Research Statement

Dwayne "Chris" Brown, Jr.
North Carolina State University

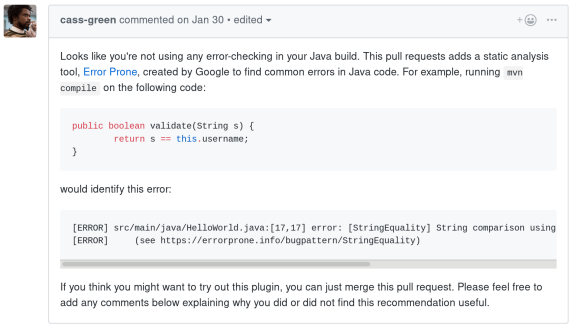
Research Overview

Decision-making plays a vital role in software engineering, and the choices developers make impact the technology we use everyday. Unfortunately, software engineers frequently make bad decisions. For example, studies show software engineers often ignore development tools, ethical programming guidelines, and other useful *developer behaviors*, which is costly for software users and producers. The motivating question of my research is: ***"How do we encourage software engineers make better decisions in their work?"***

My research interests lie in the intersection of empirical software engineering, human factors, and automation. To improve developer decision-making, my work spans disciplines to incorporate concepts from behavioral science into software engineering. Nudge theory is a framework for improving human behavior by influencing the environment surrounding decisions, or *choice architecture*, without 1) providing incentives to adopt the target behavior and 2) banning alternative choices [6]. My research introduces ***developer recommendation choice architectures***, a framework for creating automated recommendations that nudge developers towards better software engineering behaviors and practices.

My research philosophy is two-fold, to: 1) *perform empirical studies* characterizing software engineering problems; and 2) *develop tools and techniques* to overcome these problems. In my dissertation work, I explore what makes effective recommendations, construct a framework to improve automated recommendations, examine existing systems through the lens of this framework, and develop bots integrating ***developer recommendation choice architectures***. To provide evidence supporting this framework, I conducted multi-methodological studies collecting quantitative and qualitative data observing the behavior of CS students, open source software developers, and software engineers in industry. As a researcher, I will continue using this framework to observe developer actions and motivate the design of future tools for improving the productivity, decision-making, and behavior of software engineers.

Effective Developer Recommendations: To determine what makes effective recommendations to developers, I analyzed *peer interactions*, or the process of learning from colleagues during work activities, and the naive *telemarketer design*. To discover what makes peer interactions effective, we observed 13 pairs of participants completing data analysis tasks. For each session, we determined if interactions contained characteristics such as politeness, persuasiveness, and receptiveness, and if each recommendation was effective. Overall, we analyzed 142 peer interactions and found receptiveness, or *desire* and *familiarity*, was the only significant characteristic (Wilcoxon, $p = 0.0002$, $\alpha = .05$) [1]. To analyze automated recommendations, I developed the naive *telemarketer design* , a baseline approach which "calls" users to recommend and add tools to projects without the ability to customize messages or respond to users (Figure 1). We evaluated this approach in `tool-recommender-bot` on 52 GitHub repositories, and found only two recommendations (4%) were effective due to its poor *social context* and *developer workflow* [2].



(a) naive *telemarketer design* recommendation

(b) Adding tools to configuration files

Constructing a Framework: To improve automated recommendations to developers, I introduce ***developer recommendation choice architectures***, a framework for creating effective recommendations that nudge developers towards better behaviors and practices in their work. This framework consists of three principles: 1) **actionability**, or the ease with which developers can adopt the target behavior, 2) **feedback**, or the clarity and relevance of the information provided, and 3) **locality**, or the placement and timing of recommendations. In a preliminary evaluation, I surveyed professional software engineers and found 100% of participants prefer actionable recommendations over static ones [4].

Evaluating Existing Tools: To evaluate my framework, we studied recommendation styles and developer impact of GitHub *suggested changes*. This recently introduced feature allows users to recommend code improvements to pull requests, and incorporates ***developer recommendation choice architectures***. To study recommendation style, I conducted a think-aloud study where 14 professional developers interacted with static analysis tool recommendations from emails, issues, pull requests, and suggested changes (see Figure 2). We found developers significantly preferred the suggested changes recommendation (Kruskal-Wallis, $p = 0.00079$) [3].

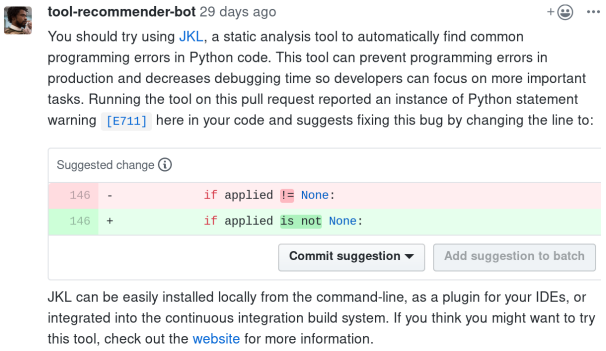


Figure 2: Suggested changes tool recommendation

To examine developer impact, I designed a study divided in two phases. Phase 1 is an empirical study analyzing suggested changes to discover types of suggestions, effectiveness compared to PR review comments, and their impact on pull-based software development. The results show most suggested changes are *non-functional*, not impacting code. Suggested changes are also more accepted and, while reviews take longer, the time to make and respond to recommendations is much faster than review comments. Furthermore, PRs with suggested changes have more commits, code churn, review comments, discussion comments, and participants in the review process. For Phase 2, we surveyed developers to qualitatively analyze feedback on the system. 98% of respondents found suggested changes at least moderately useful because of *user-driven communication* and *workflow integration* [5].

Developing Bots: To further explore the impact of ***developer recommendation choice architectures*** on behavior, I implemented `class-bot`, a system that automatically updates GitHub issues to encourage students to follow software engineering processes on programming projects. We conducted a preliminary evaluation of this bot on projects for an introductory Java course and mined GitHub repositories to observe programming behaviors on projects with and without `class-bot` notifications. Our early results show automated nudges improve code quality by increasing student grades and enhanced student productivity by increasing code churn and preventing procrastination.

Future Work

My career goal is to continue exploring ways to improve the behavior and productivity of software engineers. The following research areas highlight future directions of my prior work:

- ***Developer Nudges:*** One limitation of this work is the generalization of "developers" when exploring recommendations. In reality, developers are extremely diverse and have individual preferences for effective recommendations. To increase the effectiveness of automated nudges encouraging developers to adopt useful practices, I aim to study customized *developer nudges* that use different types recommendations based on user characteristics such as recent development activity, project contributions, and experience.
- ***Proactive Recommendations:*** Most of my work is *reactive*, in that recommendations occur after developers complete programming tasks. To improve the effectiveness of automated nudges, I will develop *proactive* nudges that predict developer actions and suggest useful behaviors before bad decisions are made. To accomplish this, I will apply machine learning techniques, such as collaborative filtering, to anticipate poor decisions and proactively recommend better practices to increase adoption of beneficial behaviors.
- ***Nudge Bots:*** To continue exploring effective recommendation approaches for software engineers, I aim to create an exhaustive `nudge-bot` system. This will consist of a suite of bots that: recommend various software engineering behaviors and practices; implement diverse interventions, such as GitHub Project boards, IM platforms like Slack, and automatically repairing programming mistakes; and make recommendations to developers on online programming communities such as code-hosting platforms (i.e. GitLab and BitBucket), Q&A sites (i.e. StackOverflow), and blogs and social media.

Broader Impact

My research has been published at peer-reviewed ACM and IEEE conferences and workshops. I have also presented at industry conferences such as Red Hat QECampX and DevConf. Furthermore, the tools created for this research, including `tool-recommender-bot`[1] and `class-bot`[2], are publicly available online and future systems will be made open source for developers and researchers to use, evaluate, and extend. Lastly, as a researcher from an underrepresented minority group, I plan to enable participation and increase diversity in computing research by recruiting and mentoring a diverse team of students to further explore software engineering problems.

[1] https://github.com/chbrown13/tool-recommender-bot
[2] https://github.com/chbrown13/nudge-bot

References

[1] ***Chris Brown***, Justin Middleton, Esha Sharma, and Emerson Murphy-Hill. How software users recommend tools to each other. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing*, VL/HCC 2019, pages 129–137, Raleigh, NC, USA, Oct 2017. IEEE Press.

[2] ***Chris Brown*** and Chris Parnin. Sorry to bother you: Designing bots for effective recommendations. In *Proceedings of the 1st International Workshop on Bots in Software Engineering*, BotSE 2019, pages 54–58, Montreal, QC, Canada, May 2019. IEEE Press.

[3] ***Chris Brown*** and Chris Parnin. Comparing different developer behavior recommendation styles. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ICSEW'20, page 78–85, New York, NY, USA, 2020. Association for Computing Machinery.

[4] ***Chris Brown*** and Chris Parnin. Sorry to bother you again: Developer recommendation choice architectures for designing effective bots. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ICSEW'20, page 56–60, New York, NY, USA, 2020. Association for Computing Machinery.

[5] ***Chris Brown*** and Chris Parnin. Understanding the impact of github suggested changes on recommendations between developers. In *Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2020, Sacramento, CA, 2020. ACM.

[6] Richard H Thaler and Cass R Sunstein. *Nudge: Improving decisions about health, wealth, and happiness*. Penguin, 2009.