

Improving Software Development through Human-Centered Approaches

Brad A. Myers

Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University

<http://www.cs.cmu.edu/~bam>

bam@cs.cmu.edu



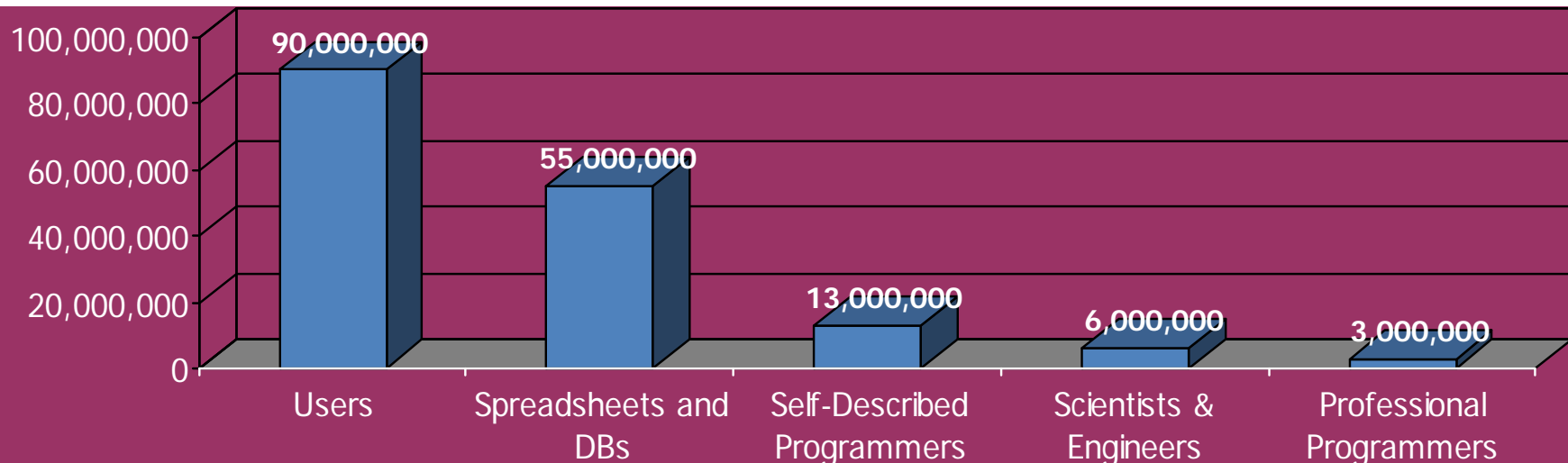
Natural Programming Project

- Researching better tools for programmers since 1978
- Natural Programming project started in 1995
- Make programming easier and more correct by making it more *natural*
 - Closer to the way that people think about algorithms and solving their tasks
- Methodology – human-centered approach
 - Perform *studies* to inform design
 - Provide new knowledge about what people do and think, & barriers
 - Guide the designs from the data
 - Design of programming *languages* and *environments*
 - Iteratively evaluate and improve the tools
- Target novice, expert and end-user programmers



End User Programming

- People whose primary job is *not* programming
- In 2012, in USA at work: — *Scaffidi, Shaw and Myers 2005*
 - 3 million professional programmers
 - 6 million scientists & engineers
 - 13 million will describe themselves as programmers
 - 55 million will use spreadsheets or databases at work (and therefore may potentially program)
 - 90 million computer users at work in US
- We should make better tools for all of these people!



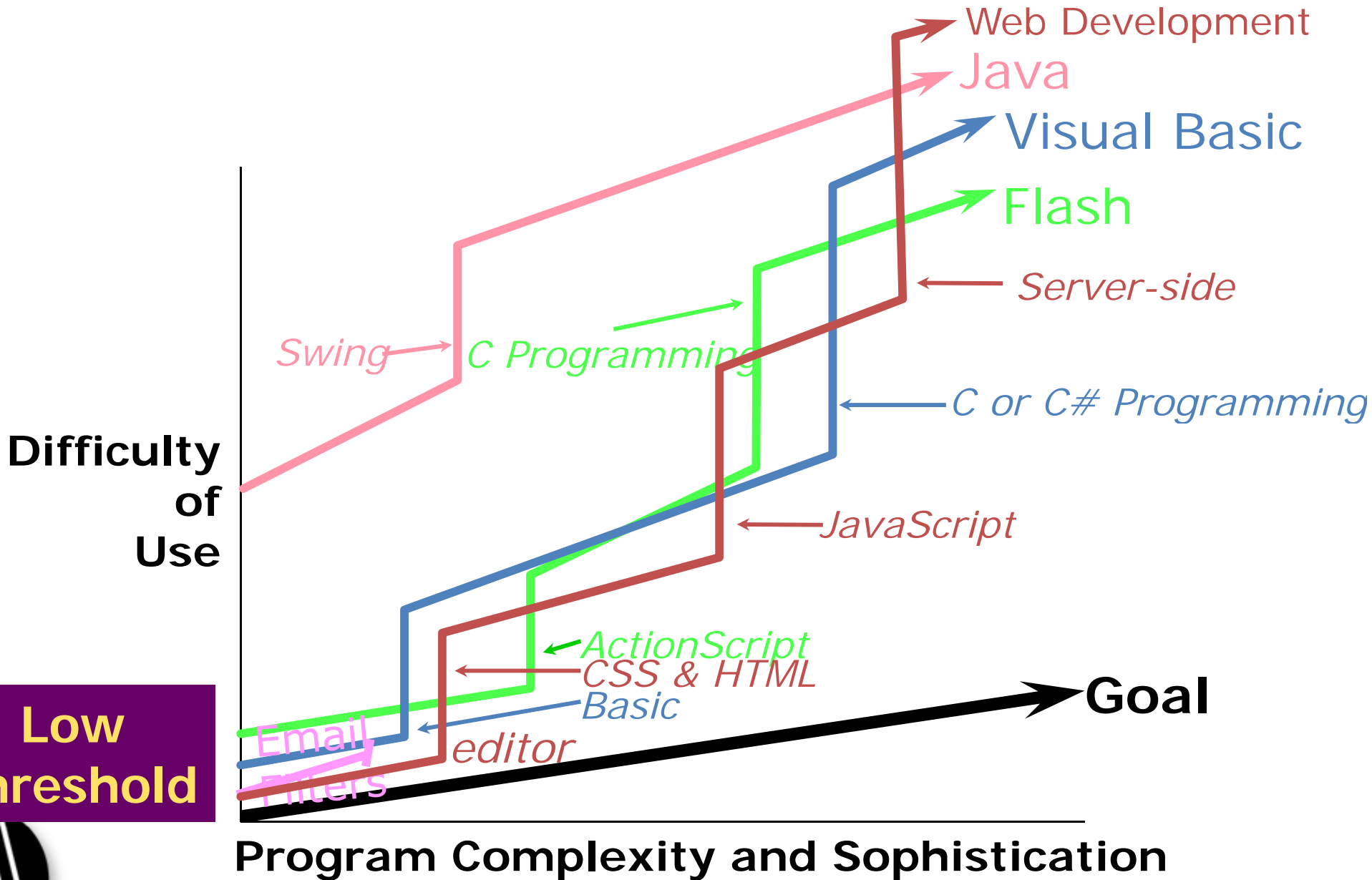
Debugging

- Study commissioned by NIST USA (2002) of 14 software vendors
 - Software errors cost ~\$60 billion annually
 - Software engineers spend 70-80% of time testing and debugging
 - Time for 1 developer to fix 1 bug was ~17.4 hours
- Current debugging techniques *same as for last 70 years*
 - Same for end-user and professional environments



Goal: Gentle Slope Systems

High Ceiling



Improve Developer Experience

- Use human centered approaches to:
 - Make developers *more effective*
 - *Reduce errors* in resulting code
 - Insure that developer tools are *useful*
 - Understand developers' *barriers* that cause *wasted time*
 - Direct efforts at *most important* issues
- Address: programming languages, APIs, tools, documentation & resources



Why Would Being Natural be Good?

- Programmers are People Too
 - Take the human into account
- Language should be close to user's plan
 - “Programming is the process of transforming a mental plan into one that is compatible with the computer.”
— *Jean-Michel Hoc*
- *Closeness of mapping*
 - “The closer the programming world is to the problem world, the easier the problem-solving ought to be.... Conventional textual languages are a long way from that goal.” — *Green and Petre*
- Depends on target population
 - Need studies



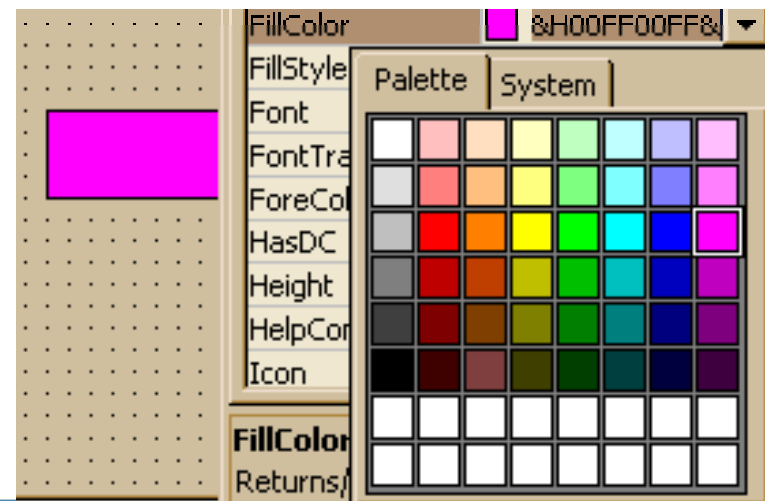
Not so Natural!

```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

- 3 kinds of parentheses and 9 special words!
- Compared to click and type: "Hello World!"

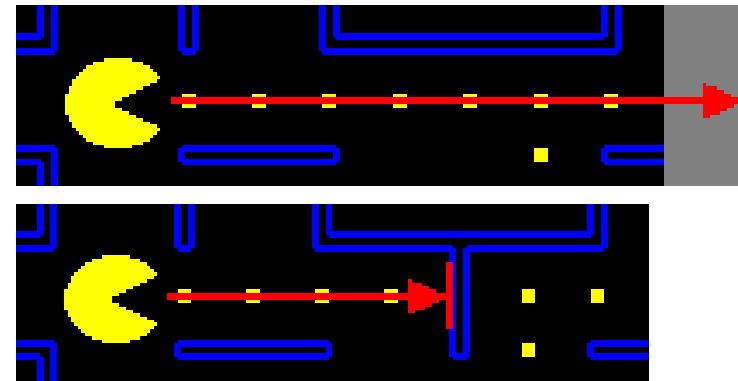


Let `Shape1.FillColor`
= `&H00FF00FF&`



First Natural Programming Studies

- John Pane, PhD 2002
- Studies:
 - How people *naturally* express programming concepts and algorithms
 - 1) Nine scenes from PacMan
 - 2) Transforming and calculating data in a spreadsheet
 - Specific issue of language design
 - 3) Selecting specific objects from a group (“and”, “or”, “not”)
 - Lots of interesting results



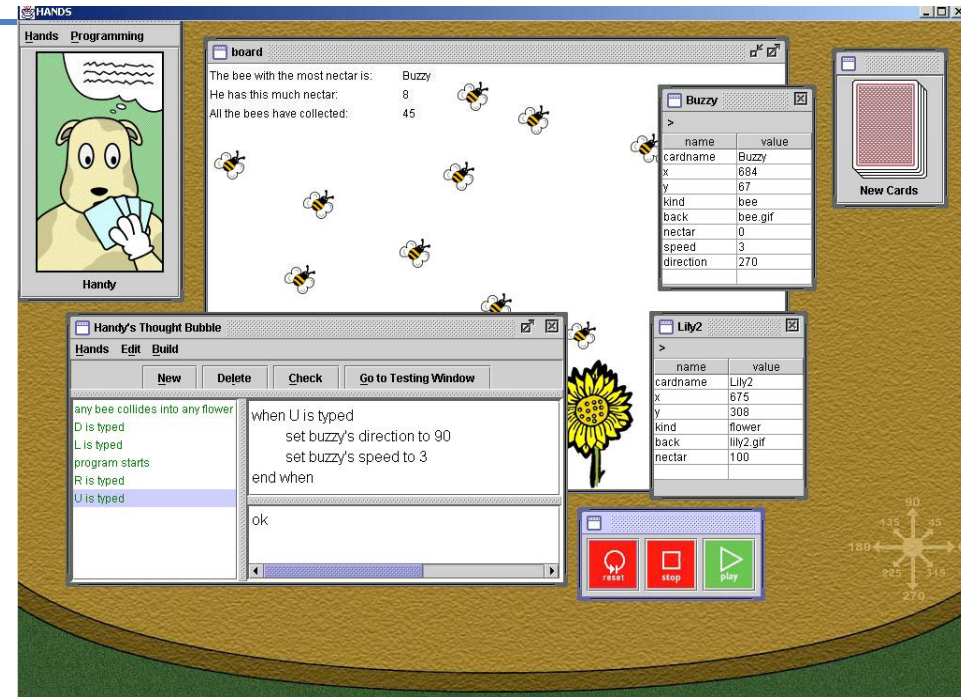
Examples of Results

- Rule-based style
 - *“If PacMan loses all his lives, its game over.”*
- “And”, “Or”, “Not” don’t match computer interpretation
 - ... men and women, ... (*not* an apple) or pear
- Operations suggest data as lists, not arrays
 - People don’t make space before inserting
- Objects normally moving
 - *“If PacMan hits a wall, he stops.”*
 - so objects remember their own state



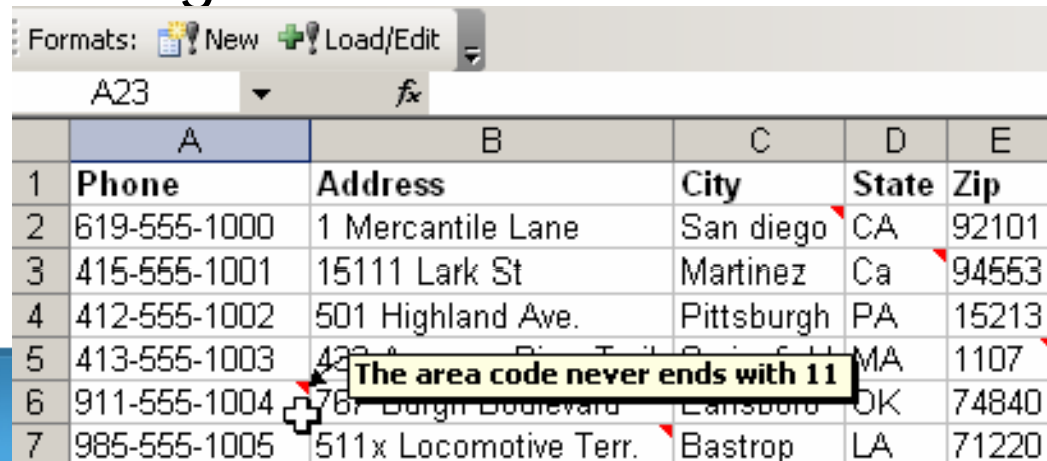
New Language and System: HANDS

- John Pane, PhD 2002
- Properties:
 - Metaphor of agent (Handy the dog) operating on cards
 - All operations can operate on single items or sets of items
 - Integrated queries with language
 - Sets can be dynamically constructed and used
 - “Set the speed of all bees to 0”
- See the video: <http://web.cs.cmu.edu/~pane/HANDS/HANDS.MPG>



Supporting “Natural” Data Types

- Chris Scaffidi, PhD 2009
- Ask users about types of data, say “Person name”, “age”, “date”, “Project code”, ...
- User-centered type system called “topes”
 - Structured
 - Constraints on the values and parts
 - May be “always” or “usually” true
 - “USA phone area code never ends in 11”
 - “USA Last names usually start with a capital letter”
- Library for verifying & transforming values
 - Can be used from JavaScript for web and from VB for Excel
- Editor for specifying



	A	B	C	D	E
1	Phone	Address	City	State	Zip
2	619-555-1000	1 Mercantile Lane	San diego	CA	92101
3	415-555-1001	15111 Lark St	Martinez	Ca	94553
4	412-555-1002	501 Highland Ave.	Pittsburgh	PA	15213
5	413-555-1003	425 A B...	...	MA	1107
6	911-555-1004	787 Burgin Boulevard	Lansboro	OK	74840
7	985-555-1005	511x Locomotive Terr.	Bastrop	LA	71220



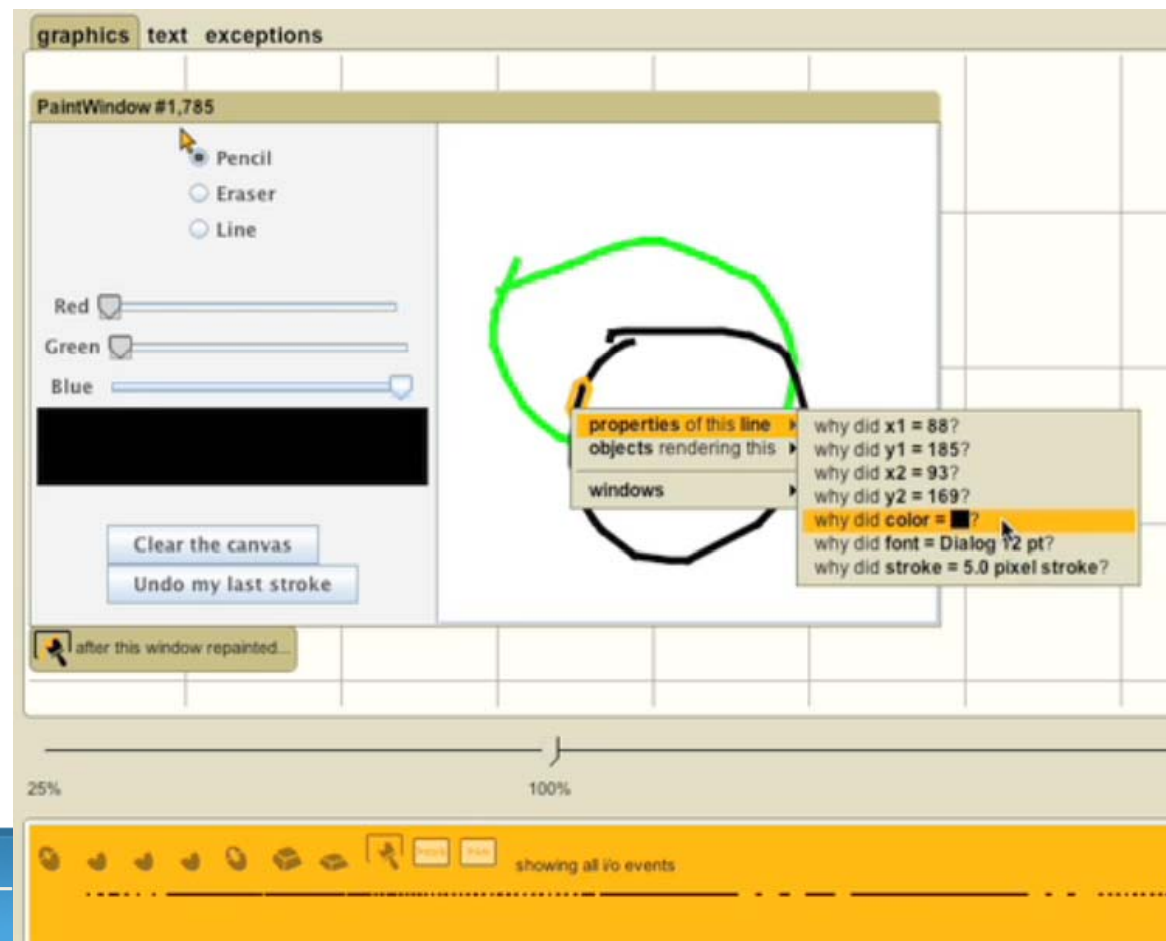
Study of Errors

- Study of novice errors and debugging
 - Created a new model of barriers & kinds of errors
 - All of the observed debugging problems could be addressed by “Why” questions
 - 32% were “Why did”; 68% were “Why didn’t”
- Current debugging techniques require user to *guess* where bug is or where to look
 - Most of initial guesses are *wrong*, even for experts



Whyline

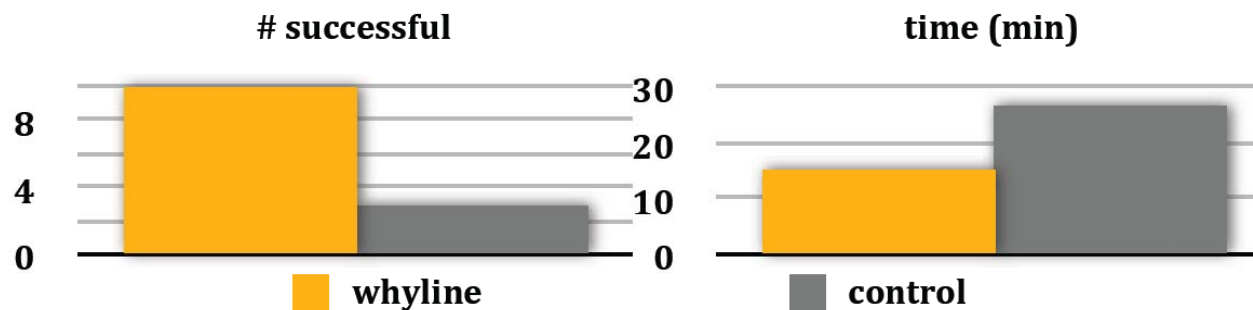
- Andy Ko, PhD 2008
- Allow users to directly ask “Why” and “Why not”



1:27

Whyline User Studies

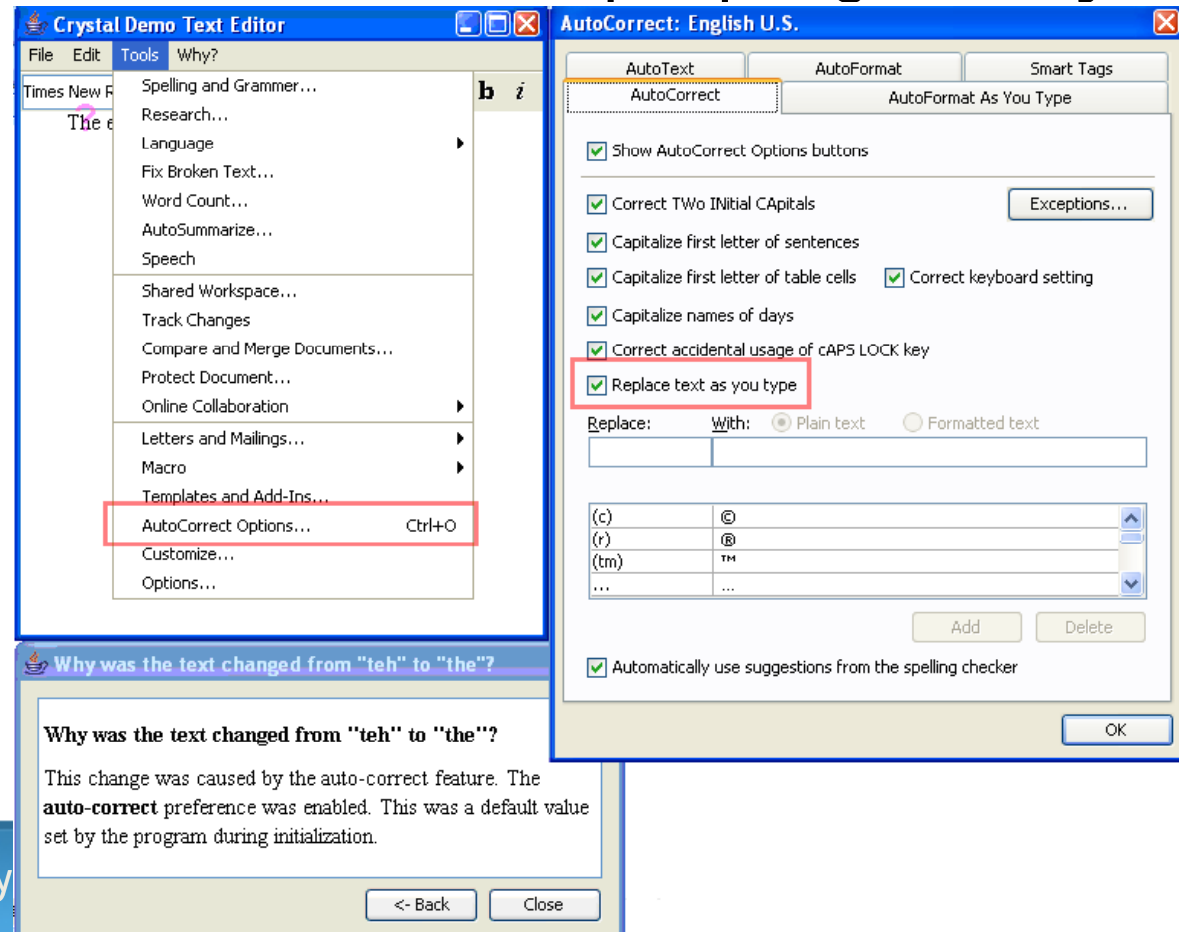
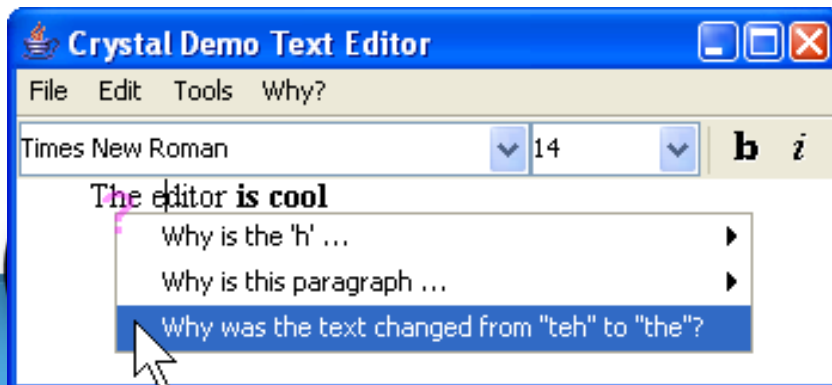
- Initial study:
 - Whyline with novices outperformed experts with Eclipse
 - Factor of **2.5** times faster
 - ($p < .05$, Wilcoxon rank sums test)
- Formal study:
 - Experts attempting 2 difficult tasks
 - Whyline over **3** times as successful, in **1/2** of the time



Crystal

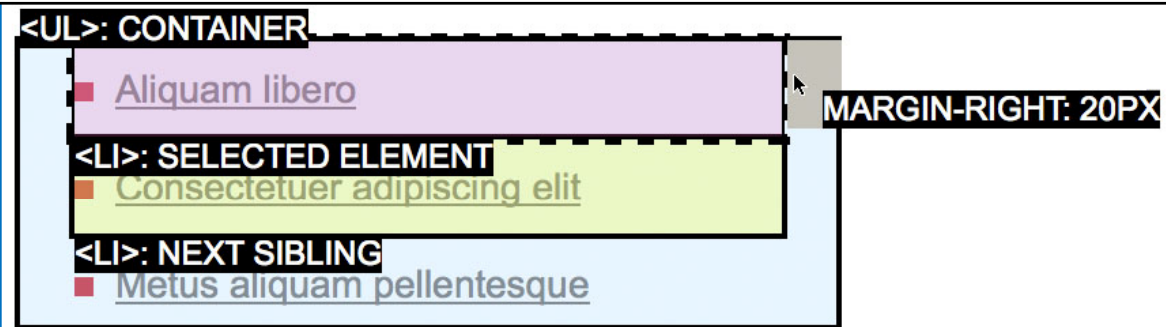


- **Crystal: Clarifications Regarding Your Software** using a **Toolkit, Architecture and Language**
- Apply WhyLine idea to regular desktop applications (Word 2003)
- Lots of complexity in powerful features that people generally like
- Ask “Why” about what recently happened
- Architecture: supports adding to application with small overhead

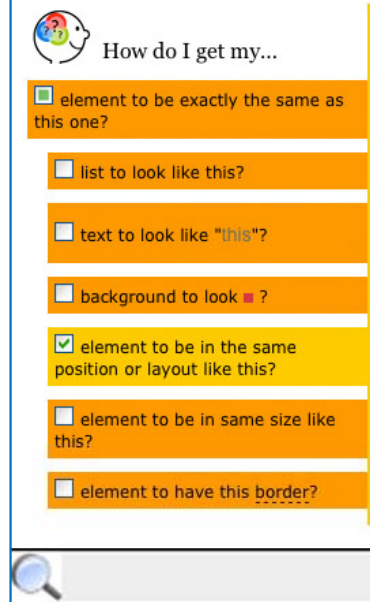


WebCrystal

- Investigate CSS and HTML responsible for example behaviors
- Navigate around HTML hierarchy
- Ask “how-do-I” questions about look, position and behavior
- Generates code in user-selected format
- Combine code for multiple elements
- CHI'2012



The diagram shows a list structure with three items. The first item, "Aliquam libero", is highlighted in purple and labeled as the "SELECTED ELEMENT". The second item, "Consectetur adipiscing elit", is highlighted in green and labeled as the "NEXT SIBLING". The entire list is enclosed in a light blue box labeled as the "CONTAINER". A callout box on the right indicates "MARGIN-RIGHT: 20PX" for the selected element.



The interface shows a search bar with the text "How do I get my...". Below it are several checkboxes for different types of questions:

- element to be exactly the same as this one?
- list to look like this?
- text to look like "this"?
- background to look like this?
- element to be in the same position or layout like this?
- element to be in same size like this?
- element to have this border?

The element is positioned like this because it is a in a list structure with respect to its container and its siblings. It uses `margin-left = 20px`, `margin-right = 20px`, `text-align = left`, and its default attributes.

- Give me an example of making my element use all these position attributes.
- Give me an example of making my margin-left = 20px.
- Give me an example of making my margin-right = 20px.
- Give me an example of making my text-align = left.

Sample Code in the **inline CSS** format:

Save this code for later use

```
<SPAN style='font-family: Arial,Helvetica,sans-serif; font-size: 46px; padding-bottom: 10px; padding-top: 12px;'>Your text.</SPAN>
```

Sample Code in the **separate CSS** format:

Save this code for later use

```
/*css*/  
SPAN.your_class {  
font-family: Arial,Helvetica,sans-serif;  
font-size: 46px;  
padding-bottom: 10px;  
padding-top: 12px;  
}  
/*html*/  
<SPAN class='your_class'>Your  
text.</SPAN>
```



Study of Design Requirements for Maintenance-Oriented IDEs

- Studied **expert** use of Java Eclipse IDE in a lab setting (2004-2006)
- Focus on day-to-day maintenance tasks such as bug repairs and feature enhancements
- Lab study with detailed analysis
- Rich dataset → multiple papers



A Programmer's Working Set

- A collection of task-relevant code fragments
- In modern software development, dependencies are distributed and non-local

```
class TestRunner {
public:
    TestRunner() {}
    TestRunner(int argc, char* argv[]) {
        m_argv = argv;
    }
    ~TestRunner() {}

    void Run() {
        // ...
    }
};

int main(int argc, char* argv[]) {
    TestRunner tr(argc, argv);
    tr.Run();
}
```

```
class TestRunner {
public:
    TestRunner() {}
    TestRunner(int argc, char* argv[]) {
        m_argv = argv;
    }
    ~TestRunner() {}

    void Run() {
        // ...
    }
};

int main(int argc, char* argv[]) {
    TestRunner tr(argc, argv);
    tr.Run();
}
```

```
class TestRunner {
public:
    TestRunner() {}
    TestRunner(int argc, char* argv[]) {
        m_argv = argv;
    }
    ~TestRunner() {}

    void Run() {
        // ...
    }
};

int main(int argc, char* argv[]) {
    TestRunner tr(argc, argv);
    tr.Run();
}
```



Times for Bottlenecks

- Each instance of an interactive bottleneck cost only a few seconds, but . . .

Interactive Bottleneck	Overall Cost
Navigating to fragment in <i>same</i> file (<i>via scrolling</i>)	~ 11 minutes
Navigating to fragment in <i>different</i> file (<i>via tabs and explorer</i>)	~ 7 minutes
Recovering working set after returning to a task	~ 1 minute
Total Costs	~19 minutes

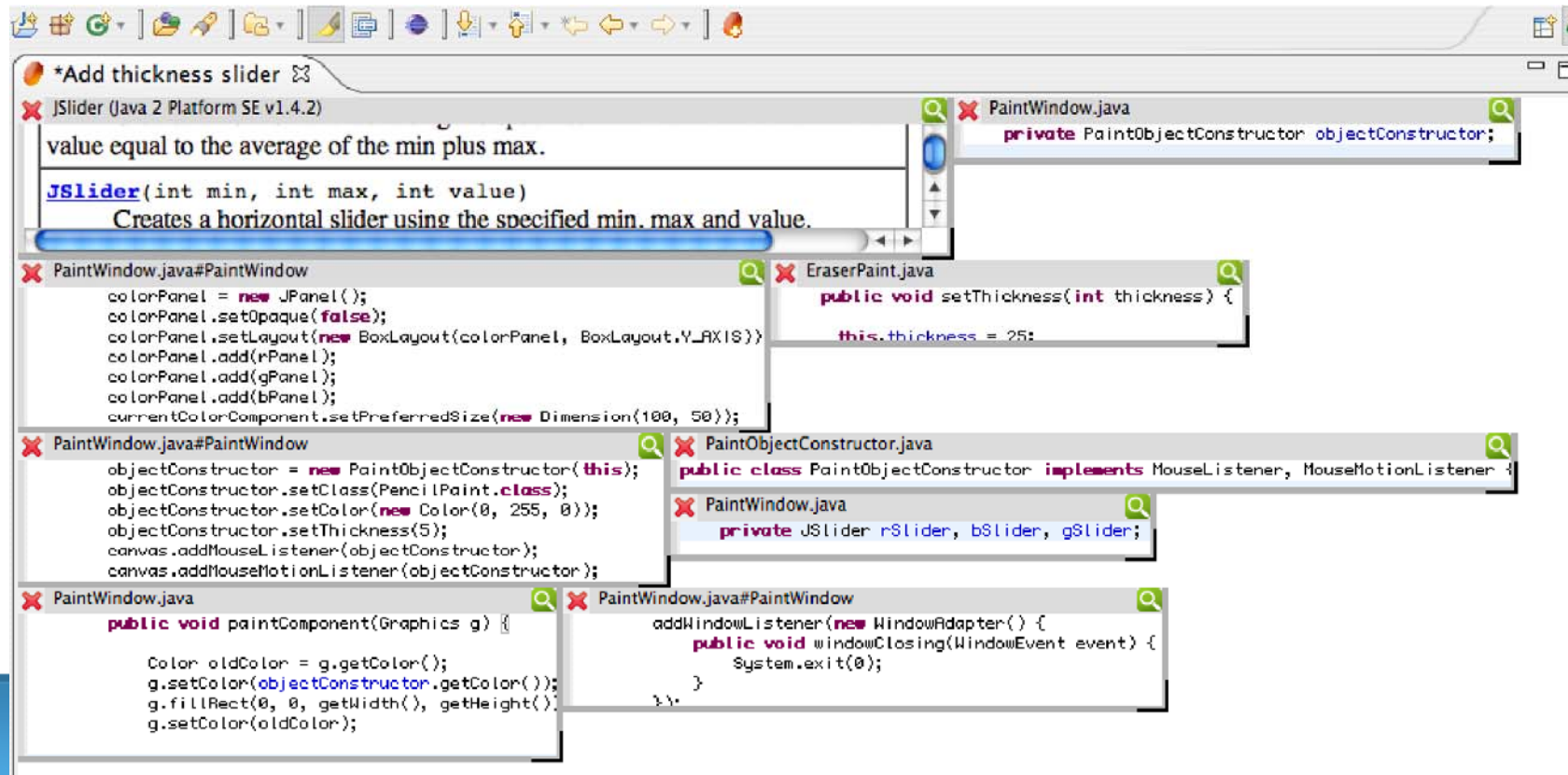
= **35%** of uninterrupted work time!



Jasper: Working Set Tool



- Jasper = Java Aid with Sets of Pertinent Elements for Recall
- Allow programmers to grab arbitrary fragments of code to represent working sets
 - Allow programmers to view in one place, one screen



Study of APIs

- Started as PhD work of Jeff Stylos, 2009
 - Inspired by Steven Clarke, Microsoft Visual Studio group
- **A**pplication **P**rogramming **I**nterface
 - Libraries, frameworks, SDKs, ...
- Which programming patterns are most usable?
- Barriers to use of APIs
- Measures: learnability, errors, preferences
- Expert and novice programmers
- Studied:
 - Default parameters in constructors
 - Factory pattern
 - Object design
 - SAP's Web Services APIs



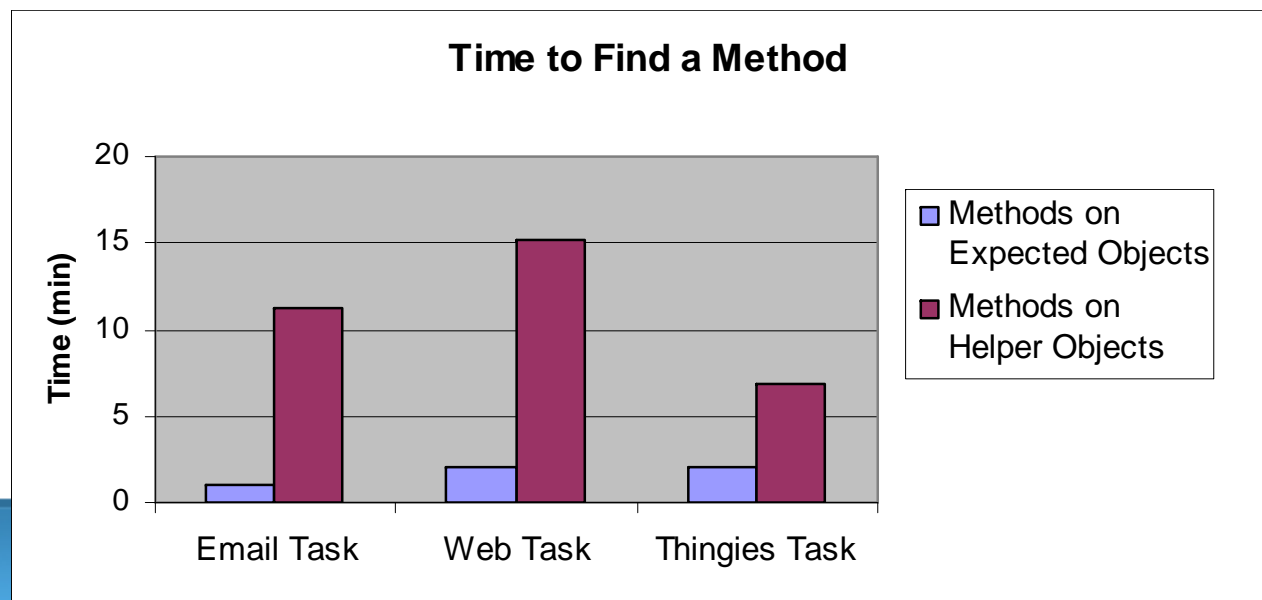
“Factory” Pattern

- Instead of “normal” creation: `Widget w = new Widget();`
- Objects must be created by *another* class:
`AbstractFactory f = AbstractFactory.getDefault();`
`Widget w = f.createWidget();`
- Used frequently in Java (>61) and .Net (>13) and SAP
- Results:
 - When asked to design on “blank paper”, **no one** designed a factory
 - Time to develop using factories took **2.1 to 5.3 times longer** compared to regular constructors (20:05 v 9:31, 7:10 v 1:20)
 - All subjects had difficulties getting using factories in APIs



Object Method Placement

- Where to put functions when doing object-oriented design of APIs when multiple classes work together
 - `mail_Server.send(mail_Message)`
VS.
`mail_Message.send(mail_Server)`
- When desired method is on the class that they start with, users were between **2.4** and **11.2 times faster** ($p < 0.05$)
- Starting class can be predicted based on user's tasks

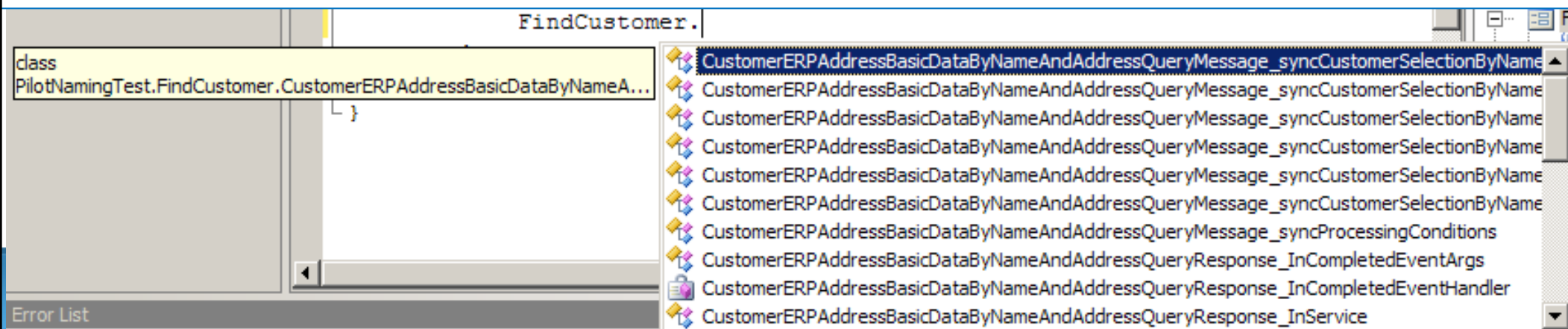


Study of APIs for SAP



- Study APIs for Enterprise Service-Oriented Architectures (“Web Services”)
- Naming problems:
 - Too long
 - Not understandable
 - Differences in *middle* are frequently missed

CustomerAddressBasicDataByNameAndAddressRequestMessageCustomerSelectionCommonName
CustomerAddressBasicDataByNameAndAddressResponseMessageCustomerSelectionCommonName



eSOA Documentation Results

- Multiple paths: unclear which one to use
- Some paths were dead ends
- Inconsistent look and feel caused immediate abandonment of paths
- Hard to find required information
- Business background helped

	Business Application Backgrounds		No Business Application Backgrounds	
	Success	Failure	Failure	Failure
Enterprise Workplace Homepage				
Enterprise Service index				
Cross Industry Solution Map				
Service Category				
Solution Map				
Process Component View				
Enterprise Service Interface				
Service Operation				

SAP ES WORKPLACE

SAP COMMUNITY NETWORK

ENTERPRISE SERVICES BY ENTERPRISE SERVICES BUNDLES

Enterprise Services Bundles group enterprise services according to business criteria. This bundles.

Please note: Enterprise Services Bundles including enterprise services from various applications are listed more than once. Browse the following Enterprise Services Bundles:

- Financials
 - Bank Communication Management
 - Credit Management

Sales Order

Added by Daniel Sass, last edited by Deborah Ga Labels: business object, customer fact sheet, cf logistics execution time integration, avail quote, no order for configurable product, order to cash for fashion integration, c

Enterprise Service Operations

- Create Sales Order
- Create Sales Order_V1
- Create Sales Order_V2
- Check Sales Order Creation
- Read Sales Order_V1
- Read Sales Order_V2

SAP's NetWeaver® Gateway Developer Tools

- Plug-in to Visual Studio 2010 for developing SAP applications
- We used the HCI methods of *heuristic evaluation* and *cognitive walkthroughs* to evaluate early prototypes
- Our recommendations were quickly incorporated due to agile software development process



Our Tools to Help with APIs

- Mica



- Jadeite



- Calcite



- Euklas



- Graphite



- Apatite





Mica Tool to Help Find Examples

- Makes Interfaces Clear and Accessible
- Use Google to find relevant pages
- Match pages with Java keywords
- Also notes which pages contain example code or definitions

The screenshot shows a Microsoft Internet Explorer browser window titled "Mica: Java full screen - Microsoft Internet Explorer". The address bar contains the URL <http://gem.pebbles.cs.cmu.edu:8080/mica/search?q=full+screen&lang=Java>. The search results page features the "Mica" logo and a search bar with the text "Java full screen" and a dropdown menu set to "Java". Below the search bar, the results are listed under "Search Completed".

Full-Screen Exclusive Mode API
Do you want to use high-performance graphics in the **Java** development environment?
... If you've been asking any of these questions, then the **full-screen** ...
[setFullScreenWindow](#)
[getDefaultScreenDevice](#)
[GraphicsDevice](#)
[GraphicsEnvironment](#)
[isFullScreenSupported](#)
[BufferCapabilities](#)
[getDefaultConfiguration](#)
[BufferStrategy](#)
[Full-Screen Exclusive Mode API](#)
[java.sun.com/docs/books/tutorial/extra/fullscreen/ - 7k - Cached](#)

Full-Screen Exclusive Mode
Full-screen exclusive mode is handled through a **java.awt.GraphicsDevice** object.
For a list of all available **screen** graphics devices (in single or ...
[java.sun.com/docs/books/tutorial/extra/fullscreen/exclusivemode.html - 8k - Cached](#)

Enabling Full-Screen Mode (Java Developers Almanac Example)
Code Examples from The **Java** Developers Almanac 1.4.
[javaalmanac.com/egs/java.awt/screen_FullWin.html - 8k - Cached](#)

Double-Buffering in Full-Screen Mode (Java Developers Almanac Example)
Code Examples from The **Java** Developers Almanac 1.4.
[javaalmanac.com/egs/java.awt/screen_Flip.html - 10k - Cached](#)

[Office of the President, Republic of China -Taiwan Rainbow Hall](#)
Taiwan Rainbow Hall: **JAVA** ■■■■■ **JAVA Full Screen** ■■■■■



Jadeite: Improved JavaDoc



- **Jadeite**: Java **API** Documentation with **Extra** Information **Tacked-on** for **Emphasis**

<http://www.cs.cmu.edu/~jadeite>

- Fix JavaDoc to help address problems
 - Focus attention on most popular packages and classes using font size
 - “Placeholders” for methods that users want to exist
 - Automatically extracted code examples for how to create classes

Packages
[com.sun.mail.dsn](#)
[com.sun.mail.handlers](#)
[com.sun.mail.lap](#)
[com.sun.mail.imap](#)
[com.sun.mail.imap.protocol](#)
[com.sun.mail.pop3](#)
[com.sun.mail.smtp](#)
[com.sun.mail.util](#)
[javax.mail](#)
[javax.mail.event](#)
[javax.mail.internet](#)
[javax.mail.search](#)
[javax.mail.util](#)

See Also (auto-generated):
[Transport](#)
[MimeMessage](#)
[InternetAddress](#)

abstract void	saveChanges () Save any changes made to this message into the message-store when the containing folder is closed, if the message is contained in a folder.
void	send () Use the Transport.send(message) method to send Messages
protected void	setExpunged (boolean expunged) Sets the expunged flag for this Message.

Most common way to construct:

```
SSLSocketFactory factory = ...;  
String host = ...;  
int port = ...;  
SSLSocket socket = (SSLSocket) factory.createSocket(host, port);  
Based on 38 examples
```





Calcite: Eclipse Plugin for Java

- **Calcite: C**onstruction **A**nd **L**anguage **C**ompletion Integrated **T**hroughout

<http://www.cs.cmu.edu/~calcite>

- Code completion in Eclipse augmented with Jadeite's information
 - How to create objects of specific classes

SSLSocket s = ???

Press 'Ctrl+Space' to show Template Proposals

Press 'Tab' from proposal table or click for focus



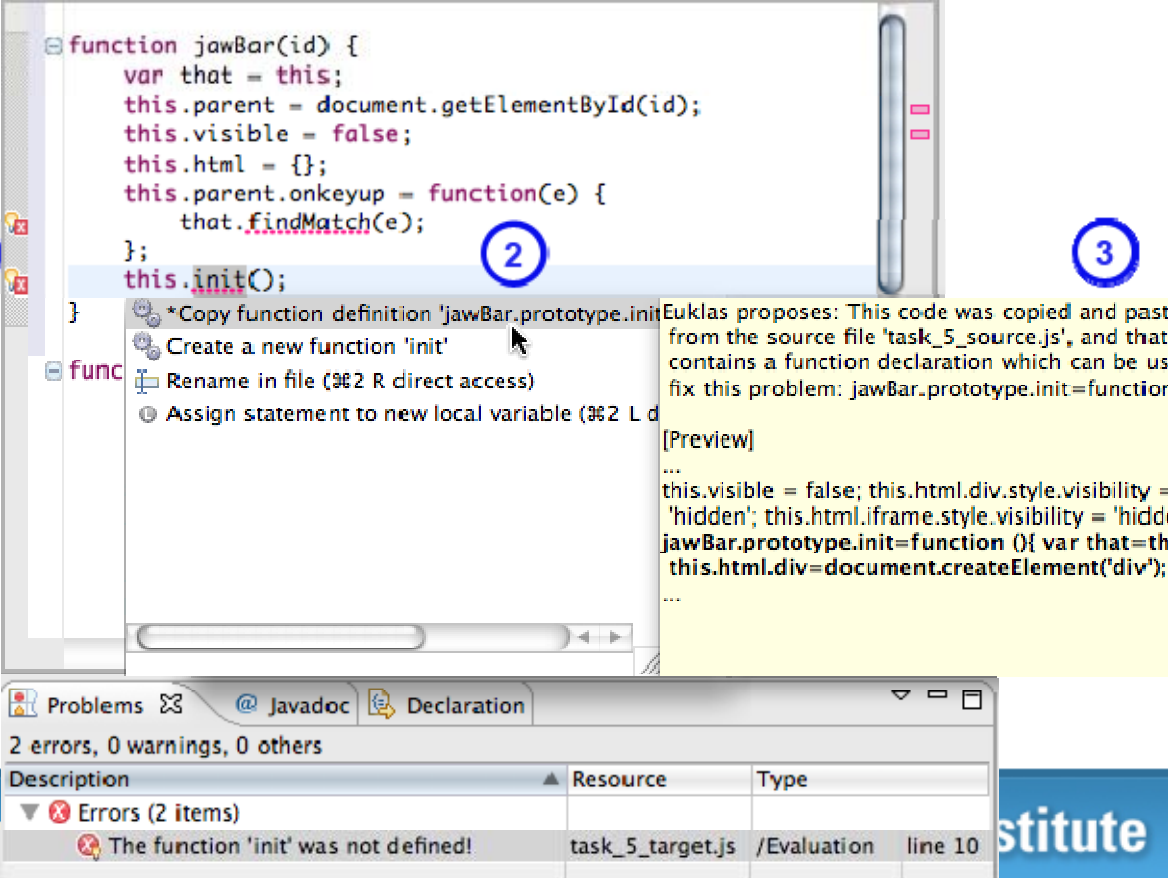
Euklas: Eclipse Plugin for JavaScript



- Euklas: Eclipse Users' Keystrokes Lessened by Attaching from Samples

<http://www.cs.cmu.edu/~euklas>

- Brings Java-like analysis to JavaScript
- Auto-correct uses copy source context for errors due to copy & paste



The screenshot shows the Eclipse IDE with a JavaScript file open. The code defines a function `jawBar(id)` with an `init()` method. A context menu is open over the `init()` call, showing options like "Copy function definition", "Create a new function 'init'", "Rename in file", and "Assign statement to new local variable". A yellow tooltip explains the error: "Euklas proposes: This code was copied and pasted from the source file 'task_5_source.js', and that contains a function declaration which can be used to fix this problem: `jawBar.prototype.init=function() { var that=this; this.html=document.createElement('div'); ... }`". The Problems window at the bottom shows two errors: "The function 'init' was not defined!".

```
function jawBar(id) {
    var that = this;
    this.parent = document.getElementById(id);
    this.visible = false;
    this.html = {};
    this.parent.onkeyup = function(e) {
        that.findMatch(e);
    };
    this.init();
}

function jawBar.prototype.init() {
    // ...
}
```

Problems 2 errors, 0 warnings, 0 others

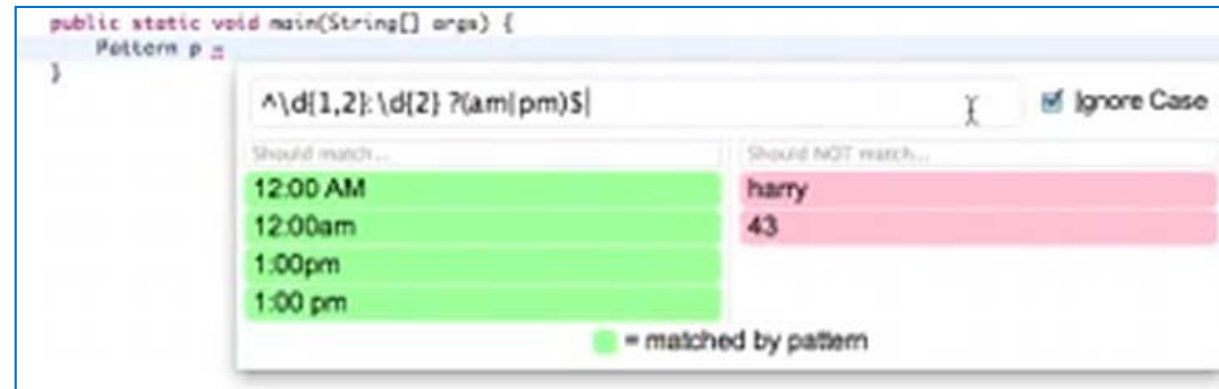
Description	Resource	Type
▼ Errors (2 items)		
The function 'init' was not defined!	task_5_target.js	/Evaluation line 10



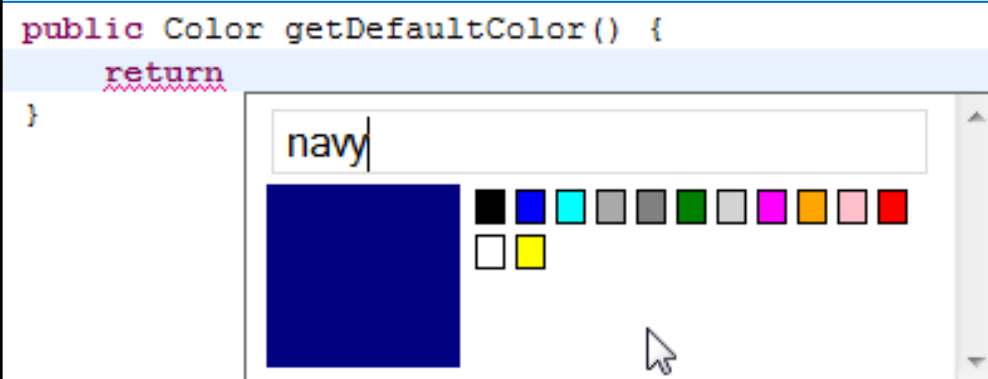
Graphite: Eclipse Plugin for Literals



- Graphite: GRAphical Palettes Help Instantiate Types in the Editor.
- Pop up a custom palette for specialized constants (literals) in Eclipse
 - Color palettes
 - Regular expression strings
- Customizable



(ICSE'2012)



(a)

```
public Color getDefaultColor() {  
    return new Color(  
        0,  
        0,  
        128); // navy  
}
```

(b)

Apatite Documentation Tool



- **Apatite**: Associative **P**erusing of **A**PIs That Identifies **T**argets **E**asily
<http://www.cs.cmu.edu/~apatite>
- Start with verbs (actions) and properties and find what classes implement them
- Find associated items
 - E.g., classes that are often used together
 - Classes that implement or are used by a method

The screenshot displays two side-by-side panels of the Apatite tool's search results. Both panels have a search bar at the top with the text 'Type here to search...'. The left panel shows a search for 'read', with the 'read' method highlighted in a green bar. The right panel shows a search for 'BufferedReader', with 'BufferedReader' highlighted in a green bar. Both panels have a 'FILTER' button between them. The results are organized into categories: Packages, Classes, Methods, Actions, and Properties. The left panel shows results for 'read' under the 'Methods' category, including 'close', 'list', 'read', and 'write'. The right panel shows results for 'BufferedReader' under the 'Classes' category, including 'BufferedReader', 'FileInputStream', and 'InputStreamReader'. The 'Methods' category on the right shows 'close', 'println', 'toString', and 'write'. The 'Actions' category on the right shows 'block', 'read', 'stream', and 'zero'. The 'Properties' category on the right shows '(No results)'. The bottom of the screenshot shows the 'Properties' category for the left panel, listing 'AbsolutePath', 'Directory', 'Name', and 'Path'.



Studies of Code Understanding

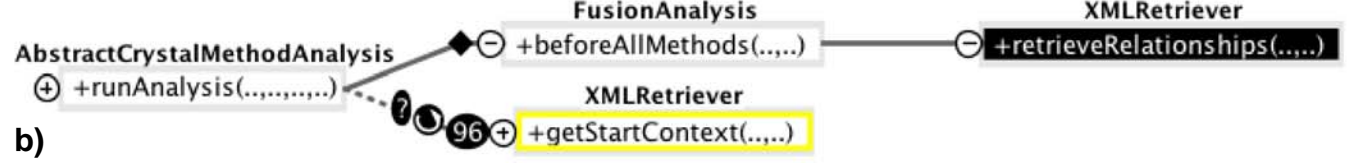
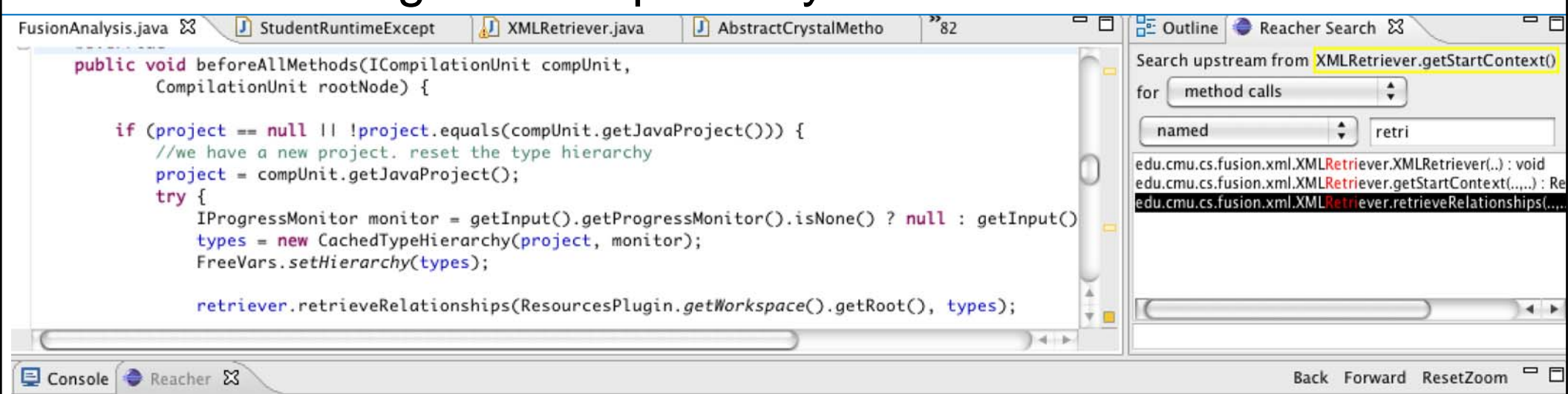
- Thomas LaToza, PhD 2012
- Studies about how experts learn unfamiliar code
- Programmers investigate *reachability questions*
 - How can this code *be reached*, either upstream or downstream
 - E.g., control flow from user scrolling → update status line
- Identified over 100 hard-to-answer questions that developers asked
 - E.g., “What method implements this trigger?”
 - “Why was this designed this way?”
- Survey shows such control flow questions are difficult and important
- No easy way to discover with current tools
 - Call graphs are too general



REACHER

- Visualize exactly the paths of interest
- **Search** along the paths
- Focused questions and answers enable effective analysis of complex codebases
- Developers with Reacher **5.6** times more **successful** than those working with Eclipse only

0:53



Fluorite Logger



- PhD work of YoungSeok Yoon (in progress)
- **Fluorite**: Full of **L**ow-level **U**ser **O**perations **R**ecorded **I**n **T**he Editor <http://www.cs.cmu.edu/~fluorite>
- Logger for *all* keystrokes & events in Eclipse
- Analyzes frequencies and patterns
- Deleting is a high percent of all the keystrokes
- Also surveyed >100 developers

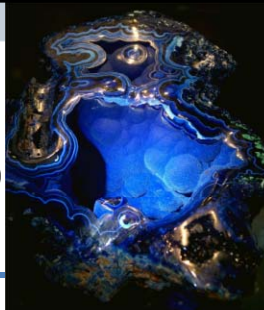
Commands		Keystrokes	
Type char.	17092 (31.8%)	Down arrow	5797 (13.7%)
Line down	5795 (10.8%)	Backspace	5693 (13.5%)
Delete prev.	5692 (10.6%)	Up arrow	4495 (10.6%)
Move caret	4686 (8.7%)	Right arrow	3586 (8.5%)
Line up	4491 (8.4%)	Left arrow	2751 (6.5%)
Col. next	3544 (6.6%)	Shift	1645 (3.9%)
Col. prev.	2715 (5.1%)	Enter	1641 (3.9%)
Select text	1975 (3.7%)	T	1289 (3.1%)
Sel. col. next	1035 (1.9%)	E	1250 (3.0%)
File open	907 (1.7%)	S	1021 (2.4%)
Sel. col. prev.	857 (1.6%)	N	1003 (2.4%)
Save	852 (1.6%)	I	881 (2.1%)
Delete	576 (1.1%)	Space	859 (2.0%)
Paste	459 (0.9%)	A	790 (1.9%)
Assist(auto)	456 (0.8%)	O	750 (1.8%)
Run	391 (0.7%)	L	610 (1.4%)
Copy	314 (0.6%)	Delete	576 (1.4%)
Undo	294 (0.5%)	C	557 (1.3%)
Assist(manual)	213 (0.4%)	.	546 (1.3%)
Sel. line down	212 (0.4%)	R	510 (1.2%)
Others	1113 (2.1%)	Others	5970 (14.1%)
Total	53669	Total	42220



Backtracking Results

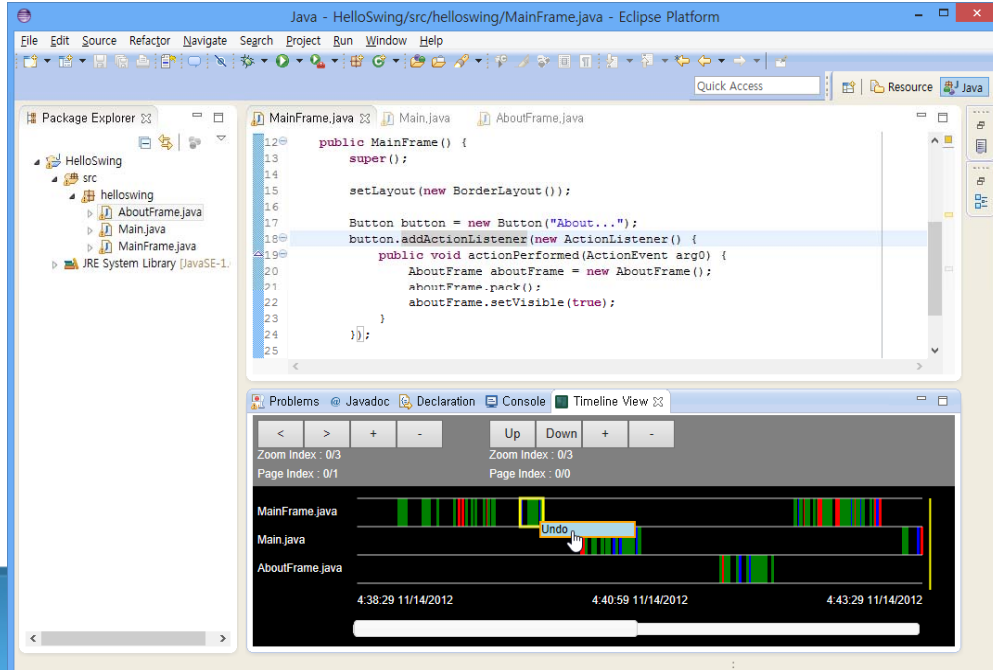
- All developers *backtrack* for many reasons
 - Explorations, investigations, iterative design
- People use comments to remove code, so they can restore it if necessary
 - But difficult to comment & uncomment correctly
 - Often non-local changes
- Undo not used for exploration, just typo fixing
- Current work: new tool to help developers backtrack





Azurite: Eclipse Plugin for Selective Undo

- PhD work of YoungSeok Yoon (in progress)
- **Azurite**: Adding **Z**est to **U**ndoing and **R**estoring **I**mproves **T**extual **E**xploration <http://www.cs.cmu.edu/~azurite>
- Work out semantics of selective undo for code
 - Conflicting edits of same code must be shown to user
- Time-line visualization of all past operations
- Search through history (time) to find appropriate points



Summary

- 30 studies; 18 systems in 17 years
- Doing studies first provides new insights that can inspire significantly new designs for programming languages and environments
- Need to understand software engineers' real issues
- New designs shown to be better



Thanks to:



- Funding:

- NSF under IIS-1116724, IIS-0329090, CCF-0811610, IIS-0757511 (Creative-IT), NSF ITR CCR-0324770 as part of the EUSES Consortium

- SAP



- Adobe



- IBM



- Microsoft Research RISE



- >30 students:

- | | | |
|-------------------|----------------------------------|---------------------------------|
| ■ Htet Htet Aung | ■ Aristiwidya B. (Ika) Hardjanto | ■ Stephen Oney |
| ■ Jack Beaton | ■ Erik Harpstead | ■ John Pane |
| ■ Ruben Carbonell | ■ Sae Young (Sophie) Jeong | ■ Sunyoung Park |
| ■ John R. Chang | ■ Andy Ko | ■ Chotirat (Ann) Ratanamahatana |
| ■ Kerry S. Chang | ■ Sebon Koo | ■ Christopher Scaffidi |
| ■ Polo Chau | ■ Thomas LaToza | ■ Jeff Stylos |
| ■ Luis J. Cota | ■ Joonhwan Lee | ■ David A. Weitzman |
| ■ Michael Coblenz | ■ Leah Miller | ■ Yingyu (Clare) Xie |
| ■ Dan Eisenberg | ■ Mathew Mooty | ■ Zizhuang (Zizzy) Yang |
| ■ Brian Ellis | ■ Gregory Mueller | ■ YoungSeok Yoon |
| ■ Andrew Faulring | ■ Yoko Nakano | |

Thank You!

Improving Software Development through Human-Centered Approaches

Brad A. Myers

Human-Computer Interaction Institute

School of Computer Science

Carnegie Mellon University

<http://www.cs.cmu.edu/~bam>

bam@cs.cmu.edu

