What We Have Here is a Failure to Communicate! Consequences & Improvement Requests

March 10, 2016

1 Introduction

We conducted a study to understand why developers encounter challenges when interpreting program analysis tool notifications. Using a miscommunication model, we defined criteria for identifying these challenges in transcripts from sessions with twenty-six developers with varying backgrounds. Although our criteria was designed to identify challenges, one criteria was general enough (inability to resolve the notification) to encapsulate some of the consequences that come with the challenges developers encounter. When participants could not explain a notification, some also provided tool improvement suggestions that might have helped them better understand the problem. In this short article, we discuss two consequences that can stem from the challenges developer encounter and tool improvements suggestions made by participants in our study.

2 Consequences

2.1 "Just Make It Go Away"

For six participants, when the text did not provide enough information regarding the problem or the fix options, they would opt to "just make it go away". By "just make it go away," we mean either applying a recommended fix or doing what the notification suggests without understanding why. Participants often mentioned that they only knew what to do about the notification because of the fix suggestions provided. $\boxed{\text{P22}}$ made a valiant effort to understand CMP4. However, he gave up on searching the notification and code for why the `serialversionUID` was needed when he realized there are recommended fixes he can choose from. Some participants immediately applied the fix when they realized it was available. Although applying the fix removed the notification, it did not clarify the problem with the code for any participants.

2.2 Lack of Trust in the Tool

For ten participants, if a tool did not communicate in a way that met their expectations, it caused their trust in the tool to waiver. For example, as P13 finished up with the set of notifications in ECL5, he began to realize how much the notifications provided by EclEmma differs from what he understands about exception handling. He mentioned his waivering trust in the tool, stating:

> Again now that I've seen this [notification] seems to be a little bit dodgy so I'm now doubting what it's telling me.

Sometimes, trust in the tool was affected when the participant did not understand or agree with the severity communicated by the tool. P4 , while explaining FB4 (Figure **??**), wanted to know why FindBugs marked the problem as scary because, to him, "it won't do any bad to check" for null. Similarly, P23 found FB5 to be much "scarier" of a problem than FB4, which has a severity of "scary" while FB5 is only "troubling." He searched some of the documentation to understand why FindBugs considers FB4 scary. However, he could not find anything to support the severity so he dismissed the notification and moved on.

3 Tool Feature Requests

Seven participants made suggestions regarding how tools could improve communication and better meet their expectations. One suggestion posed by P13 and P5 on multiple occasions during their sessions is that the tools should have more information readily available. This was even the case with FindBugs; P13 found himself in situations where he needed information FindBugs did not provide. For example, while attempting to interpret and explain FB3, P13 stated that he would "like to understand why this is a problem a bit more." He read the full description, searching for what is wrong with the way synchronization is currently being implemented, but could not find that information in the notification.

Both P5 and P13 also made mention of the compiler providing more information. P13 spoke from an experience standpoint, stating that if he was a junior Java developer, it would help him if the tool provided more information

> I would say maybe double click and it gives me the code conditions or javadoc. If I'm a junior Java developer, then definitely would help me out to read more about what this thing is.

Similarly, P22 , possibly recalling his experiences with FindBugs, expected the functionality of the compiler to be similar to that of FindBugs by providing detailed descriptions. While interpreting CMP4, he clicked the gutter icon to see if he could get any more details regarding the problem at hand. However, rather than providing more details, clicking the icon provides a list of action items.

He stated the following:

> Why doesn't this give a detailed description like...So when I hover over the mouse it gives me the error, but when I click on it it directly give me the error and the operations.

Another suggestion, made by $\boxed{\text{P15}}$ and $\boxed{\text{P16}}$, was that tools attempt to be more context-specific when communicating problems in their code. For example, while looking at ECL4, P16 attempted to explain what EclEmma was trying to tell him about coverage of the methods in that class. Eventually he began looking for where the methods in that class are called in the code and asked about viewing the test cases. P16 explained to us he was searching for where the methods are tested and in what context.

Participants also made some tool-specific suggestions. Four participants thought EclEmma could improve its notifications by improving its communication of control flow. P4, for example, was attempting to interpret ECL5 and found it challenging to understand what parts of program did not execute and why without control flow information being readily available. Most of the feature requests and suggestions came from more experienced participants, suggesting that perhaps with experience comes a stronger notion of preference regarding how the tools they use should work.