Finding Causes of Program Output with the Java WhyLine

Andrew Ko & Brad Myers

CHI 2009

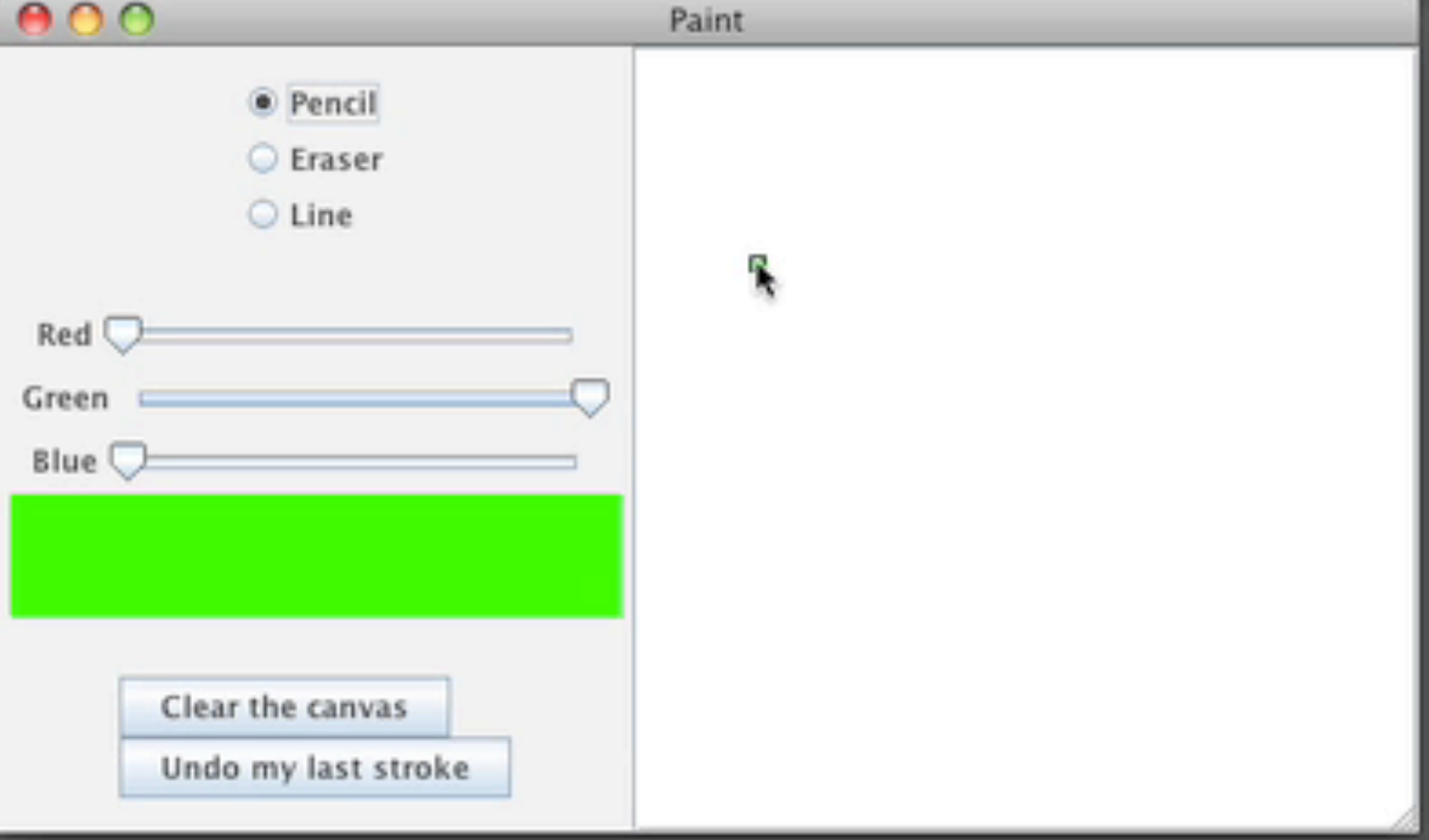
Summary by Prof. Thomas LaToza

SWE 795, Spring 2017

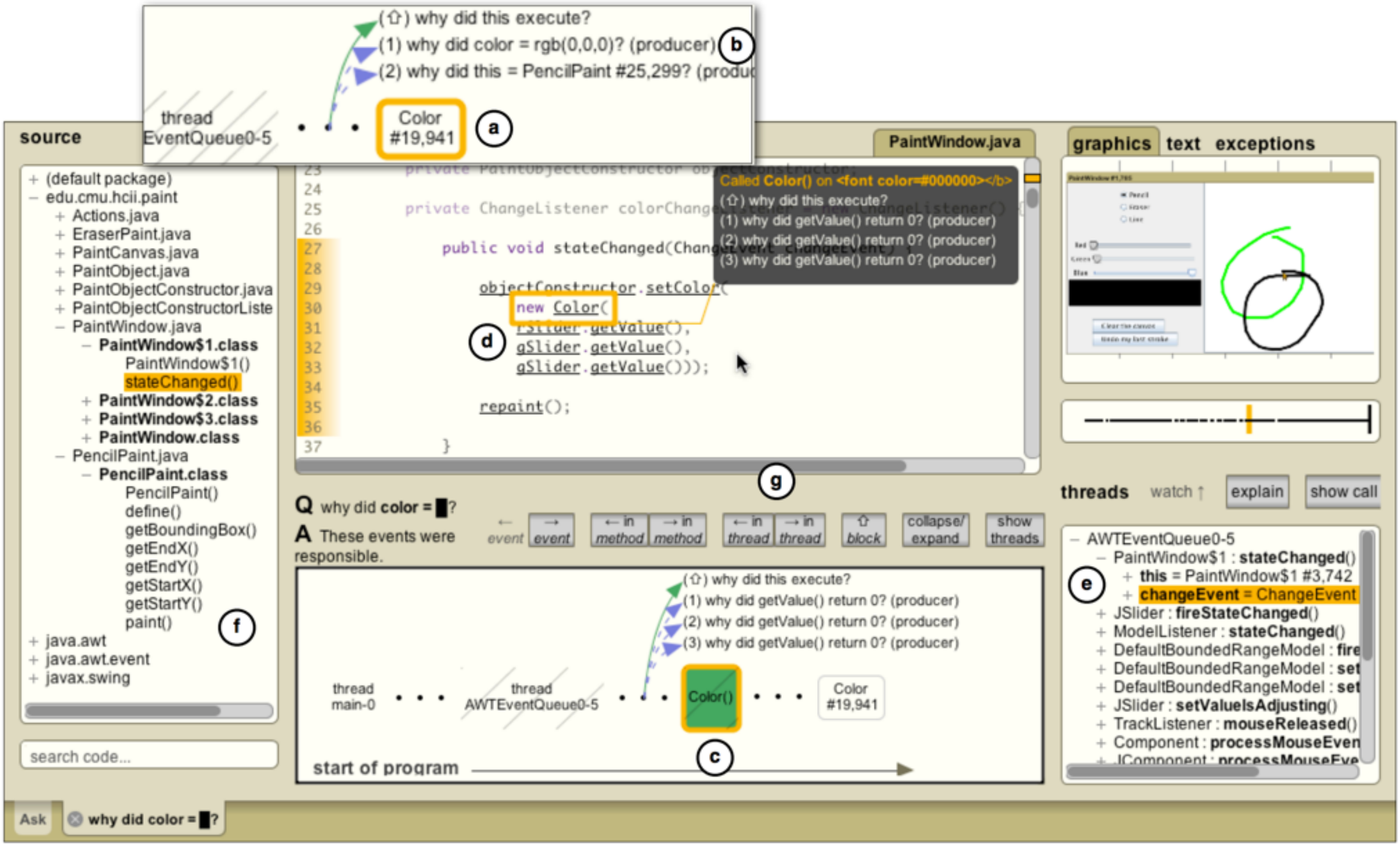
Software Engineering Environments

Finding Causes of Program Output

- Problem
  - Debugging challenging because developers must map observable symptom of failure (e.g., a button that is not displayed) to underlying **cause**
  - Developers must map incorrect output to responsible code
    - Requires **guessing** cause (hypothesizing) and checking with tools
    - Most hypotheses are wrong
- Solution
  - Enable developers to directly ask **why** and **why not** questions about output, trace back to code responsible for output


Paint
Pencil
Eraser
Line
Red
Green
Blue
Clear the canvas
Undo my last stroke

WhyLine


(⇧) why did this execute?
(1) why did color = rgb(0,0,0)? (producer)
(2) why did this = PencilPaint #25,299? (produc
b
thread
EventQueue0-5
Color
#19,941
a
source
+ (default package)
– edu.cmu.hcii.paint
+ Actions.java
+ EraserPaint.java
+ PaintCanvas.java
+ PaintObject.java
+ PaintObjectConstructor.java
+ PaintObjectConstructorListe
– PaintWindow.java
– PaintWindow$1.class
PaintWindow$1()
stateChanged()
+ PaintWindow$2.class
+ PaintWindow$3.class
+ PaintWindow.class
– PencilPaint.java
– PencilPaint.class
PencilPaint()
define()
getBoundingBox()
getEndX()
getEndY()
getStartX()
getStartY()
paint()
f
+ java.awt
+ java.awt.event
+ javax.swing
search code...
PaintWindow.java
Called Color() on <font color=#000000>
(⇧) why did this execute?
(1) why did getValue() return 0? (producer)
(2) why did getValue() return 0? (producer)
(3) why did getValue() return 0? (producer)
23 private PaintObjectConstructor objectConstructor;
24
25 private ChangeListener colorChange
26
27 public void stateChanged(Chang
28
29 objectConstructor.setColor(
30 new Color(
31 rSlider.getValue(),
32 gSlider.getValue(),
33 gSlider.getValue()));
34
35 repaint();
36
37 }
d
g
Q why did color = ?
A These events were responsible.
← event
→ event
← in method
→ in method
← in thread
→ in thread
⇧ block
collapse/ expand
show threads
(⇧) why did this execute?
(1) why did getValue() return 0? (producer)
(2) why did getValue() return 0? (producer)
(3) why did getValue() return 0? (producer)
thread main-0
thread AWTEventQueue0-5
Color()
Color #19,941
start of program
c
graphics
text
exceptions
threads
watch ↑
explain
show call
– AWTEventQueue0-5
– PaintWindow$1 : stateChanged()
+ this = PaintWindow$1 #3,742
+ changeEvent = ChangeEvent
e
+ JSlider : fireStateChanged()
+ ModelListener : stateChanged()
+ DefaultBoundedRangeModel : fire
+ DefaultBoundedRangeModel : set
+ DefaultBoundedRangeModel : set
+ JSlider : setValueIsAdjusting()
+ TrackListener : mouseReleased()
+ Component : processMouseEven
Ask
why did color = ?

Timeline visualization

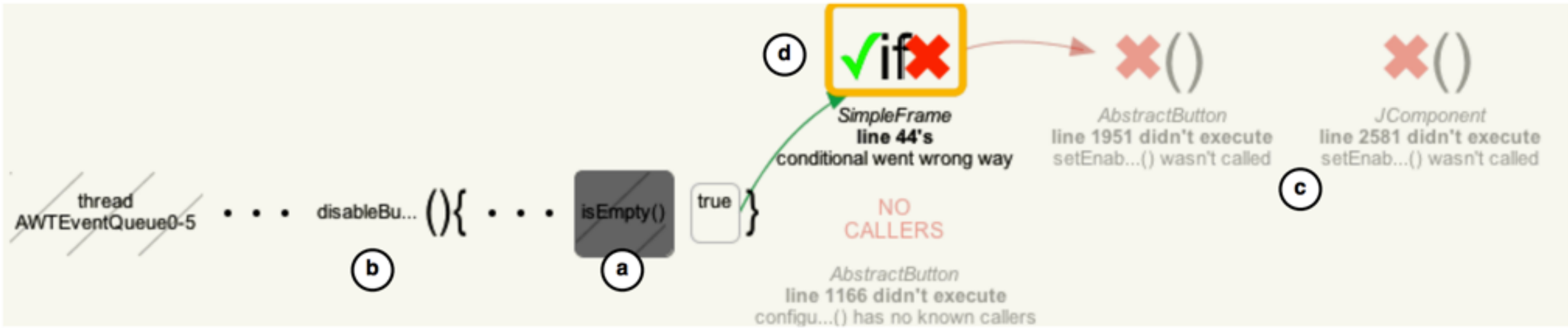


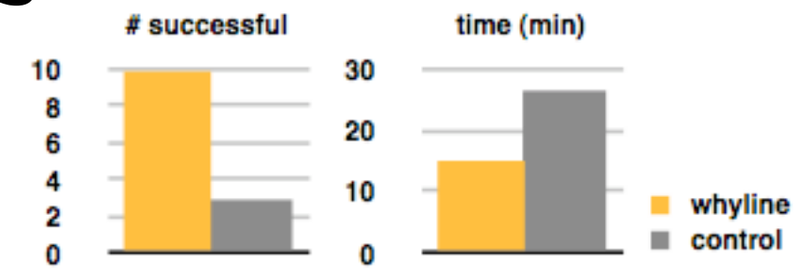
Figure 10. An answer showing (a) a collapsed invocation, (b) a hidden call context, (c) several unexecuted instructions, and (d) a conditional that evaluated in the wrong direction, preventing the desired instruction from executing.

Evaluation

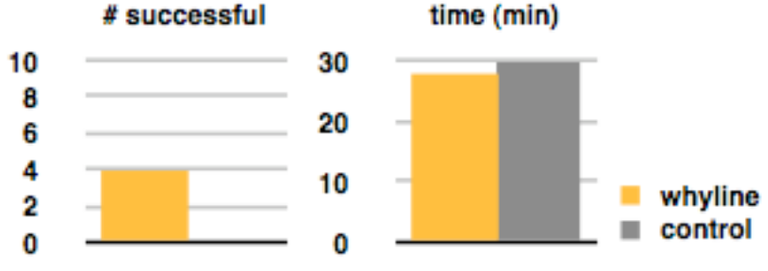
- **20** masters students did two 30 minute tasks
- Used **tutorial** to teach the tool to users
- Tasks: **debug** 2 real bug reports from ArgoUML
  Diagnose problem & write change recommendation
- **Measured** time, success, code exploration, perception

Results

Task 1



Task 2



		task 1		task 2	
		whyline	*control*	*whyline*	*control*
# of unique source files viewed per minute	*mean*	1.8	13.3	1	0.6
	σ^2	1.4	0.8	0.5	0.4
range of files viewed		8 – 39	10 – 66	16 – 72	6 – 44
median distance to key function	*mean*	2.2	3.4	3.6	3.3
	σ^2	0.6	0.5	0.5	0.5
# *why did* questions (median, range)		2, 1–4	—	4, 1–8	—
# *why didn't* questions (median, range)		0, 0–0	—	0, 0–2	—
median # debugger steps taken		—	9	—	14.5
median # text searches		0.5	7	1	8

Questions for discussion

- Overall reaction to the paper
- Are the claims about the benefits of WhyLine convincing?
  - How much evaluation is enough?
- In what contexts might WhyLine be more difficult to apply?
  - How might WhyLine be extended to support these contexts?
- What are the pros and cons of WhyLine approach to debugging vs. alternatives?
- How much time overhead does demonstrating bug for WhyLine add for developer?
- What challenges would there be in commercializing WhyLine?