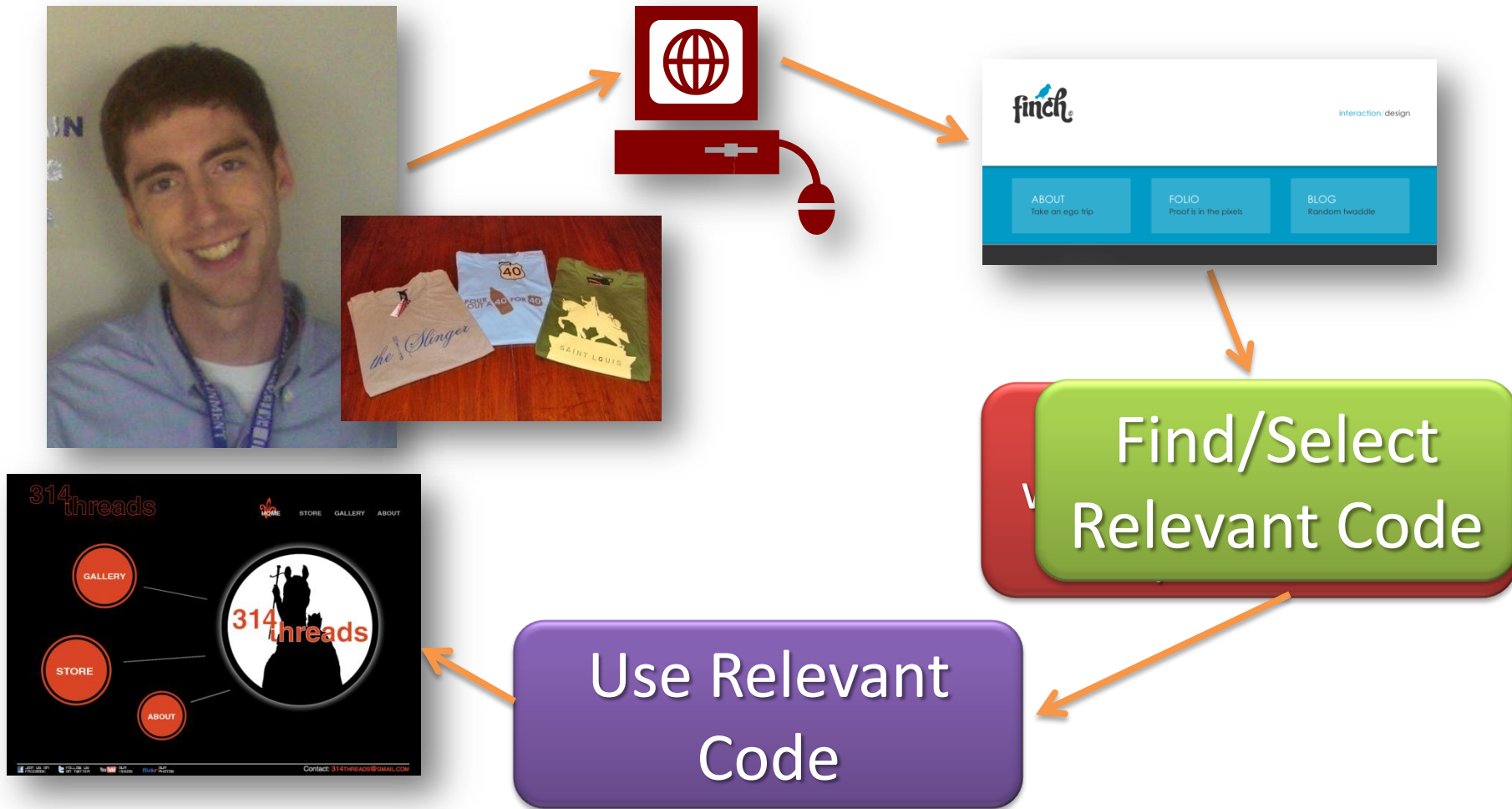


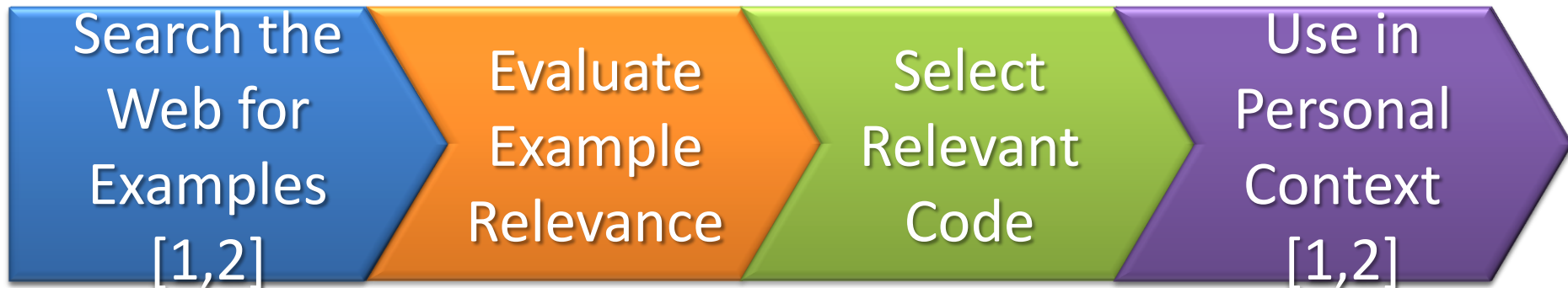
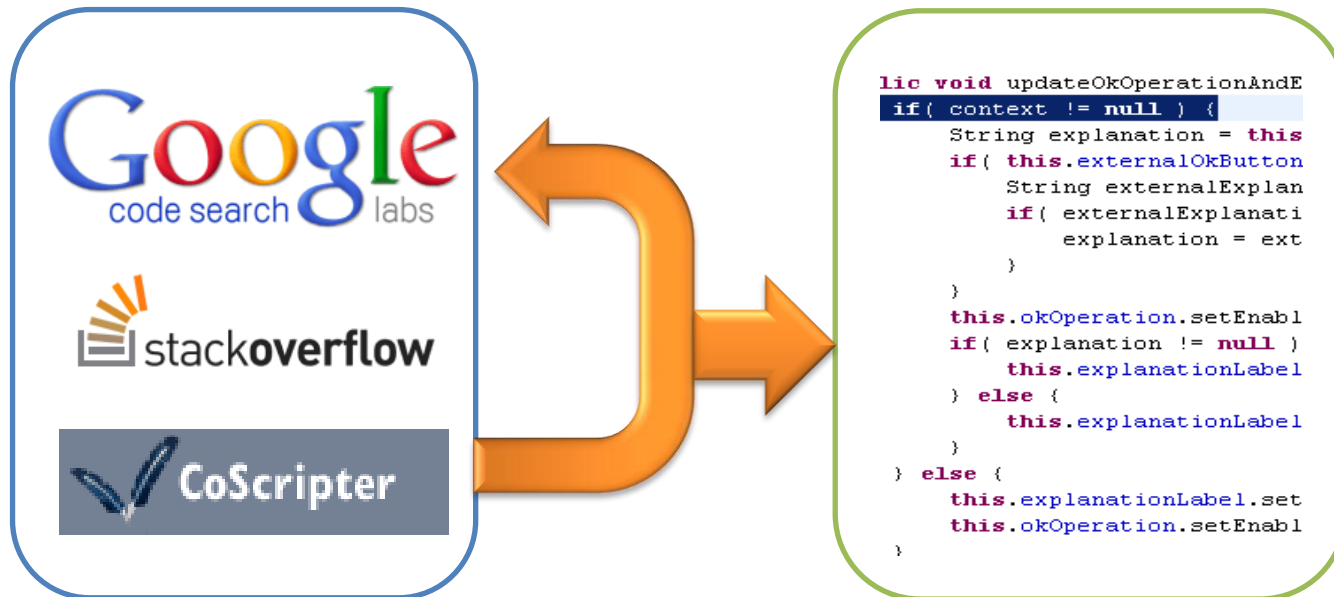
Toward Transforming Freely Available Source Code into Usable Learning Materials for End-Users

Paul Gross and Caitlin Kelleher
Washington University in St. Louis

Illustrating Available, Unusable Code



An Independent Learning Process for End-user Programmers



[1] Brandt et al., *Two studies of opportunistic programming: interleaving web...*, CHI 2009.
 [2] Rosson et al., *Who, what, and how: A survey of informal and professional...*, VL/HCC 2005.

Our Context: Looking Glass

The screenshot displays the 'Looking Glass' software interface. The window title is 'Looking Glass C:\Documents and Settings\grosspa\My Documents\LookingGlass\MyProjects\CafeteriaKiss.a3p'. The interface is divided into several sections:

- Scene View (Top Left):** A 3D environment showing a cafeteria scene with characters. A list of objects is visible on the left: 'this (a.k.a. scene)', 'this.camera', 'this.grassyGround', 'this.sunLight', 'this.cafeteria', 'this.kristen', 'this.philip', and 'this.lunchLady'. A 'Run...' button is present. Below the scene is an 'edit scene' button with a blue arrow icon.
- Procedures Panel (Bottom Left):** A list of procedures for the 'this.philip' object, including 'PlugEars', 'GalleryPerson', and 'Person' methods like 'keepTouching', 'touch', 'walk', 'walkOffscreen', 'walkTo', and 'fallDown'.
- Scripting Area (Right):** A block-based programming environment for the 'run' procedure. The code is as follows:


```

run
  AttemptToBrainwash_lg
  kiss
  declare procedure run
    current instance: this
    do in order
      this.camera showTitle /Cafeteria Romance, duration: 2.0 more
      this.lunchLady say /Crazy kids everywhere! more
      this.lunchLady leaveOverwhelmed
      this.philip say /Kristen, we are finally alone... more
      this.philip kiss whoToKiss: this.kristen
      this.lunchLady say /What are you DOING? more
      this.philip turnToFace this.lunchLady more
    do together
      this.lunchLady AttemptToBrainwash_lg whosHead: this.philip getPart Head
      do in order
        this.lunchLady delay 11.0
        this.lunchLady say /You do not like girls., duration: 2.0 more
      
```

An Independent Learning Process for End-user Programmers

```
graph LR; A[Search the Web for Examples] --> B[Evaluate Example Relevance]; B --> C[Select Relevant Code]; C --> D[Use in Personal Context]
```

Search the
Web for
Examples

Evaluate
Example
Relevance

Select
Relevant
Code

Use in
Personal
Context

Evaluation by Inexperienced End-Users

- Inexperienced end-users envision output



Evaluation by Inexperienced End-Users

Executable Program

```
graph TD; A[Executable Program] --> B[Graphical Output]; B --> C[Relevance Decision];
```

Graphical Output

Relevance Decision

Example Code Usability for Inexperienced End-Users

Evaluation

Example Programs/
Snippets

Does this exhibit
a feature I am
looking for?

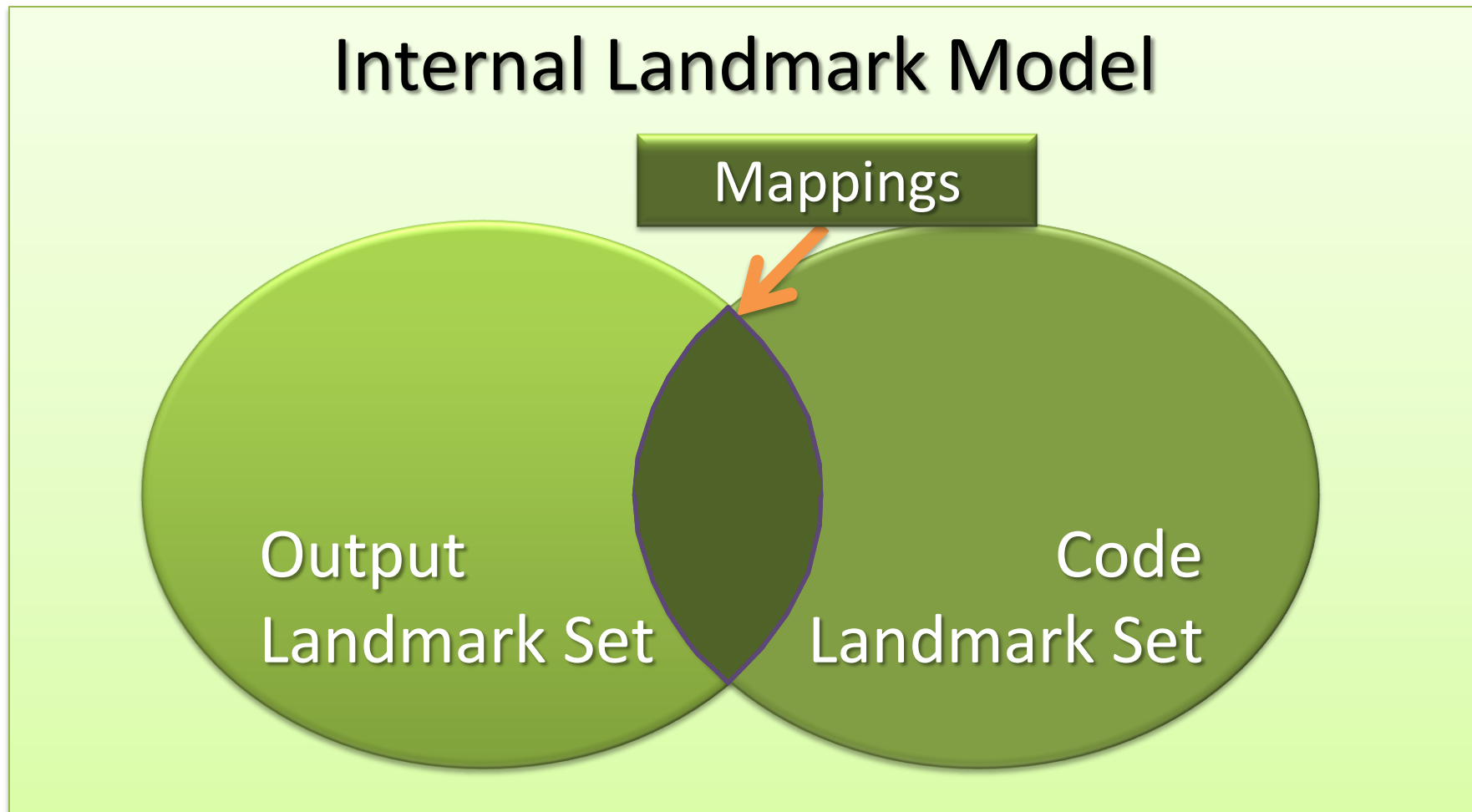
Read code,
descriptions,
execute
program

Selection

Relevant
Example

Where is the code
corresponding to the
feature I want?

Non-Programmers' Selection Search Model



Mapping Code to Output: Debuggers

The screenshot shows the Eclipse IDE in a debug state. The main editor window displays the following code from `SelectStatementPanel.java`:

```

@Override
protected JLabel createAwtComponent() {
    javax.swing.JLabel rv = new javax.swing.JLabel();
    int w = 175;
    int h = 20;
    rv.setOpaque(true);
    rv.setBackground(java.awt.Color.white);

    SwingUtilities.forceComponentDimensions(rv, w, h);

    return rv;
}

protected void refresh() {

```

The 'Variables' window on the right shows the following state:

Name	Value
this	SelectStatementPanel\$Stat...
rv	JLabel (id=111)
w	175
h	20

The 'Console' window at the bottom shows the following output:

```

IDE (1.6) (a) [Java Application] C:\Program Files\Java\jdk1.6.0_20\bin\javaw.exe (Oct 11, 2010 9:32:38 AM)
todo: RecentProjectsMenuModel handleMenuSelected
warning: createBufferedImageForUseAsColorBuffer size mismatch
project saved to: C:\Documents and Settings\grosspa\My Documents\LookingGlass\MyProjects\apprenticeAndFan.lgp
C:\Documents and Settings\grosspa\My Documents\Research\Alice\git_workspace\ide\generated\gallery\assets\edu.wustl.cse.lookingglass.ap

```

Hastings: an Output History Explorer Tool

What was happening at 10.41 seconds?

0.00 seconds 28.00 seconds

The scene at 10.41 s.

Actions in: MyScene.run()

6 do together
 (this.hoodeeny) takeADeepBreath
 (this.sam) lookAt (this.hoodeeny) more

7 do together
 (this.hoodeeny) sayTheMagicWords sayWordsTo: (this.lamp)
 (this.sam) lookAt (this.lamp) more

8 (this.lamp) turnGreen

9 (this.sam) delay 0.5

10 (this.camera) getCloseUpOf (this.sam), spatialRelation: FRONT_LEFT_OF, amount: 1.5, duration: 0.0

11 (this.camera) move BACKWARD, 1.0, duration: 0.0 more

12 (this.camera) move UP, 0.5, duration: 0.0 more

13 (this.camera) move LEFT, 1.0, duration: 0.0 more

14 (this.sam) delay 0.5

15 (this.sam) getPart Mouth turn FORWARD, 0.125, duration: 0.25 more

16 (this.sam) delay 1.0

17 (this.hoodeeny) straightenUp more

18 (this.cam) getPart Mouth turn BACKWARD, 0.125, duration: 0.25 more

zoom into DoTogether

zoom into DoTogether

explore turnGreen

Actions at 10.41 s.

- hoodeeny's Actions
 - in [hoodeeny.sayTheMagicWords\(lamp\)](#)
 - [hoodeeny.lookAt\(lamp\)](#)
- sam's Actions

Do these Tools
Adequately Assist
Non-Programmers
with Selection?

Evaluative Study: Hastings vs. a Debugger

The screenshot displays a software development environment with several key components:

- Executing Threads:** A panel on the left showing a tree view of threads. The current thread is `philip.kiss(Character whoToKiss)`, which is highlighted in green. It includes sub-threads for `scene.run()`, `Do Together [Thread-87]`, and `Do Together [Thread-88]`.
- Step Controls:** A panel below the threads with buttons for `Step into` (downward arrow) and `Step over` (curved arrow).
- Locals:** A table showing local variables for the current thread:

Local	Value
<code>whoToKiss</code>	<code>kristen</code>
- Breakpoints:** A panel at the bottom listing breakpoints: `MyScene.run: philip.say(String)`, `MyPhilip.kiss: Count up to 2`, and `MyScene.run: Do In Order`. A `Remove All Breakpoints` button is also present.
- Code Editor:** The main area shows a procedure named `kiss` with parameters `Character` and `whoToKiss`. The code includes:


```

            declare procedure kiss with parameter: Character whoToKiss
            do in order
            (this walkTo whoToKiss, spatialRelation: IN_FRONT_OF, offset: 0.0)
            do together
            (this touch whoToKiss, direction: RIGHT, touchpart: LEFT_HAND, offset: 0.0, duration: 1.0)
            (this touch whoToKiss, direction: LEFT, touchpart: RIGHT_HAND)
            count up to 2
            (this getPart Head) roll LEFT, 0.1
            (this getPart Head) roll RIGHT, 0.1
            loop
            
```

 A green box highlights the `do together` block.
- 3D Scene View:** A window on the right shows a 3D scene with a character (Philip) in a room. The character is positioned near a table and a chair. The scene is rendered in a simple, blocky style.

Looking Glass Debugger Threads

Looking Glass Program Code

run

declare procedure **run** current instance: **this**

do together

- this.hoodeeny** takeADeepBreath
- this.sam** lookAt **this.hoodeeny** more

run takeADeepBreath

declare procedure **takeADeepBreath** Add Parameter...

do together

- this** getPart **LeftUpperArm** move **UP**, **0.1** more
- this** look **UP**, amount: **0.1** more

Thread Presentation

Debug Hierarchy

D [Java Application]

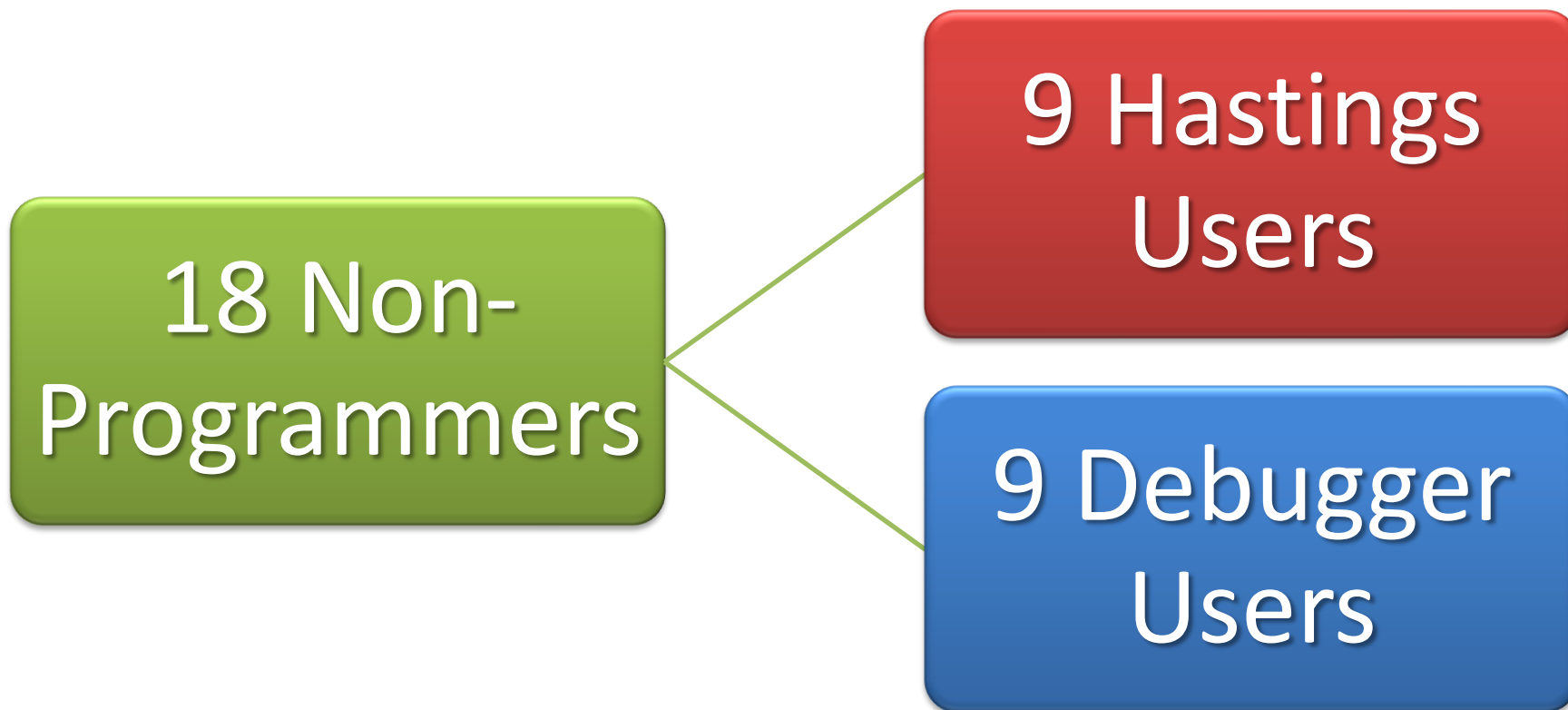
- edu.wustl.cse.lookingglass.ide.DebuggerUserStu
 - Thread [AWT-Shutdown] (Running)
 - Daemon Thread [AWT-Windows] (Running)
 - Thread [DestroyJavaVM] (Running)
 - Thread [AWT-EventQueue-0] (Running)
 - Thread [Thread-5] (Running)
 - Thread [Thread-7] (Running)
 - Thread [DoTogether-169] (Running)**
 - Thread [DoTogether-168] (Running)
 - Thread [DoTogether-170] (Running)
 - Thread [DoTogether-171] (Running)

Executing Threads

- RunThread [Thread-11]
 - scene.run()
 - Do Together [Thread-284]
 - hoodeeny.takeADeepBreath()
 - Do Together [Thread-287]**
 - Do Together [Thread-285]

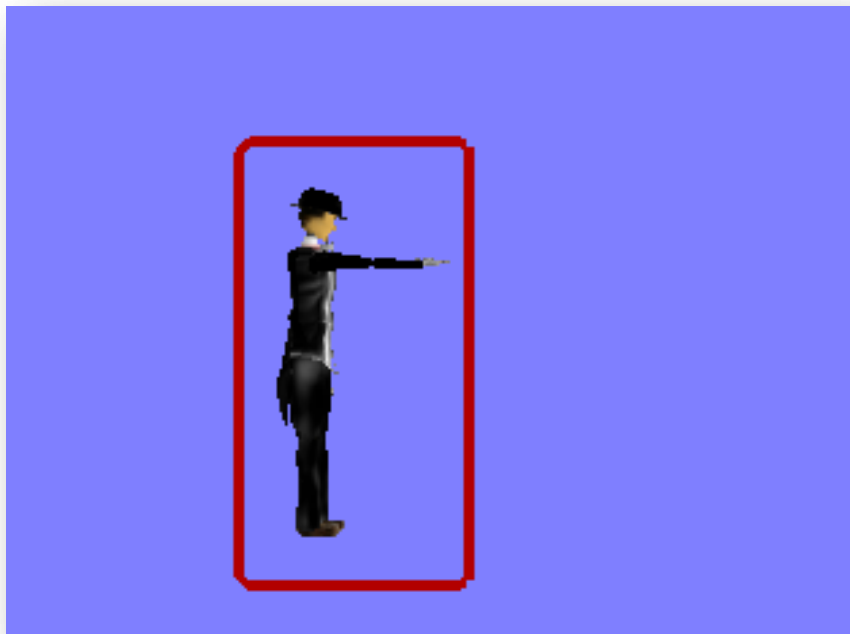
▶ || all ▶ all

Evaluative Study: Hastings vs. a Debugger





Evaluative Study: Tasks



- Find and mark code corresponding to graphical output




Drag to first statement: **First Statement Marker**

 Locate Marker  Clear Marker

Drag to last statement: **Last Statement Marker**

 Locate Marker  Clear Marker

 Submit Solution

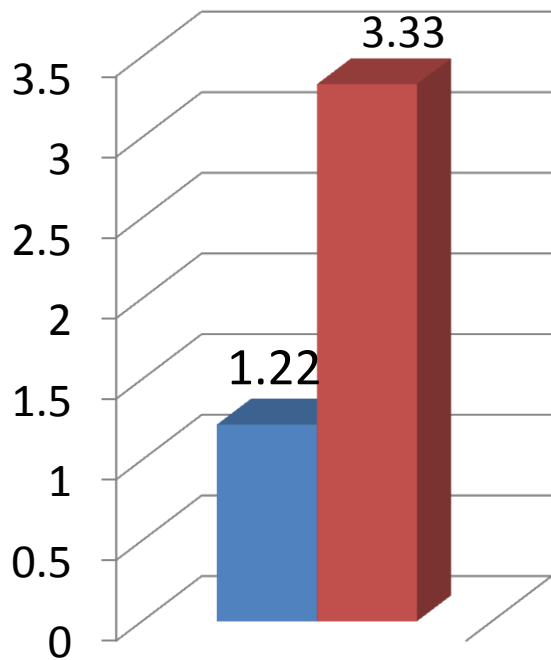
Do these Tools Adequately Assist Non-Programmers with Selection?

Quantitative Performance

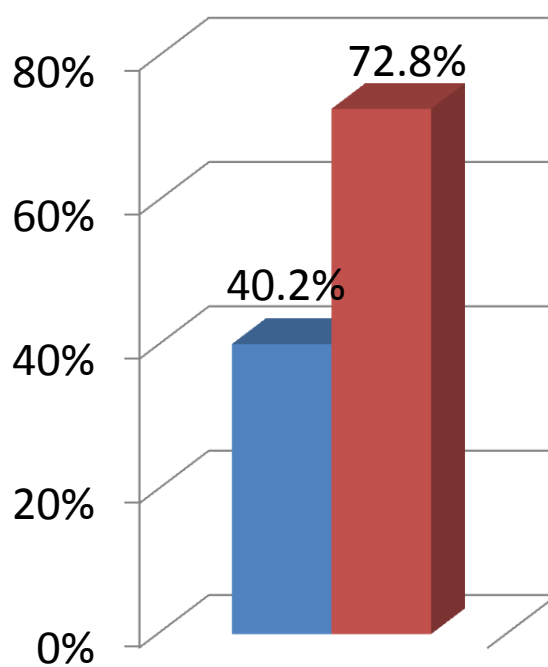
Qualitative Barriers

Quantitative Performance

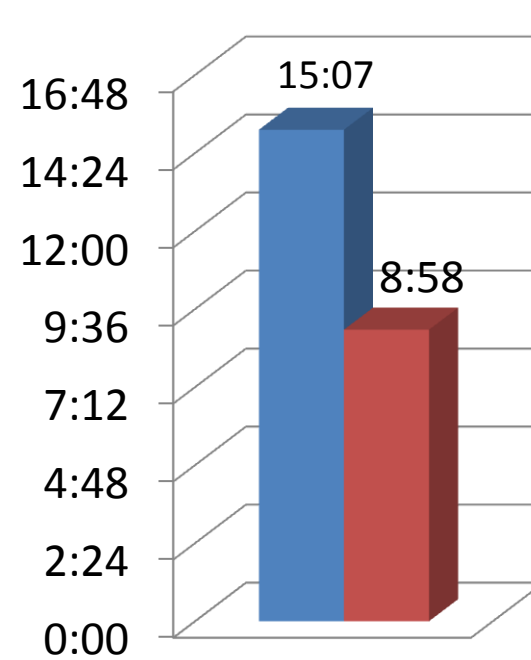
Average Tasks Completed
p < .01



Percentage of Correct Tasks
p < .05



Average Time on Task
p = .06



■ Debugger ■ Hastings

Qualitative Barriers

Debugger Barriers

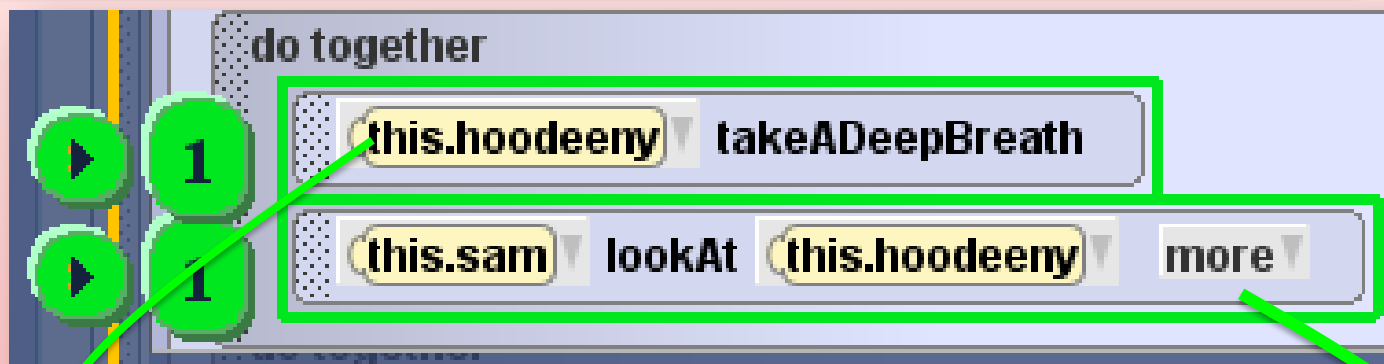
- Thread Misunderstanding
- Step Over Misinterpretation

Hastings Barriers

- Concurrency Ambiguity
- Playback Fidelity

A Qualitative Barrier

Concurrency Ambiguity [Hastings]



Do these Tools
Adequately Assist Non-
Programmers with
Selection?

Room for improvement

Future Design Guidelines

Do not rely on execution abstractions

Provide affordances for replaying code elements

Use direct code interactions

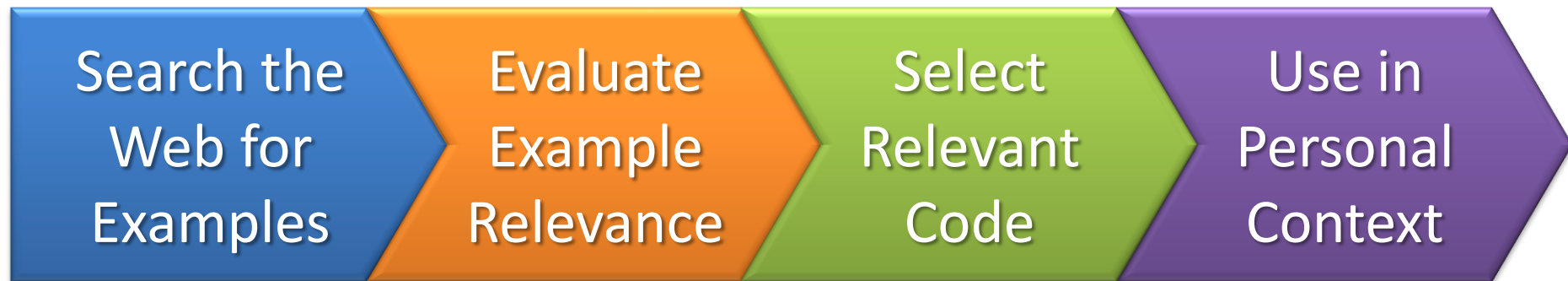
An Independent Learning Process for End-user Programmers

Search the
Web for
Examples

Evaluate
Example
Relevance

Select
Relevant
Code

Use in
Personal
Context



Further Challenges

Search the
Web for
Examples

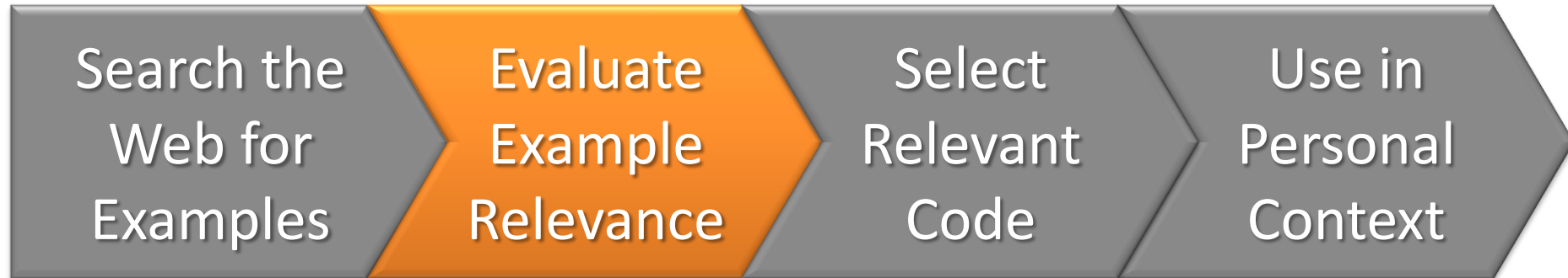
Evaluate
Example
Relevance

Select
Relevant
Code

Use in
Personal
Context

- Searching with inexperienced end-users' evaluative vocabulary

Further Challenges



- Supporting efficient evaluation of many search results
- Enabling snippet evaluation

Further Challenges

Search the
Web for
Examples

Evaluate
Example
Relevance

Select
Relevant
Code

Use in
Personal
Context

- Supporting complex selection of code areas

Further Challenges

Search the
Web for
Examples

Evaluate
Example
Relevance

Select
Relevant
Code

Use in
Personal
Context

- Promoting learning from personally motivating examples
- Assisting with reuse and adaptation

Questions?

- {grosspa, ckelleher}@cse.wustl.edu
- <http://lookingglass.wustl.edu>

References

- [1] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer. *Two studies of opportunistic programming: interleaving web foraging, learning, and writing code*. In Proc. of CHI, pages 1589–1598, 2009.
- [2] M. B. Rosson, J. Ballin, and J. Rode. *Who, what, and how: A survey of informal and professional web developers*. In Proc. of VL/HCC, pages 199–206, 2005.
- [3] P. Gross and C. Kelleher. *Non-programmers identifying functionality in unfamiliar code: strategies and barriers*. JVLC, 21 (5):263–276, Aug. 2010.