

# Strategies for Confirming we are in the “right place” in the Source Code



```
2776
2777 //{{{ getListener() method
2778 private BufferListener getListener(int index)
2779 {
2780     return bufferListeners.get(index).listener;
2781 } //}}}
2782
2783 //{{{ contentInserted() method
2784 private void contentInserted(int offset, int length,
2785     IntegerArray endOffsets)
2786 {
2787     try
2788     {
2789         transaction = true;
2790
2791         int startLine = lineMgr.getLineOfOffset(offset);
2792         int numLines = endOffsets.getSize();
2793
2794         if (!loading)
2795         {
2796             firePreContentInserted(startLine, offset, numLines, length);
2797         }
2798
2799         lineMgr.contentInserted(startLine, offset, numLines, length,
2800             endOffsets);
2801         positionMgr.contentInserted(offset, length);
2802
2803         setDirty(true);
2804
2805         if (!loading)
2806         {
2807             fireContentInserted(startLine, offset, numLines, length);
2808
2809             if (!undoInProgress && !insideCompoundEdit())
2810                 fireTransactionComplete();
2811         }
2812     }
2813     finally
2814     {
2815         transaction = false;
2816     }
2817 } //}}}
2818
2819 //{{{ parseBufferLocalProperties() method
2820 private void parseBufferLocalProperties(CharSequence prop)
```

# Strategies for Confirming we are in the “right place” in the Source Code

```
2776
2777 //{{{ getListener() method
2778 private BufferListener getListener(int index)
2779 {
2780     return bufferListeners.get(index).listener;
2781 } //}}}
2782
2783 //{{{ contentInserted() method
2784 private void contentInserted(int offset, int length,
2785     IntegerArray endOffsets)
2786 {
2787     try
2788     {
2789         transaction = true;
2790
2791         int startLine = lineMgr.getLineOfOffset(offset);
2792         int numLines = endOffsets.getSize();
2793
2794         if (!loading)
2795         {
2796             firePreContentInserted(startLine, offset, numLines, length);
2797         }
2798
2799         lineMgr.contentInserted(startLine, offset, numLines, length,
2800             endOffsets);
2801         positionMgr.contentInserted(offset, length);
2802
2803         setDirty(true);
2804
2805         if (!loading)
2806         {
2807             fireContentInserted(startLine, offset, numLines, length);
2808
2809             if (!undoInProgress && !insideCompoundEdit())
2810                 fireTransactionComplete();
2811         }
2812     }
2813     finally
2814     {
2815         transaction = false;
2816     }
2817 } //}}}
2818
2819 //{{{ parseBufferLocalProperties() method
2820 private void parseBufferLocalProperties(CharSequence prop)
```

- Breakpoint
- `System.out.println("message");`
- Change observable behavior

# Example Concept Location Explanation for Assignment

- *'I started by locating the **loading screen image** in the **resources folder**. Once I identified the loading screen image as **"splash.png"**, I searched exactly for that string in the entire workspace and found the file **"SplashScreen.java"** [...]'*

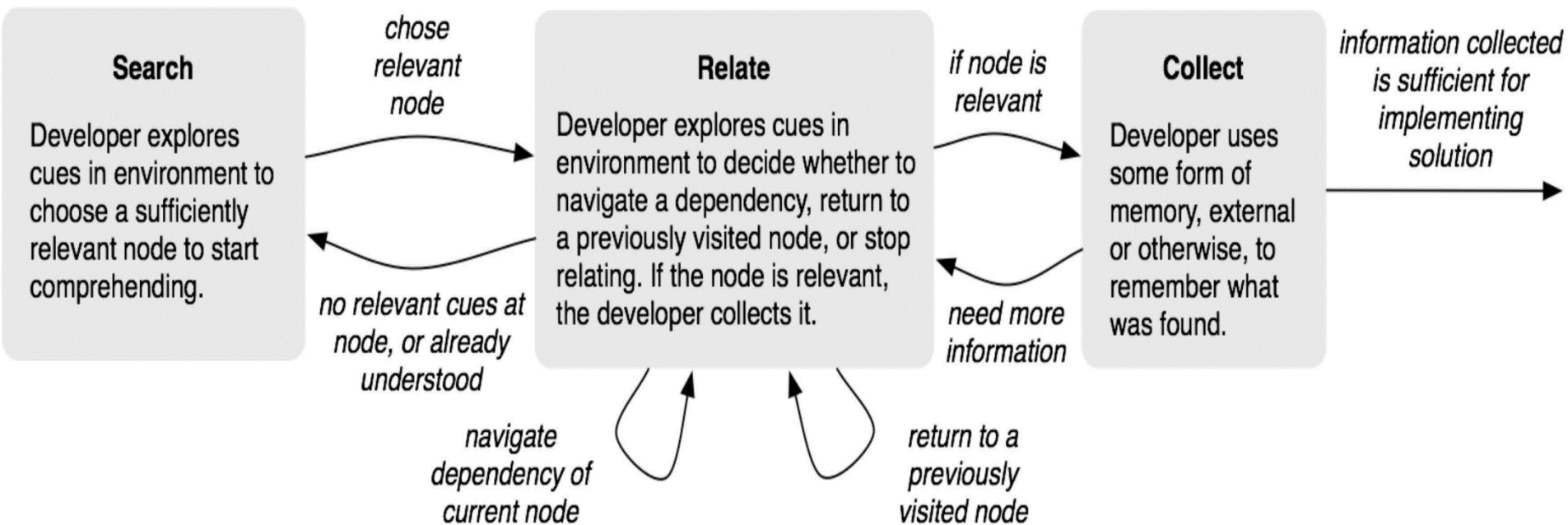
# Program Understanding Model

“Developers work on software maintenance tasks by

- searching for relevant code,
- navigating dependencies, and
- constructing a mental model of the dependencies of a feature,

suggesting the need for environments that explicitly support feature location through dependency navigation.”

# Program Understanding Model



“Developers work on software maintenance tasks by

- searching for relevant code,
  - navigating dependencies, and
  - constructing a mental model of the dependencies of a feature,
- suggesting the need for environments that explicitly support feature location through dependency navigation.”

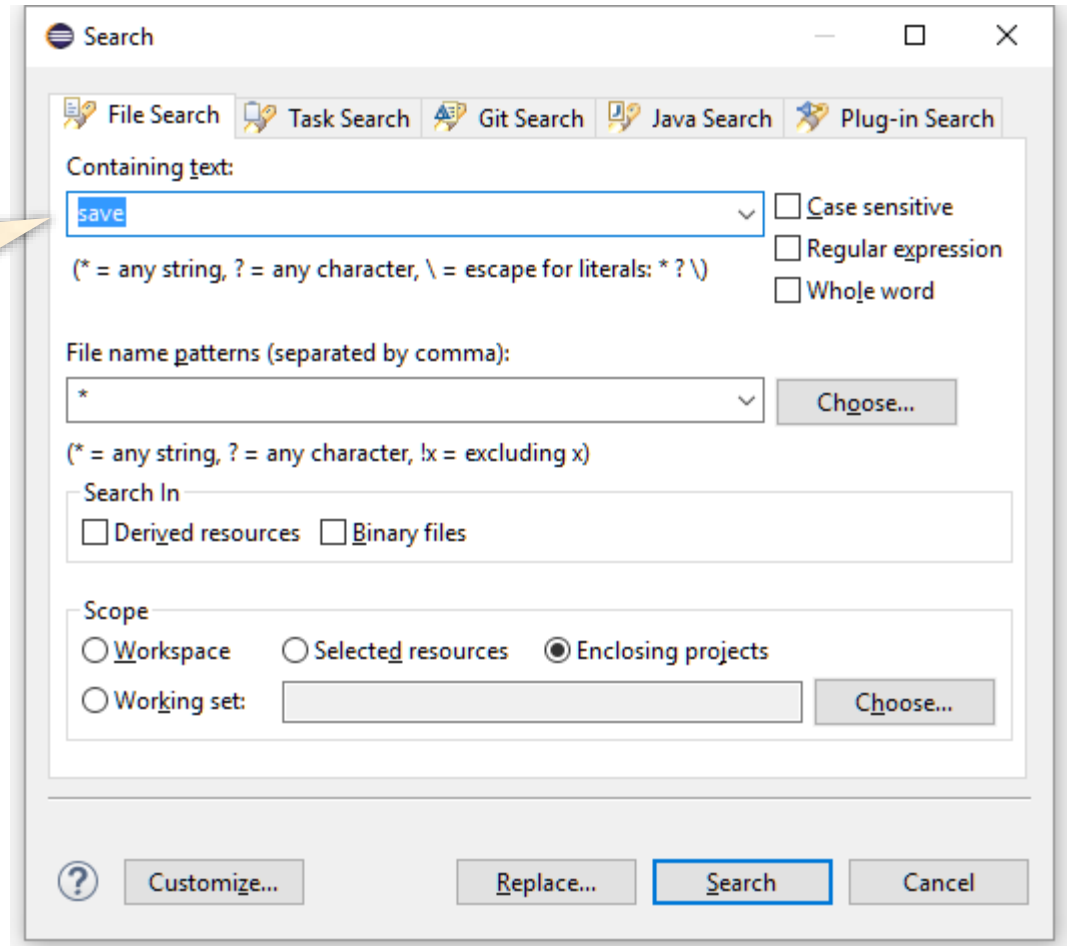
# Grep-based Concept Location

# Grep-based Concept Location

- Source code is not processed (treated as “text”) and search mechanism is regular expression matching
- Queries are regular expressions (i.e., formal language):
  - `[hc]at`, `.at`, `*at`, `^[^b]at`, `^[hc]at`, `[hc]at$`, etc.
- Results are **unordered lines of text** where the query is matched
- Works great for developers who **know exactly what to search for**

# Grep-based Concept Location Example

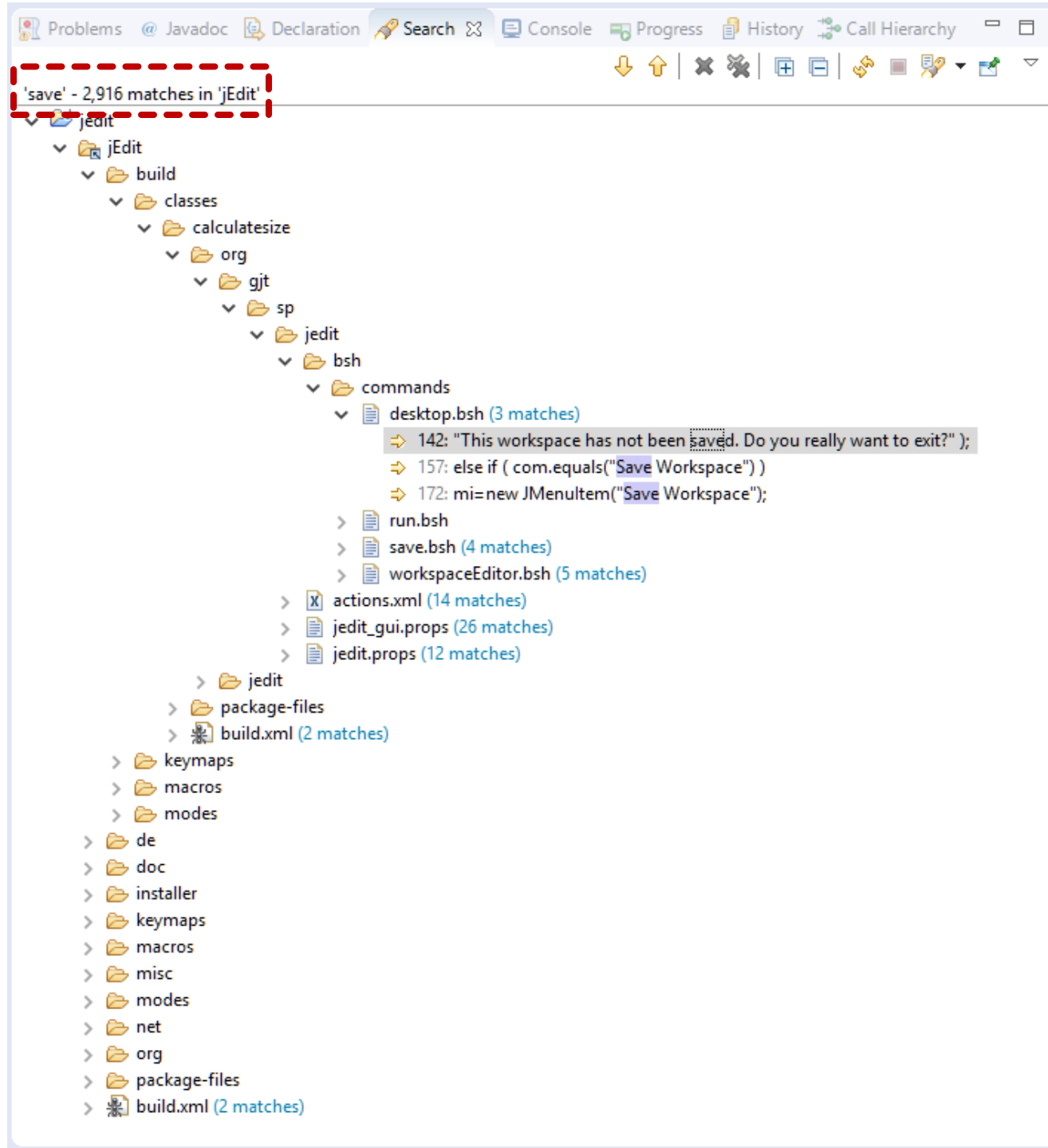
query = "save"





# Grep-based Concept Location Example

~3,000 results



# Limitations of Grep-based search

- Query formulation (**regular expression**) is more difficult than natural language (e.g., **google searches**)
- Results **unranked**
- **Exact matches only**
  - i.e., cannot find synonyms, or conceptually similar topics

# How Can We Do Better?

Google

Google Search

I'm Feeling Lucky

bing

Web | Images | Videos | More

YAHOO!

Web

Images

Video

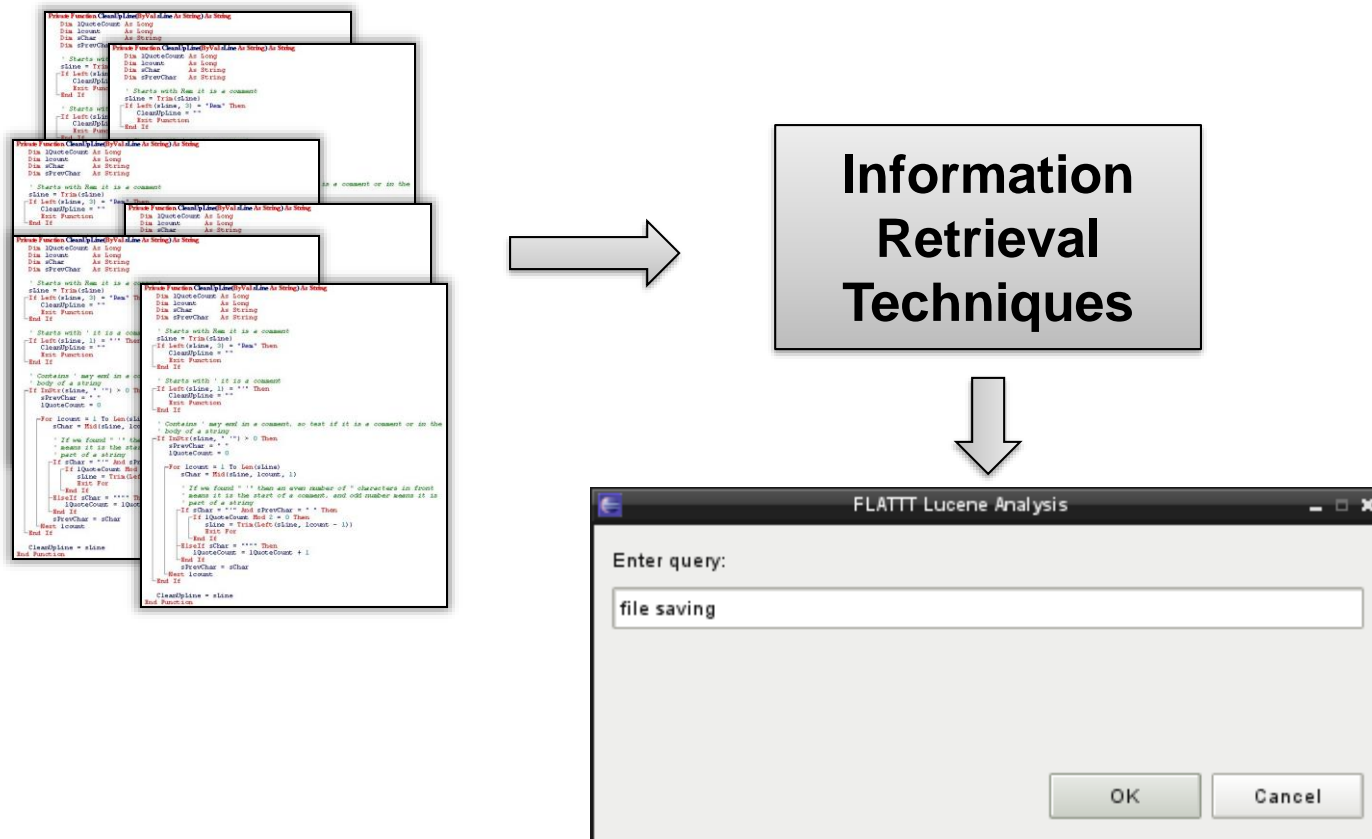
Local

Shopping

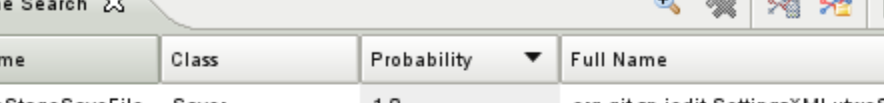
More ▾

Web Search





# FLAT3




The screenshot shows the 'Lucene Search' application window. It contains a table with the following columns: Name, Class, Probability, and Full Name. The table lists several search results, with the 'save' entry highlighted in blue.

	Name	Class	Probability	Full Name
❑	twoStageSaveFile	Saver	1.0	org.gjt.sp.jedit.SettingsXML::twoStageSaveFile
⊙	SAVE_DIALOG	VFSBrowser	0.98176175	org.gjt.sp.jedit.browser.VFSBrowser::SAVE_DIALOG
⊙	save	KillRing	0.8797569	org.gjt.sp.jedit.buffer.KillRing::save
⊙	saveAs	Buffer	0.7820565	org.gjt.sp.jedit.Buffer::saveAs
❑	Saver	Saver	0.75175285	org.gjt.sp.jedit.SettingsXML::Saver
⬆	Saver	Saver	0.7118754	org.gjt.sp.jedit.SettingsXML::Saver



# FLAT<sup>3</sup>: Feature Location and Textual Tracing Tool

- <http://www.cs.wm.edu/semeru/flat3/>
  - Contains 5 minute demo video
- Eclipse plug-in  (can be easily extended)
- **Static** feature location (uses Information Retrieval)
- **Dynamic** feature location (uses execution traces)
- Feature annotation



# 13

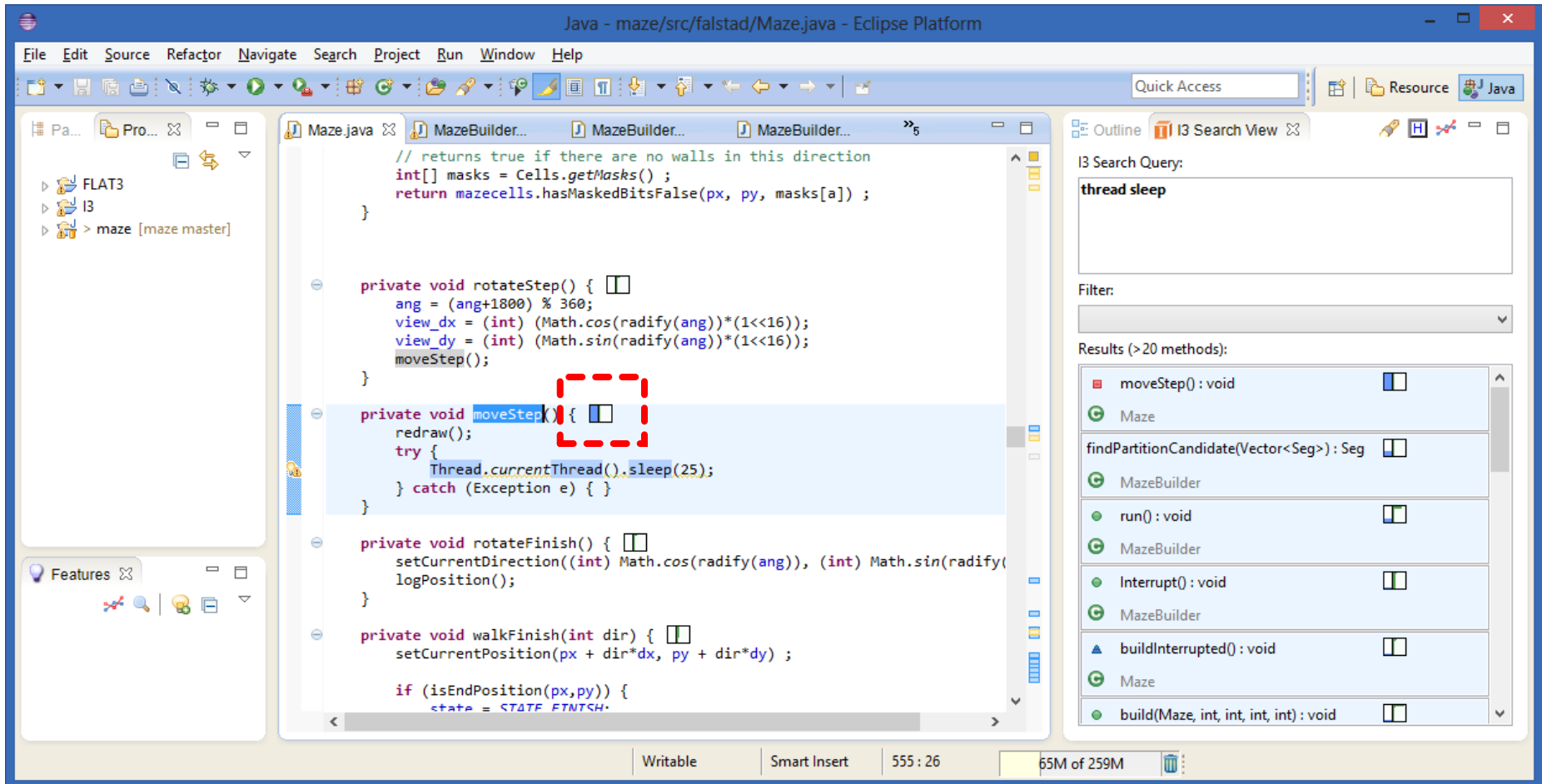
## Supports

- Feature Location
- Impact Analysis

In-situ visualization

Focus on user interface

# I3



findPartitionCandidate(Vector<Seg>): Seg

## Textual Similarity to Search Query

similarity: 0.34

### Keyword Frequencies:

grade<sub>7</sub> partit<sub>7</sub> sl<sub>6</sub> int<sub>5</sub> seg<sub>5</sub> segment<sub>5</sub> skip<sub>5</sub> i<sub>4</sub> pk<sub>4</sub> sl.size<sub>4</sub>  
0<sub>3</sub> bestgrad<sub>3</sub> pe<sub>3</sub> thread<sub>3</sub> valu<sub>3</sub> 50<sub>2</sub> current<sub>2</sub> determin<sub>2</sub> find<sub>2</sub>  
findpartitioncandid<sub>2</sub> maxtri<sub>2</sub> maze<sub>2</sub> maze.increasepercentag<sub>2</sub>  
minimum<sub>2</sub> pk.partit<sub>2</sub> size<sub>2</sub> sl.elementat<sub>2</sub> thread.currentthread<sub>2</sub>  
use<sub>2</sub> vector<sub>2</sub> 1<sub>1</sub> 100<sub>1</sub> 100/expect<sub>1</sub> 31<sub>1</sub> 5000<sub>1</sub> access<sub>1</sub> all<sub>1</sub> bit<sub>1</sub>  
candid<sub>1</sub> chanc<sub>1</sub> comput<sub>1</sub> consid<sub>1</sub> constant<sub>1</sub> consum<sub>1</sub> dure<sub>1</sub> e<sub>1</sub>  
element<sub>1</sub> event<sub>1</sub> except<sub>1</sub> generat<sub>1</sub> give<sub>1</sub> has<sub>1</sub> here<sub>1</sub> high<sub>1</sub>  
increas<sub>1</sub> initi<sub>1</sub> keyboard<sub>1</sub> main<sub>1</sub> most<sub>1</sub> need<sub>1</sub> number<sub>1</sub> observ<sub>1</sub>  
occasion<sub>1</sub> onli<sub>1</sub> param<sub>1</sub> part<sub>1</sub> percentag<sub>1</sub> process<sub>1</sub> proport<sub>1</sub>  
random<sub>1</sub> screen<sub>1</sub> seem<sub>1</sub> set<sub>1</sub> sleep<sub>1</sub> smallest<sub>1</sub> some<sub>1</sub> subset<sub>1</sub>  
time<sub>1</sub> tri<sub>1</sub> updat<sub>1</sub> which<sub>1</sub>

## Textually Similar Methods

genNodes(Vector<Seg>): BSPNode  
MazeBuilder  
grade\_partition(Vector<Seg>, Seg): int  
MazeBuilder  
countNonPartitions(Vector<Seg>): int  
MazeBuilder  
run(): void  
MazeBuilder  
setPartitionBitForC...Vector<Seg>): void  
MazeBuilder  
traverse\_ssector(BSPLeaf): void

```
private void moveStep() {  
    redraw();  
    try {  
        Thread.currentThread().sleep(25);  
    } catch (Exception e) { }  
}  
  
private void rotateFinish() {  
    setCurrentDirection((int) Math.cos(radify(ang)), (int) Math.sin(radify(  
    logPosition();  
})  
  
private void walkFinish(int dir) {  
    setCurrentPosition(px + dir*dx, py + dir*dy);  
  
    if (isEndPosition(px,py)) {  
        state = STATE_FINISH;  
    }  
}
```

## I3 Search Query:

thread sleep

Filter:

Results (> 20 methods):

moveStep(): void  
Maze  
findPartitionCandidate(Vector<Seg>): Seg  
MazeBuilder  
run(): void  
MazeBuilder  
Interrupt(): void  
MazeBuilder  
buildInterrupted(): void  
Maze  
build(Maze, int, int, int, int): void

Writable

Smart Insert

555 : 26

65M of 259M



findPartitionCandidate(Vector<Seg>): Seg

similarity: 0.34

Keyword Frequencies:

grade<sub>7</sub> partit<sub>7</sub> sl<sub>6</sub> int<sub>5</sub> seg<sub>5</sub> segment<sub>5</sub> skip<sub>5</sub> i<sub>4</sub> pk<sub>4</sub> sl.size<sub>4</sub>  
 0<sub>3</sub> bestgrad<sub>3</sub> pe<sub>3</sub> thread<sub>3</sub> valu<sub>3</sub> 50<sub>2</sub> current<sub>2</sub> determin<sub>2</sub> find<sub>2</sub>  
 findpartitioncandid<sub>2</sub> maxtri<sub>2</sub> maze<sub>2</sub> maze.increasepercentag<sub>2</sub>  
 minimum<sub>2</sub> pk.partit<sub>2</sub> size<sub>2</sub> sl.elementat<sub>2</sub> thread.currentthread<sub>2</sub>  
 use<sub>2</sub> vector<sub>2</sub> 1<sub>1</sub> 100<sub>1</sub> 100/expect<sub>1</sub> 31<sub>1</sub> 5000<sub>1</sub> access<sub>1</sub> all<sub>1</sub> bit<sub>1</sub>  
 candid<sub>1</sub> chanc<sub>1</sub> comput<sub>1</sub> consid<sub>1</sub> constant<sub>1</sub> consum<sub>1</sub> dure<sub>1</sub> e<sub>1</sub>  
 element<sub>1</sub> event<sub>1</sub> except<sub>1</sub> generat<sub>1</sub> give<sub>1</sub> has<sub>1</sub> here<sub>1</sub> high<sub>1</sub>  
 increas<sub>1</sub> initi<sub>1</sub> keyboard<sub>1</sub> main<sub>1</sub> most<sub>1</sub> need<sub>1</sub> number<sub>1</sub> observ<sub>1</sub>  
 occasion<sub>1</sub> onli<sub>1</sub> param<sub>1</sub> part<sub>1</sub> percentag<sub>1</sub> process<sub>1</sub> proport<sub>1</sub>  
 random<sub>1</sub> screen<sub>1</sub> seem<sub>1</sub> set<sub>1</sub> sleep<sub>1</sub> smallest<sub>1</sub> some<sub>1</sub> subset<sub>1</sub>  
 time<sub>1</sub> tri<sub>1</sub> updat<sub>1</sub> which<sub>1</sub>

Textually Similar Methods

- genNodes(Vector<Seg>): BSPNode
- MazeBuilder
- grade\_partition(Vector<Seg>, Seg): int
- MazeBuilder
- countNonPartitions(Vector<Seg>): int
- MazeBuilder
- run(): void
- MazeBuilder
- setPartitionBitForC...Vector<Seg>): void
- MazeBuilder
- traverse\_ssector(BSPLeaf): void

Shows other ranked methods that are textually similar to the current method

Shows why the current method is similar to the search query

moveStep() {

currentThread().sleep(25);

ception e) { }

ateFinish() {

rection((int) Math.cos(radify(ang)), (int) Math.sin(radify(

);

kFinish(int dir) {

sition(px + dir\*dx, py + dir\*dy) ;

sition(px,py)) {

STATE FINISH

Writable Smart Insert 555 : 26 65M of 259M

Filter:

Results (> 20 methods):

- moveStep(): void
- Maze
- findPartitionCandidate(Vector<Seg>): Seg
- MazeBuilder
- run(): void
- MazeBuilder
- Interrupt(): void
- MazeBuilder
- buildInterrupted(): void
- Maze
- build(Maze, int, int, int, int): void

Analyzes historical information and shows a **ranked list of methods** that were **changed frequently** with the current method across multiple commits

The screenshot shows an IDE interface with a 'Change History' and 'Co-Changed Methods' panel. The 'Change History' panel displays a list of changes for the method 'createPartControl(Composite) : void'. The 'Co-Changed Methods' panel displays a list of methods that were changed frequently with the current method across multiple commits.

**Change History**

5 changes

**Last 7 days**

Rev. 37 (09/02/13 11:23) by Fabian.Beck@visus.uni-stuttgart.de  
simple list of co-changed methods (not yet ordered)

**Last 4 weeks (excluding last 7 days)**

Rev. 15 (08/22/13 12:52) by Fabian.Beck@visus.uni-stuttgart.de  
bugfix: result list refresh; search independent of flat3

Rev. 14 (08/21/13 10:18) by Fabian.Beck@visus.uni-stuttgart.de  
result list improved

Rev. 10 (08/20/13 12:43) by Fabian.Beck@visus.uni-stuttgart.de  
IRToolTips tag cloud and list of methods (dummy version)

Rev. 8 (08/16/13 01:34) by Fabian.Beck@visus.uni-stuttgart.de  
packages restructured; started to implement tooltips for search

**Co-Changed Methods**

updateResultsLabel...st<Method>) : void  
G SearchView

update(SearchJob) : void  
G SearchView

update(SearchJob) : void  
G I3

createContentRight(...posite) : Composite  
G IRToolTip

createDialogArea(Composite) : Control  
G A ToolTip

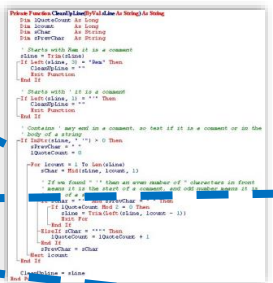
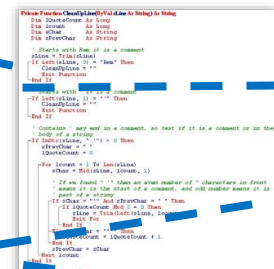
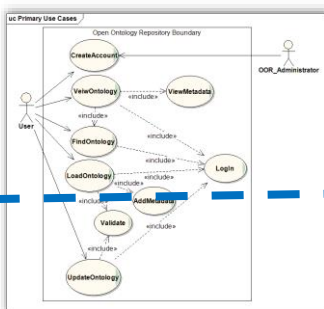
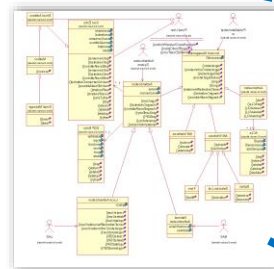
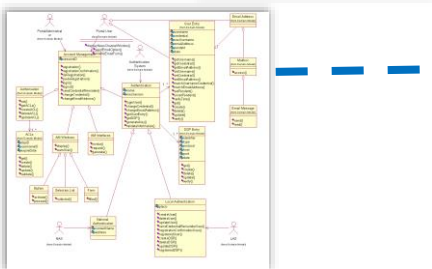
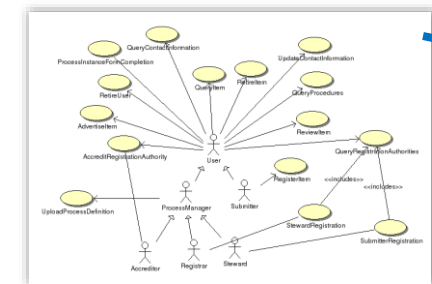
createContentLeft(C...posite) : Composite

# Traceability Link Recovery





# Traceability Link Recovery



```

Python Program to Check if a String is a Palindrome or Not
// Python Program to Check if a String is a Palindrome or Not
// Importing the sys module
import sys

// Function to check if a string is a palindrome or not
def is_palindrome(s):
    // Base case: if the string is empty or has one character, it is a palindrome
    if len(s) <= 1:
        return True

    // Check if the first and last characters are the same
    if s[0] != s[-1]:
        return False

    // Recursive call to check the substring from index 1 to len(s)-2
    return is_palindrome(s[1:-1])

// Main function
def main():
    // Taking input from the user
    s = input("Enter a string: ")

    // Calling the is_palindrome function
    result = is_palindrome(s)

    // Printing the result
    if result:
        print("The string is a palindrome.")
    else:
        print("The string is not a palindrome.")

// Calling the main function
if __name__ == '__main__':
    main()

```

[illegible]

```

#Find the word(s) which is/are the longest in the
#the input string
def longestWord(s):
    #split the string into words
    words = s.split()
    #store the longest word in a variable
    longestWord = ""
    #iterate over the words
    for word in words:
        #if the word is longer than the current longest word
        if len(word) > len(longestWord):
            #update the longest word
            longestWord = word
    #return the longest word
    return longestWord

#Test the function
s = "The quick brown fox jumps over the lazy dog"
print(longestWord(s))

```

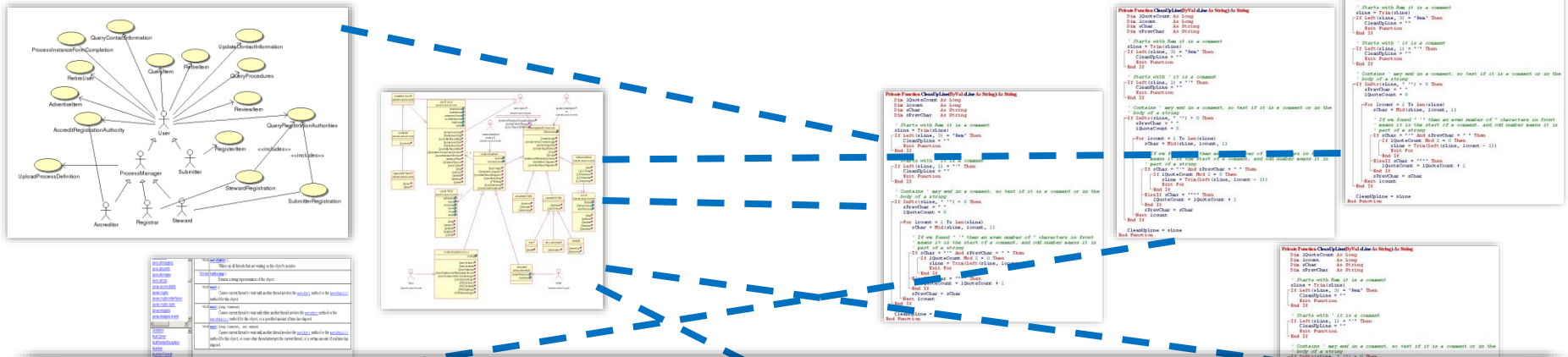
```

1 # Python Program to Check if a String is Palindrome or Not
2
3 def isPalindrome(s):
4     # Convert string to lower case
5     s = s.lower()
6
7     # Remove all spaces from string
8     s = s.replace(" ", "")
9
10    # Check if string is equal to its reverse
11    if s == s[::-1]:
12        return True
13    else:
14        return False
15
16 # Example usage
17 s = "A man a plan a canal Panama"
18
19 # Check if the string is a palindrome
20 result = isPalindrome(s)
21
22 # Print the result
23 if result:
24     print("The string is a palindrome.")
25 else:
26     print("The string is not a palindrome.")
27
28 # Output: The string is a palindrome.
29
30
31 # Python Program to Check if a String is Palindrome or Not
32
33 def isPalindrome(s):
34     # Convert string to lower case
35     s = s.lower()
36
37     # Remove all spaces from string
38     s = s.replace(" ", "")
39
40     # Check if string is equal to its reverse
41     if s == s[::-1]:
42         return True
43     else:
44         return False
45
46 # Example usage
47 s = "A man a plan a canal Panama"
48
49 # Check if the string is a palindrome
50 result = isPalindrome(s)
51
52 # Print the result
53 if result:
54     print("The string is a palindrome.")
55 else:
56     print("The string is not a palindrome.")
57
58 # Output: The string is a palindrome.
59
60
61 # Python Program to Check if a String is Palindrome or Not
62
63 def isPalindrome(s):
64     # Convert string to lower case
65     s = s.lower()
66
67     # Remove all spaces from string
68     s = s.replace(" ", "")
69
70     # Check if string is equal to its reverse
71     if s == s[::-1]:
72         return True
73     else:
74         return False
75
76 # Example usage
77 s = "A man a plan a canal Panama"
78
79 # Check if the string is a palindrome
80 result = isPalindrome(s)
81
82 # Print the result
83 if result:
84     print("The string is a palindrome.")
85 else:
86     print("The string is not a palindrome.")
87
88 # Output: The string is a palindrome.
89
90
91 # Python Program to Check if a String is Palindrome or Not
92
93 def isPalindrome(s):
94     # Convert string to lower case
95     s = s.lower()
96
97     # Remove all spaces from string
98     s = s.replace(" ", "")
99
100    # Check if string is equal to its reverse
101    if s == s[::-1]:
102        return True
103    else:
104        return False
105
106 # Example usage
107 s = "A man a plan a canal Panama"
108
109 # Check if the string is a palindrome
110 result = isPalindrome(s)
111
112 # Print the result
113 if result:
114     print("The string is a palindrome.")
115 else:
116     print("The string is not a palindrome.")
117
118 # Output: The string is a palindrome.
119
120
121 # Python Program to Check if a String is Palindrome or Not
122
123 def isPalindrome(s):
124     # Convert string to lower case
125     s = s.lower()
126
127     # Remove all spaces from string
128     s = s.replace(" ", "")
129
130     # Check if string is equal to its reverse
131     if s == s[::-1]:
132         return True
133     else:
134         return False
135
136 # Example usage
137 s = "A man a plan a canal Panama"
138
139 # Check if the string is a palindrome
140 result = isPalindrome(s)
141
142 # Print the result
143 if result:
144     print("The string is a palindrome.")
145 else:
146     print("The string is not a palindrome.")
147
148 # Output: The string is a palindrome.
149
150
151 # Python Program to Check if a String is Palindrome or Not
152
153 def isPalindrome(s):
154     # Convert string to lower case
155     s = s.lower()
156
157     # Remove all spaces from string
158     s = s.replace(" ", "")
159
160     # Check if string is equal to its reverse
161     if s == s[::-1]:
162         return True
163     else:
164         return False
165
166 # Example usage
167 s = "A man a plan a canal Panama"
168
169 # Check if the string is a palindrome
170 result = isPalindrome(s)
171
172 # Print the result
173 if result:
174     print("The string is a palindrome.")
175 else:
176     print("The string is not a palindrome.")
177
178 # Output: The string is a palindrome.
179
180
181 # Python Program to Check if a String is Palindrome or Not
182
183 def isPalindrome(s):
184     # Convert string to lower case
185     s = s.lower()
186
187     # Remove all spaces from string
188     s = s.replace(" ", "")
189
190     # Check if string is equal to its reverse
191     if s == s[::-1]:
192         return True
193     else:
194         return False
195
196 # Example usage
197 s = "A man a plan a canal Panama"
198
199 # Check if the string is a palindrome
200 result = isPalindrome(s)
201
202 # Print the result
203 if result:
204     print("The string is a palindrome.")
205 else:
206     print("The string is not a palindrome.")
207
208 # Output: The string is a palindrome.
209
210
211 # Python Program to Check if a String is Palindrome or Not
212
213 def isPalindrome(s):
214     # Convert string to lower case
215     s = s.lower()
216
217     # Remove all spaces from string
218     s = s.replace(" ", "")
219
220     # Check if string is equal to its reverse
221     if s == s[::-1]:
222         return True
223     else:
224         return False
225
226 # Example usage
227 s = "A man a plan a canal Panama"
228
229 # Check if the string is a palindrome
230 result = isPalindrome(s)
231
232 # Print the result
233 if result:
234     print("The string is a palindrome.")
235 else:
236     print("The string is not a palindrome.")
237
238 # Output: The string is a palindrome.
239
240
241 # Python Program to Check if a String is Palindrome or Not
242
243 def isPalindrome(s):
244     # Convert string to lower case
245     s = s.lower()
246
247     # Remove all spaces from string
248     s = s.replace(" ", "")
249
250     # Check if string is equal to its reverse
251     if s == s[::-1]:
252         return True
253     else:
254         return False
255
256 # Example usage
257 s = "A man a plan a canal Panama"
258
259 # Check if the string is a palindrome
260 result = isPalindrome(s)
261
262 # Print the result
263 if result:
264     print("The string is a palindrome.")
265 else:
266     print("The string is not a palindrome.")
267
268 # Output: The string is a palindrome.
269
270
271 # Python Program to Check if a String is Palindrome or Not
272
273 def isPalindrome(s):
274     # Convert string to lower case
275     s = s.lower()
276
277     # Remove all spaces from string
278     s = s.replace(" ", "")
279
280     # Check if string is equal to its reverse
281     if s == s[::-1]:
282         return True
283     else:
284         return False
285
286 # Example usage
287 s = "A man a plan a canal Panama"
288
289 # Check if the string is a palindrome
290 result = isPalindrome(s)
291
292 # Print the result
293 if result:
294     print("The string is a palindrome.")
295 else:
296     print("The string is not a palindrome.")
297
298 # Output: The string is a palindrome.
299
300
301 # Python Program to Check if a String is Palindrome or Not
302
303 def isPalindrome(s):
304     # Convert string to lower case
305     s = s.lower()
306
307     # Remove all spaces from string
308     s = s.replace(" ", "")
309
310     # Check if string is equal to its reverse
311     if s == s[::-1]:
312         return True
313     else:
314         return False
315
316 # Example usage
317 s = "A man a plan a canal Panama"
318
319 # Check if the string is a palindrome
320 result = isPalindrome(s)
321
322 # Print the result
323 if result:
324     print("The string is a palindrome.")
325 else:
326     print("The string is not a palindrome.")
327
328 # Output: The string is a palindrome.
329
330
331 # Python Program to Check if a String is Palindrome or Not
332
333 def isPalindrome(s):
334     # Convert string to lower case
335     s = s.lower()
336
337     # Remove all spaces from string
338     s = s.replace(" ", "")
339
340     # Check if string is equal to its reverse
341     if s == s[::-1]:
342         return True
343     else:
344         return False
345
346 # Example usage
347 s = "A man a plan a canal Panama"
348
349 # Check if the string is a palindrome
350 result = isPalindrome(s)
351
352 # Print the result
353 if result:
354     print("The string is a palindrome.")
355 else:
356     print("The string is not a palindrome.")
357
358 # Output: The string is a palindrome.
359
360
361 # Python Program to Check if a String is Palindrome or Not
362
363 def isPalindrome(s):
364     # Convert string to lower case
365     s = s.lower()
366
367     # Remove all spaces from string
368     s = s.replace(" ", "")
369
370     # Check if string is equal to its reverse
371     if s == s[::-1]:
372         return True
373     else:
374         return False
375
376 # Example usage
377 s = "A man a plan a canal Panama"
378
379 # Check if the string is a palindrome
380 result = isPalindrome(s)
381
382 # Print the result
383 if result:
384     print("The string is a palindrome.")
385 else:
386     print("The string is not a palindrome.")
387
388 # Output: The string is a palindrome.
389
390
391 # Python Program to Check if a String is Palindrome or Not
392
393 def isPalindrome(s):
394     # Convert string to lower case
395     s = s.lower()
396
397     # Remove all spaces from string
398     s = s.replace(" ", "")
399
400     # Check if string is equal to its reverse
401     if s == s[::-1]:
402         return True
403     else:
404         return False
405
406 # Example usage
407 s = "A man a plan a canal Panama"
408
409 # Check if the string is a palindrome
410 result = isPalindrome(s)
411
412 # Print the result
413 if result:
414     print("The string is a palindrome.")
415 else:
416     print("The string is not a palindrome.")
417
418 # Output: The string is a palindrome.
419
420
421 # Python Program to Check if a String is Palindrome or Not
422
423 def isPalindrome(s):
424     # Convert string to lower case
425     s = s.lower()
426
427     # Remove all spaces from string
428     s = s.replace(" ", "")
429
430     # Check if string is equal to its reverse
431     if s == s[::-1]:
432         return True
433     else:
434         return False
435
436 # Example usage
437 s = "A man a plan a canal Panama"
438
439 # Check if the string is a palindrome
440 result = isPalindrome(s)
441
442 # Print the result
443 if result:
444     print("The string is a palindrome.")
445 else:
446     print("The string is not a palindrome.")
447
448 # Output: The string is a palindrome.
449
450
451 # Python Program to Check if a String is Palindrome or Not
452
453 def isPalindrome(s):
454     # Convert string to lower case
455     s = s.lower()
456
457     # Remove all spaces from string
458     s = s.replace(" ", "")
459
460     # Check if string is equal to its reverse
461     if s == s[::-1]:
462         return True
463     else:
464         return False
465
466 # Example usage
467 s = "A man a plan a canal Panama"
468
469 # Check if the string is a palindrome
470 result = isPalindrome(s)
471
472 # Print the result
473 if result:
474     print("The string is a palindrome.")
475 else:
476     print("The string is not a palindrome.")
477
478 # Output: The string is a palindrome.
479
480
481 # Python Program to Check if a String is Palindrome or Not
482
483 def isPalindrome(s):
484     # Convert string to lower case
485     s = s.lower()
486
487     # Remove all spaces from string
488     s = s
```

# Design Documents

## Source Code

# Traceability Link Recovery



**The Problem Traceability Link Recovery is trying to solve:**

What are the **documents/artifacts** associated with a given **source code** component?

# Design Documents

## Source Code

# Traceability can be Between

- Requirements and code
- Design and code
- Requirements and design
- Requirements and test cases
- Design and test cases
- Bug report and maintainer
- Manual page to code
- ....

# Traceability Link Recovery “Definition”\*

- The ability to describe and follow the life of a requirement, in both a forward and backward direction:
  - from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases.

# Why Traceability?

# Why Traceability?

- It is **required** or **suggested** by many standards:
  - MIL-STD-498, IEEE/EIA 12207 (Military)
  - DO178B, DO254 (Avionic), EN50128 (Railways)
  - Medical, financial, etc.
- Contractual agreements to ensure that:
  - **all required functionalities are there**
  - **there is no EXTRA functionality**
- Facilitates:
  - **bottom-up** and **top-down** program comprehension
  - **Impact analysis**
    - Identification of **reusable software components**
- Easy to implement with modern tools (e.g., GitHub)

# The basic Assumption

- Developers use **consistent naming conventions** to
  - create identifiers
  - write comments
  - name artifacts
  - write manual pages or e-mails
- Developers use **domain concepts** and **knowledge** in a **uniform** and **consistent** way

# Example – A Requirement

It shall be possible to insert a new book into the library. The user shall insert the following information:

- The **title** of the book;
- The names of the **authors**;
- The **number** of **pages** of the book.

After confirming the data inserted, data are stored into the database. In case of abort, nothing shall happen.



# Example of Mapping

```
public class Book  
{
```

```
String Auth;
```

```
String Title;
```

```
int NPages;
```

authors

title

number of pages

```
public class AddBook extends  
JFrame {
```

```
JPanel contentPane;
```

```
GridBagLayout gridBagLayout1 = new  
GridBagLayout();
```

```
JTextField Auth = new JTextField();
```

```
JLabel jLabel1 = new JLabel();
```

```
JTextField T = new JTextField();
```

```
JLabel jLabel2 = new JLabel();
```

```
JTextField Npages = new JTextField();
```

```
JLabel jLabel3 = new JLabel();
```

```
JButton Cancel = new JButton();
```

```
JButton Add = new JButton();
```

```
Border border1;
```

```
....
```

Authors:

Title:

N. of pages:

# Challenges for Traceability Link Recovery Techniques

- High level documents mostly in natural language
  - source code
  - acronyms
  - abbreviations
  - etc.
- Automatic-generated code

# Challenges for Traceability Link Recovery Techniques

- Conceptual distance between different artifacts
  - High level requirement vs. code or test cases
- Vocabulary inconsistency
  - fault, defect, bug, issue, ...
- Text sparseness
  - there is no better data than more data

How will Traceability be Implemented in  
CS471/CS481?

# Linking Tasks to Stories

- Always link a task to a story (e.g., #123), where 123 is the story ID

# Linking Tasks to Stories

- Always link a task to a story (e.g., #123), where 123 is the story ID
  - but my task doesn't have a story to link...

# Linking Tasks to Stories

- Always link a task to a story (e.g., #123), where 123 is the story ID
  - but my task doesn't have a story to link...
    - Did you forget to create the story?

# Linking Tasks to Stories

- Always link a task to a story (e.g., #123), where 123 is the story ID
  - but my task doesn't have a story to link...
    - Did you forget to create the story?
  - working on an “independent” task (i.e., not linked to a story) is a good sign you're going off on a tangent...so stop it and reassess



# Referencing a new task to existing story #1

The screenshot shows a web browser window with the URL <https://github.com/BoiseState/CS481-Test/issues/new>. The page is titled "BoiseState/CS481-Test New Issue" and is powered by ZenHub. A ZenHub overlay is visible, showing a task titled "Setup database" with the description "Create a MySQL database schema." A red dashed box highlights the text "Links to #1" in the description, with a red annotation "//sample linking task to story 1" next to it. The right sidebar shows the issue's metadata, including Pipeline (New Issues), Assignees (bgdndit), Labels (None yet), Milestone (Sprint 1), Estimate (2), and Epics (Not inside an Epic). At the bottom of the overlay, there are buttons for "Create an epic" and "Submit new issue".

BoiseState/CS481-Test New Issue

powered by | ZenHub

Setup database

Write Preview

Create a MySQL database schema.

Links to #1 //sample linking task to story 1

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

Create an epic Submit new issue

Pipeline New Issues

Assignees bgdndit

Labels None yet

Milestone Sprint 1

Estimate 2

Epics Not inside an Epic

# Task #10 linked to Story #1

BoiseState/CS481-Test#10 Setup database

powered by ZenHub

## Setup database #10

Open bgdndit opened this issue just now · 0 comments

bgdndit commented just now

Create a MySQL database schema.

Links to #1

bgdndit added this to the Sprint 1 milestone just now

bgdndit self-assigned this just now

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

Close issue Comment

Pipeline

New Issues

Assignees

bgdndit

Labels

None yet

Projects

None yet

Milestone

Sprint 1

Estimate

2

Epics

Not inside an Epic

# Benefits of Linking (story #1 points to task #10)

The screenshot shows a GitHub issue page for "Create account · Issue #1". The issue description is: "As a new user, I need to create an account so that the server can authenticate potential users." The acceptance criteria are listed as follows:

- ☐ Given a user name in the form of an email address, when the button to sign up is clicked, an email will be sent to the email address to confirm the user owns that email address.
- ☐ Given a user name in the form of an email address, when the button to sign up is clicked, a verification on the server will be performed to ensure that the email address provided is unique and it was not already associated with any other account.
- ☐ Given a password, when the user clicks the sign up button, a verification of the password containing at least 6 characters is performed.
- ☐ Given a password, when the user clicks the sign up button, a verification of the password containing both upper and lower case characters is performed.

The issue has a "story" label and is part of the "Sprint 1" milestone. The estimate is set to 5. The issue is currently "Closed".

The issue history shows the following actions:

- bgdndit added the story label a day ago
- bgdndit added this to the Sprint 1 milestone a day ago
- bgdndit set the estimate to 5 a day ago
- bgdndit changed the pipeline from New Issues to Backlog a day ago

The issue was referenced a day ago. The tasks are:

- Task 1 for Story 1 - Illustrate Linking Commit #4 (Closed)
- Task 2 for Story 1 #6 (Closed)
- Task 3 for Story 1 #7 (Closed)
- Setup database #10 (Open)

The right sidebar shows the issue details, including the Assignees (No one—assign yourself), Labels (story), Projects (None yet), Milestone (Sprint 1), Estimate (5), Epics (Not inside an Epic), Notifications (Unsubscribe), and 1 participant.

# A story is split into multiple tasks (each with its owner): It's easy to track progress when using linking between software artifacts

**Acceptance Criteria:**

- ☐ Given a user name in the form of an email address, when the button to sign up is clicked, an email will be sent to the email address to confirm the user owns that email address.
- ☐ Given a user name in the form of an email address, when the button to sign up is clicked, a verification on the server will be performed to ensure that the email address provided is unique and it was not already associated with any other account.
- ☐ Given a password, when the user clicks the sign up button, a verification of the password containing at least 6 characters is performed.
- ☐ Given a password, when the user clicks the sign up button, a verification of the password containing both upper and lower case characters is performed.

bgdndit added the **story** label a day ago

bgdndit added this to the **Sprint 1** milestone a day ago

bgdndit set the estimate to **5** a day ago

bgdndit changed the pipeline from **New Issues** to **Backlog** a day ago

This was referenced a day ago

Task 1 for Story 1 - Illustrate Linking Commit #4

Task 2 for Story 1 #6

Task 3 for Story 1 #7

Setup database #10

**Assignees**

No one—assign yourself

**Labels**

**story**

**Projects**

None yet

**Milestone**

**Sprint 1**

**Estimate**

**5**

**Epics**

Not inside an Epic

**Notifications**

**Unsubscribe**

You're receiving notifications because you authored the thread.

1 participant

Lock conversation

# In the commit log message specify the task/bug ID to associate the implementation with the ID

The screenshot shows a GitHub pull request interface for a repository named "CS481-Test". The pull request is titled "testPRBranch" and is currently in the "Changes" tab. The "Changes" tab shows a single change in the file "src/Main.java". The change is highlighted in green and shows the following code:

```
@@ -1,5 +1,10 @@
public class Main
{
+   private void demoMethodTask19()
+   {
+       //implementation of task #19 in testPRBranch
+   }
+   private void demoMethodIllustrateCommitLink()
+   {
+       //initial implementation of
```

The commit message is being edited in the "Commit" section. The current message is "implementation of task #". A dropdown menu is open, showing a list of tasks/bugs that can be linked to the commit message:

- #19 Sample Task to Illustrate a Pull Request
- #17 Task 4 Story 1
- #1 Create account
- #15 Customize ZenHub
- #11 Story Six

The "Commit" button is visible at the bottom of the commit message section.

Description of PR should include a reference to the task ID. Title only is not sufficient to make the link

The screenshot displays a GitHub Pull Request (PR) interface. On the left, a sidebar shows repository filters: 'Filter repositories', 'GitHub', 'CS481-Test', and 'Other' with a 'Tutorial' link. The main area is titled 'testPRBranch' and shows a comparison between 'master' and 'testPRBranch'. A commit titled 'implementation of task #19' by 'bgdndit' is selected. The diff for 'src/Main.java' is shown, with changes highlighted in green. The right sidebar shows the 'Pull request' details, including the title 'implementation of task #19' and a description box. A red dashed rectangle highlights the title field. Below the description box, a message states 'This pull request can be automatically merged.' and a 'Send pull request' button is visible, with a mouse cursor hovering over it.

Filter repositories

GitHub

CS481-Test

Other

Tutorial

testPRBranch

implementation of task #19  
1 minute ago by bgdndit

implementation of task #19  
bgdndit 04d51ef GitHub Revert Cc

src/Main.java

...	...	@@ -1,5 +1,10 @@
1	1	public class Main
2	2	{
	3 +	private void demoMethodTask19()
	4 +	{
	5 +	//implementation of task #19 in testPRBranch
	6 +	}
	7 +	
3	8	private void demoMethodIllustrateCommitLink()
4	9	{
5	10	//initial implementation of

Pull request

from testPRBranch into master

implementation of task #19

implementation of task #19

This pull request can be automatically merged.

Send pull request

Your local commits will be synced to GitHub.

# How will Traceability be Implemented in CS<sub>471</sub>/CS<sub>481</sub>?

- Link tasks to stories
- Link commit log messages to tasks/bugs
- Link Pull-Requests to tasks