Reuse

SWE 795, Spring 2017

Software Engineering Environments

Partially adapted from Kerry Chang, Finding Code to Reuse, 05-899D Human Aspects of Software Development

Today

- Part 1 (Discussion)(15 mins)
  - Preventing defects reading from last week

- Part 2 (Lecture)(~70 mins)

- Break!

- Part 3 (Discussion)(~60 mins)
  - Discussion of readings

What is reuse?

- Making use of previously written code rather than writing new code

- Often, reuse takes form of reusing a *library* or a *framework*

- Once made choice to reuse a library or framework, need to understand how to achieve specific behavior with library or framework
  - Often finding code *snippets* that achieve desired behavior

Reuse of Uses

- Developers rely extensively on *examples* to understand how to instantiate objects

Reuse Activity	Specific Strategies Observed
Finding a Usage Context	Find senders of messages defined for target class, focusing on "interesting" ones
Evaluating a Usage Context	
Executing the Context	Look for references to application data objects in the openOn: method.
Assessing Similarity	Open example application "on" a basic data object from the project.
Studying Bits of Context	Reason by analogy from familiar syntactic construction, e.g., button1Down:
Deciding to Subclass	Look for use of unmappable instance variables or many messages to "self."
Debugging a Usage Context	
Getting an Instance Running	Focus first on the openOn: code for starting up a window.
Borrowing the Context	Use multiple browsers to work from related pieces of context. Carry out step-by-step replacement of message parameters. Edit what does not compile. Develop a method to substitute one data object for another.
Analysis by Testing	Adapt or develop the method identified in the notification "message not understood."

Mary Beth Rosson and John M. Carroll. 1996. The reuse of uses in Smalltalk programming. *ACM Trans. Comput.-Hum. Interact.* 3, 3 (September 1996), 219-253.

Some possible reuse strategies

- Read the documentation
- Find tutorials
- Find StackOverflow snippets
- Find similar code in your own codebase
- Call API functions, see what they return

Opportunistic vs. systematic

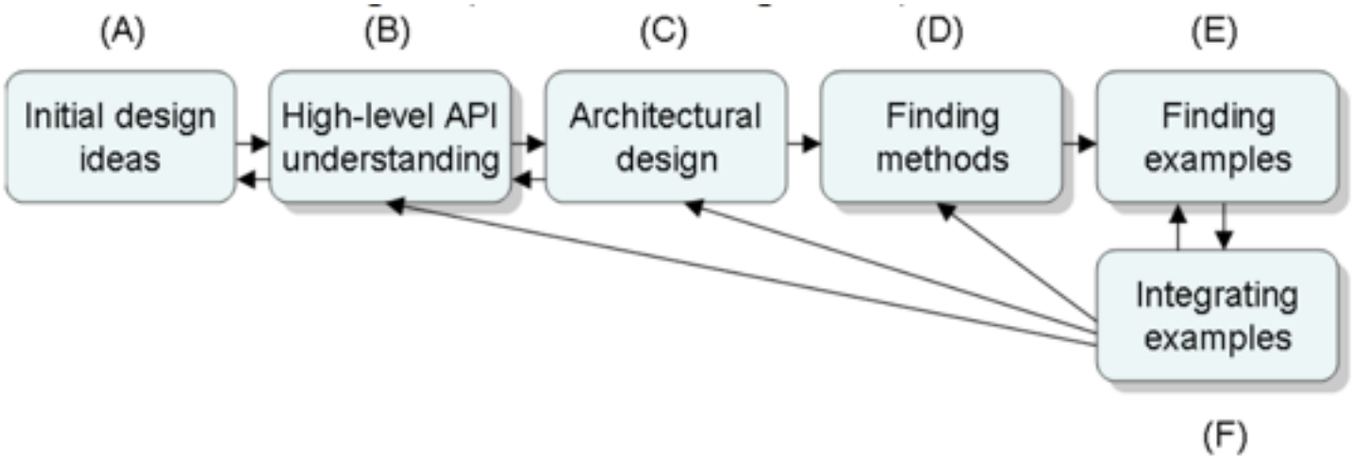
- Opportunistic developers more likely to start with example code
- Systematic developers more likely to read the documentation first

Steven Clarke.

Reuse topics today

- How do developers do it?
  - What are the key steps? What challenges do developers face?
  - Challenges working with APIs
- Techniques for supporting software reuse
  - Browsing APIs
  - Code search

Example reuse process



B: read tutorials, articles, projects to understand domain

D: searched Google, often seeking descriptions in API of specific classes & methods to use

E: looked for examples of how to use specific methods

J. Stylos and B. A. Myers, "Mica: A Web-Search Tool for Finding API Components and Examples," *Visual Languages and Human-Centric Computing (VL/HCC'06)*, Brighton, 2006, pp. 195-202.

Types of reuse

WEB SESSION INTENTION:	LEARNING	CLARIFICATION	REMINDER
Reason for using Web	Just-in-time learning of unfamiliar concepts	Connect high-level knowledge to implementation details	Substitute for memorization (*e.g.*, language syntax or function usage lookup)
Web session length	Tens of minutes	$\sim$ 1 minute	$<$ 1 minute
Starts with web search?	Almost always	Often	Sometimes
Search terms	Natural language related to high-level task	Mix of natural language and code, cross-language analogies	Mostly code (*e.g.*, function names, language keywords)
Example search	"ajax tutorial"	"javascript timer"	"mysql_fetch_array"
Num. result clicks	Usually several	Fewer	Usually zero or one
Num. query refinements	Usually several	Fewer	Usually zero
Types of webpages visited	Tutorials, how-to articles	API documentation, blog posts, articles	API documentation, result snippets on search page
Amount of code copied from Web	Dozens of lines (*e.g.*, from tutorial snippets)	Several lines	Varies
Immediately test copied code?	Yes	Not usually, often trust snippets	Varies

Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. *Conference on Human Factors in Computing Systems* (CHI '09), 1589-1598.

Types of reuse

- Learning—relies on selecting highest quality tutorials tutorials
  - e.g., “update web page without reloading php”
- Clarification—learning syntax based on exiting understanding of the domain concepts
  - e.g., reminding use of syntax of HTML forms
  - Often search by analogy to domain concepts in other languages / frameworks
    - e.g., Perl has a function to format dates as strings, what’s the one for PHP?
- Reminder—using web as external memory aid
  - e.g., forgot a word in a long function name
  - e.g., 6 lines of code necessary to connect and disconnect from MySQL database copied hundreds of times by individual

Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. *Conference on Human Factors in Computing Systems* (CHI ’09), 1589-1598.

Design implications

- Web tutorials used for just in time learning
  - —> Tutorials should be tightly coupled to code, where developers can play in sandbox then read tutorial content to understand problems when do not work
- Web search used as translator from intention to terminology & syntax
  - —> tools could compare code from search results to users code to automatically locate errors
  - —> search should be integrated into autocomplete
- Developers delay testing, esp for routine functionality
  - —> Tools should assist with adaption by highlighting variables and literals in reused snippets & provide link back to original source

Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. *Conference on Human Factors in Computing Systems* (CHI '09), 1589-1598.

Challenges with reuse

- Mapping an abstract conceptual solution into the appropriate elements
  - *“How do I create a rectangle? Why is there no Rectangle tool?”*
- Understanding control & data flow, hidden dependencies due to run-time binding or inheritance, between classes in the API
  - *“I’m over-riding SelectionTool, and in particular mouseDown() so that when the figure is clicked the box is drawn. This bit works, however when trying to drag the figure, if I do something similar the rectangle flickers like mad.”*
- Understanding how functionality works
  - *“How does ... work?”, “What does ... do?” or, “Where is ... defined/created/called?”*
- Making changes consistent w/ architectural constrains of API
  - Violating constraints of MVC architecture by passing references in prohibited ways

Douglas Kirk, Marc Roper, and Murray Wood. 2007. Identifying and addressing problems in object-oriented framework reuse. *Empirical Softw. Eng.* 12, 3 (June 2007), 243-274.

Challenges with reuse

- **Design** barriers—inherent cognitive difficulties of the programming problem, separate from notation used
  - I don't know what I want the computer to do
- **Selection** barriers—finding programming interfaces available to achieve a particular behavior
  - I don't know what to use
- **Coordination** barriers—constraints governing how languages & libraries can be combined
  - I don't know how to make them work together
- **Use** barriers—determining how API how to use API
  - I don't know how to use it
- **Understanding** barriers—environment properties such as compile & runtime errors that prevent seeing behavior
  - It didn't do what I expected
- **Information** barriers—environment properties that prevent understanding runtime execution state
  - I think I know why didn't behave as expected, but don't know how to check

Vocabulary problem

- Developers are familiar with concepts using one set of terminology.
- API, tutorials, or other resources use different terminology

- How do developers find the right concepts with alternative terms?

Challenges may vary by context

- Size of desired snippet
  - Reusing a line of code? A whole algorithm?

- Alternatives
  - How many alternatives are there? How important is it to find the best alternative?

- Integration
  - What libraries or frameworks does a snippet require? How can they be integrated?

Challenges working with API documentation

- Goal: Parse a Java source file w/ Eclipse
- Answer:

```
IFile file = ...;
ICompilationUnit cu =
  JavaCore.createCompilationUnitFrom(file);
ASTNode ast = AST.parseCompilationUnit(cu, false);
```

- Challenges
  - Want to work with files and ASTNodes, but key class is JavaCore
  - No connection from what you might know about ASTNode and IFile to JavaCore

Techniques for reuse

- Browsing API documentation
- Searching with ….
- Integrating search into autocomplete
- Enabling sample adaption & exploratory programming w/ examples

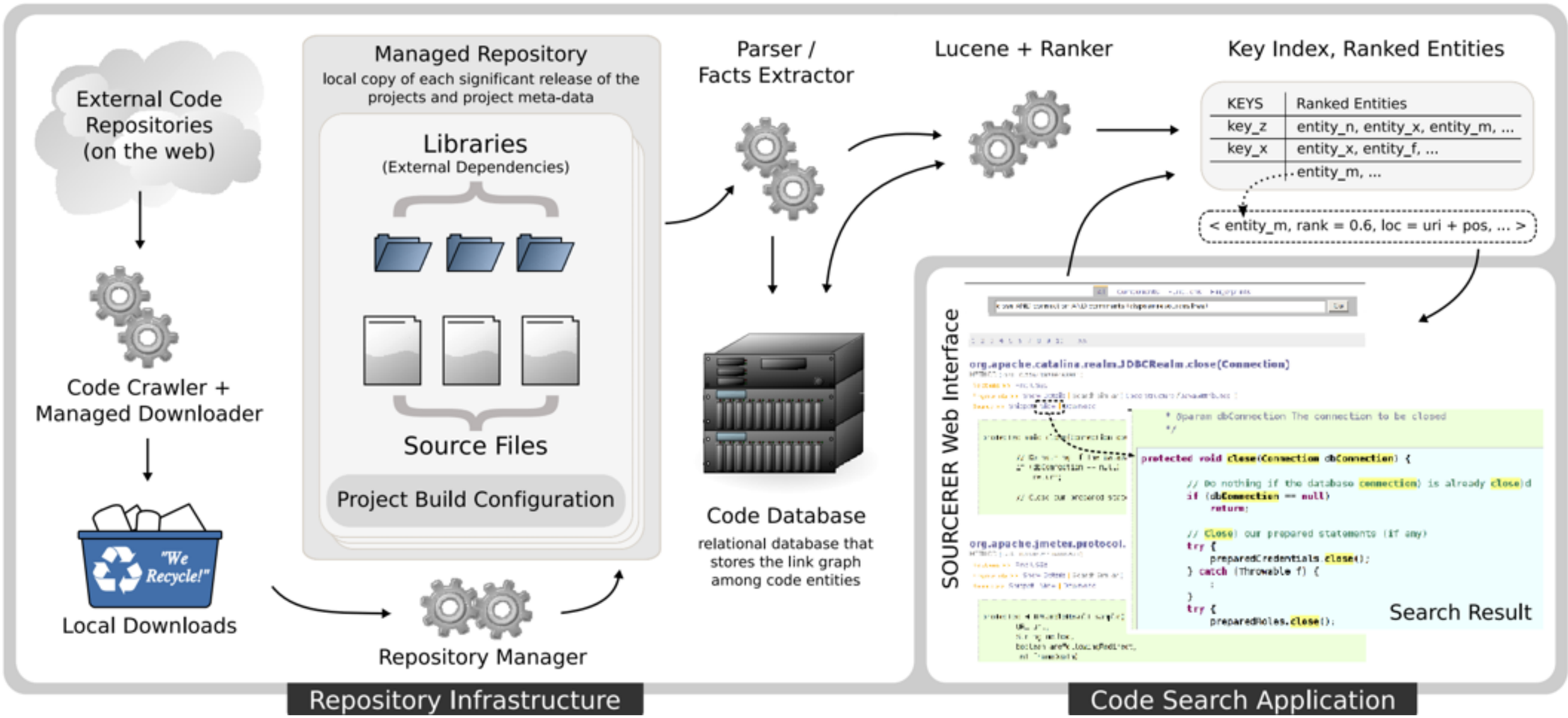
Filtering & browsing documentation

Apatite: A New Interface for Exploring APIs

Daniel S. Eisenberg, Jeffrey Stylos, and Brad A. Myers

Carnegie Mellon Univeristy

Indexing OSS projects in code search



Sushil Bajracharya, Trung Ngo, Erik Linstead, Yimeng Dou, Paul Rigor, Pierre Baldi, and Cristina Lopes. Sourcerer: a search engine for open source code supporting structure-based search. In OOPSLA '06: Companion to the 21st ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications, pages 681–682, New York, NY, USA, 2006. ACM Press.

Searching by inputs and outputs

Name	Test Cases	
	Input	**Output**
Simple Tokenizer	“this is a test”	[“this”, “is”, “a”, “test”]
Quote Tokenizer	“this is a test”	[“this”, “is”, “a”, “test”]
	“this is a ‘test with’ quoted \”string types\” in it”	[“this”, “is”, “a”, “test with”, “quoted”, “string types”, “in”, it”]
Robots.txt	“http://www.cs.brown.edu/people/spr”	true
	“http://www.cnn.com/topics”	true
	“http://www.nytimes.com/college/students”	false
Log2	0	RuntimeException
	1	0
	4	2
	32	5
To Roman	13	xiii
From Roman	VIII	8
	xxvi	26
Primes	5	true
	39	false
	59	true
Perfect Numbers	6	true
	12	false
	28	true
Day of Week	“08/07/08”	“Thursday”
Easter	2008	new Date(108,2,23)

Steven P. Reiss. 2009. Semantics-based code search. In *Proceedings of the 31st International Conference on Software Engineering* (ICSE '09). IEEE Computer Society, Washington, DC, USA, 243-253.

S6 Search Page - Iceweasel

File Edit View History Bookmarks Tools Help

http://conifer.cs.brown.edu:8180/S6Search/s6search.html

Most Visited Getting Started Latest Headlines

S6

Look for: METHOD In Remote Archives Using Google

Description: roman numeral
(keywords)

Method

Declaration: java.lang.String convert(int a1)

Tests: (17) == "XVII" CALL

( ) == CALL

User Context Security Contracts Threading

Find it!

Results:

Order By: Code Size Format Using: Brown

Source: programs/roman-numerals/bio98q1.java @ http://www.olympiad.org.uk License

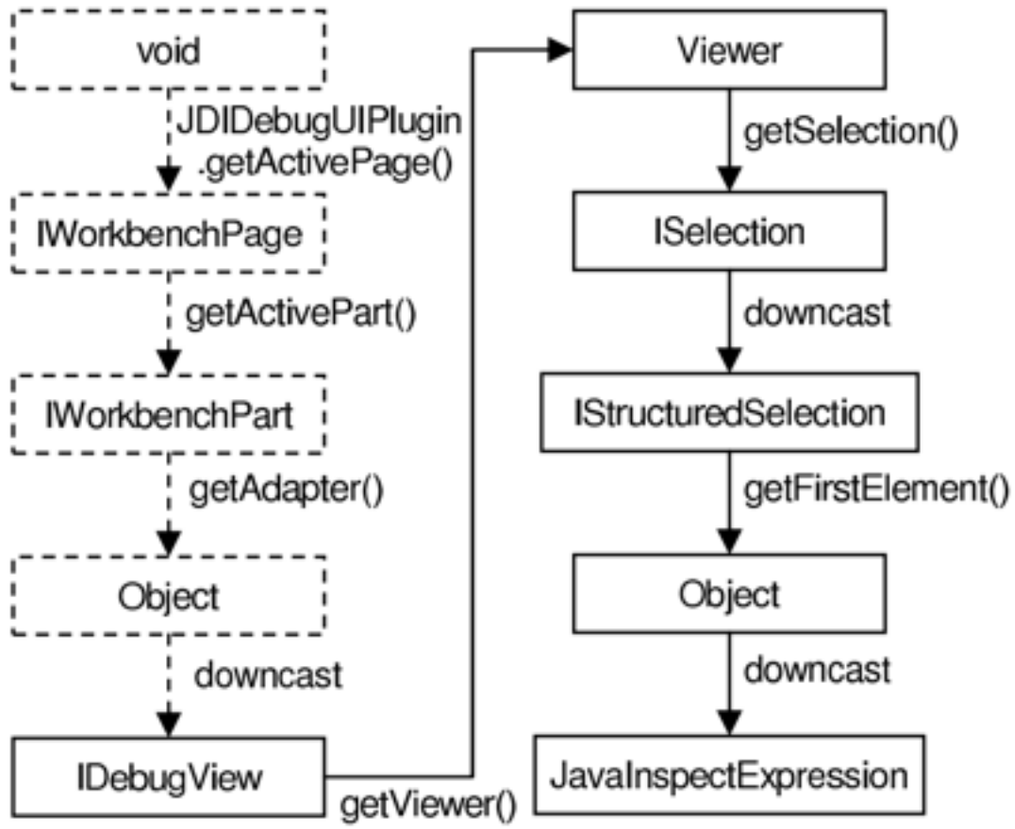
```
static String [] hundreds = {"", "c", "cc", "ccc", "cd", "d", "dc", "dcc", "dccc", "cm" };

static String [] tens = {"","x","xx","xxx","xl","l","lx","lxx","lxxx","xc" };
static String [] thousands = {"","m","mm","mmm" };
static String [] units = {"","i","ii","iii","iv","v","vi","vii","viii","ix"};


public static String convert(int n)
{
   return (thousands[(n/1000)] + hundreds[(n/100)% 10] + tens[(n/10)% 10] +units[(n)% 10]).toUpperCase();
}
```

Source: XQuisitor/saxonb8-4+Folder/net/sf/saxon/number/Numberer_en.java @ http://www.cafeconleche.org/xquisitor/xquisitor-1.0a5.zip License

Searching by input and output *types*

Programming problem	τ_{in}	τ_{out}
Read lines from an input stream (Tester)	`InputStream`	`BufferedReader`
Open a named file for memory-mapped I/O (Almanac)	`String`	`MappedByteBuffer`
Get table widget from an Eclipse view (FAQs)	`TableViewer`	`Table`
Get the active editor (Eclipse FAQs)	`IWorkbench`	`IEditorPart`
Retrieve canvas from scrolling viewer (Author)	`ScrollingGraphicalViewer`	`FigureCanvas`
Get window for MessageBox (Author)	`KeyEvent`	`Shell`
Convert legacy class (Author)	`Enumeration`	`Iterator`

Mine *Jungoloids* describing paths by which types can be converted



David Mandelin, Lin Xu, Rastislav Bodík, and Doug Kimelman. 2005. Jungloid mining: helping to navigate the API jungle. *Conference on Programming language design and implementation* (PLDI '05), 48-61.

Searching by output

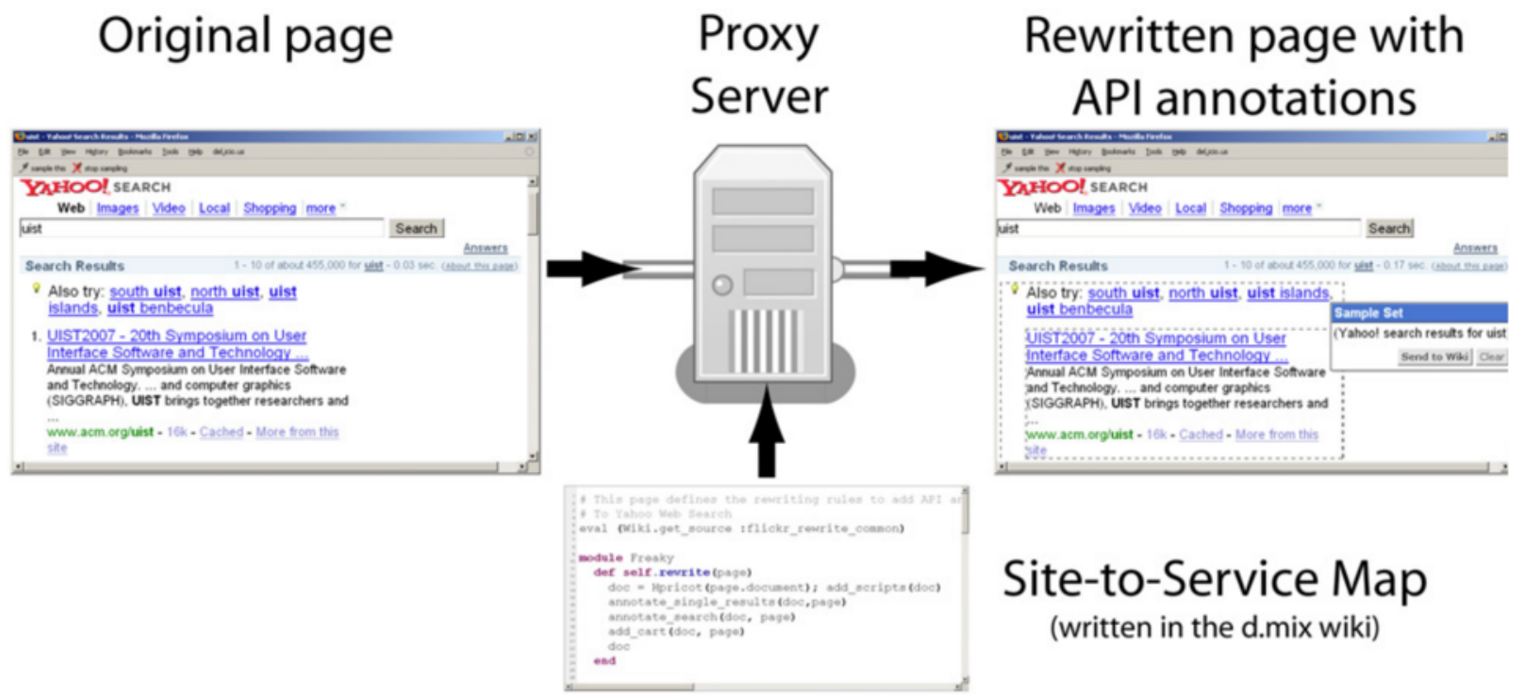


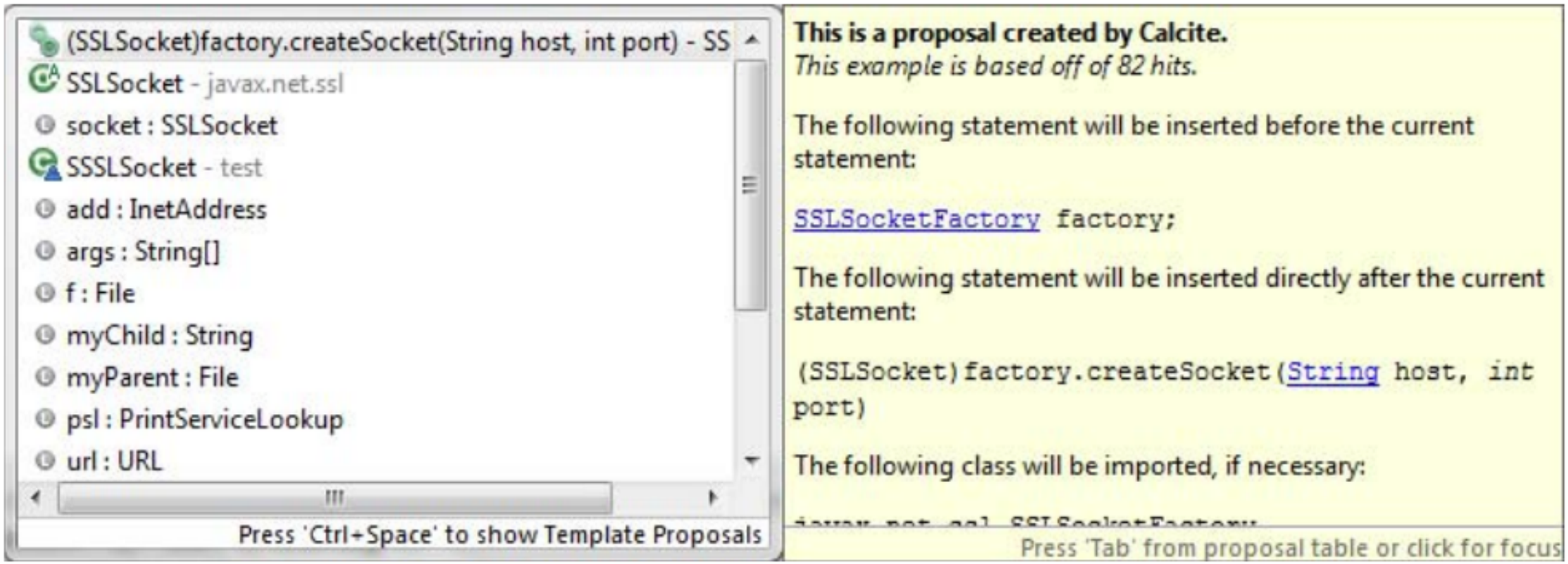
Figure 1. With d.mix, users browse web sites through a proxy that marks API-accessible content. Users select marked elements they wish to copy. Through a site-to-service map, d.mix composes web service calls that yield results corresponding to the user's selection. This code is copied to the d.mix wiki for editing and hosting.

http://dl.acm.org/citation.cfm?doid=1294211.1294254

Björn Hartmann, Leslie Wu, Kevin Collins, and Scott R. Klemmer. 2007. Programming by a sample: rapidly creating web applications with d.mix. *Symposium on User interface software and technology,* 241-250.

Searching for instantiation snippets

- Classes are often created through factories rather than constructors, making construction snippets harder to find
- Integrate construction snippet search into autocomplete



M. Mooty, A. Faulring, J. Stylos and B. A. Myers, "Calcite: Completing Code Completion for Constructors Using Crowds," *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, Leganes, 2010, pp. 15-22.

Labeling snippets with keywords

- Problem: how do you ensure that there's high quality labels explaining the intention of code snippets?
- Idea: enable search from keywords to code **and** from code to keywords
  - Log associations to support future queries

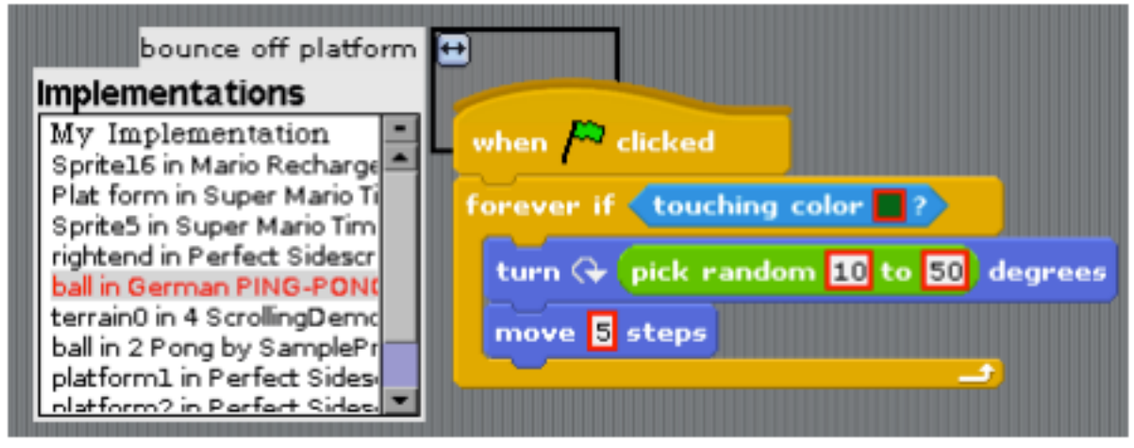


Figure 2. Given a purpose statement, the Zone sidebar (left) shows code that might fulfill it. Selecting an implementation from the list on the left shows its code on the right. As a simulation of future functionality, red boxes surround values that vary among otherwise similar code, highlighting what might need to be changed.

Figure 3. The Zone sidebar can suggest possible purpose statements for a code fragment (simulated for this illustration).

Kenneth C. Arnold and Henry Lieberman. 2010. Managing ambiguity in programming by finding unambiguous examples. *Conference on Object oriented programming systems languages and applications*, 877-884.

Grouping diverse search results

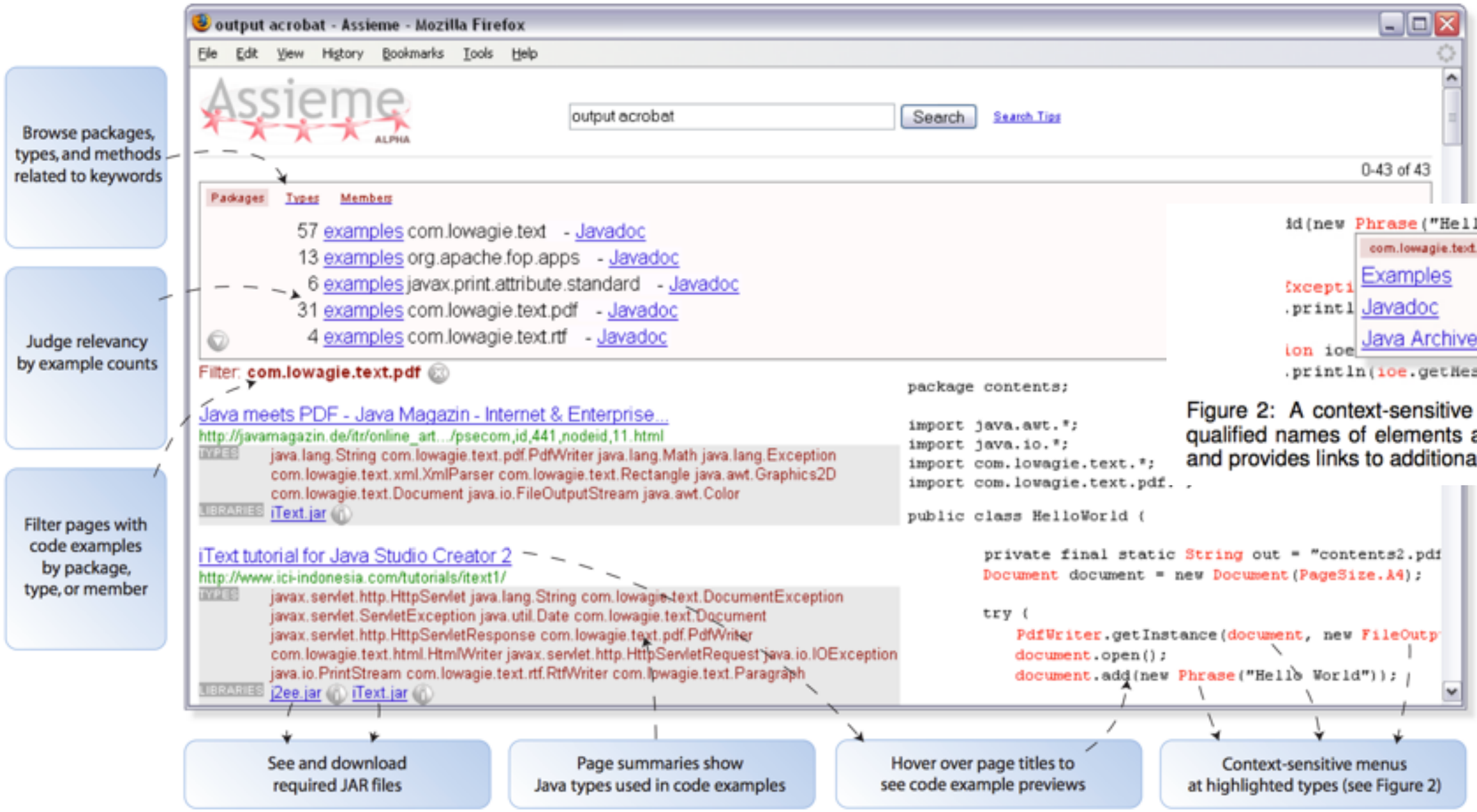
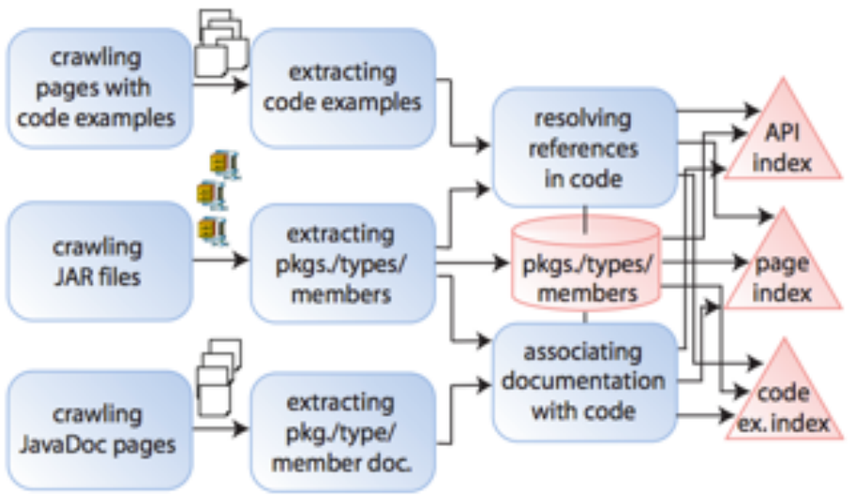


Figure 2: A context-sensitive menu reveals the fully qualified names of elements appearing in Java code and provides links to additional information.



Raphael Hoffmann, James Fogarty, and Daniel S. Weld. 2007. Assieme: finding and leveraging implicit references in a web search interface for programmers. *Symposium on User interface software and technology* (UIST '07), 13-22.

Adapt snippets

SnipMatch Demonstration

Doug Wightman[1], Zi Ye[1], Joel Brandt[2], Roel Vertegaal[1]

[1]Human Media Lab, Queen's University
Kingston, ON, K7L 3N6, Canada
{wightman, zi, roel}@cs.queensu.ca

[2]Advanced Technology Labs, Adobe
San Francisco, CA 94103
joel.brandt@adobe.com

Doug Wightman, Zi Ye, Joel Brandt, and Roel Vertegaal. 2012. SnipMatch: using source code context to enhance snippet retrieval and parameterization. *Symposium on User interface software and technology,* 219-228.

Adapt snippets

Stephen Oney and Joel Brandt. 2012. Codelets: linking interactive documentation and example code in the editor. *Conference on Human Factors in Computing Systems* (CHI ’12), 2697-2706.

Link code back to snippets

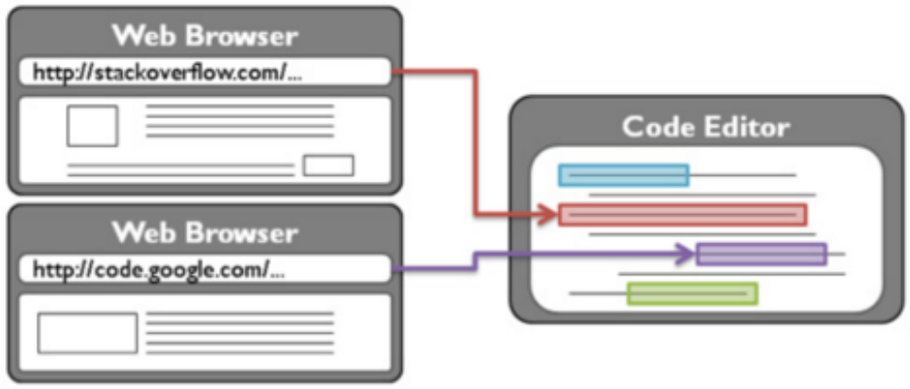


Figure 1: HyperSource associates Web pages visited by programmers with subsequent code edits.

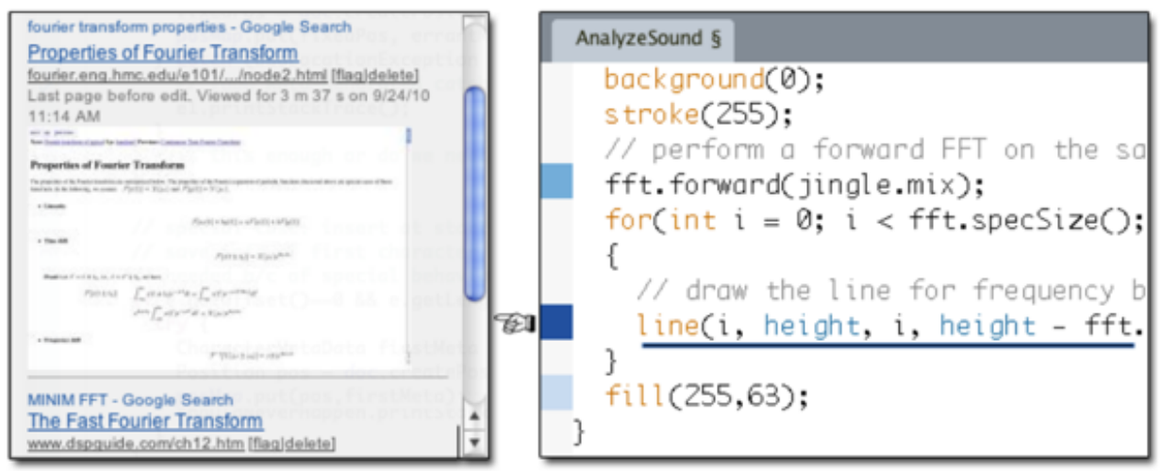


Figure 2: Web history (L) associated with the current line of code (R). Gutter highlights provide scent.

https://www.youtube.com/watch?v=_PYvPlv4OQw

Björn Hartmann, Mark Dhillon, and Matthew K. Chan. 2011. HyperSource: bridging the gap between source and code-related web sites. *Conference on Human Factors in Computing Systems* (CHI '11), 2207-2210.