Producing Productive Programmers

Using research to improve developer productivity
public class Test{
    public static void main(){
        String test = "test";
    }
}
Goal

Design better tools.
Topic #1:

Understanding program analysis tool use

Research Questions

RQ₁: What reasons do developers have for using and not using static analysis tools to find bugs?

RQ₂: How well do current tools integrate with developer workflows?

RQ₃: What improvements would developers like to see made to their tools?
Methodology

**Question**
Tell me about your 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 ehhh 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 5th 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

```java
public String getAddress() throws UnknownHostException {
    String result = null;
    try {
        InetAddress address = InetAddress.getByName(
            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;
```
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
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
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
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.
“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
“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
"Yeah, this (description) is helpful. This (tooltip) 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
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., 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

```java
private static Func1<List<Character>, Cast> asCast() {
    return new Func1<List<Character>, Cast>() {
        @Override
        public Cast call(List<Character> characters) {
            return new Cast(characters);
        }
    };
}
```

************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:

```java
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){ ... }
Methodology
Methodology

Attributes

Variables – Public Variables

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

Generics - Generic Type Declarations, Total LOC
Source Code as Predictor

Generics

Exceptions

Variables

0 10 20 30 40 50 60 70 80 90 100

False Positive
True Positive
Concept-specific Improvements

Overall Classification Accuracy

- Combination: 78%
- Generics-specific: 74%
- LOC: 70%
What change(s) needs to be made to properly bind the generic type parameter U to String in the following code:

```java
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
@drbrittjay