Research Statement

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

Research Overview

Decision-making plays an important role in software engineering, however developers often require help when making decisions. For example, software engineering researchers and toolsmiths create and evaluate a wide variety of tools and practices to help accomplish programming tasks, but these useful *developer behaviors* often go ignored by software engineers in industry. While my research focuses on software engineering, poor developer behavior can impact a variety of computer science research areas that rely on programming such as data science, HCI, AI, computer systems, and data science. For example, studies show software engineers often ignore using security tools and practices which leads to vulnerabilities in software. This problem can be summed up in the following question posed by Greg Wilson: ***"How do we get working programmers to actually adopt better practices?"***[1]

My research goal is, given a developer who is unaware of a useful developer behavior in a development situation, to identify the most effective strategy to convince them to adopt the behavior. To encourage developers to adopt better practices, my dissertation research involves interdisciplinary work focused on improving developer adoption of behaviors with *nudge theory*, a behavioral science framework to improve human decision-making without providing incentives or banning selections [3], by introducing ***developer recommendation choice architectures*** to improve the framing and design of decisions for software engineers.

My research experiences involve numerous undergraduate and graduate opportunities studying a variety of computer science topics. Overall, my research philosophy is two-fold: 1) to perform empirical studies to collect and analyze data for *characterizing software engineering problems*, and 2) to develop *tools and techniques* to help solve these problems. This is shown in my doctoral research, which incorporates include several multi-methodology studies analyzing qualitative and quantitative data to explore effective automated recommendations for improving developer behavior.

Research Contributions

My research contributions include a *conceptual framework* characterizing effective developer recommendations, evaluating *existing tools*, and developing an *automated recommender system* that uses nudges to make effective behavior recommendations to software engineers.

Characterizing effective developer recommendations

To determine what makes effective recommendations for software engineers, we analyzed *peer interactions* and developed a baseline naive *telemarketer design* approach.

Research suggests learning from colleagues during work activities, or *peer interactions*, is the most effective mode of tool discovery for developers. To discover what makes peer interactions effective, we conducted a user study observing tool recommendations between 13 pairs of participants working together to complete data analysis tasks and conducted

[1] https://twitter.com/gvwilson/status/1142245508464795649?s=20

(a) naive *telemarketer design* recommendation

(b) `tool-recommender-bot` configuration changes

semi-structured interviews to explore recommendations between peers. For each study, we analyzed sessions to determine 1) when peer interactions occurred, 2) if interactions contained characteristics such as politeness, persuasiveness, receptiveness, and time pressure based on a set of criteria we developed for each, and 3) if recommended tools were used or ignored by users who received suggestions. Overall, we analyzed 142 peer interactions and found that *receptiveness* was the only significant characteristic that impacted peer interactions (Wilcoxon, $p = 0.0002$, $\alpha = .05$). To define receptiveness, we used two criteria: ***demonstrate desire*** and ***familiarity***. This work was published and presented at the 2017 Visual Languages and Human-Centric Computing conference [1].

While peer interactions are effective for recommending tools, they do not scale with larger distributed development teams. I developed the naive *telemarketer design* , a baseline approach for automated recommendations which "calls" users to deliver generic messages without the ability to customize recommendations or respond to users. I implemented this design in `tool-recommender-bot`, a system that generates automated pull requests recommending static analysis tools with examples of common programming errors (Figure 1a) and changing build configuration files (Figure 1b). To evaluate our approach, we recommended the Error Prone Java static analysis tool[2] to 52 GitHub repositories. Ultimately, the system only made two effective recommendations (4%) while most PRs were closed or ignored. Based on comments from developers, we found that our approach failed because it lacked ***social context*** and poor ***developer workflow*** integration. The results of this study were published at the 2019 International Workshop on Bots in Software Engineering [2].

In summary, we uncovered four concepts to consider for designing effective recommendations to developers: **desire**, **familiarity**, **social context**, and **developer workflow**. We suggest that integrating these into automated suggestions to improve developer behavior and increase adoption of beneficial programming tools and practices.

Evaluating existing tools

To analyze existing systems for developer recommendations, we studied the new GitHub *suggested changes* feature. This recently introduced feature fits into the nudge theory framework and allows users to recommend code changes to developers on pull requests. To study GitHub suggestions, we conducted a study divided into two phases. Phase 1 is an empirical study where we mined GitHub for instances of suggested changes to categorize types of

[2]http://errorprone.info

changes developers propose and determine how often suggestions are accepted. Our results show that developers suggest *corrective*, *improvement*, *formatting*, and *non-functional* code changes. While PRs are still the most effective recommendation system on GitHub, we found suggested changes are accepted significantly faster by developers.

For Phase 2, we collected feedback from developers by surveying users who received or made suggestions and conducting a user study with 14 professional developers to evaluate a prototype recommendation system with this feature, presented in Figure 2. Users reported suggested changes are useful primarily because they improve communication, are concise, and save time. Additionally, developers significantly preferred tool recommendations from suggested changes over emails, PRs, and issues (Kruskal-Wallis, $p = .00079$, $\alpha = .05$). Based on our results and prior work in nudge theory, we propose the following ***developer recommendation choice architectures*** to explain the effectiveness of suggested changes and to provide implications for designing automated developer behavior recommendations: **actionability**, **feedback**, and spatial and temporal **locality**. These results will be submitted for publication at FSE 2020.

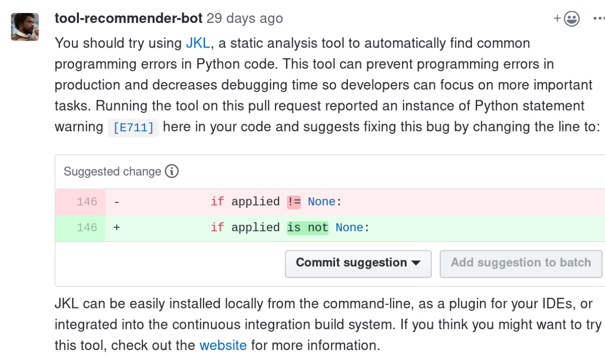

Figure 2: Example suggested changes tool recommendation

Developing an automated recommender system

To evaluate my framework, I implemented `class-bot`, a bot designed to encourage Computer Science students to adopt better software engineering behaviors. A primary reason for poor performance and high attrition in introductory programming classes is poor project management skills, so we implemented this system on projects for an introductory Java programming course to encourage students to follow better software engineering processes. Data collection consisted of mining GitHub repositories to observe programming behaviors for projects with and without `class-bot` to compare how nudges impacted student productivity and collecting feedback from students on the automated system. The results show our system significantly improved student productivity and project quality based on observed metrics. I plan to submit this work to a peer-reviewed research conference.

Future Work

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

- ***Developer Nudges:*** One limitation of this work is that we generalize "developers" when exploring what makes effective recommendations. In reality, developers are extremely diverse and each developer has individual preferences for what makes suggestions effective, and which types of interventions are useful in certain situations. To create better strategies for encouraging developers to adopt useful practices, I aim to study customized *developer* *nudges*. To start this work, I plan to evaluate using different intervention types for GitHub users based on characteristics such as recent development activity and experience.

- ***Proactive Recommendations:*** The recommendations implemented for my work so far have been *reactive*, in that they suggest behaviors after developers have started or completed programming tasks in a manner that could be improved. To improve the effectiveness of developer nudges, I plan to develop *proactive* recommendations to predict developer actions while they complete coding tasks and suggest useful behaviors before developers start. To begin this work, I plan to apply machine learning techniques such as collaborative filtering into recommender systems to anticipate poor developer behaviors and send proactive nudges to recommend better practices.

- ***Integrating Research in Industry:*** Additionally, I am passionate about finding ways to bridge the gap between software engineering research and industry. To increase awareness of research tools and techniques for software engineers, I plan to continue to studying effective recommendation approaches and expand `class-bot` to include additional practices and tools created and evaluated by the research community, as well as integrating these behaviors into CS classes to instill in future software engineers. To increase the relevance of industry problems for research, I aim to pursue collaborations with industry developers, design empirical studies and tools that impact real-world practices, gather feedback from professional developers, and present results to industry developers in a way that increases awareness and motivates adoption of useful developer behaviors.

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 2018 and DevConf 2019. Finally, the tools created for this research are publicly available for developers and researchers to use, evaluate, and extend, including *tool-recommender-bot*[3] and *nudge-bot*[4].

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

[3] https://github.com/chbrown13/tool-recommender-bot
[4] https://github.com/chbrown13/nudge-bot