

A Small Matter of Programming

- Software engineering is a highly complex task, a microcosm of many challenges in HCI
- Making software engineering more accessible could empower millions to customize applications and write programs

Research agenda: HCI and programming

- Understand the challenges in programming
- Design more effective software engineering interfaces
- Aid novices in learning to program or writing programs
- Abstract best practices into toolkits

Understanding programmers and programming

Information Needs in Programming

[Ko, DeLine and Venolia , ICSE '07]

- Observed 17 developers in 90-minute sessions and transcribed all activities

Information Needs in Programming

[Ko, DeLine and Venolia, ICSE '07]

- Observed 17 developers in 90-minute sessions and transcribed all activities
- Thematic coding of information needs
 - **Writing code** e.g., how do I use this method?
 - **Submitting a change** e.g., which files are included?
 - **Triaging bugs** e.g., is the problem worth fixing?
 - **Reproducing failure** e.g., what are failure conditions?
 - **Understanding execution** e.g., what caused this behavior?
 - **Design** e.g., why is the code implemented this way?
 - **Awareness** e.g., what are my collaborators working on?

Information Needs in Programming

[Ko, DeLine and Venolia, ICSE '07]

- Observed 17 developers in 90-minute sessions and transcribed all activities
- Thematic coding of information needs
 - **Writing code** e.g., how do I use this method?
 - **Submitting a change** e.g., which files are included?
 - **Triaging bugs** e.g., is the problem worth fixing?
 - **Reproducing failure** e.g., what are failure conditions?
 - **Understanding execution** e.g., what caused this behavior?
 - **Design** e.g., why is the code implemented this way?
 - **Awareness** e.g., what are my collaborators working on?
- Most common need: collaborator awareness

Obstacles to learning APIs

[Robillard and DeLine, Empir. Software Engineering 2011]

- Survey and in-person interviews, combined reaching 440 professional software engineers

Obstacles to learning APIs

[Robillard and DeLine, Empir. Software Engineering 2011]

- Survey and in-person interviews, combined reaching 440 professional software engineers
- Biggest challenge: inadequate documentation

Obstacles to learning APIs

[Robillard and DeLine, Empir. Software Engineering 2011]

- Survey and in-person interviews, combined reaching 440 professional software engineers
- Biggest challenge: inadequate documentation
 - **API intent:** how it was intended to be used
“Nowhere in there does it say, and we intended to be used for a few graphics of small size because the memory footprint is going to be this.”

Obstacles to learning APIs

[Robillard and DeLine, Empir. Software Engineering 2011]

- Survey and in-person interviews, combined reaching 440 professional software engineers
- Biggest challenge: inadequate documentation
 - **API intent:** how it was intended to be used
“Nowhere in there does it say, and we intended to be used for a few graphics of small size because the memory footprint is going to be this.”
 - **Code examples:** snippets, tutorials, working apps

Obstacles to learning APIs

[Robillard and DeLine, Empir. Software Engineering 2011]

- Survey and in-person interviews, combined reaching 440 professional software engineers
- Biggest challenge: inadequate documentation
 - **API intent:** how it was intended to be used
“Nowhere in there does it say, and we intended to be used for a few graphics of small size because the memory footprint is going to be this.”
 - **Code examples:** snippets, tutorials, working apps
 - **Penetrability:** how much detail and implementation to expose?

Web foraging and programming

[Brandt et al., CHI '09]

- Laboratory study: ask programmers to implement a chat room in PHP
- Programmers make heavy use of the web
 - **JIT learning** of new skills
 - **Clarifying** existing skills
 - **Reminding** themselves of details
- Average participant spent **19%** of their programming time on the web

Software engineering interfaces

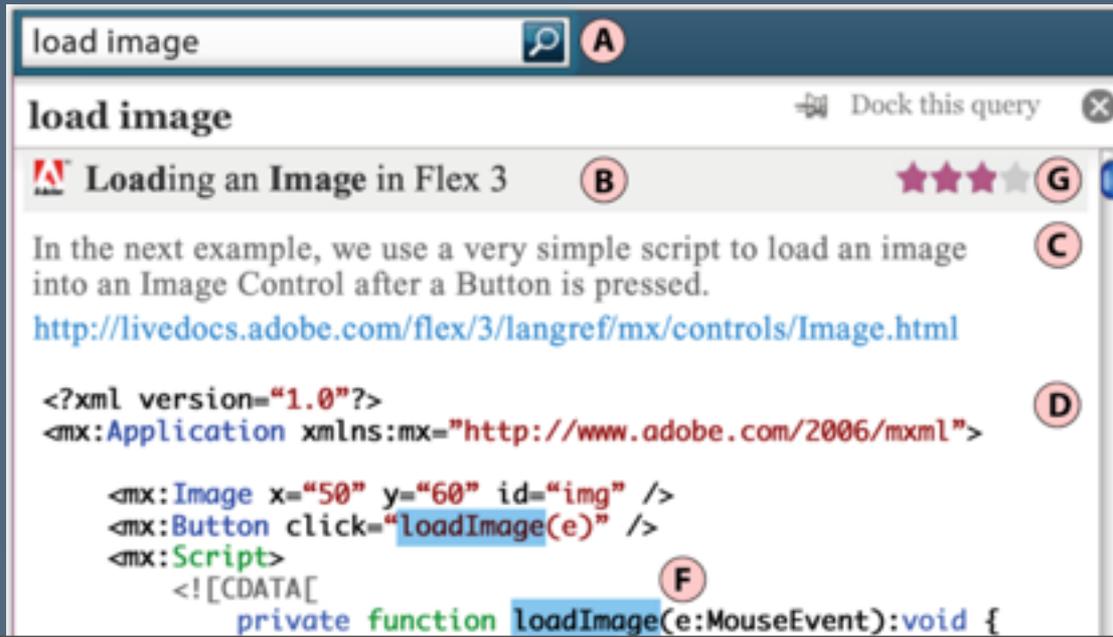
Goals of software engineering interface research

- Design a better toolbench, produce a better programmer
- Typically assume that the programming language is static, but the interface of the IDE can be molded

Example-centric programming

[Brandt et al., CHI '10]

- Close the loop between the development environment and web search
- Autocomplete code via web examples
- [video]



Asking 'why' questions of code

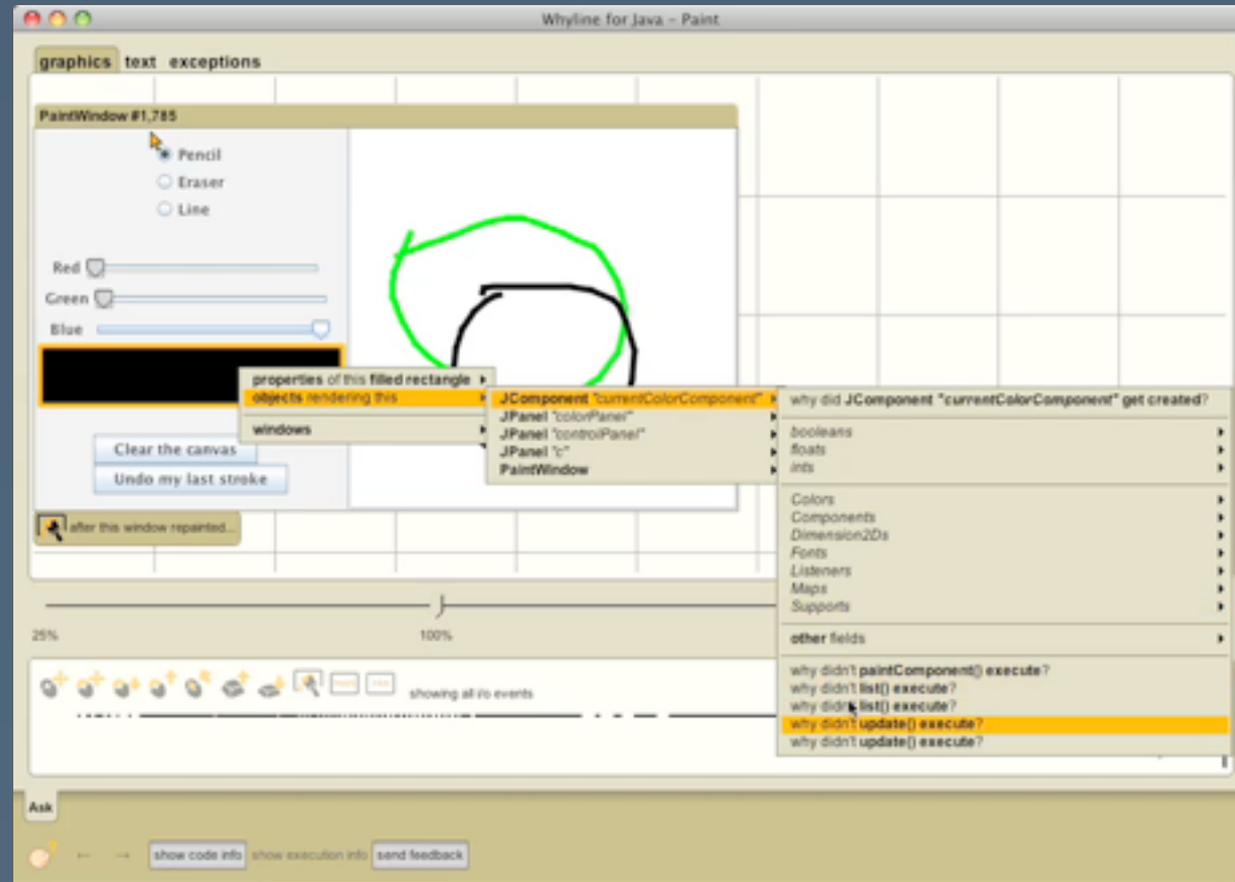
[Ko and Myers CHI '04, ICSE '09]

- Debugging problems often reduce to “why” questions
- Analyze program traces to answer them

Asking 'why' questions of code

[Ko and Myers CHI '04, ICSE '09]

- Debugging problems often reduce to “why” questions
- Analyze program traces to answer them



Whyline for Java

Missing user-facing feedback

[Ko and Zhang, CHI '11]

- Usability heuristic: all user inputs should produce some form of feedback
- Statically analyze code to identify user inputs that produce no feedback

Feedlack!

project **Calculator**

Feedlack found **54** places in your code that appear to be missing feedback:

`nd()` at `overlib.js` 927 **may not produce feedback**

`script()` at `Calculator.html` 90 **may not produce feedback**

`func(f)` at `newcalc.js` 919

```
602         'return overlib('Sets the t
603         onmouseout='nd()';'
604         onmousedown=
605         'if(base==10){topbar.trigme
606         style='cursor: default'>Ra
607         type='radio'
```

`nd()` at `overlib.js` 927

Keyword programming

[Little and Miller, UIST '06, ASE '09]

- Many applications have macro interfaces, but they are difficult to learn
- Allow keyword search over an API: e.g., `click` search button or `left margin 2 inches`

```
public List<String> getLines(BufferedReader src) throws Exception {  
    List<String> array = new ArrayList<String>();  
    while (src.ready()) {  
        add line  
    }  
    return array;  
}
```

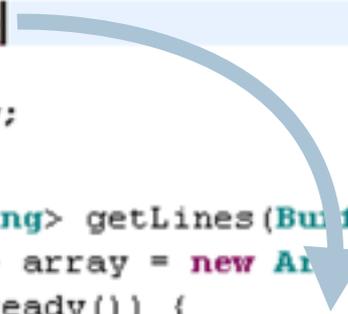
Keyword programming

[Little and Miller, UIST '06, ASE '09]

- Many applications have macro interfaces, but they are difficult to learn
- Allow keyword search over an API: e.g., click search button or left margin 2 inches

```
public List<String> getLines(BufferedReader src) throws Exception {
    List<String> array = new ArrayList<String>();
    while (src.ready()) {
        add line
    }
    return array;
}

public List<String> getLines(BufferedReader src) throws Exception {
    List<String> array = new ArrayList<String>();
    while (src.ready()) {
        array.add(src.readLine());
    }
    return array;
}
```



Visual layout of code snippets

[Bragdon et al., CHI '10]

- Most engineering time is spent navigating across multiple related code snippets
- So, design for many small windows into files

Visual layout of code snippets

[Bragdon et al., CHI '10]

The image shows a screenshot of an IDE window titled "ShapeDraw - Code Snippets". The main area displays a visual layout of code snippets for a Java Swing application. The snippets are arranged in a hierarchical structure, with lines indicating dependencies or relationships between them. The snippets are:

- Snippet 1 (Leftmost, largest):** Contains the main method and initialization code for the application. It includes calls to `setLayout()`, `createPropertyButtons()`, `randomShapes()`, `generateStatisticsMessages()`, `handleStatisticsOutput()`, `createShapeButtons()`, `createHandedShapeButton()`, `createMoreFunctionsPanel()`, and `createMenuBar()`.
- Snippet 2 (Middle-left):** Contains the `createPropertyButtons()` method, which creates a `DeleteShape` button and sets its focusable property to `false`.
- Snippet 3 (Middle-right):** Contains the `createMenuBar()` method, which creates a `Dropdown` button and sets its focusable property to `false`.
- Snippet 4 (Rightmost):** Contains the `TextMenuButton` class, which is a `MenuItem` that sets its text to "Text Input".

The snippets are connected by lines, indicating that the main method depends on the other three, and the `createMenuBar()` method depends on the `TextMenuButton` class. The IDE interface includes a menu bar at the top with "Menu", "ShapeDraw", and "Code Snippets". A toolbar on the right contains "Debug", "Test", and "Run" buttons. A sidebar on the right shows the "ShapeDraw" project structure.

Learning programming

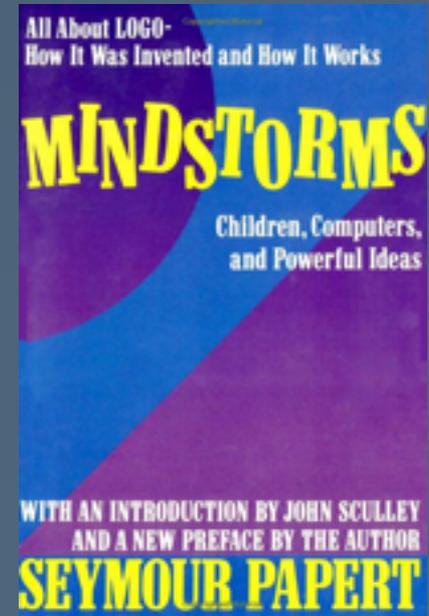
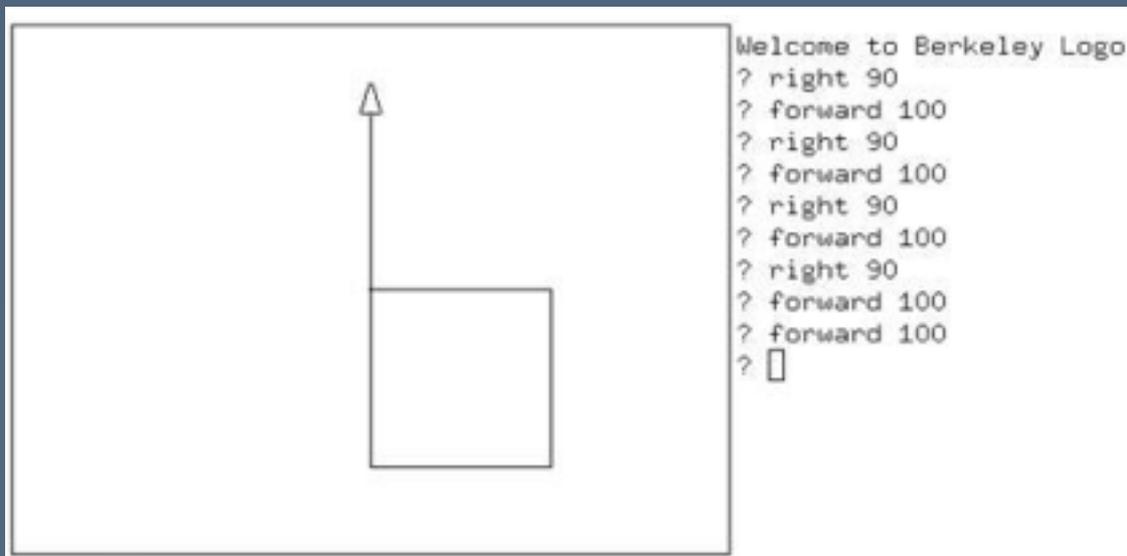
Goals of programming education

- Make programming accessible to new populations: children, scripters, interested amateurs
- Tools and innovations depend on the population

Logo: programming for children

[Papert '93]

- **Constructionist** learning: learning happens most effectively when people are making tangible objects
- Lego Mindstorms followed this mold and was named after it



Alice: 3D animations for intro programming

[Cooper, Dann, and Pausch CCSC '00]

- Visualization and animation help new programmers understand program execution
- Evolution of Logo and Karel the Robot into a three-dimensional environment



Scratch: kids remix and create

[Resnick et al., CACM '09]

Scratch: kids remix and create

[Resnick et al., CACM '09]

The screenshot displays the Scratch programming environment. The main stage shows a scene with a tree, a blue bird, a brown owl, and a yellow mouse. The interface includes a menu bar (File, Edit, Tips), a toolbar with navigation and save icons, and a top right area with 'Want to save? Click remix' and 'ScratchTeam' options. The 'Scripts' panel on the left lists categories: Motion, Looks, Sound, Pen, Data, Events, Control, Sensing, and Operators. A 'Make a Block' button is visible, and a 'jump' block is shown in the 'Operators' category. The code editor on the right contains the following script:

```
when green flag clicked
  go to x: 150 y: 100
  forever loop
    jump 100

define jump height
  change y by height
  wait 0.3 secs
  change y by 10 - height
  wait 0.3 secs
```

The 'Sprites' panel at the bottom left shows a collection of sprites: Stage (1 backdrop), Sprites (5), and a 'New backdrop:' section with a 'New sprite:' section. The 'New sprite:' section shows a 'New sprite:' button and a 'New sprite:' button. The 'Sprites' panel also shows a 'New sprite:' button and a 'New sprite:' button.

Programming by demonstration

Goals of PBD

- Teach a computer to program simply by demonstrating what should be done
- Challenges
 - There is an infinite, and hugely branching, space of programs that might be inferred
 - Inferred macros can be extremely brittle

Recall: EAGER

[Cypher, CHI '91]

- Infer a macro by watching the user's behavior

Creating a Subject List

A user has a stack of message cards (a) and wants to make a list of the subjects of the messages. The user copies the subject from the first message, goes to the ****Subject List**** card, types "1.", and pastes in the first subject (b). The user then goes to the second message, copies its subject, and adds it to the Subject List.

At this point, the Eager icon pops up (c), since Eager has detected a pattern in the user's actions. Eager highlights the right-arrow button in green (c), since it anticipates that the user will click here next. Eager continues anticipating that the user will navigate to the third message, select (d) and copy its subject, go to the Subject List, type "3." (e) and then paste in the subject (f).

The user is now confident that Eager knows what to do, and clicks on the Eager icon. It completes the task automatically (g).

(a) (b) (c) (d) (e) (f) (g)

Simultaneous structured editing

[Miller and Myers, USENIX '01]

- Utilize lightweight structure in text
- Today, versions of this exist in Sublime Text

Simultaneous structured editing

[Miller and Myers, USENIX '01]

- Utilize lightweight structure in text
- Today, versions of this exist in Sublime Text

columns by similarity. As you edit a line, your changes are applied to the other lines in the same column in a similar way. Click the link icon on that line to unlink it.

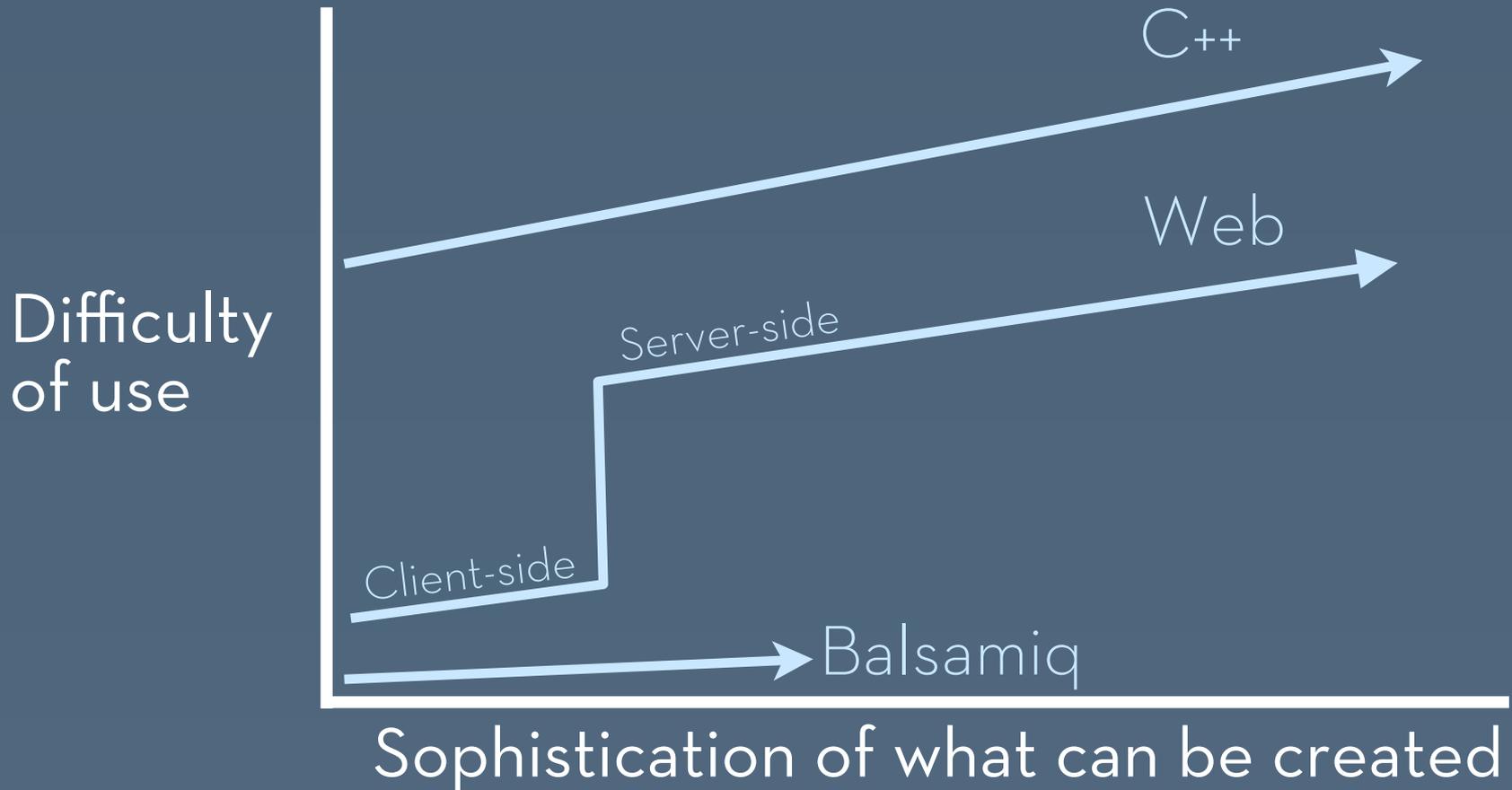
[Screenshot for a demonstration.](#)

To Uppercase	To Lowercase	Fill
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.ForeachCmd",</code>	<code>"foreach");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.ListCmd",</code>	<code>"list");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.SocketCmd",</code>	<code>"socket");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.TellCmd",</code>	<code>"tell");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.ScanCmd",</code>	<code>"scan");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.FileCmd",</code>	<code>"file");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.LindexCmd",</code>	<code>"lindex");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.SubstCmd",</code>	<code>"subst");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.BreakCmd",</code>	<code>"break");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.ContinueCmd",</code>	<code>"continue");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.LinsertCmd",</code>	<code>"linsert");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.LrangeCmd",</code>	<code>"lrange");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.SetCmd",</code>	<code>"set");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.ErrorCmd",</code>	<code>"error");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.ConcatCmd",</code>	<code>"concat");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.ExprCmd",</code>	<code>"expr");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.CloseCmd",</code>	<code>"close");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.PackageCmd",</code>	<code>"package");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.AppendCmd",</code>	<code>"append");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.ReadCmd",</code>	<code>"read");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.EvalCmd",</code>	<code>"eval");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.FormatCmd",</code>	<code>"format");</code>
<code>Extension.loadOnDemand(this,</code>	<code>"tcl.lang.LappendCmd",</code>	<code>"lappend");</code>

Toolkits

Recall: Threshold/Ceiling

[Myers, Hudson and Pausch, TOCHI 2000]



Research agenda: toolkits

- Crystallize and formalize a perspective on a difficult engineering problem
- If successful, shift the entire programming practice for the area

Sikuli: programming with screenshots

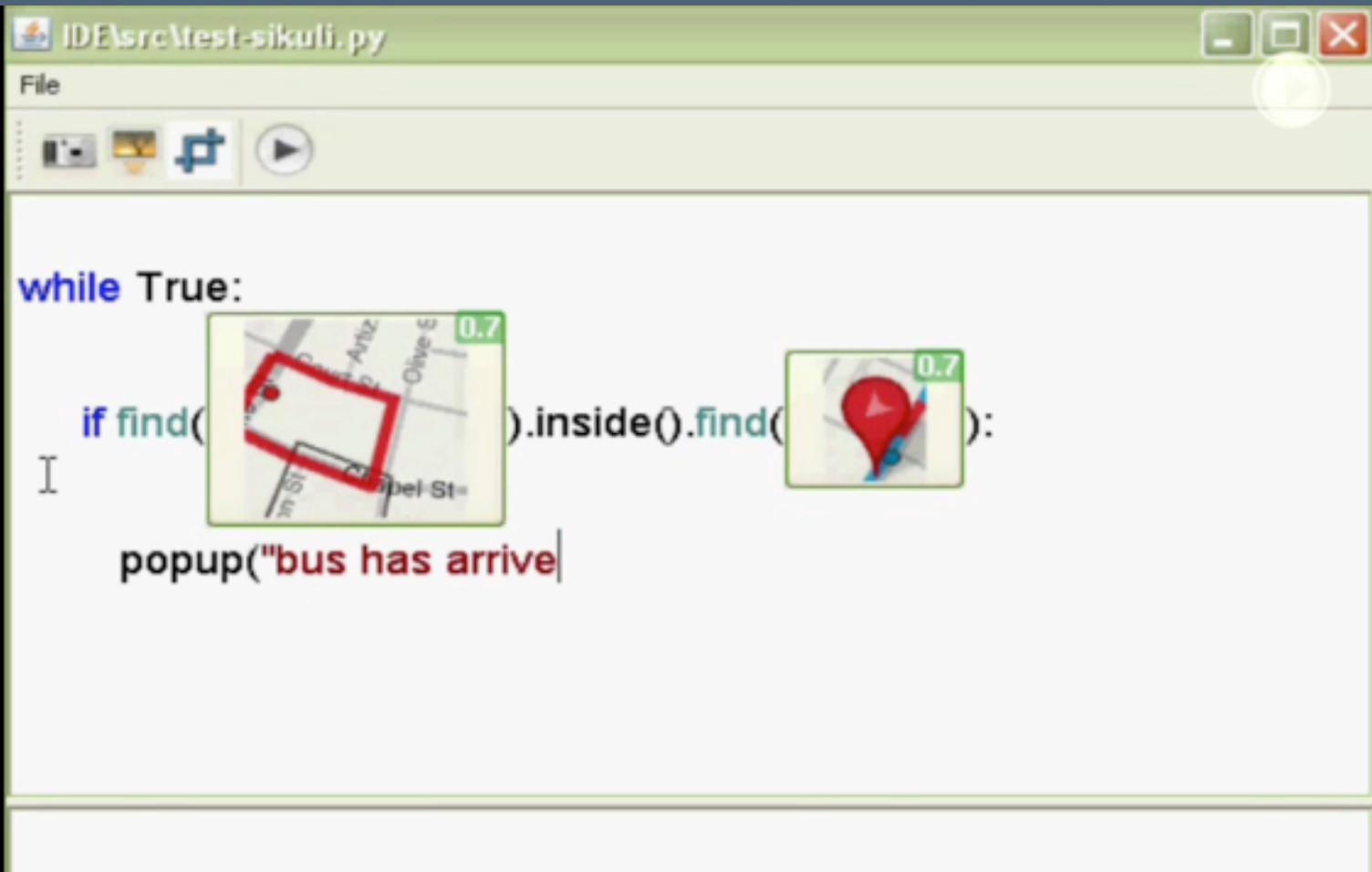
[Yeh, Chang, and Miller, UIST '09]

- Template search in desktop scripting

Sikuli: programming with screenshots

[Yeh, Chang, and Miller, UIST '09]

- Template search in desktop scripting



Recall: Chickenfoot

[Bolin et al., UIST '05]

- Client-side scripting, automation and customization of rendered web pages

```
isbn = find('number just after isbn')
with (fetch('libraries.mit.edu')) {
  pick('Keywords');
  enter(isbn)
  click('Search')
  link=find('link just after Location')
}
// back to Amazon
if (link.hasMatch) {
  insert(before('first rule after "Buying"'),
  link.html)
}
```



Research agenda: HCI and programming

- Understand the challenges in programming
- Design more effective software engineering interfaces
- Aid novices in learning to program or writing programs
- Abstract best practices into toolkits

Final Presentations

- Wednesday, June 12, 8:30am - 11:30am
- Wallenberg 124
- 4 minutes to present, 2 minutes for questions
 - Next group plugs in laptop during questions
- Make sure to test your slides in the room before presentation day
- Presentation order TBD