To Fix or to Learn? How Production Bias Affects Developers’ Information Foraging during Debugging

David Piorkowski¹, Scott D. Fleming², Christopher Scaffidi¹, Margaret Burnett¹, Irwin Kwan¹, Austin Z. Henley², Jamie Macbeth³, Charles Hill¹, Amber Horvath¹

¹Oregon State University
Corvallis, OR, USA
{piorkoda, cscaffid, burnett, kwan, hillc, horvatha}
@eecs.oregonstate.edu

²University of Memphis
Memphis, TN, USA
{Scott.Fleming, azhenley}
@memphis.edu

³Clemson University
Clemson, SC, USA
{jmacbet}
@clemson.edu

Abstract—Developers performing maintenance activities must balance their efforts to learn the code vs. their efforts to actually change it. This balancing act is consistent with the “production bias” that, according to Carroll’s minimalist learning theory, generally affects software users during everyday tasks. This suggests that developers’ focus on efficiency should have marked effects on how they forage for the information they think they need to fix bugs. To investigate how developers balance fixing versus learning during debugging, we conducted the first empirical investigation of the interplay between production bias and information foraging. Our theory-based study involved 11 participants: half tasked with fixing a bug, and half tasked with learning enough to help someone else fix it. Despite the subtlety of difference between their tasks, participants foraged remarkably differently—making foraging decisions from different types of “patches,” with different types of information, and succeeding with different foraging tactics.

Keywords—debugging; information foraging; theory meets tools

I. INTRODUCTION

Software maintenance invariably requires some learning of the current code before making changes, so as to plan correct and efficient changes. For example, one study suggested that methodical exploration of code prior to making maintenance changes can cut the time of actually coding in half [34]. Because of this connection between comprehension and maintenance, a wide range of tools aim at aiding program exploration and comprehension during maintenance and evolution (e.g., [2][11][12][13][21][25][42][43][44]).

It is therefore troubling that a recent observational field study of work practices revealed that “wherever possible, developers seem to prefer strategies that avoid comprehension” of existing code [35]. Specifically, this study found that developers frequently tried to move forward with coding as quickly as possible (what could be considered a form of satisficing [37]), with a minimal amount of activity invested ahead of time in exploring the code they were about to modify. Such results are consistent with those of other studies [3][17][23].

Based on such findings, Maalej et al. concluded:

Software comprehension is a hard and time-consuming task and consequently is avoided whenever possible. This indicates that Carroll’s minimalist theory [Carroll 1998], which suggests people put in the minimum effort to maximize

their outcome, is applicable … We think that researchers should consider developers as users and investigate how ‘user-developers’ analyze application behavior, how they relate observations to code, and how this behavior can be supported by tools [23].

This tendency of software developers to view learning as a costly task detracting from their efficiency is called production bias in Carroll’s theory [5][6]. To date, the effects of production bias on developers have not been investigated in detail.

In this paper, we investigate these effects and their implications for tool design. Our investigation is grounded not only in Carroll’s theory, but also in Information Foraging Theory (IFT) [29]. IFT provides a conceptual framework describing how people in an information environment, such as an IDE, seek information. For developers faced with a bug, the information they seek can include how to reproduce the bug, what causes the bug, where to fix the bug, and whether similar bugs were fixed elsewhere [23].

We performed our investigation by conducting a qualitative laboratory study. One group of developers was tasked with fixing a bug, whereas the other group was tasked with learning enough about the bug to help someone else fix it. We assigned people these differing tasks to reveal and analyze differences in their behaviors from both an IFT and a production bias perspective to address four research questions:

- **RQ1 (information goals):** How does trying to fix a bug versus trying to learn about the bug affect the types of information that developers seek?
- **RQ2 (information patches):** How does trying to fix a bug versus trying to learn about the bug affect where in the environment developers make foraging decisions?
- **RQ3 (information cues):** How does trying to fix a bug versus trying to learn about the bug affect the types of cues developers attend to when making foraging decisions?
- **RQ4 (foraging tactics):** How does trying to fix a bug versus trying to learn about the bug affect the tactics that developers use in making their foraging decisions?

Refer to the IEEE website for the remainder of the paper.