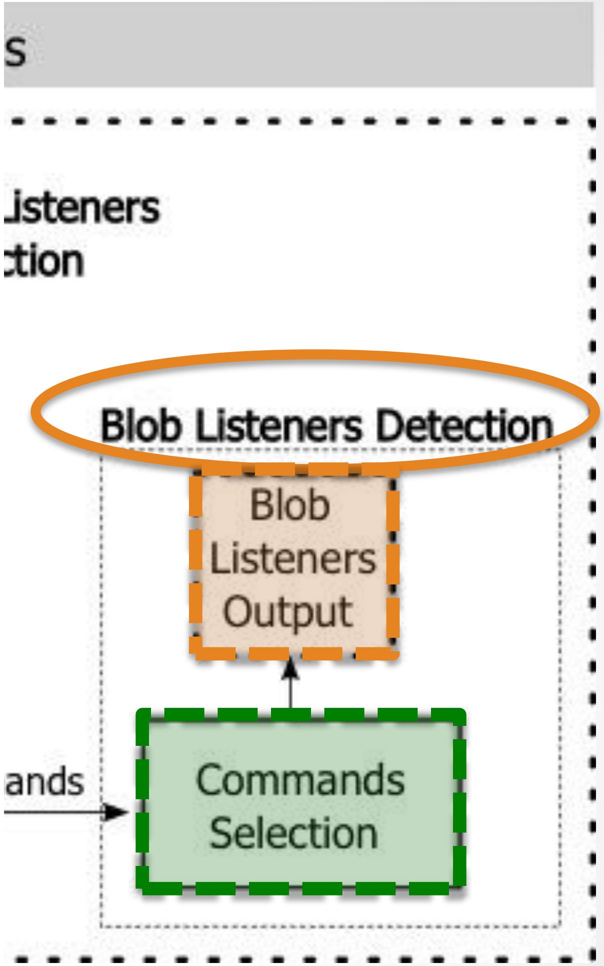


```
public class CalculatorLayout extends JFrame implements ActionListener{
  private JButton bOne = new JButton("1");
  private JButton bTwo = new JButton("2");
  private JButton bSin = new JButton("sin");
  private JButton bCos = new JButton("cos");
  private JTextField tfDisplay = new JTextField();//result displaying screen
  private JTextField tfRawInput = new JTextField();
  //... more 41 declarations/instantiation of swing widgets variables

 @Override public void actionPerformed(ActionEvent e) {
    Object src = e.getSource();
  if (e.getSource() == bOne){ //Command#1
    if(operation == '='){
      sDisplay = "1";
      sRawInput = "1";
      tfRawInput.setText(sRawInput);
      operation = '';
      }
    else{
      sDisplay = sDisplay + "1";
      sRawInput += "1";
      tfRawInput.setText(sRawInput);
    }
  } //... more 15 GUI commands to handle buttons bTwo, etc.
  else if (e.getSource() == bEqual && !sDisplay.equals("")){//Command #18
    number2 = Double.parseDouble(sDisplay);
      if(operation == '+'){
        result = number1 + number2;
      }
      else if(operation == '-'){
        result = number1 - number2;
      }
      else if(operation == '*') {
        result = number1 * number2;
      }//... more 3 "else if" conditional statements
      String temp = "";
      if(isPoint  operation == '/'){
        tfDisplay.setText(""+result);
        temp = ""+result;
      }
      else if(!isPoint){
        tfDisplay.setText(""+(long)result);
        temp = ""+(long)result;
      }
      sDisplay = "";
      number1 = result;
      isPlus = true;
      isPoint = false;
      isOperation = true;
      sRawInput += "=";
      tfRawInput.setText(sRawInput);
      sRawInput = temp;
      operation = '=';
  }//...more 21 GUI commands to handle buttons bsin, bcos, etc.
  }
}
```

Command #1

Command #2

1 *Blob listener* with 39 GUI commands



two GUI commands

InspectorGuidget[7]

- Open-source tool as an Eclipse plug-in dedicated to Java GUI systems

Software System	***Successfully Detected* Blob listeners (#)**	**FN (#)**	**FP (#)**	**Recall (%)**	**Precision (%)**
FastPhotoTagger	3	0	0	100.00	100.00
GanttProject	2	0	0	100.00	100.00
JaxoDraw	7	0	1	100.00	87.50
Jmol	11	1	0	91.67	81.82
TerPaint	3	0	0	100.00	100.00
TripleA	11	0	0	100.00	100.00
Overall	**37**	**1**	**1**	**97.59**	**97.37**

✓ 37 out of 38 *Blob listeners* were detected

[7] https://github.com/diverse-project/InspectorGuidget

Conclusion

- **A new type of GUI design smell**
  - *Blob listener* has an negative impact on fault-proneness of GUI listeners
- **A novel static analysis approach**
  - InspectorGuidget dedicated to Java systems
  - 37 out of 38 instances of *Blob listeners* on six real-world GUI systems
- **Good coding practices** to avoid the presence of *Blob listeners*

Conclusions & Perspectives

Conclusions

- **GUI fault model**
  - 279 GUI-related bug reports of five interactive open-source systems
  - Evaluation of GUI testing frameworks against real GUI failures and GUI mutants

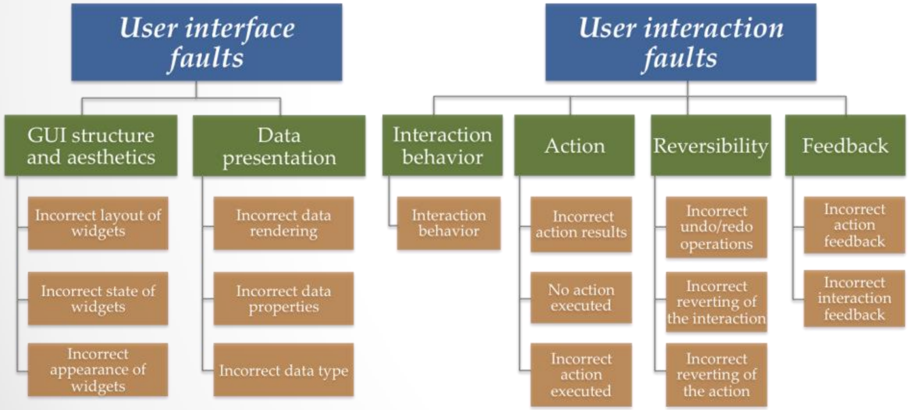
- **An automatic detection of a new type of GUI design smell**
  - *Blob listener* that degrades the GUI code quality
  - InspectorGuidget detected 37 out 38 instances of *Blob listeners*

Conclusions

- **Experiment and tools**
  - A complete data set
  - GUI systems that have several interactive features
  - Empirial studies of GUI implementations

Perspectives

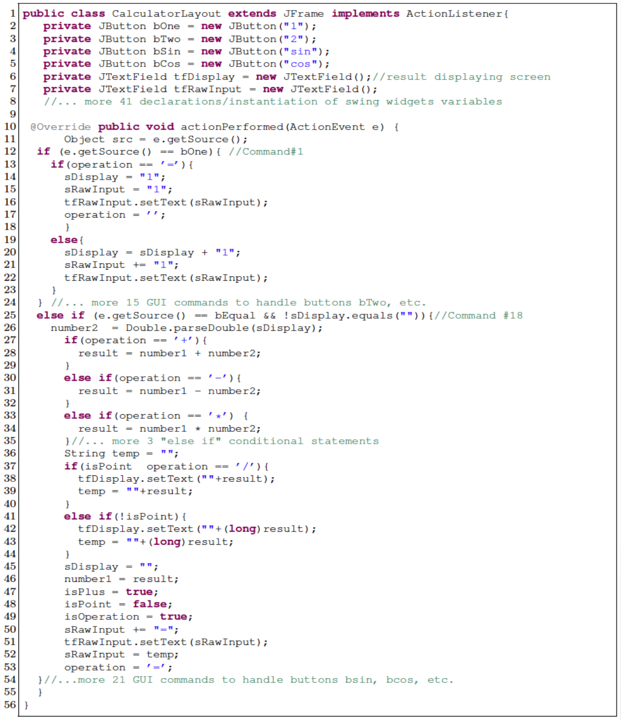
- Domain-specific mutants
  - Mapping between GUI faults and specific GUI toolkits


User interface faults
User interaction faults
GUI structure and aesthetics
Data presentation
Interaction behavior
Action
Reversibility
Feedback
Incorrect layout of widgets
Incorrect state of widgets
Incorrect appearance of widgets
Incorrect data rendering
Incorrect data properties
Incorrect data type
Interaction behavior
Incorrect action results
No action executed
Incorrect action executed
Incorrect undo/redo operations
Incorrect reverting of the interaction
Incorrect reverting of the action
Incorrect action feedback
Incorrect interaction feedback

GUI Fault Model

Mapping

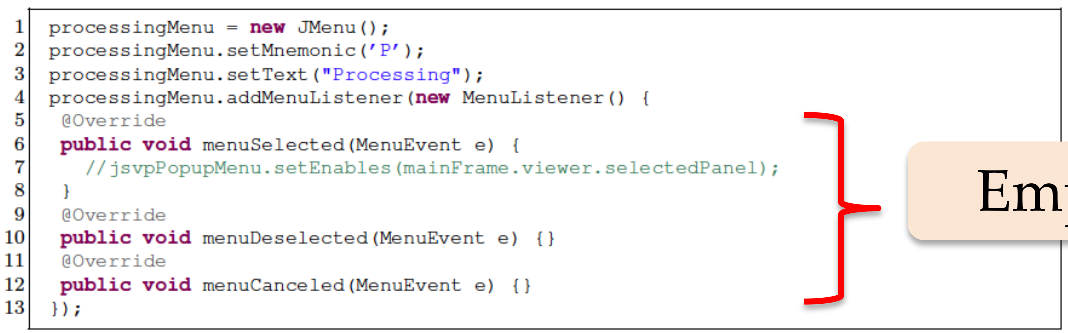
Java Swing mutants



```java
public class CalculatorLayout extends JFrame implements ActionListener{
   private JButton bOne = new JButton("1");
   private JButton bTwo = new JButton("2");
   private JButton bSin = new JButton("sin");
   private JButton bCos = new JButton("cos");
   private JTextField tfDisplay = new JTextField();//result displaying screen
   private JTextField tfRawInput = new JTextField();
   //... more 41 declarations/instantiation of swing widgets variables

 @Override public void actionPerformed(ActionEvent e) {
      Object src = e.getSource();
  if (e.getSource() == bOne){ //Command#1
    if(operation == '=')
      sDisplay = "1";
      sRawInput = "1";
      tfRawInput.setText(sRawInput);
      operation = '';
      }
    else{
      sDisplay = sDisplay + "1";
      sRawInput += "1";
      tfRawInput.setText(sRawInput);
    }
  } //... more 15 GUI commands to handle buttons bTwo, etc.
  else if (e.getSource() == bEqual && !sDisplay.equals("")){//Command #18
    number2  = Double.parseDouble(sDisplay);
      if(operation == '+'){
        result = number1 + number2;
      }
      else if(operation == '-'){
        result = number1 - number2;
      }
      else if(operation == '*') {
        result = number1 * number2;
      }//... more 3 "else if" conditional statements
      String temp = "";
      if(isPoint  operation == '/'){
        tfDisplay.setText(""+result);
        temp = ""+result;
      }
      else if(!isPoint){
        tfDisplay.setText(""+(long)result);
        temp = ""+(long)result;
      }
      sDisplay = "";
      number1 = result;
      isPlus = true;
      isPoint = false;
      isOperation = true;
      sRawInput += "=";
      tfRawInput.setText(sRawInput);
      sRawInput = temp;
      operation = '=';
  }//...more 21 GUI commands to handle buttons bsin, bcos, etc.
  }
}
```

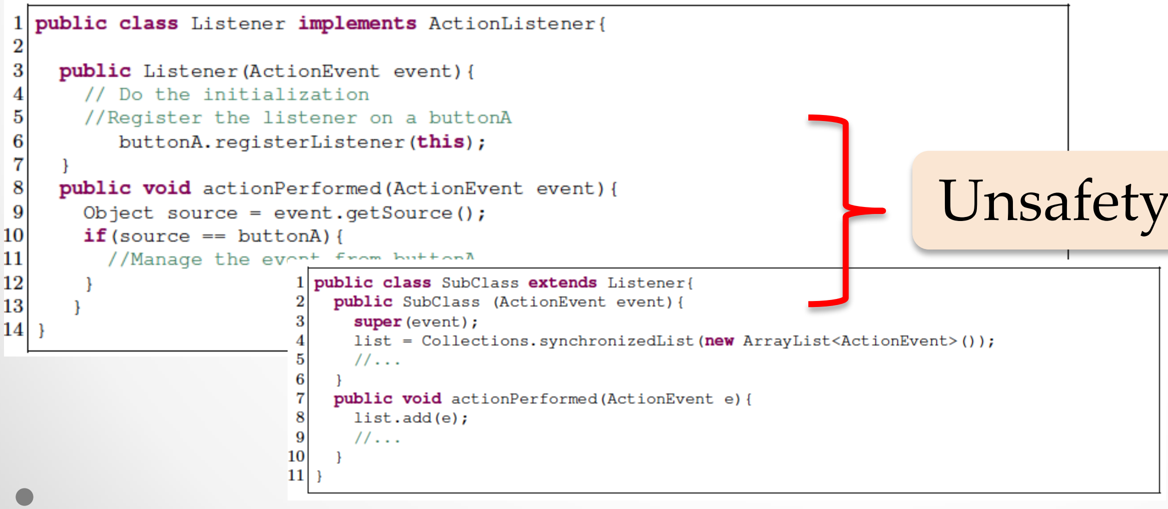
Perspectives

- **GUI design smells**
  - A set of checking rules to check automatically for potential defects in GUI code



```java
processingMenu = new JMenu();
processingMenu.setMnemonic('P');
processingMenu.setText("Processing");
processingMenu.addMenuListener(new MenuListener() {
 @Override
 public void menuSelected(MenuEvent e) {
   //jsvpPopupMenu.setEnables(mainFrame.viewer.selectedPanel);
 }
 @Override
 public void menuDeselected(MenuEvent e) {}
 @Override
 public void menuCanceled(MenuEvent e) {}
});
```

Empty listener bodies



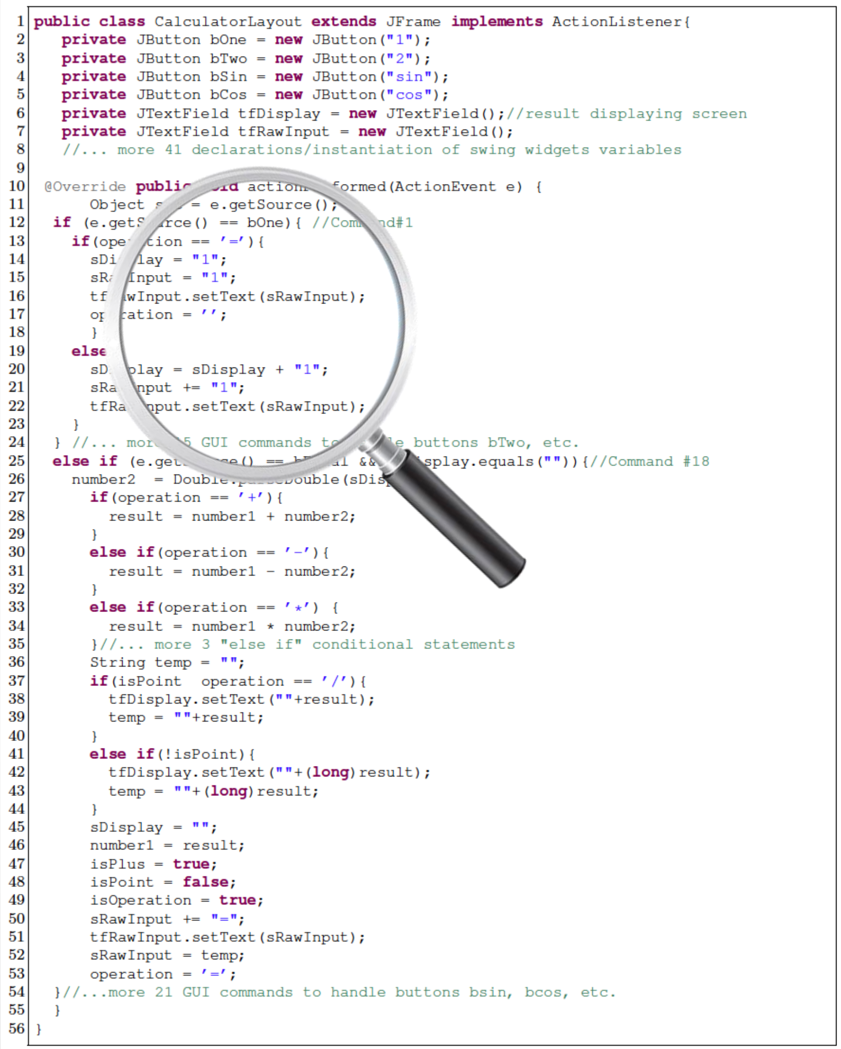
```java
public class Listener implements ActionListener{

  public Listener(ActionEvent event){
    // Do the initialization
    //Register the listener on a buttonA
       buttonA.registerListener(this);
  }
  public void actionPerformed(ActionEvent event){
    Object source = event.getSource();
    if(source == buttonA){
      //Manage the event from buttonA
    }
  }
}
```

```java
public class SubClass extends Listener{
  public SubClass (ActionEvent event){
    super(event);
    list = Collections.synchronizedList(new ArrayList<ActionEvent>());
    //...
  }
  public void actionPerformed(ActionEvent e){
    list.add(e);
    //...
  }
}
```

Unsafety listener registration

Perspectives

- **GUI design smells**



```java
public class CalculatorLayout extends JFrame implements ActionListener{
   private JButton bOne = new JButton("1");
   private JButton bTwo = new JButton("2");
   private JButton bSin = new JButton("sin");
   private JButton bCos = new JButton("cos");
   private JTextField tfDisplay = new JTextField();//result displaying screen
   private JTextField tfRawInput = new JTextField();
   //... more 41 declarations/instantiation of swing widgets variables

 @Override public void actionPerformed(ActionEvent e) {
      Object src = e.getSource();
  if (e.getSource() == bOne){ //Command#1
    if(operation == '='){
      sDisplay = "1";
      sRawInput = "1";
      tfRawInput.setText(sRawInput);
      operation = '';
      }
    else
      sDisplay = sDisplay + "1";
      sRawInput += "1";
      tfRawInput.setText(sRawInput);
    }
  } //... more 15 GUI commands to handle buttons bTwo, etc.
  else if (e.getSource() == bEqual && !sDisplay.equals("")){//Command #18
    number2  = Double.parseDouble(sDisplay);
      if(operation == '+'){
        result = number1 + number2;
      }
      else if(operation == '-'){
        result = number1 - number2;
      }
      else if(operation == '*') {
        result = number1 * number2;
      }//... more 3 "else if" conditional statements
      String temp = "";
      if(isPoint  operation == '/'){
        tfDisplay.setText(""+result);
        temp = ""+result;
      }
      else if(!isPoint){
        tfDisplay.setText(""+(long)result);
        temp = ""+(long)result;
      }
      sDisplay = "";
      number1 = result;
      isPlus = true;
      isPoint = false;
      isOperation = true;
      sRawInput += "=";
      tfRawInput.setText(sRawInput);
      sRawInput = temp;
      operation = '=';
  }//...more 21 GUI commands to handle buttons bsin, bcos, etc.
  }
}
```

Bug finders

- **Findbugs**
- **PMD**, *etc.*

Publications

- Valéria Lelli, Arnaud Blouin, and Baudry Benoit. **Classifying and qualifying GUI defects**. In Software Testing, Verification and Validation (ICST), 2015 IEEE Eighth International Conference, pages 1–10, April 2015.

- Valéria Lelli, Arnaud Blouin, Baudry Benoit, and Fabien Coulon. **On model-based testing advanced GUIs.** In 11th Workshop on Advances in Model Based Testing (A-MOST), pages 1–10, April 2015. *Best paper award*.

- Valéria Lelli. **Challenges of testing for critical interactive systems**. In Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference. Doctoral Symposium, pages 509–510, March 2013.

- Valéria Lelli, Arnaud Blouin, Baudry Benoit, Fabien Coulon, and Olivier Beaudoux. **Automatic Detection of GUI Design Smells: The Case of Blob Listener**. *Submitted* to Software Testing, Verification and Validation Conference (ICST) 2016.