Producing Productive Programmers

Using research to improve developer productivity
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?

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

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

Suggested Improvements

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.
Primary Reason for Underuse

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

Findings Validation – Member Check

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

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.

Findings - Themes

Knowledge Gaps
- 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

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

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

“Yes, 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

Problem #3:

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

---

**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

H1: We can predict concept knowledge using source code contributions.
H2: Concept-specific contributions increases model performance.
**Methodology**

**Source code as experience (predictor)**

- Concept-specific source code
- Code ownership

**Concept Inventories for knowledge assessment (ground truth)**

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

---

**Attributes**

- **Variables** - Public Variables
- **Exceptions** - Throws Method, Try Statements, Finally Blocks, Advanced
- **Generics** - Generic Type Declarations, Total LOC
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
@dbrbrittjay