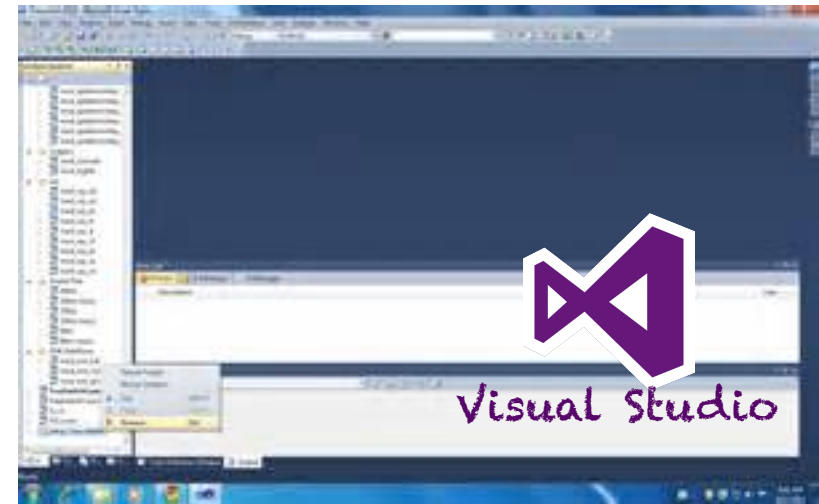
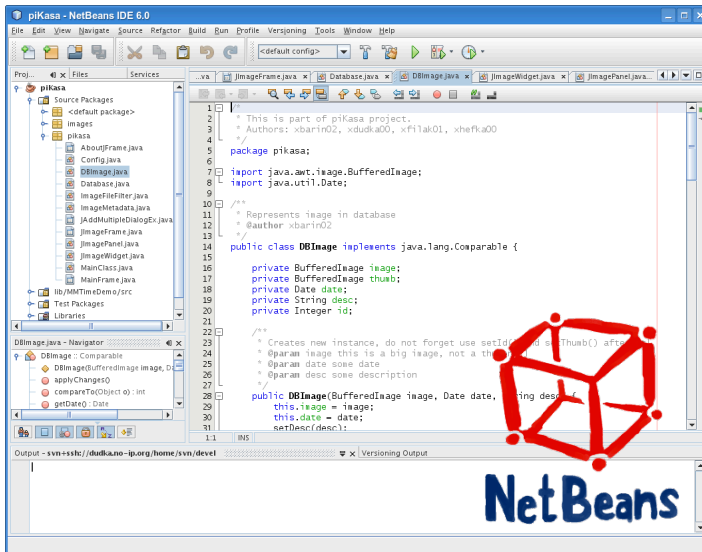
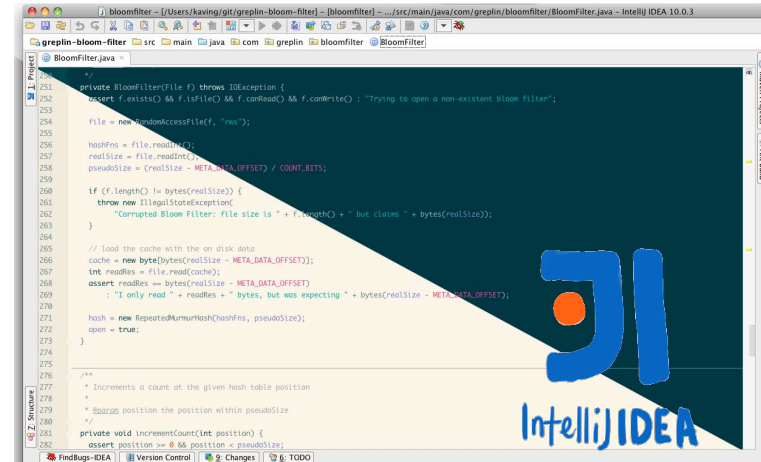
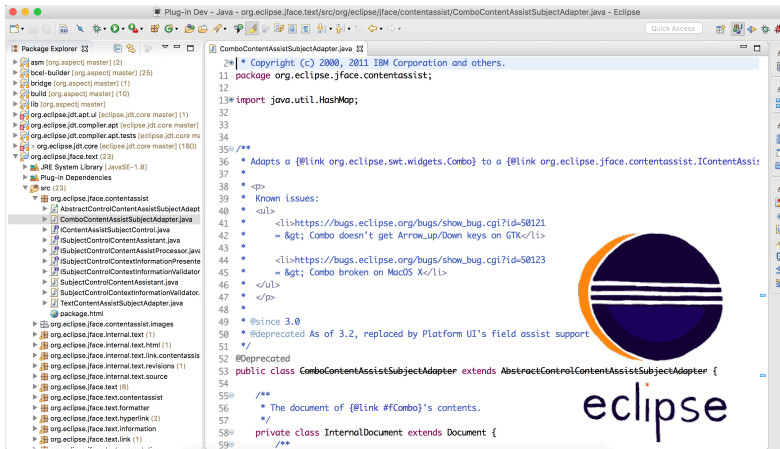


# Producing Productive Programmers

Using research to improve developer productivity








```

test.java + (~) - VIM1
public class test{
    public static void main(){
        String test = "test";
    }
}

```

5 lines indented

1,1 All



```

emacs [abc.py -- /home/gbrj/dev/python/Lib/abc.py]
File Edit Options Buffers Tools IM-Python Python Outline ECB Help
collections.py abc.py os.py
# Copyright 2007 Google, Inc. ALL RIGHTS RESERVED.
# Licensed to PSF under a Contributor Agreement.

"""Abstract Base Classes (ABCs) according to PEP 3119."""

def abstractmethod(funcobj):
    """A decorator indicating abstract methods.

    Requires that the metaclass is ABCMeta or derived from it. A
    class that has a metaclass derived from ABCMeta cannot be
    instantiated unless all of its abstract methods are overridden.
    The abstract methods can be called using any of the normal
    'super' call mechanisms.

    Usage:

        class C(metaclass=ABCMeta):
            @abstractmethod
            def my_abstract_method(self, ...):
                ...

    """
    funcobj.__isabstractmethod__ = True
    return funcobj

class abstractproperty(property):
    """A decorator indicating abstract properties.


    Requires that the metaclass is ABCMeta or derived from it. A
    class that has a metaclass derived from ABCMeta cannot be
    instantiated unless all of its abstract properties are overridden.
    The abstract properties can be called using any of the normal
    'super' call mechanisms.

    Usage:

        class C(metaclass=ABCMeta):
            @abstractproperty
            def my_abstract_property(self):
                ...

    """

```



Global list of 2000 items of type: ALL  
 Available with "M": (0)ALL (1)TODO (2)DONE  
 python: 2000 Fix 2000 items in the documentation  
 python: 2000 Reduce the number of open issues  
 python: 2000 Backport the ABC docs


Python 2.5.1 (r251:54863, Jan 21 2008, 18:23:45)  
 [GCC 4.2.2 (Ge... 2.2 pl-0)] on linux2  
 Type "help()", "copyright()", "credits()" or "license()" for  
 >>>

Emacs

```

*D:\source\notepad4ever.cpp - Note
Notepad_plus.cpp notepad4ever.cpp
1 #include <GPL.h>
2 #include <free_software.h>
3
4 void notepad4ever ()
5 {
6     while (true)
7     {
8         Notepad++;
9     }
10
11

```



FingerText



Jedi



Clang



Smex

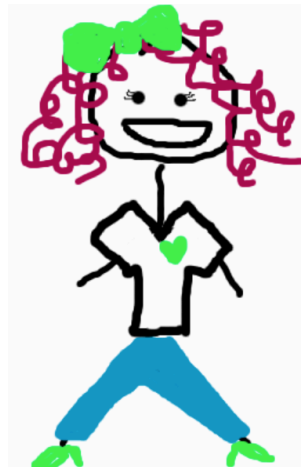


ctrlp.vim



# Goal

Design better tools.





# Topic #1 :

## Understanding program analysis tool use



# Research Questions

RQ<sub>1</sub>: What reasons do developers have for using and not using static analysis tools to find bugs?

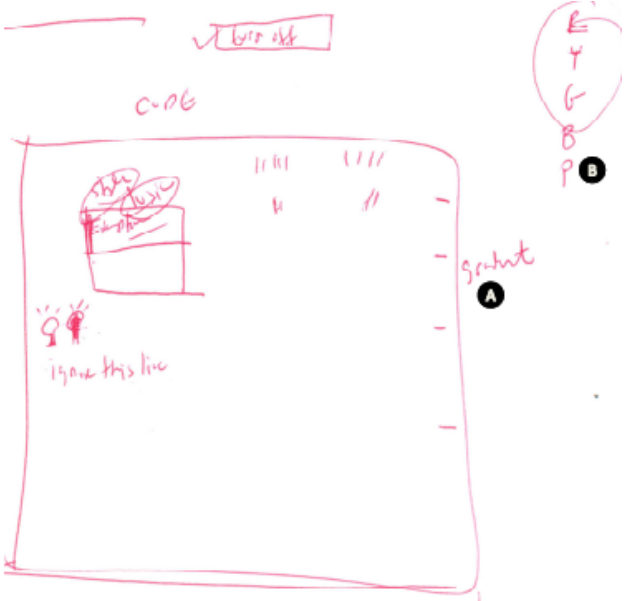
RQ<sub>2</sub>: How well do current tools integrate with developer workflows?

RQ<sub>3</sub>: What improvements would developers like to see made to their tools?

# Methodology

**Question**  
*Tell me about particularly easy for you first experience ...*

**Short Response**



Reasons for using and not using  
Integration with Workflow  
Suggested Improvements

# Reasons for Use and Underuse

## Tool Output

### **Jake**

...like I mentioned with FlexLint it gives you so many warnings and sifting through them is so... arduous that whenever I just look at it I'm like eh hh forget this.

## User Input and Customizability

### **Andy**

Like you know it's like is this list prioritized by you know what's important to me? No. You know? And there may be a default listing that should be prioritized because like this one's inefficient

# Reasons for Use and Underuse

## Supporting Teamwork

### **John**

The only reason I like the batch results is to communicate, broadcast to the team a sense of progress or lack of progress.

## Result Understandability

### **Matt**

so I click in there I think and it gives me a light bulb and it says ok so now I wanna know why raising a string exception is bad. Like what should I be doing instead? Since it thinks it's a problem. And so none of these really help me.

# Workflow Integration

## Workflows

### **Mike**

Clang is my favorite. It's built in , into the compiler. You don't have to invoke anything special

# Suggested Improvements

## Tool Design

### **Chris**

I don't mind the idea of the actual source code itself having some plasticity to it so that let's say the fourth line there was some error here...having the 5<sup>th</sup> line drop down and having the content expand with maybe all sorts of annotations about my code.

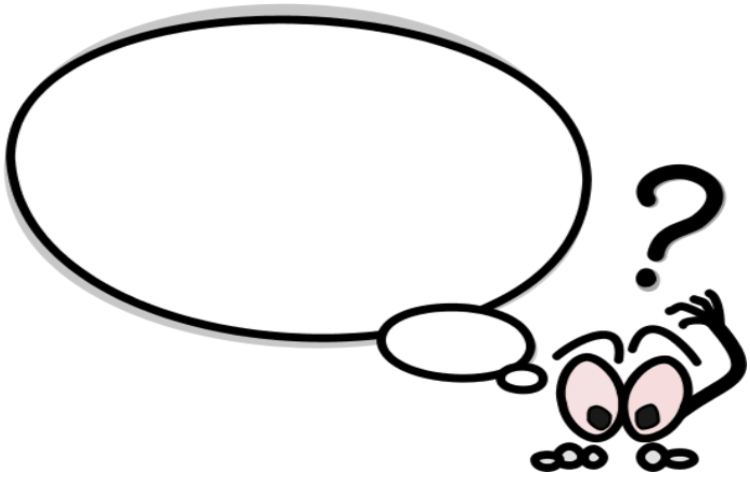
Tool Output

Result Understandability

User Input/Customizability

Collaboration Support

Workflow



# Interesting Improvements

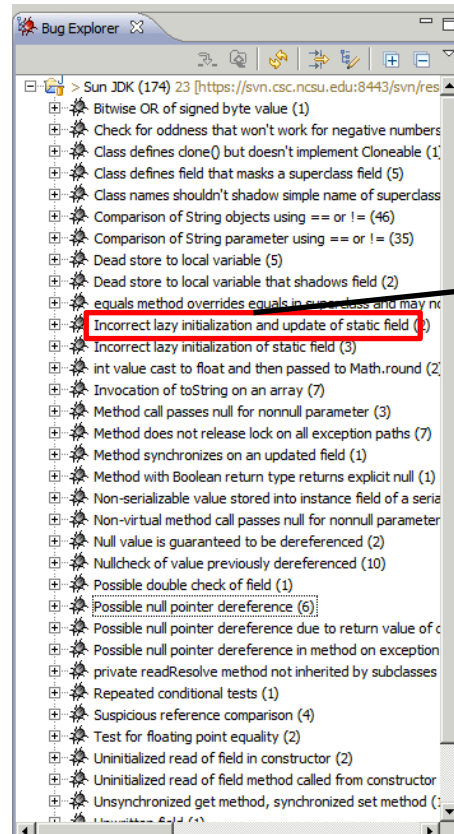
```
public String getAddress() throws  
String result = null;  
try {  
    InetAddress address = InetAddress  
    byte[] ip = address.getAddress()  
    int i = 4;  
    String ipAddress = "";  
    StringBuffer stringBuffer = new  
        ipAddress);  
    for (byte b : ip) {  
        stringBuffer.append((b & 0xFF));  
        if (--i > 0) {  
            stringBuffer.append(".");  
        }  
    }  
    ipAddress = stringBuffer.toString();  
    result = ipAddress;  
} catch (UnknownHostException e) {  
    e.printStackTrace();  
}  
return result;  
}
```

(Original)  
 Use StringBuffer  
 Use StringBuilder

**i** You can:  
- select an option (●).  
- rename `variable` names.



# Primary Reason for Underuse



```
95 if (managingFocusForwardTraversalKeys == null) {  
96     managingFocusForwardTraversalKeys = new HashSet<KeyStroke>();  
97     managingFocusForwardTraversalKeys.add(  
98         KeyStroke.getKeyStroke(KeyEvent.VK_TAB, 0));  
99 }
```

This method contains an unsynchronized lazy initialization of a static field. After the field is set, the object stored into that location is further updated or accessed. The setting of the field is visible to other threads as soon as it is set. If the further accesses in the method that set the field serve to initialize the object, then you have a *very serious multi-threading bug*, unless something else prevents any other thread from accessing the stored object until it is fully initialized.

Even if you feel confident that the method is never called by multiple threads, it might be better to not set the static field until the value you are setting it to is fully populated/initialized.

# Problem #2:

We know developers have trouble with tool output, but we don't know why.

# Research Question

RQ: Why do developers encounter challenges when interpreting program analysis tool notifications?

# Methodology



```
Java Browsing - Sun JDK/src/javawx/swing/JTree.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help

JTree.java
4601 // AccessibleContext methods
4602
4603 /**
4604  * Get the accessible name of this object.
4605  *
4606  * @return the localized name of the object; null if this
4607  * object does not have a name
4608  */
4609 //TODO 1 FB Task 1
4610 public String getAccessibleName() {
4611     AccessibleContext ac = getCurrentAccessibleContext();
4612     if (ac != null) {
4613         String name = ac.getAccessibleName();
4614         if ((name != null) && (name.equals("")) ) {
4615             return ac.getAccessibleName();
4616         } else {
4617             return null;
4618         }
4619     }
4620     if ((accessibleName != null) && (accessibleName != "")) {
4621         return accessibleName;
4622     } else {
4623         // fall back to the client property
4624         return (String)getClientProperty(AccessibleContext.ACCESS
4625     }
4626 }
4627
4628 /**
4629  * Set the localized accessible name of this object.
4630  *
4631  * @param s the new localized name of the object.
4632  */
4633 public void setAccessibleName(String s) {
4634     AccessibleContext ac = getCurrentAccessibleContext();
```

!	Description	Resource
	TODO 1 FB Task 1	JTree.jav
	TODO 2 FB Task 2	SynthSpli
	TODO 3 FB Task 3	DefaultSt
	TODO 3 FB Task 4	DefaultSt
	TODO 5 FB Task 5	Wrappedi
	TODO COMP Task 1	ContextLi
	TODO COMP Task 2	ContextLi
	TODO COMP Task 3	DirectByt
	TODO COMP Task 4	DynAnyBi
	TODO COMP Task 5	AbstractI
	TODO COMP Task 6	AbstractI
	TODO ECL Class 1	PlotUtili
	TODO ECL Class 2	XYCrossh.
	TODO ECL Class 3	NormalDis
	TODO ECL Class 4	XYLine3Df
	TODO ECL Class 5	WeekTest
	TODO ECL Class 6	CustomCe

Writable Smart Insert 4614 : 49

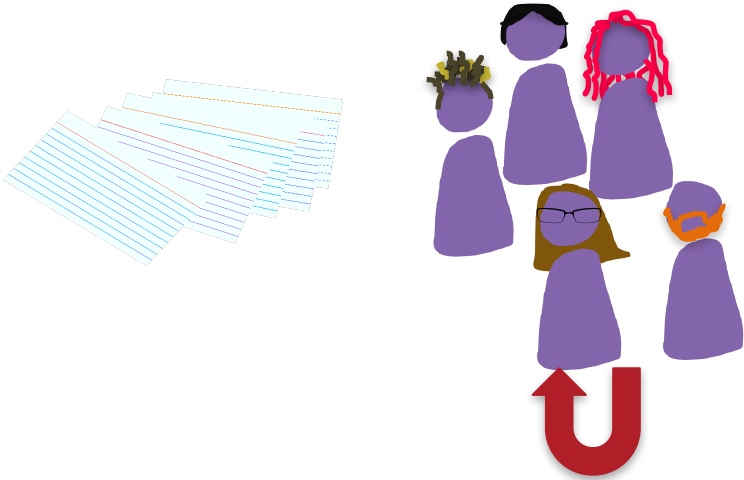


8:30 AM 11/4/2016

# Methodology



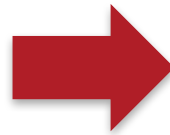
- Explicit challenge statement
- Unable to explain or interpret
- Info needed outside notification



- General Knowledge Gaps
- Conceptual Knowledge Gaps
- Notification Experience Gaps
- Problem Importance Gaps
- Problem Resolution Gaps
- General Problem Description Mismatches
- Information Salience Mismatches
- Visual Communication Mismatches
- Consistent Communication Mismatches
- Familiar Communication Mismatches

# Findings Validation – Member Check

General Knowledge Gaps  
Conceptual Knowledge Gaps  
Notification Experience Gaps  
Problem Importance Gaps  
Problem Resolution Gaps  
General Problem Description Mismatches  
Information Salience Mismatches  
Visual Communication Mismatches  
Consistent Communication Mismatches  
Familiar Communication Mismatches



Our findings align with your day to day experiences with interpreting and resolving tool notifications. \*

1      2      3      4      5

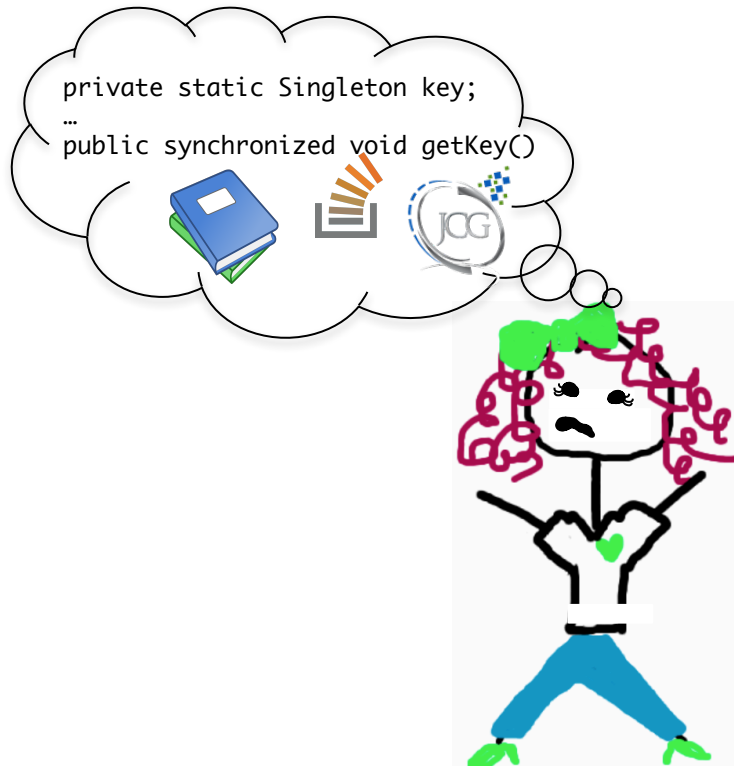
Strongly Disagree                  Strongly Agree

Anything you would like to add regarding how our findings align with your experiences?

Long answer text

---

# Findings - Theory



This method contains an unsynchronized lazy initialization of a static field. After the field is set, the object stored into that location is further updated or accessed. The setting of the field is visible to other threads as soon as it is set. If the further accesses in the method that set the field serve to initialize the object, then you have a *very serious multi-threading bug*, unless something else prevents any other thread from accessing the stored object until it is fully initialized.

Even if you feel confident that the method is never called by multiple threads, it might be better to not set the static field until the value you are setting it to is fully populated/initialized.

If she hasn't experienced it, or can't make the connection to her experience, she struggles.

# Findings - Themes

## Knowledge Gaps

General Knowledge Gaps  
Conceptual Knowledge Gaps  
**Notification Experience Gaps**  
Problem Importance Gaps  
**Problem Resolution Gaps**

## Knowledge Mismatches

General Problem Description Mismatches  
**Information Salience Mismatches**  
**Visual Communication Mismatches**  
Consistent Communication Mismatches  
Familiar Communication Mismatches



```
95  if (managingFocusForwardTraversalKeys == null) {
96      managingFocusForwardTraversalKeys = new HashSet<KeyStroke>();
97      managingFocusForwardTraversalKeys.add(
98          KeyStroke.getKeyStroke(KeyEvent.VK_TAB, 0));
99  }
```

### **Incorrect lazy initialization and update of static field javax... managingFocusForwardTraversalKeys in javax...installDefaults()**

This method contains an unsynchronized lazy initialization of a static field. After the field is set, the object stored into that location is further updated or accessed. The setting of the field is visible to other threads as soon as it is set. If the further accesses in the method that set the field serve to initialize the object, then you have a *very serious multi-threading bug*, unless something else prevents any other thread from accessing the stored object until it is fully initialized.

Even if you feel confident that the method is never called by multiple threads, it might be better to not set the static field until the value you are setting it to is fully populated/initialized.

# Problem Resolution Gaps

*“It’s not immediately clear to me how you would fix it. I mean what they say is well don’t initialize it until you have the value to store in it ready **but I’m not sure.**” – P24*

# Notification Experience Gaps

*“I’m not really sure what I’m looking at, mainly because I’m not really familiar ... I’m not sure if I’ve ran into this once...update of static field...I can’t really recollect exactly.” – P13*

# Information Saliency Mismatches

*“Yeah, this (description) is helpful. This (tool tip) per se is not very helpful but this (description) is...in my case, I may not have particularly used this type of or static variables...It’s like oh yeah okay...it’s a couple of clicks away.” – P13*

# Visual Communication Mismatches

*“As for the reason why this is yellow, maybe it’s because you can enter the finally block either from a try or from an exception or something. I don’t know and it’s indicating we’re only coming through when an exception is thrown. Maybe...um...why are the colors different?” – P16*

```
296         try {
297             Week w = new Week(new Date(1136109830000L),
298                 TimeZone.getTimeZone("GMT"));
299             assertEquals(2005, w.getYearValue());
300             assertEquals(52, w.getWeek());
301             assertFalse(true);
302         }
303         finally {
304             Locale.setDefault(saved);
305         }
306     }
```

# Problem #3 :

We know why developers have trouble with tool output, but not how we can fix it.

Johnson, B., Pandita, R., Murphy-Hill, E., Heckman, S. "Bespoke tools: adapted to the concepts developers know." Proceedings of 2015 10<sup>th</sup> Joint Meeting on Foundations of Software Engineering – ESEC/FSE 2015, pp. 878-881

Johnson, B., Pandita, R., Murphy-Hill, E., Heckman, S., Menzies, T., Knowing How Much Developers Know (Without Having to Ask!): Predicting Developer Conceptual Knowledge Using Public Code Repositories. ICSE 2018 (in submission)

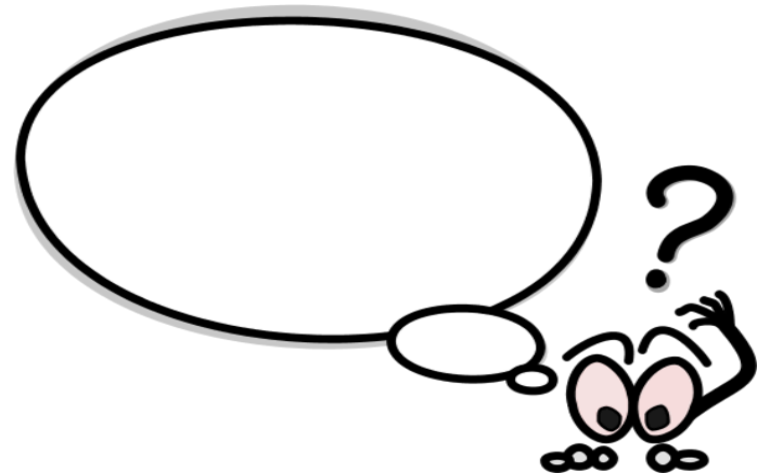
## Knowledge

### Gaps

General Knowledge Gaps  
Conceptual Knowledge Gaps  
Notification Experience Gaps  
Problem Importance Gaps  
Problem Resolution Gaps

## Knowledge Mismatches

General Problem Description Mismatches  
Information Salience Mismatches  
Visual Communication Mismatches  
Consistent Communication Mismatches  
Familiar Communication Mismatches

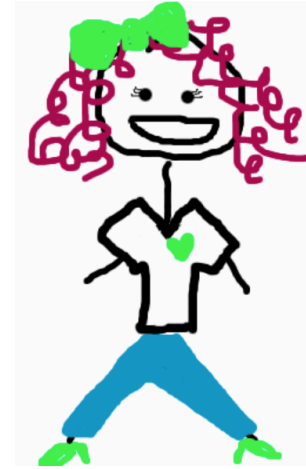


# Hypotheses

Primary source of knowledge is experience

Developer experience = source code

Assess what developers know



$H_1$ : We can predict concept knowledge using source code contributions.

$H_2$ : Concept-specific contributions increases model performance.



# Methodology

## Source code as experience (predictor)

Concept-specific source code  
Code ownership

```
watson -- ataulm -- Ataul Munim
https://github.com/ataulm/watson.git
git clone https://github.com/ataulm/watson.git
.\watson\
.\watson\.git
0
Project cloned!
Ataul Munim is responsible for commit a93ea917564873b42f9e23f711a73d1100107c62
Ataul Munim is responsible for commit e7005c7968b5effe07638c297541ddf83f27a42d
Ataul Munim is responsible for commit 7af039b421f6b17d1abfc74d8c9748983ab065de
Ataul Munim is responsible for commit 43b1490951f4d26f87e96daa6a69e3aadaac21eb
Ataul Munim is responsible for commit 8a96459a419b1c0959024833ab9a1925e8fd2b0f
Ataul Munim is responsible for commit 00571eb528d28d7b4ade33993fce8ba760c64b0c
Ataul Munim is responsible for commit af2ef98f0d93af746fd07f4d903d47ab0456101b
Ataul Munim is responsible for commit 16ba7897a8563a99d7ef29fb2e15473b4b7c38
```



```
385     private static Func1<List<Character>, Cast> asCast() {
386         return new Func1<List<Character>, Cast>() {
387
388             @Override
389             public Cast call(List<Character> characters) {
390                 return new Cast(characters);
391             }
392
393         };
394     }
```

```
*****Analysis complete*****|
Ataul Munim added type argument method count = 249
--> recency = months
Ataul Munim added wildcard count = 37
--> recency = months
Ataul Munim added type declaration count = 363
--> recency = months
Ataul Munim added type parameter method count = 38
--> recency = months
Ataul Munim added type parameter field count = 0
--> recency = null
Ataul Munim added diamond count = 27
--> recency = months
Ataul Munim added method invocation count = 3
--> recency = months
Ataul Munim added implicit method invocation count = 30
--> recency = months
Ataul Munim added class instantiation count = 130
--> recency = months
Ataul Munim added nested count = 69
--> recency = year
Ataul Munim added bounds count = 0
--> recency = null
```

# Methodology

## Concept Inventories for knowledge assessment (ground truth)

Define conceptual content  
Build bank of test questions  
Pilot questions  
Establish validity and reliability

What change(s) needs to be made to properly bind the generic type parameter U to String in the following code:

```
public <U> void method(U u){ ... }
```

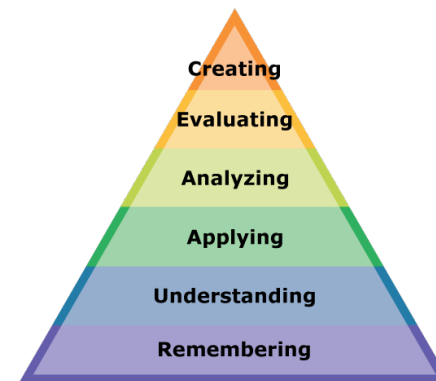
- `public <U> void method(String u){ ... }`
- `public <String> void method(U u){ ... }`
- `public <U extends String> void method(U u){ ... }`
- `public <U> void method((String)U u){ ... }`
- `public <U implements String> void method(U u){ ... }`

### Lesson: Generics

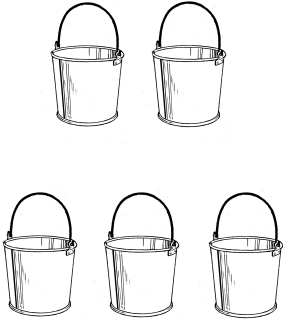
*by Gilad Bracha*

Introduced in J2SE 5.0, this long-awaited enhancement to adds compile-time type safety to the Collections Framework

[Introduction](#)  
[Defining Simple Generics](#)  
[Generics and Subtyping](#)  
[Wildcards](#)  
[Generic Methods](#)  
[Interoperating with Legacy Code](#)  
[The Fine Print](#)  
[Class Literals as Runtime-Type Tokens](#)  
[More Fun with Wildcards](#)  
[Converting Legacy Code to Use Generics](#)  
[Acknowledgements](#)



# Methodology



Novice

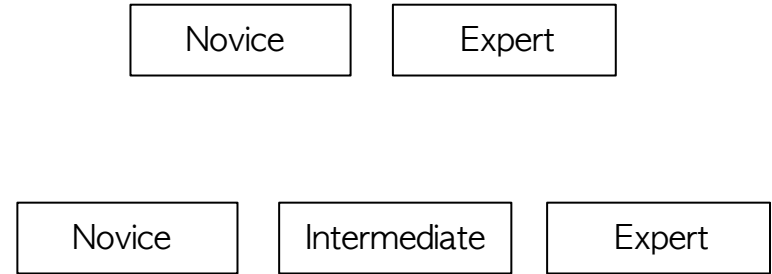
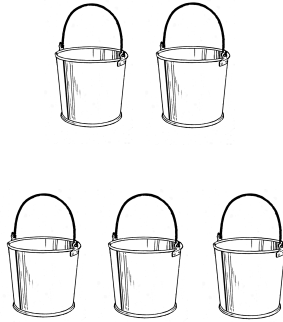
Expert

Novice

Intermediate

Expert

# Methodology



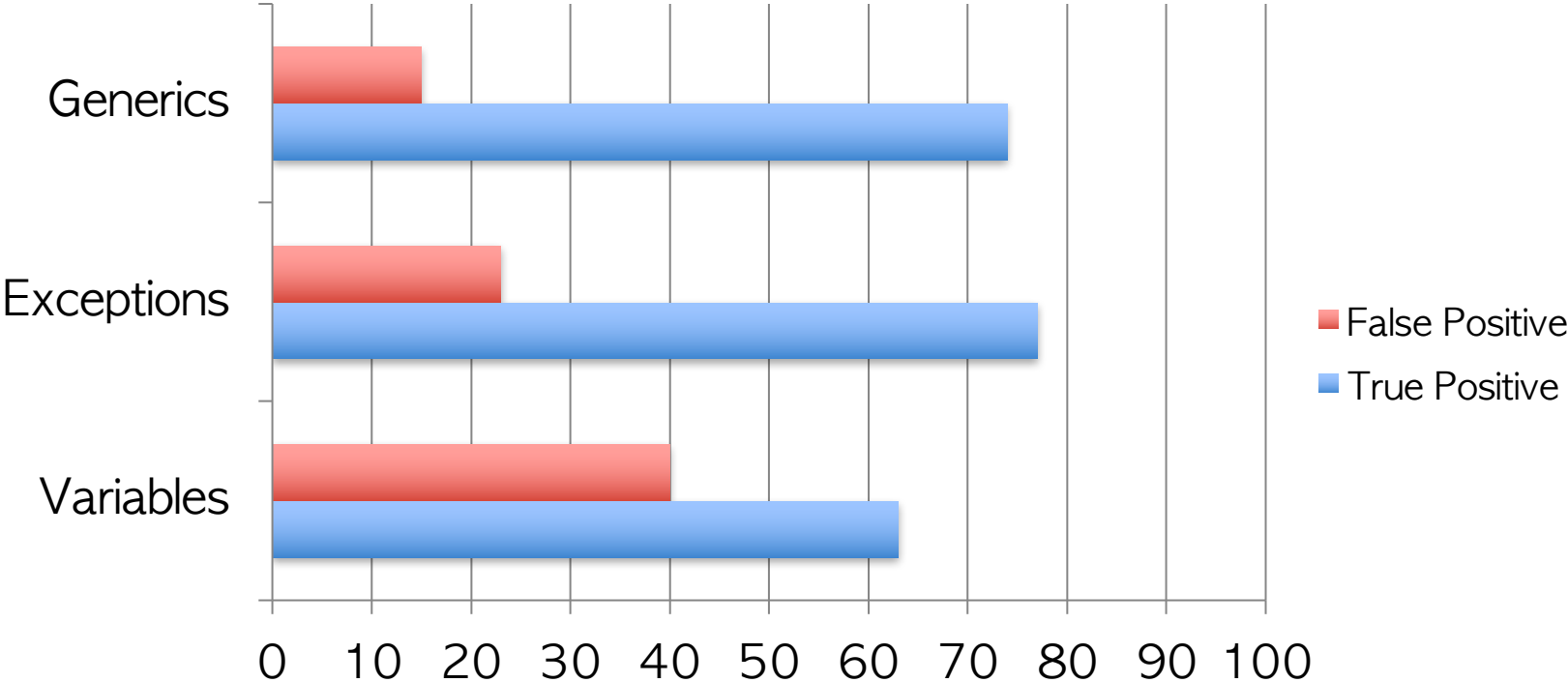
## Attributes

Variables – Public Variables

Exceptions - Throws Method, Try Statements, Finally Blocks, Advanced

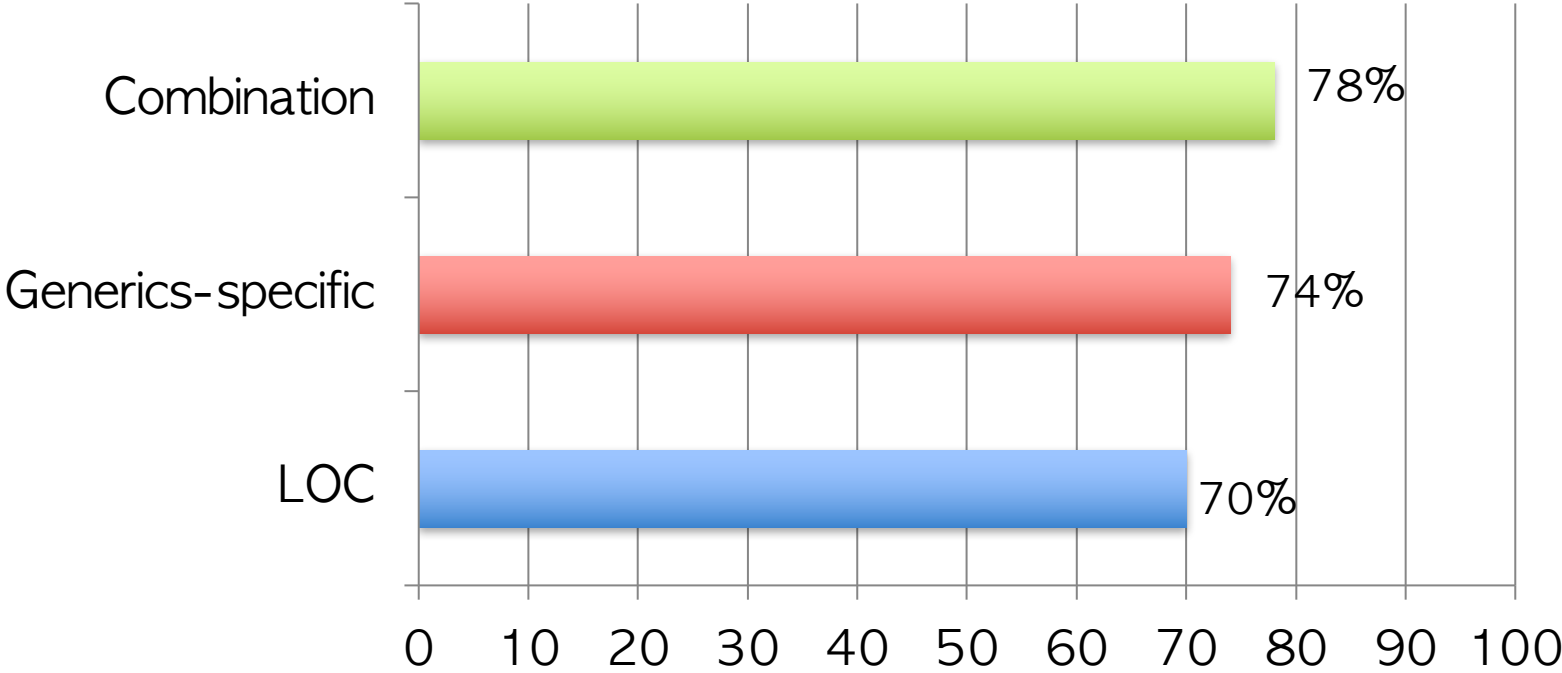
Generics - Generic Type Declarations, Total LOC

# Source Code as Predictor



# Concept-specific Improvements

Overall Classification Accuracy



## Source code (predictor)

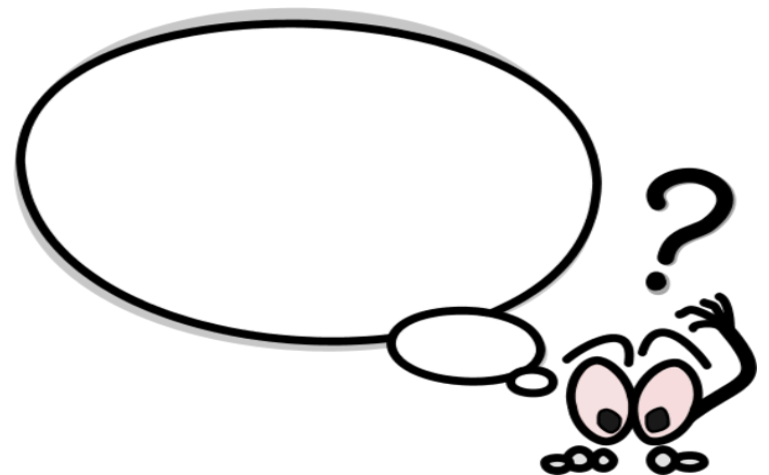
```
*****Analysis complete*****|
Ataul Munim added type argument method count = 249
--> recency = months
Ataul Munim added wildcard count = 37
--> recency = months
Ataul Munim added type declaration count = 363
--> recency = months
Ataul Munim added type parameter method count = 38
--> recency = months
Ataul Munim added type parameter field count = 0
--> recency = null
Ataul Munim added diamond count = 27
--> recency = months
Ataul Munim added method invocation count = 3
--> recency = months
Ataul Munim added implicit method invocation count = 30
--> recency = months
Ataul Munim added class instantiation count = 130
--> recency = months
Ataul Munim added nested count = 69
--> recency = year
Ataul Munim added bounds count = 0
--> recency = null
```

## Concept Inventories (ground truth)

What change(s) needs to be made to properly bind the generic type parameter U to String in the following code:

```
public <U> void method(U u){ ... }
```

- public <U> void method(String u){ ... }
- public <String> void method(U u){ ... }
- public <U extends String> void method(U u){ ... }
- public <U> void method((String)U u){ ... }
- public <U implements String> void method(U u){ ... }



# In-Class Activity!

Get into groups of 4

Come up with (software-related) topic group interested in

Examples:

Tools

Agile

Ethics

Come up with specific topic to research

Come up with 1-2 research questions

How might you answer each question?



Developers need tools that understand them.  
And now we know we can make it happen!



[bjohnson@cs.umass.edu](mailto:bjohnson@cs.umass.edu)



[@drbrittjay](https://twitter.com/drbrittjay)