# Digital Nudges for Encouraging Developer Behaviors

**Chris Brown**

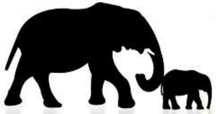
dcbrow10@ncsu.edu

**Committee:**
Dr. Chris Parnin (Chair)
Dr. Anne McLaughlin (PSY, GSR)
Dr. Sarah Heckman
Dr. Kathryn Stolee

Oral Preliminary Exam
North Carolina State University

NC STATE UNIVERSITY

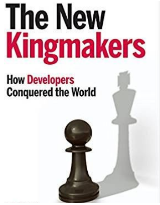
# Outline

- Motivation
- Background 
- Thesis Statement
- Experiments and Evaluations
  - Completed
  - Proposed
- Research Plan

# Motivation

## ***Decision-making is a vital part of software engineering.***

- *“[Software engineers] have the power to make or break business...Developers are now the real decision makers in technology.”* [O’Grady, 2013]



The New Kingmakers
How Developers Conquered the World

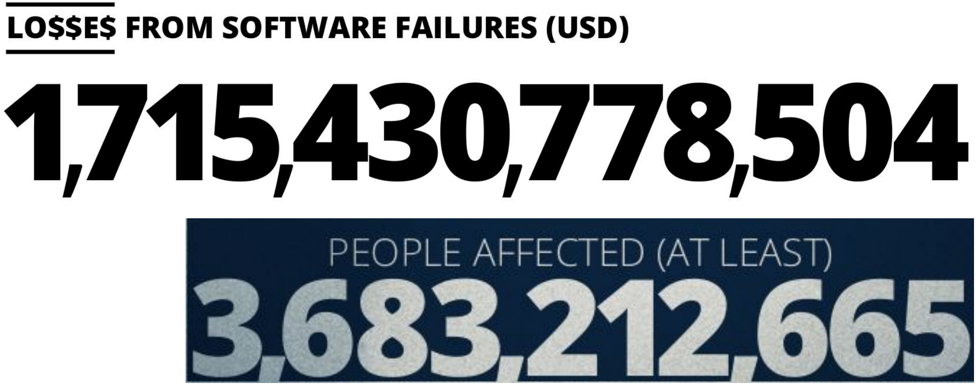
- *“The most important skill in software development is not how good your coding skills are or how much you know about machine learning and data science. It’s decision-making!”* [Woo, 2019]

HACKERNOON

- *“Though rarely discussed in the software engineering literature, [our] results suggest effective decision-making is critical...as engineers grow in their careers, they are tasked with making decisions in increasingly more complex and ambiguous situations, often with significant ramifications.”* [Li, 2015]

# Problem

***Software engineers need help making decisions...***



LO$$E$ FROM SOFTWARE FAILURES (USD)

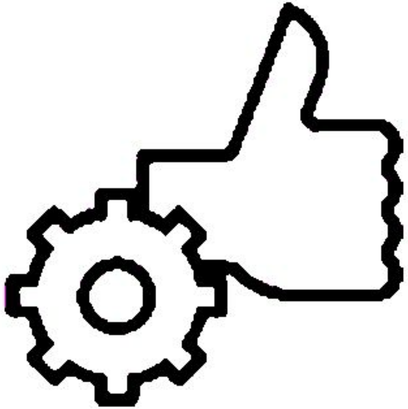
1,715,430,778,504

PEOPLE AFFECTED (AT LEAST)

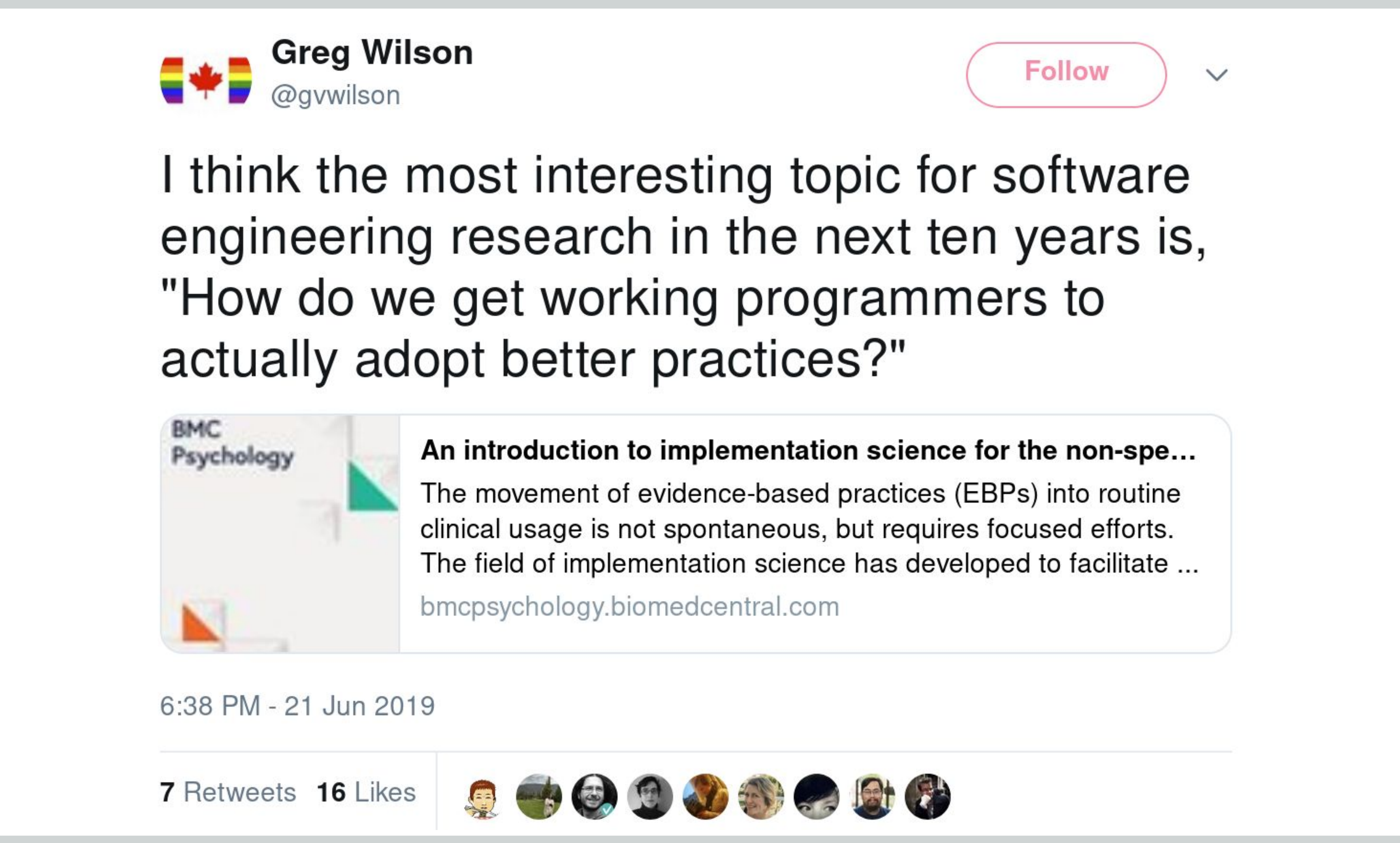
3,683,212,665

[Tricentis, 2017]

*Recommendation Systems for Software Engineering*



[Robillard, 2010]



**Greg Wilson**
@gvwilson

I think the most interesting topic for software engineering research in the next ten years is, "How do we get working programmers to actually adopt better practices?"

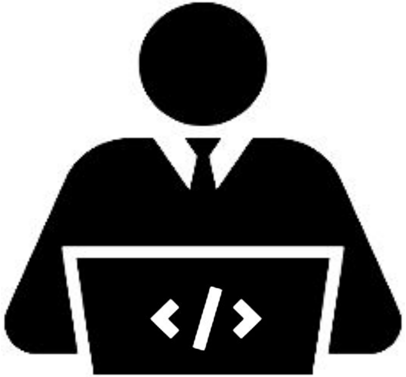
**An introduction to implementation science for the non-spe…**
The movement of evidence-based practices (EBPs) into routine clinical usage is not spontaneous, but requires focused efforts. The field of implementation science has developed to facilitate ...
bmcpsychology.biomedcentral.com

6:38 PM - 21 Jun 2019

**7** Retweets **16** Likes

# Research Goal

*Given a developer who is unaware of a useful behavior during a development situation, identify the most effective strategy to convince them to adopt the behavior.*

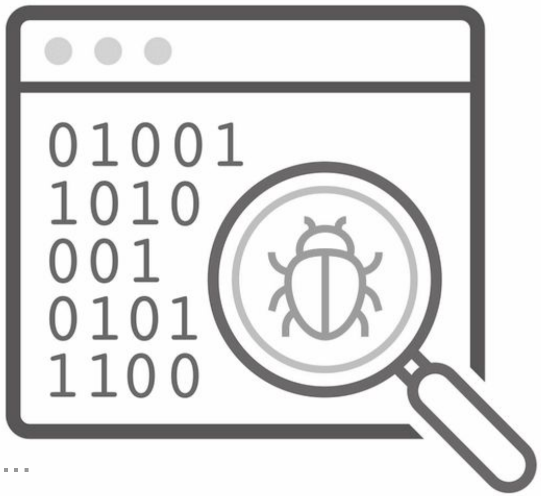


# Background: *Developer Behavior*

***Tools and practices designed to help developers complete programming tasks.***

👍

Improve code quality [Ayewah, 2010],
Prevent errors [Bessey, 2010],
Reduce developer effort [Singh, 2017],...



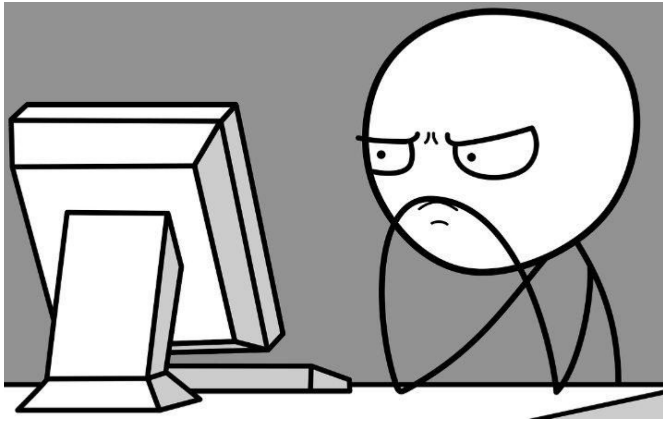
👎

**Result Understandability,**
**Customizability,**
**Tool Output,...**
**[Johnson, 2013]**

# Background: *Developer Behavior*

## ***Developer Behavior Adoption Problem***

***Decision Fatigue***



[Makabee, 2011]

***Developer Inertia***

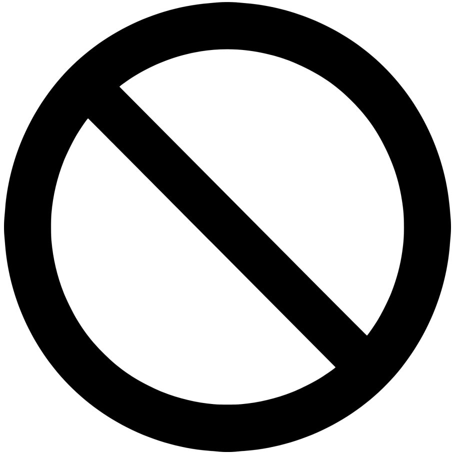
[Murphy-Hill, 2015]

***Research-Practice Gap***

[Norman, 2010]

# Background: *Nudge Theory*

***Any factor that impacts human decision-making without providing incentives or banning alternatives***



[Thaler and Sunstein, 2009]

# Background: *Digital Nudges*

***The use of nudges to guide users' behavior in digital choice environments.***

[Weinmann, 2016]

# Background: *Choice Architecture*

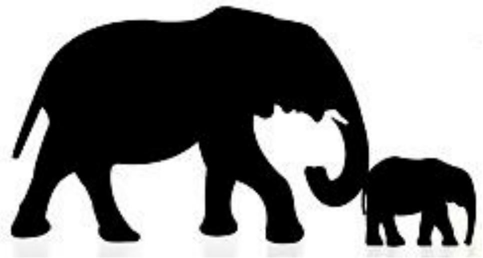
***The framing and presentation of choices to decision-makers***

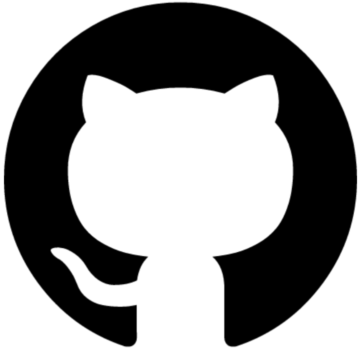
*“There is no such thing as a ‘neutral’ design...Choice architecture, both good and bad, is pervasive and unavoidable, and it greatly affects our decisions.”*

[Thaler, 2009]

D'oh, the donuts are wayy over there.
Think I'll have fruit
My cunning choice architecture will soon have Homer eating healthy
JACK POT $$$$
KEEP OUT
HEALTHY FRESH FRUIT
$$$

# Scope of Work





# Thesis Statement

By incorporating ***developer recommendation choice architectures*** into recommendations for software engineers, we can ***nudge*** developers to adopt behaviors useful for improving code quality and developer productivity.

# Plan of Work

Determine effective strategies

Examine existing systems

Develop new tool

# Expected Contributions

1. A *conceptual framework* for using concepts from nudge theory to make effective developer recommendations.
2. A *set of experiments* to evaluate and provide evidence for the conceptual framework.
3. An *automated recommender system* to nudge software engineers to adopt developer behaviors.

# Thesis: *Effective Strategies*

By incorporating ***developer recommendation choice architectures*** into recommendations for software engineers, we can ***nudge*** developers to adopt behaviors useful for improving code quality and developer productivity.

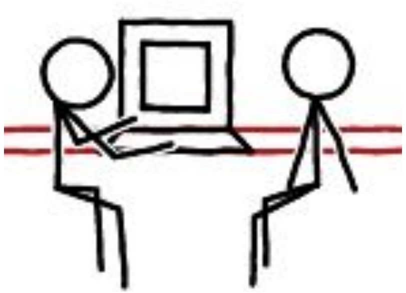
# [Completed] Peer Interactions

→ ***"How Software Users Recommend Tools to Each Other"*** [Brown, 2017]

**RQ.** What characteristics of peer interactions make recommendations effective?

# Peer Interactions

***The process of discovering tools from colleagues during normal work activities*** [Murphy-Hill, 2011]



# Peer Interactions: Methodology

## Study Design

- 26 participants (13 pairs)
    - Professionals and Students
- Tasks
    - Kaggle ML Competition
- Setup
    - Software Usage
    - Internet Restriction

## Data Analysis

- Screen and audio recordings
    1. Politeness [Leech, 1983]
    2. Persuasiveness [Shen, 2012]
    3. Receptiveness [Fogg, 2009]
    4. Time Pressure [Andrews, 1996]
    5. Tool Observability [Murphy-Hill, 2015]
- Effectiveness
    - Tool used
    - Tool ignored
    - Unknown

# **Peer Interactions:** Results

| | **Effective** | **Ineffective** | **Unknown** | **Total** |
|---|---|---|---|---|
| ***n*** | 71 | 35 | 36 | 142 |

1. Politeness
2. Persuasiveness
3. **Receptiveness*** (Wilcoxon, p = 0.0002, OR = 0.2840)
4. Time Pressure
5. Tool Observability

# **Peer Interactions:** Receptiveness

## ***Demonstrate Desire***

*"Oh! Add level!*
*Yes, awesome!"*
- L14

## ***Familiarity***

*"I don't know R."* - S9

[Fogg, 2009]

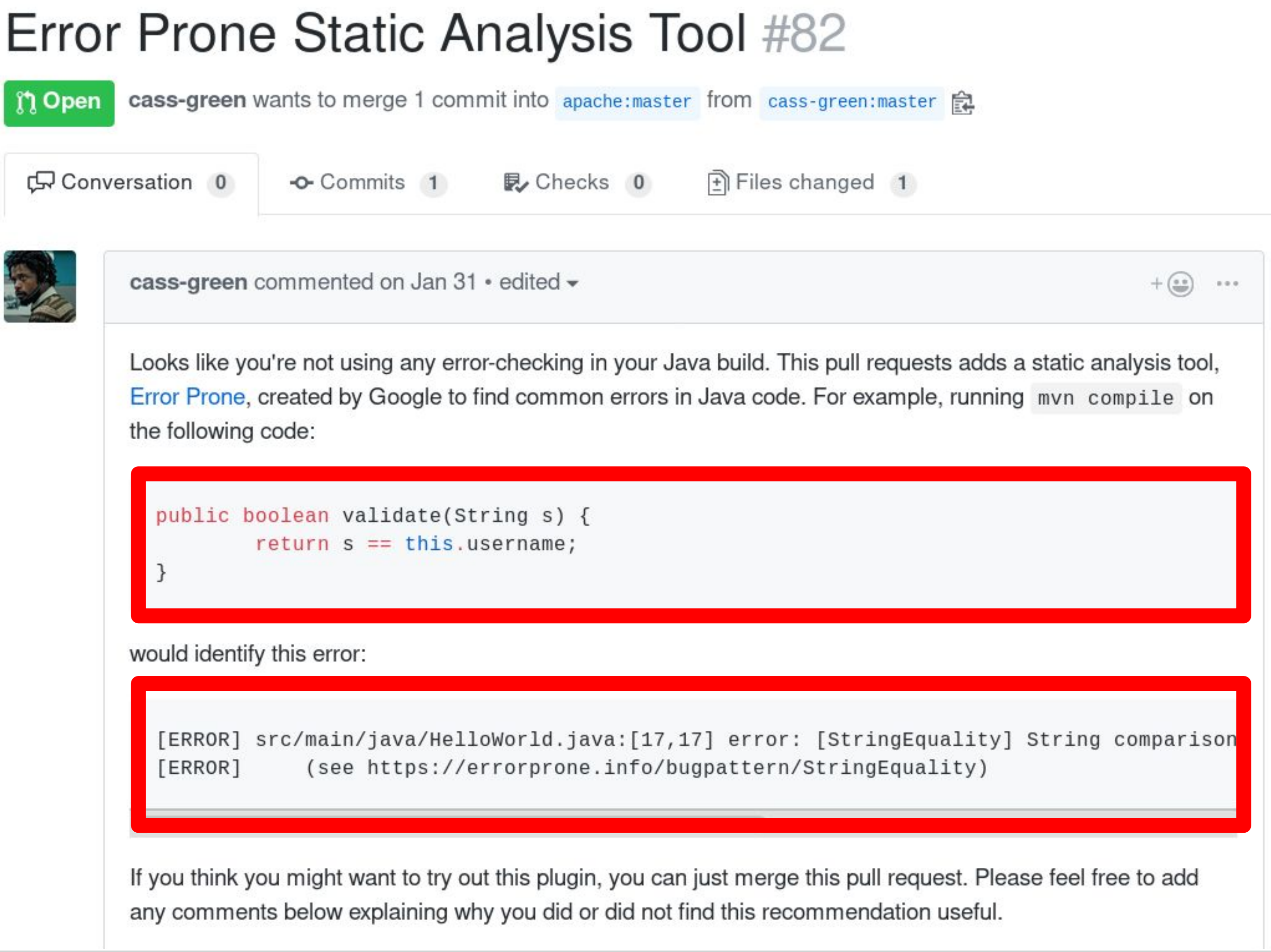
# [Completed] Sorry to Bother You

➔ ***"Sorry to Bother You: Designing Bots for Effective Recommendations"*** [Brown, 2019]

**Goal:** Identify and evaluate a baseline approach for automated developer recommendations.

## **Naive *telemarketer design***

- Static Recommendations
- Generic Messages
- Socially Inept



# Error Prone Static Analysis Tool #82

cass-green wants to merge 1 commit into apache:master from cass-green:master

cass-green commented on Jan 31 • edited

Looks like you're not using any error-checking in your Java build. This pull requests adds a static analysis tool, Error Prone, created by Google to find common errors in Java code. For example, running `mvn compile` on the following code:

```
public boolean validate(String s) {
        return s == this.username;
}
```

would identify this error:

```
[ERROR] src/main/java/HelloWorld.java:[17,17] error: [StringEquality] String comparison
[ERROR]     (see https://errorprone.info/bugpattern/StringEquality)
```

If you think you might want to try out this plugin, you can just merge this pull request. Please feel free to add any comments below explaining why you did or did not find this recommendation useful.

# Sorry to Bother You: Methodology

## Study Design

Error Prone

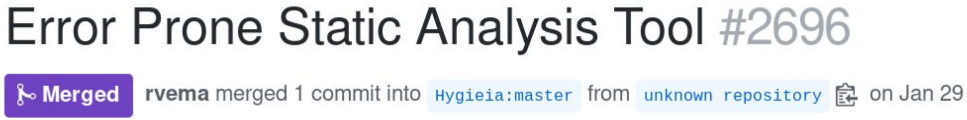
- 52 GitHub projects
  - Java 8+
  - Maven
  - No Error Prone
- `tool-recommender-bot`
  - Build configuration files
  - Automated pull requests

## Data Analysis

- Effectiveness
  - Merged
  - Closed/No Response
- Developer Feedback
  - 24 Pull Request Comments

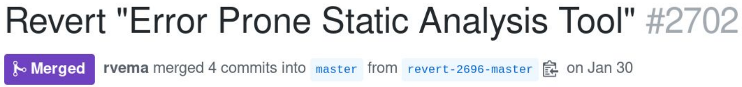
# Sorry to Bother You: Results

| | *n* | Percent |
|---|---|---|
| Merged | 2 | 4% |
| Closed | 10 | 19% |
| No Response | 40 | 77% |



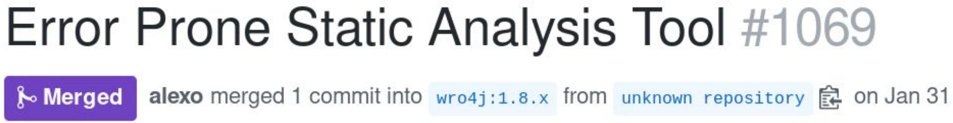
Error Prone Static Analysis Tool #2696

Merged rvema merged 1 commit into `Hygieia:master` from `unknown repository` on Jan 29



Revert "Error Prone Static Analysis Tool" #2702

Merged rvema merged 4 commits into `master` from `revert-2696-master` on Jan 30



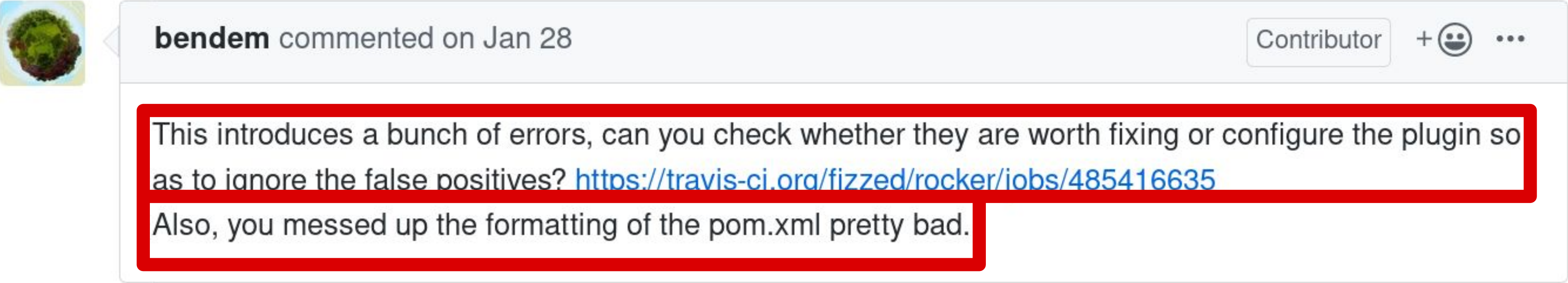
Error Prone Static Analysis Tool #1069

Merged alexo merged 1 commit into `wro4j:1.8.x` from `unknown repository` on Jan 31

# Sorry to Bother You: Feedback

## ***Social Context***

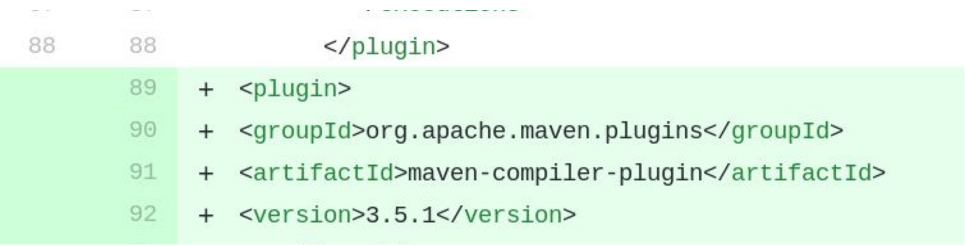
## ***Developer Workflow***



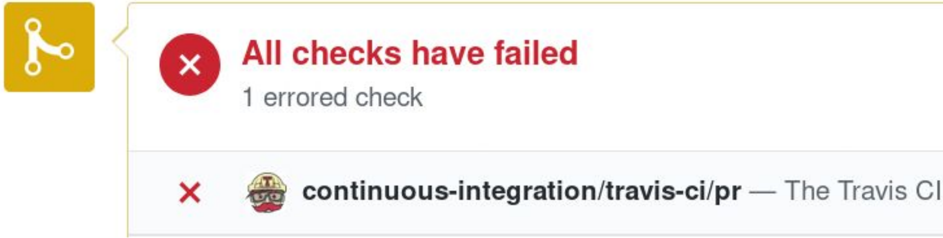
**bendem** commented on Jan 28

This introduces a bunch of errors, can you check whether they are worth fixing or configure the plugin so as to ignore the false positives? https://travis-ci.org/fizzed/rocker/jobs/485416635

Also, you messed up the formatting of the pom.xml pretty bad.



```
            </plugin>
+ <plugin>
+ <groupId>org.apache.maven.plugins</groupId>
+ <artifactId>maven-compiler-plugin</artifactId>
+ <version>3.5.1</version>
```



**All checks have failed**

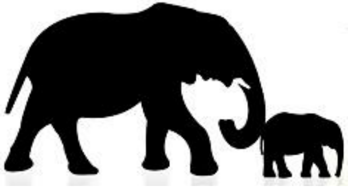
1 errored check

✕ **continuous-integration/travis-ci/pr** — The Travis CI

# Conceptual Framework

1. Desire
2. Familiarity
3. Social Context
4. Developer Workflow

1. *Actionability*
2. *Feedback*
3. *Locality*
   a. *Spatial*
   b. *Temporal*



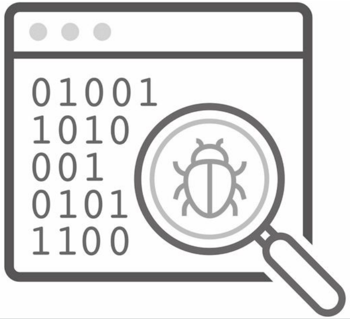
[Johnson, 2012]

# Actionability

***The ease with which users can act on recommendations***

**Default Rule**
*Automatic Enrollment*
[Madrian, 2001]

**Static Analysis**
*Splint (Secure Programming Lint)*
[Evans, 2002]

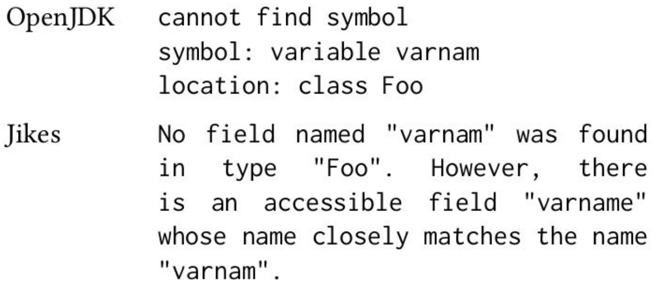


# Feedback

***Information provided to users in recommendations to encourage adoption***

**Customized Information**
*Daily caloric intake*
[Wisdom, 2010]

**Compiler Error Messages**
*Argument structure*
[Barik, 2018]



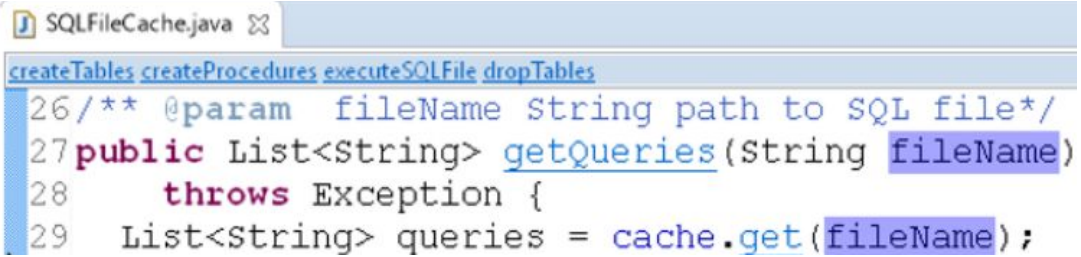
| | |
|---|---|
| OpenJDK | `cannot find symbol`<br>`symbol: variable varnam`<br>`location: class Foo` |
| Jikes | `No field named "varnam" was found in type "Foo". However, there is an accessible field "varname" whose name closely matches the name "varnam".` |

# Locality: *Spatial*

***The setting of recommendations to improve user behavior***

**Decision Staging**
Healthy Convenience Lines
[Hanks, 2012]

**Flower**
*In situ navigation*
[Smith, 2017]



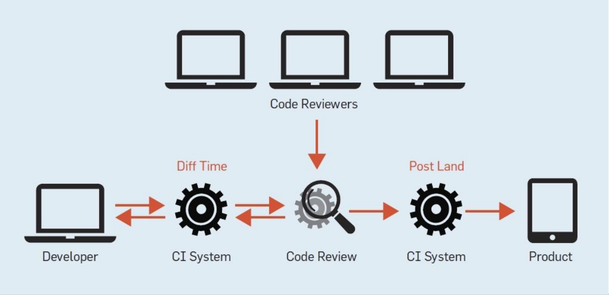
```
/** @param  fileName String path to SQL file*/
public List<String> getQueries(String fileName)
      throws Exception {
   List<String> queries = cache.get(fileName);
```

# Locality: *Temporal*

***The setting of recommendations to improve user behavior***

**Time-limited windows**
Present-biased farmers
[Duflo, 2011]

**Scaling Static Analyses at Facebook**
*"diff time"*
[Distefano, 2019]



# Thesis: *Existing Systems*

By incorporating ***developer recommendation choice architectures*** into recommendations for software engineers, we can ***nudge*** developers to adopt behaviors useful for improving code quality and developer productivity.

# [Proposed] Suggestions

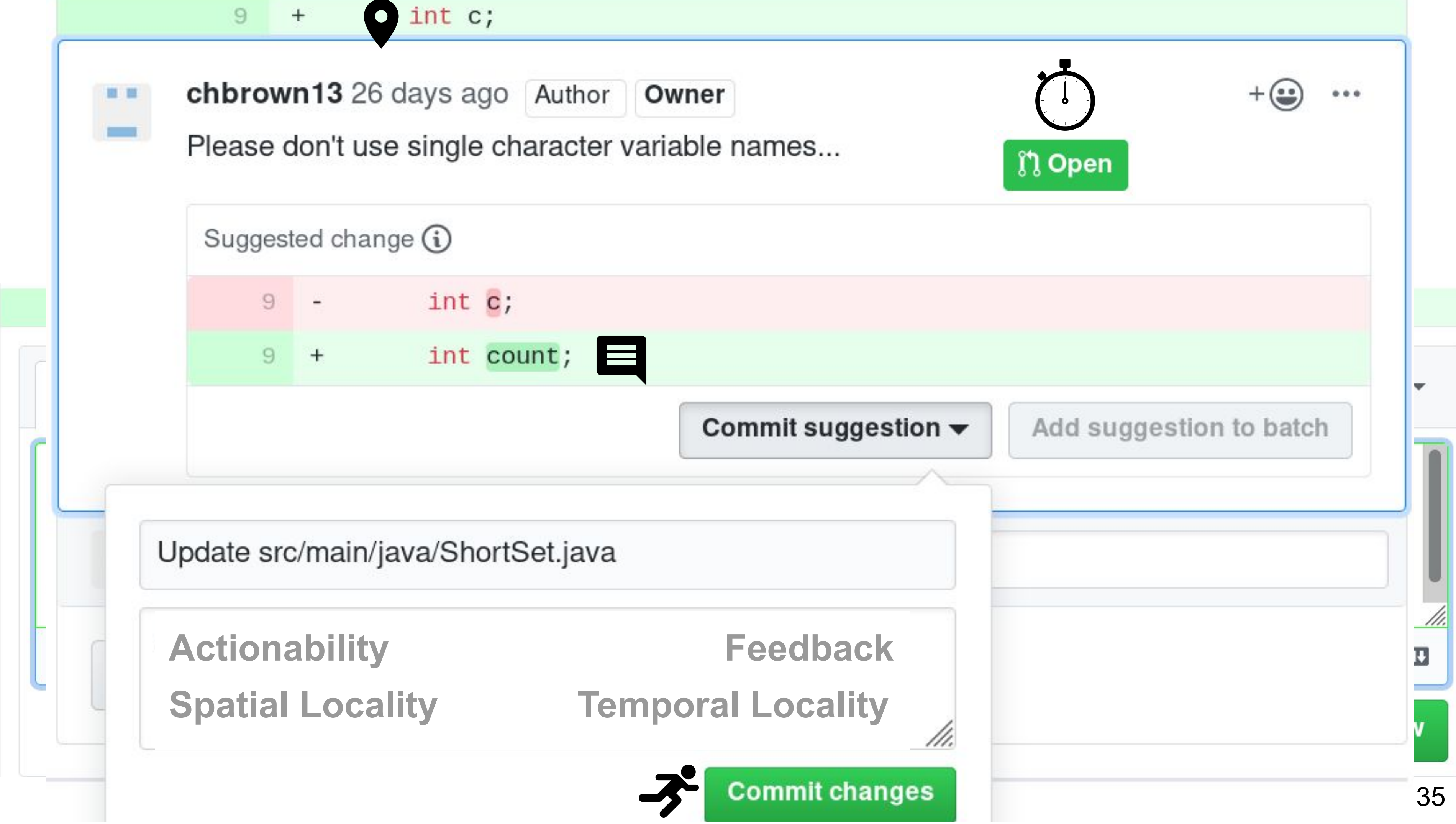
➔ ***“Understanding the Impact of GitHub Suggested Changes on Recommendations Between Developers”***

**RQ1.** What suggestions do developers make with suggested changes?

**RQ2.** How effective is the suggested changes feature on GitHub?

**RQ3.** How useful is the suggested changes feature for developers?

**RQ4.** How well does the suggested changes feature generalize to other types of recommendations?



```
9   +       int c;
```

**chbrown13** 26 days ago Author Owner

Please don't use single character variable names...

Suggested change

```
9   -       int c;
9   +       int count;
```

Update src/main/java/ShortSet.java

Actionability Feedback

Spatial Locality Temporal Locality

# **Suggestions:** Methodology

## Phase 1: *An Empirical Study on GitHub Suggested Changes*

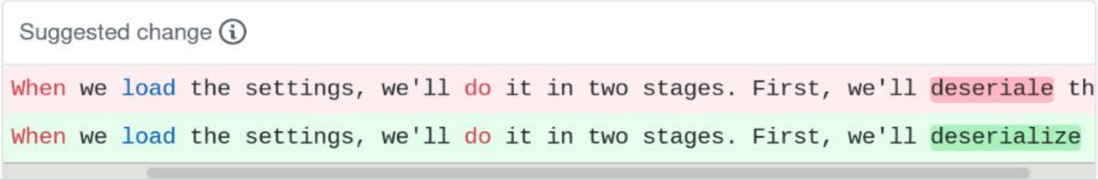
*RQ1. Categorizing Suggested Changes:*

- Detecting Suggested Changes
  - Most recently updated repositories
  - ```suggestion{...}```
  - 100 suggested changes

*Open Coding*

(IRR = 71%, Cohen's κ= 0.5942)

**(a) Non-Functional:**



Suggested change ⓘ

```
When we load the settings, we'll do it in two stages. First, we'll deseriale th
When we load the settings, we'll do it in two stages. First, we'll deserialize
```

# **Suggestions:** Methodology

## Phase 1: *An Empirical Study on GitHub Suggested Changes*

*RQ2. Defining Effectiveness:*

- Detecting Suggested Changes
  - Top-forked repositories
  - ```suggestion{...}```
  - Line of code exists in subsequent commit

*Criteria*

**Acceptance** [Middleton, 2018]

**Timing** [Layman, 2007]

*GitHub Recommendation Systems*

Pull Requests [Gousious, 2014]

Issues [Bissyandé, 2013]

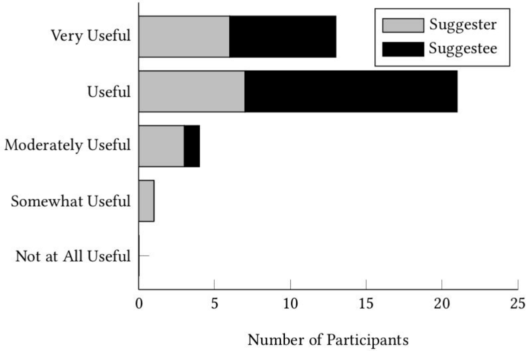
Suggested Changes

# **Suggestions:** Methodology

## Phase 2: *Developer Feedback on Suggested Changes*

*RQ3. Determining Usefulness:*

- Suggesters and Suggestees
- 5-point Likert and open-ended
- 39 responses



*Open Coding*

Useful (IRR = 72%, κ = 0.6828)

Unuseful (IRR = 77%, κ = 0.7125)

**Communication.** *“I find it *so* useful. It completely removes all ambiguity about what I'm asking for if I can just directly put the code there.”* - R14

**Unsupported features.**

- *Multi-line suggestions*

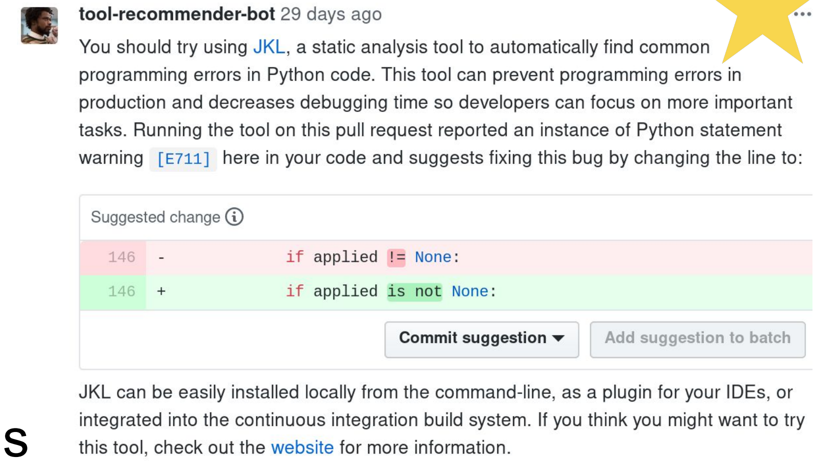
# Suggestions: Methodology

## Phase 2: *Developer Feedback on Suggested Changes*

*RQ4. Determining Generalizeability:*

- 14 professional developers
- Tool Recommendations

- Screen and audio recordings
- Think-aloud
- Likelihood of adoption
- Semi-Structured Interview



**tool-recommender-bot** 29 days ago

You should try using JKL, a static analysis tool to automatically find common programming errors in Python code. This tool can prevent programming errors in production and decreases debugging time so developers can focus on more important tasks. Running the tool on this pull request reported an instance of Python statement warning `[E711]` here in your code and suggests fixing this bug by changing the line to:

Suggested change ⓘ

```
-            if applied != None:
+            if applied is not None:
```

JKL can be easily installed locally from the command-line, as a plugin for your IDEs, or integrated into the continuous integration build system. If you think you might want to try this tool, check out the website for more information.

# **Suggestions:** Expected Results

1) Suggested changes are an effective system for a different types of recommendations,

2) Developers find this feature useful and applicable for various recommendations, and

3) Suggested changes can provide design implications for developing effective automated recommender systems.

# Thesis: *New Tool*

By incorporating ***developer recommendation choice architectures*** into recommendations for software engineers, we can ***nudge*** developers to adopt behaviors useful for improving code quality and developer productivity.

# [Proposed] Nudge-Bot

→ ***"Nudging Students Toward Better Software Engineering Behaviors"***

**RQ1.** How do nudges influence software engineering student productivity?

**RQ2.** How do nudges impact the quality of software engineering student projects?

# **Nudge-Bot:** Methodology

## Study Design

- *Software Engineering* (CSC326)
- Final Team Project
- iTrust
- `nudge-bot`

NC STATE UNIVERSITY

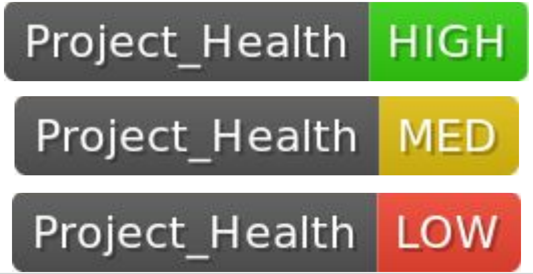
### *Defining Behavior:*

**Project Management**
[Beaubouef, 2005]
[Charette, 2005]

### *Nudging Behaviors:*

Active

Passive



Project_Health HIGH

Project_Health MED

Project_Health LOW

# Nudge-Bot: Methodology

## RQ1. Defining Productivity:

[Beaubouef, 2005]

- Time before milestone deadline
- Total time to complete project
- Total functional requirements met

## RQ2. Defining Code Quality:

[Figas, 2013]

- Final grade
- Total process requirements met

- Student feedback

# **Nudge-Bot:** Expected Results

1) Increase the number of functional and process requirements utilized by teams,

2) Improve productivity and reduce procrastination time by encouraging students to complete work on their projects sooner, and

3) Enhance the overall software quality and raise student grades for the final team project.

# Research Plan

## Completed

- Peer Interactions [VL/HCC 2017]
- Sorry to Bother You [BotSE 2019]

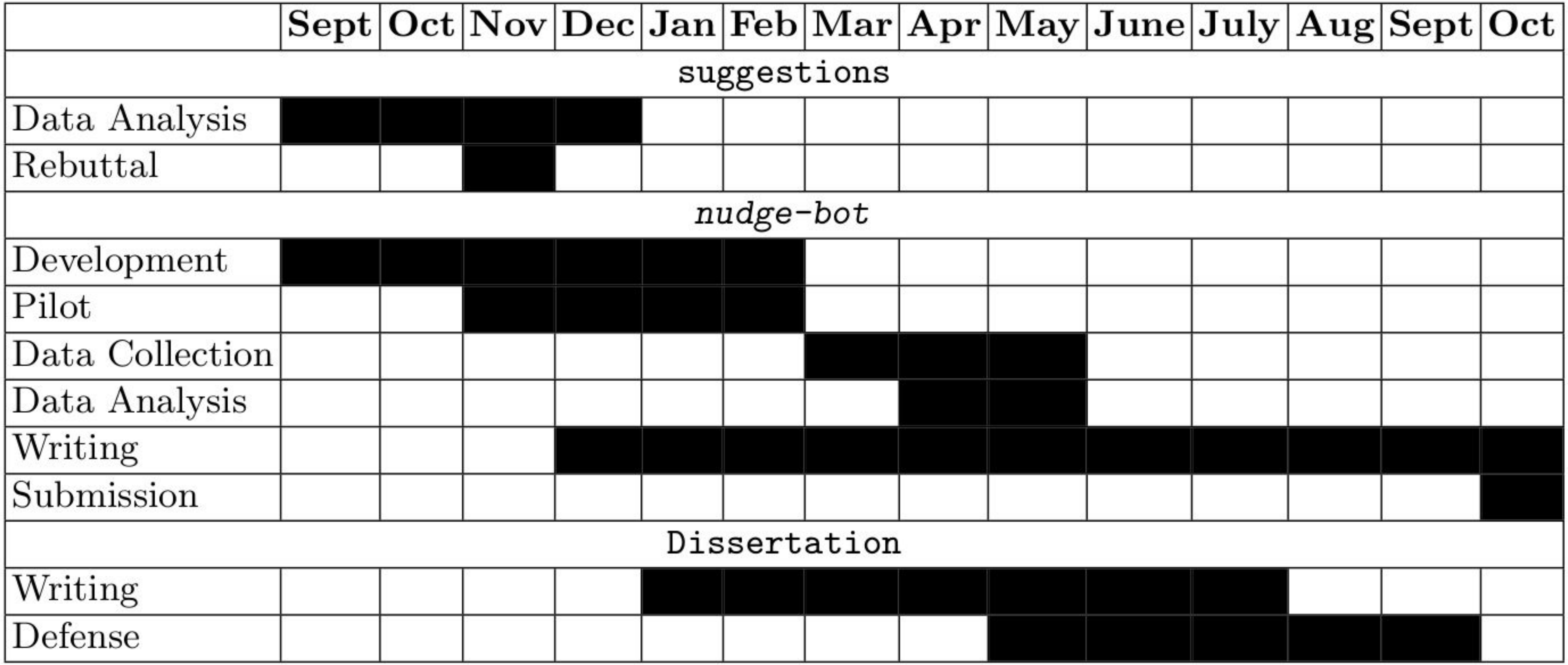
## Upcoming

- suggestions [ICSE 2020 (in submission)]
- nudge-bot [ICSE SEET 2021]
- Dissertation [Summer/Fall 2020]

# Publication List

1. **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
2. **Chris Brown**. “Digital nudges for encouraging developer actions”. In Proceedings of the 41st *International Conference on Software Engineering (ICSE 2019)*: Companion Proceedings, pages 202–205, Montreal,QC, Canada, May 2019. IEEE Press
3. Peng Sun, **Chris Brown**, Ivan Beschastnikh, and Kathryn T. Stolee.“Mining specifications from documentation using a crowd”. In 2019 IEEE 26th International Conference on *Software Analysis, Evolution and Reengineering (SANER 2019),* pages 275–286, Hangzhou, China, Feb 2019. IEEE Press
4. **Chris Brown**, Justin Middleton, Esha Sharma, and Emerson Murphy-Hill. “How software users recommend tools to each other”. In2017 IEEE Symposium on *Visual Languages and Human-Centric Computing (VL/HCC 2019)* pages 129–137, Raleigh, NC, USA, Oct 2017. IEEE Press
5. Justin Smith, **Chris Brown**, and Emerson Murphy-Hill. “Flower: navigating program flow in the ide”. In 2017 IEEE Symposium on *Visual Languages and Human-Centric Computing (VL/HCC 2017)* pages 19–23, Raleigh, NC, USA, Oct 2017. IEEE Press

# Research Plan



| | Sept | Oct | Nov | Dec | Jan | Feb | Mar | Apr | May | June | July | Aug | Sept | Oct |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| suggestions | | | | | | | | | | | | | | |
| Data Analysis | ■ | ■ | ■ | ■ | | | | | | | | | | |
| Rebuttal | | | ■ | | | | | | | | | | | |
| *nudge-bot* | | | | | | | | | | | | | | |
| Development | ■ | ■ | ■ | ■ | ■ | ■ | | | | | | | | |
| Pilot | | | ■ | ■ | ■ | ■ | | | | | | | | |
| Data Collection | | | | | | | ■ | ■ | ■ | | | | | |
| Data Analysis | | | | | | | | ■ | ■ | | | | | |
| Writing | | | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ |
| Submission | | | | | | | | | | | | | | ■ |
| Dissertation | | | | | | | | | | | | | | |
| Writing | | | | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | | |
| Defense | | | | | | | | | ■ | ■ | ■ | ■ | ■ | |

# Thanks

By incorporating ***developer recommendation choice architectures*** into  recommendations for software engineers, we can ***nudge*** developers to adopt behaviors useful for improving code quality and developer productivity.

**Chris Brown**

[dcbrow10@ncsu.edu](mailto:dcbrow10@ncsu.edu)

[https://chbrown13.github.io](https://chbrown13.github.io)

[https://github.com/chbrown13](https://github.com/chbrown13)

alt-code

NC STATE UNIVERSITY

# Citations

- Alós-Ferrer, C., Hügelschäfer, S., and Li, J.:. "Inertia and decision making." *Frontiers in psychology* 7 (2016)
- Ayewah, N., Pugh, W.: The google findbugs fixit. In: Proceedings of the 19th International symposium on Software testing and analysis. pp. 241–252. ACM (2010)
- Barik, T., Ford, D., Murphy-Hill, E., Parnin, C.: How should compilers explain problems to developers? In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 633–643. ACM (2018)
- Beaubouef, T., Mason, J.: Why the high attrition rate for computer science students: some thoughts and observations. ACM SIGCSE Bulletin 37(2), pp. 103–106 (2005)
- Bessey, A., Block, K., Chelf, B., Chou, A., Fulton, B., Hallem, S., Henri-Gros, C., Kamsky, A., McPeak, S., Engler, D.: A few billion lines of code later: using static analysis to find bugs in the real world. Communications of the ACM 53(2), 66–75 (2010)
- Bissyandé T. F., Lo D., Jiang L., Réveillere L., Klein J., and Traon, Y. T. Got issues? who cares about it? a large scale investigation of issue trackers from github. In 2013 IEEE 24th international symposium on software reliability engineering (ISSRE). IEEE Press (2013)
- **Brown, C.**, Middleton, J., Sharma, E., Murphy-Hill, E.: How software users recommend tools to each other. In: Visual Languages and Human-Centric Computing (2017)
- **Brown, C.**, Parnin, C.: Sorry to bother you: designing bots for effective recommendations. In: Proceedings of the 1st International Workshop on Bots in Software Engineering. pp. 54–58. IEEE Press (2019)
- Charette, R.N.: Why software fails [software failure]. IEEE spectrum 42(9), pp. 42–49 (2005)
- Chin, C.: For JavaScript Developers, More Choices Mean Hard Choices. *Wired* (2018) https://www.wired.com/story/javascript-developers-more-choices-mean-hard-choices/
- Distefano, D., Fähndrich, M., Logozzo, F., O’Hearn, P.W.: Scaling static analyses at facebook. Commun. ACM 62(8), pp. 62–70 (2019)
- Duflo, E., Kremer, M., Robinson, J.: Nudging farmers to use fertilizer: Theory and experimental evidence from kenya. American economic review 101(6), pp. 2350–90 (2011)
- Evans, D. and Larochelle, D., 2002. Improving security using extensible lightweight static analysis. *IEEE software*, *19*(1), pp.42-51.

# Citations

- Fogg, B.: Creating persuasive technologies: An eight-step design process. In: Proceedings of the 4th International Conference on Persuasive Technology. pp. 44:1–44:6. Persuasive ’09, ACM, New York, NY, USA (2009)
- Gousios, G., Pinzger, M., Deursen, A.v.: An exploratory study of the pull-based software development model. In: Proceedings of the 36th International Conference on Software Engineering. pp. 345–355. ACM (2014)
- Hanks, A.S., Just, D.R., Smith, L.E., Wansink, B.: Healthy convenience: nudging students toward healthier choices in the lunchroom. Journal of Public Health 34 (3), pp. 370–376 (2012)
- Johnson, B., Song, Y., Murphy-Hill, E., Bowdidge, R.: Why Don’t Software Developers Use Static Analysis Tools to Find Bugs? In: Proceedings of the 2013 International Conference on Software Engineering (ICSE). pp. 672–681. ICSE ’13, IEEE Press, Piscataway, NJ, USA (2013)
- Johnson, E.J., Shu, S.B., Dellaert, B.G., Fox, C., Goldstein, D.G., Häubl, G., Larrick, R.P., Payne, J.W., Peters, E., Schkade, D. and Wansink, B.: Beyond nudges: Tools of a choice architecture. *Marketing Letters*, *23*(2), pp.487-504 (2012)
- Layman, L., Williams, L., Amant, R.S.: Toward reducing fault fix time: Understanding developer behavior for the design of automated fault detection tools. In: Empirical Software Engineering and Measurement, 2007. ESEM 2007. pp. 176–185. IEEE (2007)
- Leech, G.: Principles of Pragmatics. Longman linguistics library ; title no. 30, Longman (1983)
- Li, P.L., Ko, A.J., Zhu, J.: What makes a great software engineer? In: Proceedings of the 37th International Conference on Software Engineering-Volume 1. pp. 700–710. IEEE Press (2015)
- Madrian, B.C., Shea, D.F.: The power of suggestion: Inertia in 401 (k) participation and savings behavior. The Quarterly journal of economics 116(4), 1149–1187 (2001)
- Makabee, H.: How Decision Fatigue Affects the Efficacy of Programmers. Effective Software Design (2011) https://effectivesoftwaredesign.com/2011/08/23/how-decision-fatigue-affects-the-efficacy-of-programmers/
- Middleton, J., Murphy-Hill, E., Green, D., Meade, A., Mayer, R., White, D., McDonald, S.: Which contributions predict whether developers are accepted into github teams. In: Proceedings of the 15th International Conference on Mining Software Repositories. pp. 403–413. MSR ’18, ACM, New York, NY, USA (2018)
- Murphy-Hill, E., Murphy, G.C.: Peer interaction effectively, yet infrequently, enables programmers to discover new tools. In: Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work. pp. 405–414. CSCW ’11, ACM, New York, NY, USA (2011).

# Citations

- Murphy-Hill, E., Murphy, G.C. and McGrenere, J.:How Do Users Discover New Tools in Software Development and Beyond?. Computer Supported Cooperative Work (CSCW), 24(5), pp.389-422 (2015)
- Norman, D. A.: The research-Practice Gap: The need for translational developers. Interactions, 17(4), (2010).
- O'Grady,  S.:  The New Kingmakers: How Developers Conquered the World. "O'Reilly Media, Inc." (2013)
- Robillard, M., Walker, R., Zimmermann, T.: Recommendation systems for software engineering. IEEE software 27(4), 80–86 (2010)
- Singh, D., Sekar, V.R., Stolee, K.T., Johnson, B.: Evaluating how static analysis tools can reduce code review effort. In: 2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). pp. 101–105. IEEE (2017)
- Smith, J., **Brown, C.**, Murphy-Hill, E.: Flower: Navigating program flow in the ide. In: 2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). pp. 19–23 (2017)
- Thaler, R.H., Sunstein, C.R.: Nudge: Improving decisions about health, wealth,and happiness. Penguin (2009)
- Tricentis.: Software Fail Watch: 5th edition. Tricentis (2017) https://www.tricentis.com/resources/software-fail-watch-5th-edition/
- Weinmann, M., Schneider, C., vom Brocke, J.: Digital nudging. Business & Information Systems Engineering 58(6), 433–436 (2016)
- Wisdom, J., Downs, J.S., Loewenstein, G.: Promoting healthy choices: Information versus convenience. American Economic Journal: Applied Economics 2(2), 164–78 (2010)
- Woo, A. Decision-making: The most undervalued skill in software engineering. HackerNoon (2019) https://hackernoon.com/decision-making-the-most-undervalued-skill-in-software-engineering-f9b8e5835ca6

# Back-Up

# Background: *Developer Behavior*

## ***Developer Behavior Adoption Problem***

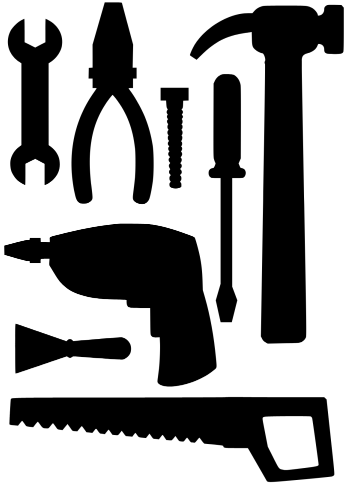
### ***Developer Inertia***

[Murphy-Hill, 2015]

### ***Research-Practice Gap***

[Norman, 2010]

### ***Choice Overload***



[Chin, 2018]

# Background: *Decision-Maker Behavior*

## ***Decision Problems*** [Johnson, 2012]

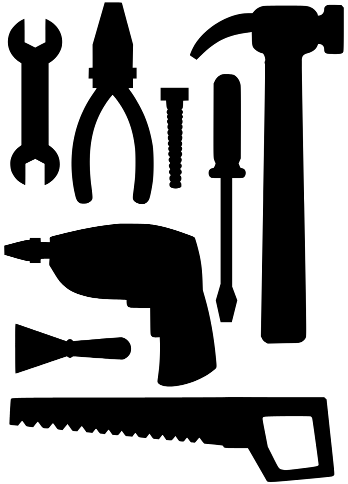
### *Decision Inertia*

[Madrian, 2001]

### *Individual Differences*

[Costa, 2010]

### *Alternative Overload*



[Kling, 2011]

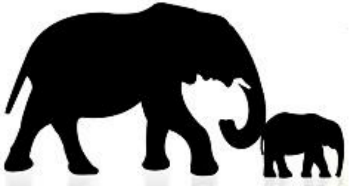
# Developer Recommendation Choice Architectures

*Choice Architecture Tools*

1. Reduce alternatives
2. Technology aids
3. Use defaults
4. Focus on satisficing
5. Limited time windows
6. Decision staging
7. Partitioning of options
8. Attribute labelling
9. Translate for evaluability
10. Customized information
11. Focus on experience

[Johnson, 2012]

1. ***Actionability***
2. ***Feedback***
3. ***Locality***
   a. ***Spatial***
   b. ***Temporal***



# Developer Recommendation Choice Architectures

***Actionability***

- Reduce alternatives
- Technology aids
- Use defaults (Default Rule)

***Feedback***

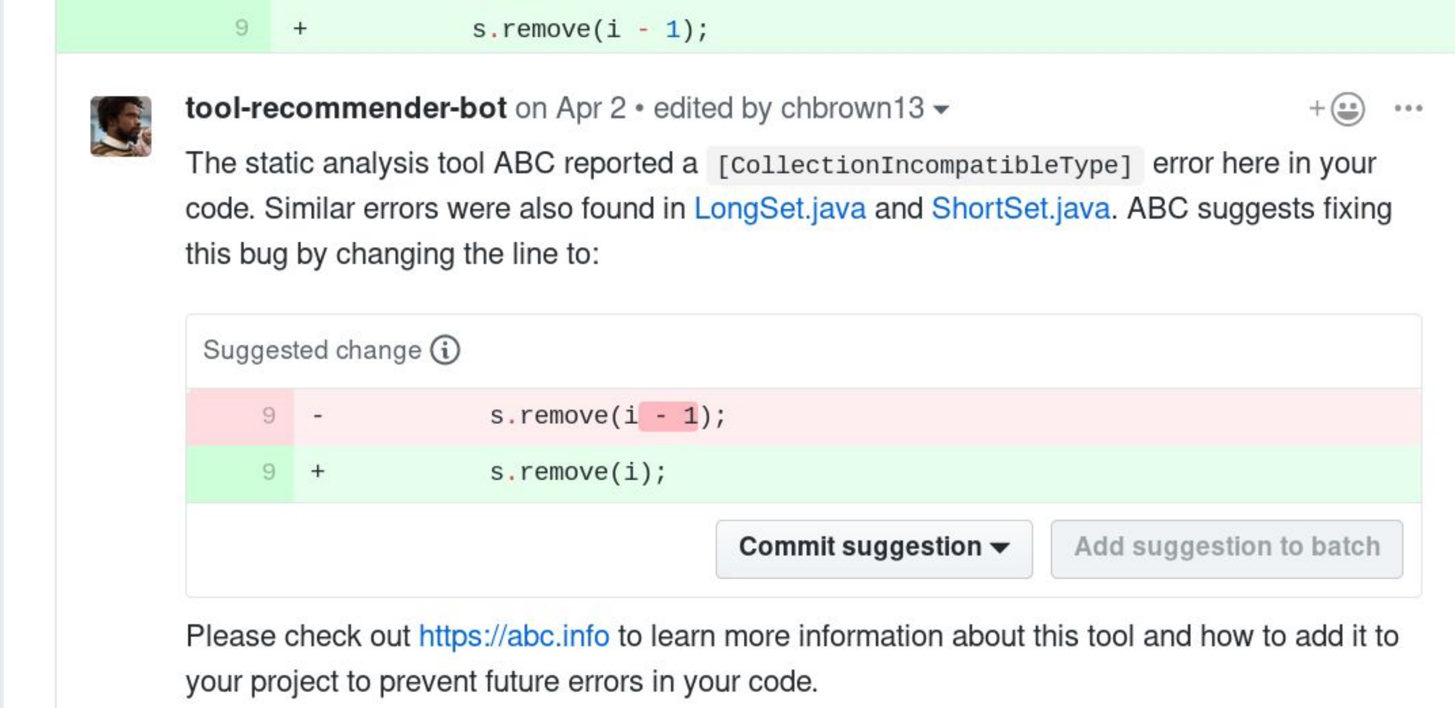
- Focus on satisficing
- Translate for evaluability
- Customized information
- Attribute labelling
- Focus on experience

***Locality***

- Limited time windows
- Decision staging
- Partitioning of options

[Johnson, 2012]

# **Proposed:** Sorry to Bother You 2



```
9  +            s.remove(i - 1);
```

**tool-recommender-bot** on Apr 2 • edited by chbrown13

The static analysis tool ABC reported a `[CollectionIncompatibleType]` error here in your code. Similar errors were also found in LongSet.java and ShortSet.java. ABC suggests fixing this bug by changing the line to:

Suggested change

```
9  -            s.remove(i - 1);
9  +            s.remove(i);
```

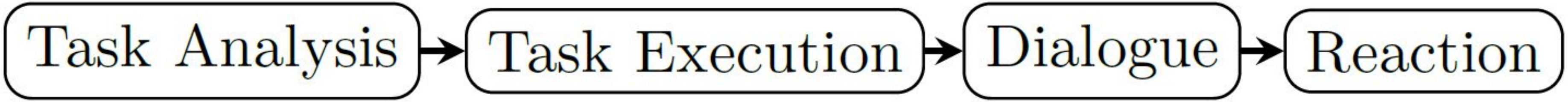
Please check out https://abc.info to learn more information about this tool and how to add it to your project to prevent future errors in your code.

[BotSE, FSE, ASE]

# Nudge-Bot

# Peer Interactions

# Recommendation Model



# 1. Task Analysis

Peers analyze goal and define operations to reach desired state.

# 2. Task Execution

Driver applies selection rule and begins executing their method.

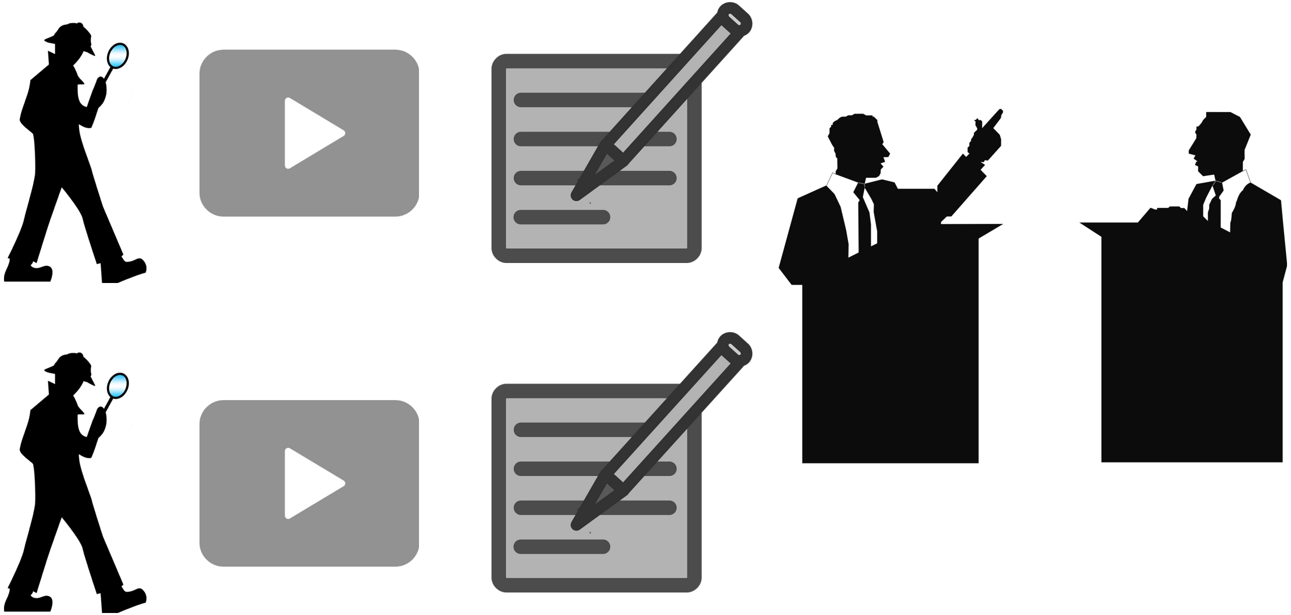
# 3. Dialogue

- *Unexpected Recommendation:* Navigator interrupts to ask about unexpected tool.
- *Expected Recommendation:* Driver asks for help from navigator.
- *Unexpected Observation:* Driver explains actions and navigator reacts.
- *Expected Observation:* Navigator asks question concerning tool used.

# 4. Reaction

The recommendee decides whether or not to adopt the new tool.

# Data Analysis



| | Cohen’s Kappa |
|---|---|
| **Pol.** | 0.50 |
| **Per.** | 0.28 |
| **Rec.** | 0.51 |

# Characteristics of Interactions

1. Politeness [Leech, 1983]
2. Persuasiveness [Shen, 2012]
3. Receptiveness [Fogg, 2009]
4. Time Pressure [Andrews, 1996]
5. Tool Observability [Murphy-Hill, 2015]

[Murphy-Hill, 2015]

# *Politeness*

| **Criteria** | **Definition** |
|---|---|
| Tact | Minimize cost and maximize benefit to peer |
| Generosity | Minimize benefit and maximize cost to self |
| Approbation | Minimize dispraise and maximize praise of peer |
| Modesty | Minimize praise and maximize dispraise of self |
| Agreement | Minimize disagreement and maximize agreement between peers |
| Sympathy | Minimize antipathy and maximize sympathy between peers |

[Leech, 1983]

# *Persuasiveness*

| **Criteria** | **Definition** |
|---|---|
| Content | Recommender provides credible sources to verify use of the tool |
| Structure | Messages are organized by climax-anticlimax order of arguments and conclusion explicitness |
| Style | Messages should avoid hedging, hesitating, questioning intonations, and powerless language |

[Shen, 2012]

# *Receptiveness*

| Criteria | Definition |
|---|---|
| Demonstrate Desire | User showed interest in discovering, using, or learning more information about the suggested tool |
| Familiarity | User explicitly expresses familiarity with the environment |

[Fogg, 2009]

# *Time Pressure*

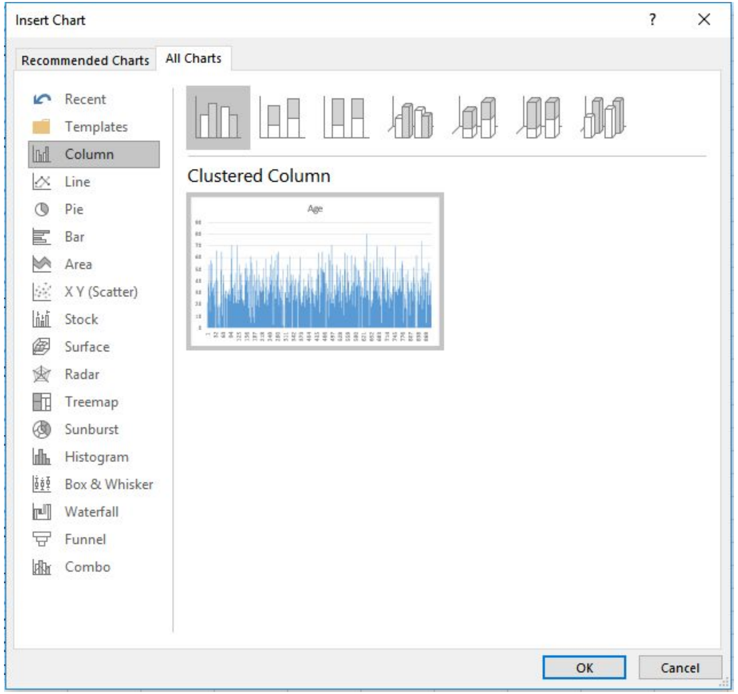
| **Criteria** | **Definition** |
|---|---|
| Time Pressure | Driver or navigator makes a statement about time before, during, or after a recommendation |

[Andrews, 1996]

# Types of Tools

1. Observable

2. Non-Observable



[Murphy-Hill, 2015]

# Methodology: *Scoring*

**Types of Tools**

**Observable** Proposed tool has user interface

**Non-Observable** Proposed tool does not have a user interface

**Politeness, Persuasiveness, Receptiveness**

**3** [illegible] recommended tool

**2** [illegible]

**-1** Participant violates a specific criteria

**1** Recommendee mostly ignores or never uses recommended tool

**Effectiveness**

**Yes** [illegible] interaction

**No** [illegible] the study after [illegible]

# **Results:** *Interaction Characteristics*

| | Polite | Neutral | Impolite |
|---|---|---|---|
| *n* | 27 | 104 | 11 |

***($p$ = 0.4936)*** $^{W}$

| | Persuasive | Unpersuasive |
|---|---|---|
| *n* | 14 | 128 |

***($p$ = 0.4556)*** $^{W}$

| | Receptive | Neutral | Unreceptive |
|---|---|---|---|
| *n* | 64 | 56 | 22 |

***($p$ = 0.0002)**** $^{W}$

| Time Pressure? | Yes | No |
|---|---|---|
| *n* | 19 | 123 |

***($p$ = 0.1470)*** $^{C}$

***W*** = Wilcoxon rank sum, ***C*** = Pearson's chi-squared, ***** = significant

# Results: *Tool Observability*

| | **Observable** | **Non-Observable** |
|---|---|---|
| ***n*** | 115 | 27 |

***(p = 0.4928)*** $^{C}$

***W*** = Wilcoxon rank sum, ***C*** = Pearson’s chi-squared, ***** = significant

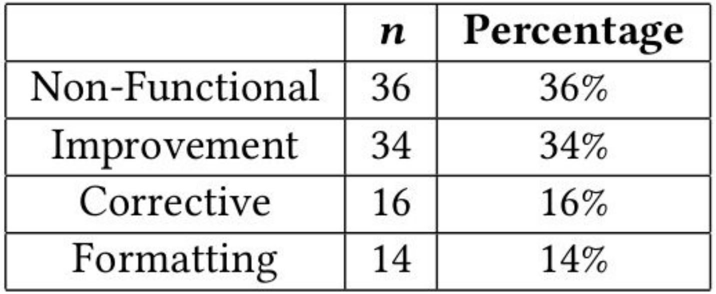
# Sorry to Bother You

# Developer Feedback

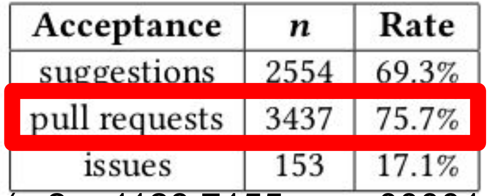
- 24 comments on 17 projects
  - 6 bot comments for first-time contributors, Contributing License Agreement signatures, test coverage
  - 18 developer comments (non-automated)
    - Positive: 5
    - Pom.xml format: 5
    - Breaking builds: 8

# Suggestions

# **Suggestions:** Results (Phase 1)

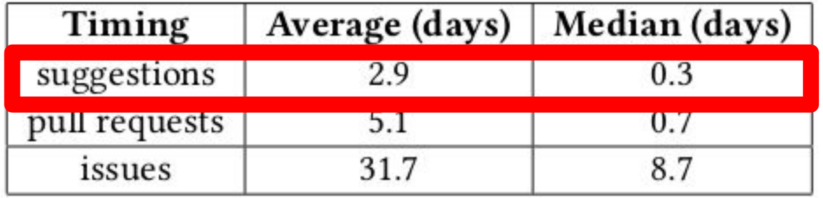


|  | *n* | Percentage |
|---|---|---|
| Non-Functional | 36 | 36% |
| Improvement | 34 | 34% |
| Corrective | 16 | 16% |
| Formatting | 14 | 14% |



| Acceptance | *n* | Rate |
|---|---|---|
| suggestions | 2554 | 69.3% |
| pull requests | 3437 | 75.7% |
| issues | 153 | 17.1% |

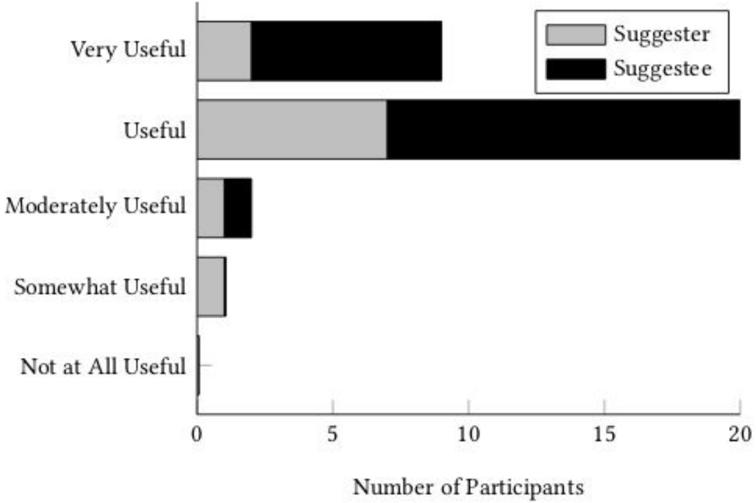
(χ 2 = 1128.7155, p < .00001, α = .05)

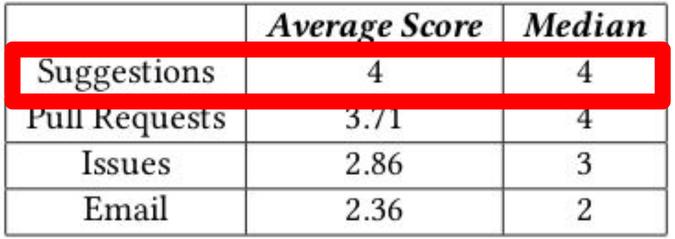


| Timing | Average (days) | Median (days) |
|---|---|---|
| suggestions | 2.9 | 0.3 |
| pull requests | 5.1 | 0.7 |
| issues | 31.7 | 8.7 |

(Kruskal-Wallis = 391.844102, p < .0001, α = .05)

# **Suggestions:** Results (Phase 2)

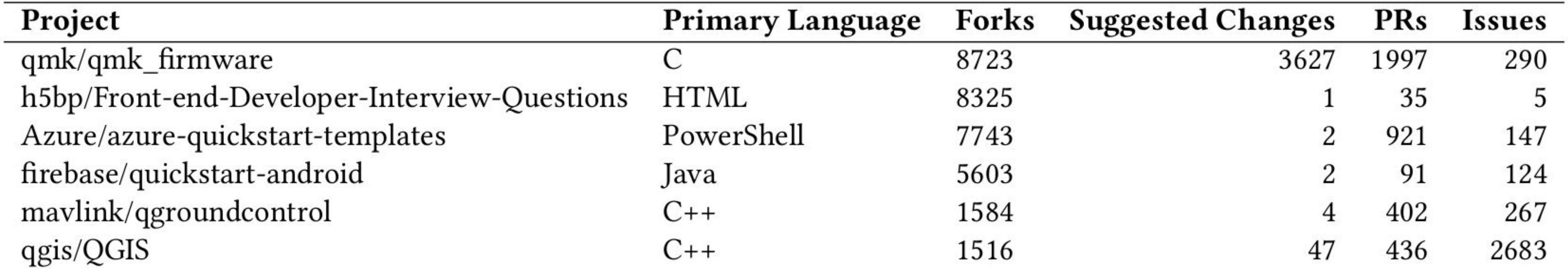




| | *Average Score* | *Median* |
|---|---|---|
| Suggestions | 4 | 4 |
| Pull Requests | 3.71 | 4 |
| Issues | 2.86 | 3 |
| Email | 2.36 | 2 |

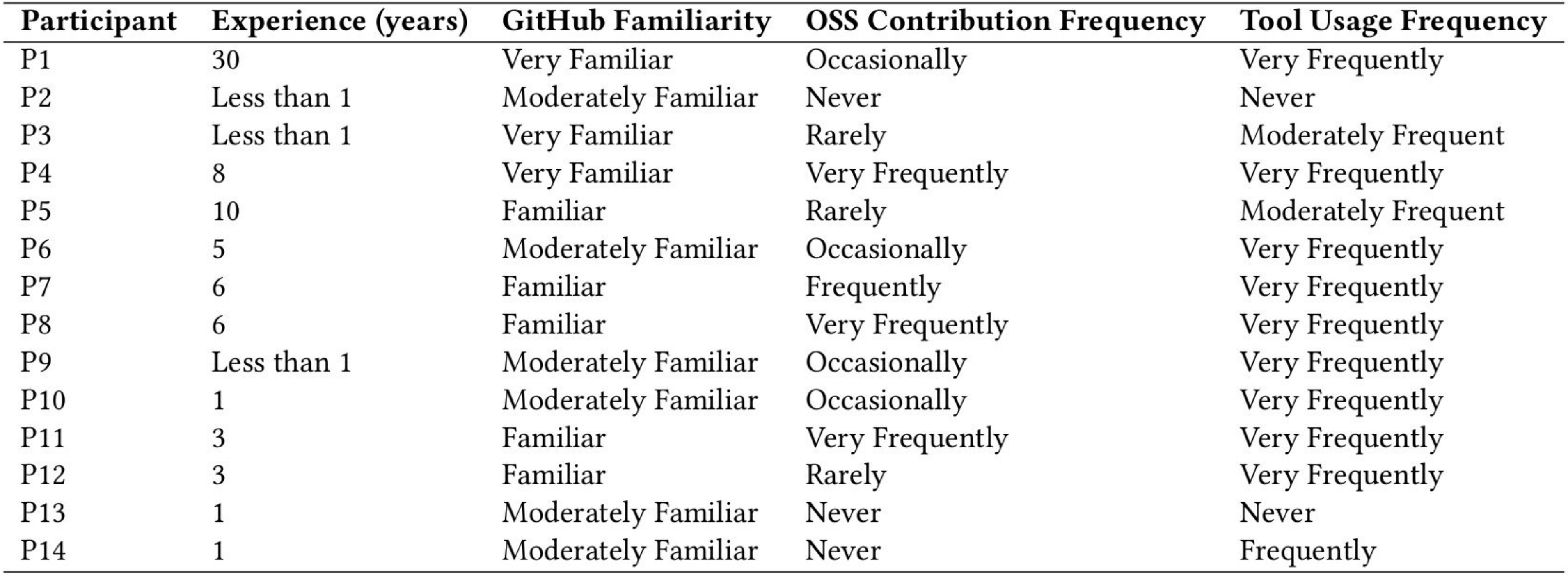
(Kruskal-Wallis, p = .00079, α = .05)

# Study Projects



| Project | Primary Language | Forks | Suggested Changes | PRs | Issues |
|---|---|---|---|---|---|
| qmk/qmk_firmware | C | 8723 | 3627 | 1997 | 290 |
| h5bp/Front-end-Developer-Interview-Questions | HTML | 8325 | 1 | 35 | 5 |
| Azure/azure-quickstart-templates | PowerShell | 7743 | 2 | 921 | 147 |
| firebase/quickstart-android | Java | 5603 | 2 | 91 | 124 |
| mavlink/qgroundcontrol | C++ | 1584 | 4 | 402 | 267 |
| qgis/QGIS | C++ | 1516 | 47 | 436 | 2683 |

# Study Participants

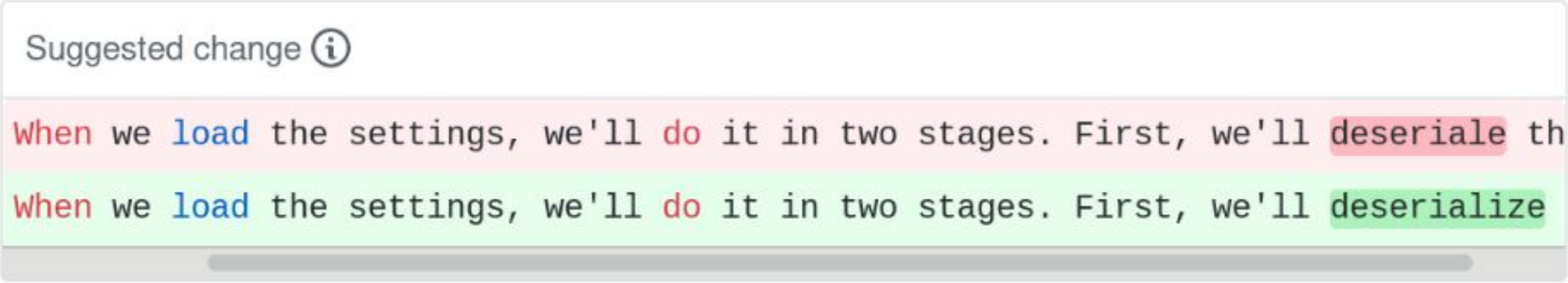


| Participant | Experience (years) | GitHub Familiarity | OSS Contribution Frequency | Tool Usage Frequency |
|---|---|---|---|---|
| P1 | 30 | Very Familiar | Occasionally | Very Frequently |
| P2 | Less than 1 | Moderately Familiar | Never | Never |
| P3 | Less than 1 | Very Familiar | Rarely | Moderately Frequent |
| P4 | 8 | Very Familiar | Very Frequently | Very Frequently |
| P5 | 10 | Familiar | Rarely | Moderately Frequent |
| P6 | 5 | Moderately Familiar | Occasionally | Very Frequently |
| P7 | 6 | Familiar | Frequently | Very Frequently |
| P8 | 6 | Familiar | Very Frequently | Very Frequently |
| P9 | Less than 1 | Moderately Familiar | Occasionally | Very Frequently |
| P10 | 1 | Moderately Familiar | Occasionally | Very Frequently |
| P11 | 3 | Familiar | Very Frequently | Very Frequently |
| P12 | 3 | Familiar | Rarely | Very Frequently |
| P13 | 1 | Moderately Familiar | Never | Never |
| P14 | 1 | Moderately Familiar | Never | Frequently |

# Types of Suggested Changes

**Non-functional:** changes that don’t impact code, i.e. rewording or fix spelling and grammar issues in documentation and code comments.

**(a) Non-Functional:**



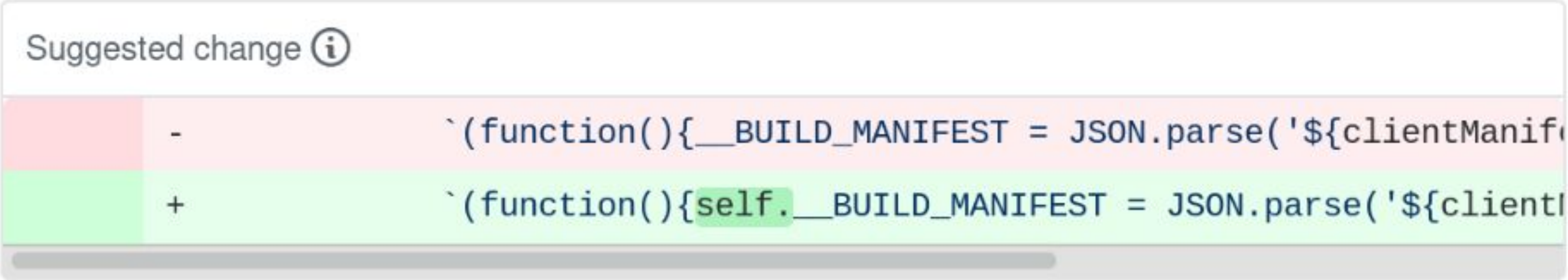
Suggested change ⓘ

```
When we load the settings, we'll do it in two stages. First, we'll deseriale th
When we load the settings, we'll do it in two stages. First, we'll deserialize
```

# Types of Suggested Changes

**Corrective:** changes to fix bugs and issues found in the code.

(b) Corrective:



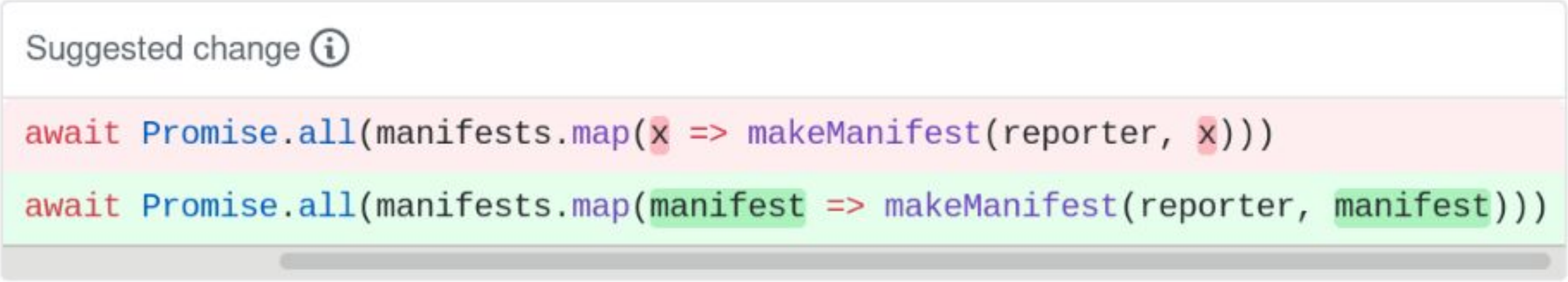
Suggested change ⓘ

```
-         `(function(){__BUILD_MANIFEST = JSON.parse('${clientManif
+         `(function(){self.__BUILD_MANIFEST = JSON.parse('${client
```

# Types of Suggested Changes

**Improvement:** changes to refactor or optimize code.

**(c) Improvement:**

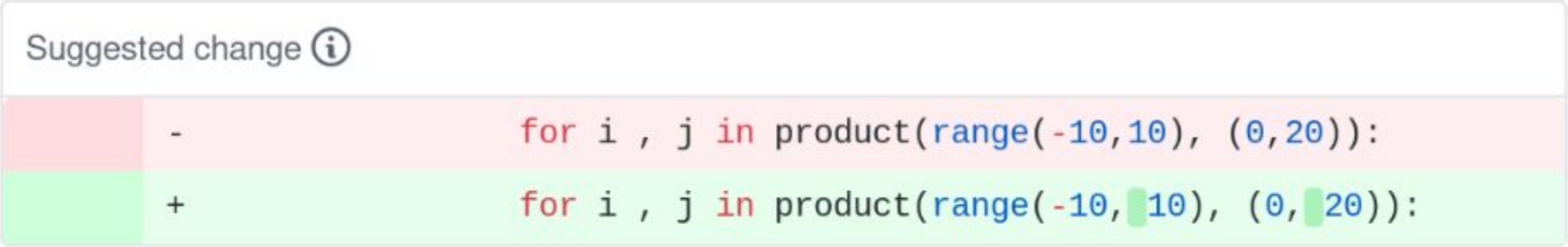


```
await Promise.all(manifests.map(x => makeManifest(reporter, x)))
await Promise.all(manifests.map(manifest => makeManifest(reporter, manifest)))
```

# Types of Suggested Changes

**Formatting:** changes that impact the presentation of the code without changing functionality

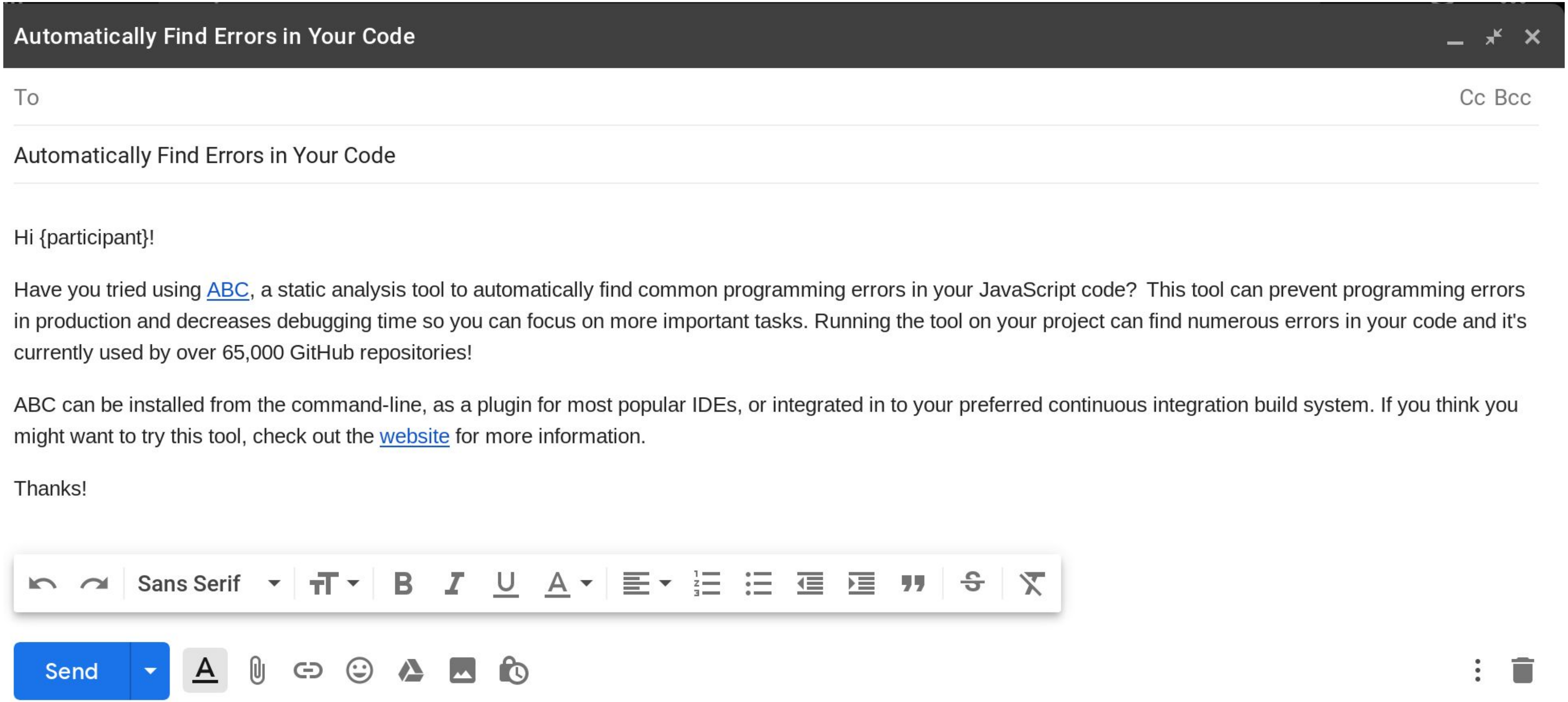
**(d) Formatting:**



Suggested change

```
-        for i , j in product(range(-10,10), (0,20)):
+        for i , j in product(range(-10, 10), (0, 20)):
```

# User Study Email



Automatically Find Errors in Your Code

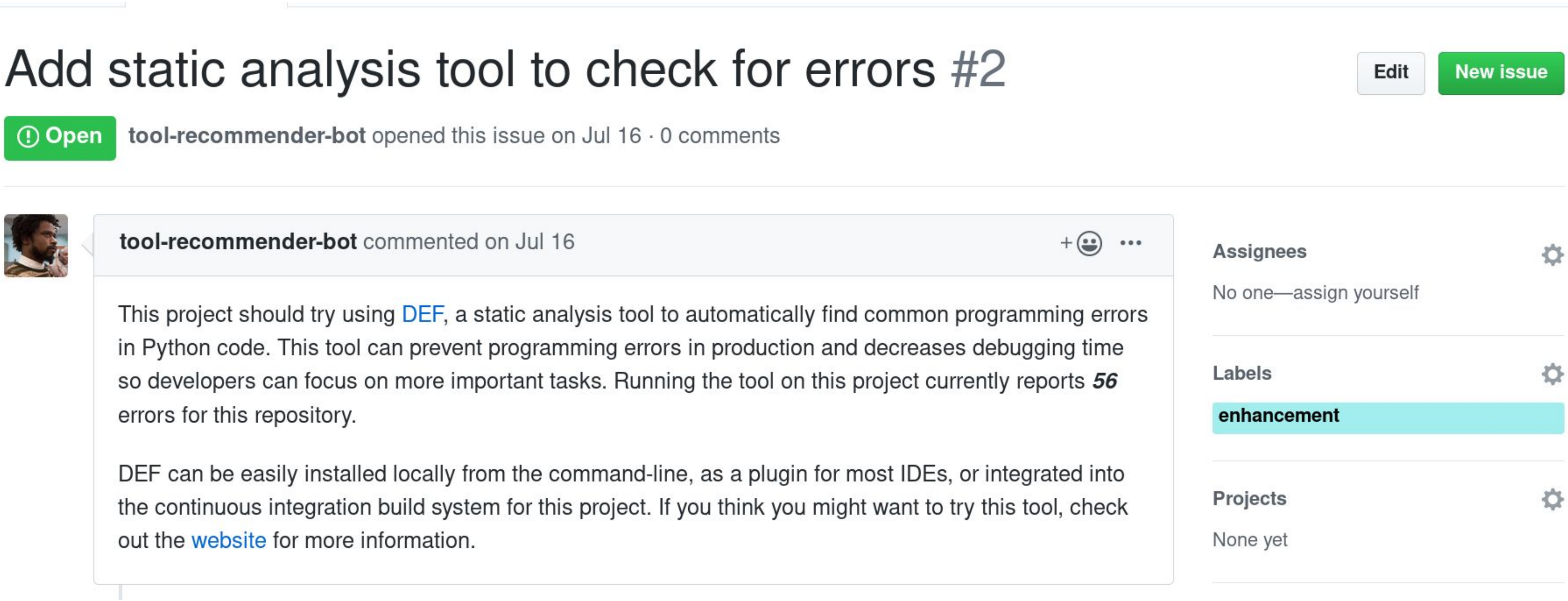
Hi {participant}!

Have you tried using ABC, a static analysis tool to automatically find common programming errors in your JavaScript code?  This tool can prevent programming errors in production and decreases debugging time so you can focus on more important tasks. Running the tool on your project can find numerous errors in your code and it's currently used by over 65,000 GitHub repositories!

ABC can be installed from the command-line, as a plugin for most popular IDEs, or integrated in to your preferred continuous integration build system. If you think you might want to try this tool, check out the website for more information.

Thanks!

# User Study Issue



Add static analysis tool to check for errors #2

**tool-recommender-bot** opened this issue on Jul 16 · 0 comments

**tool-recommender-bot** commented on Jul 16

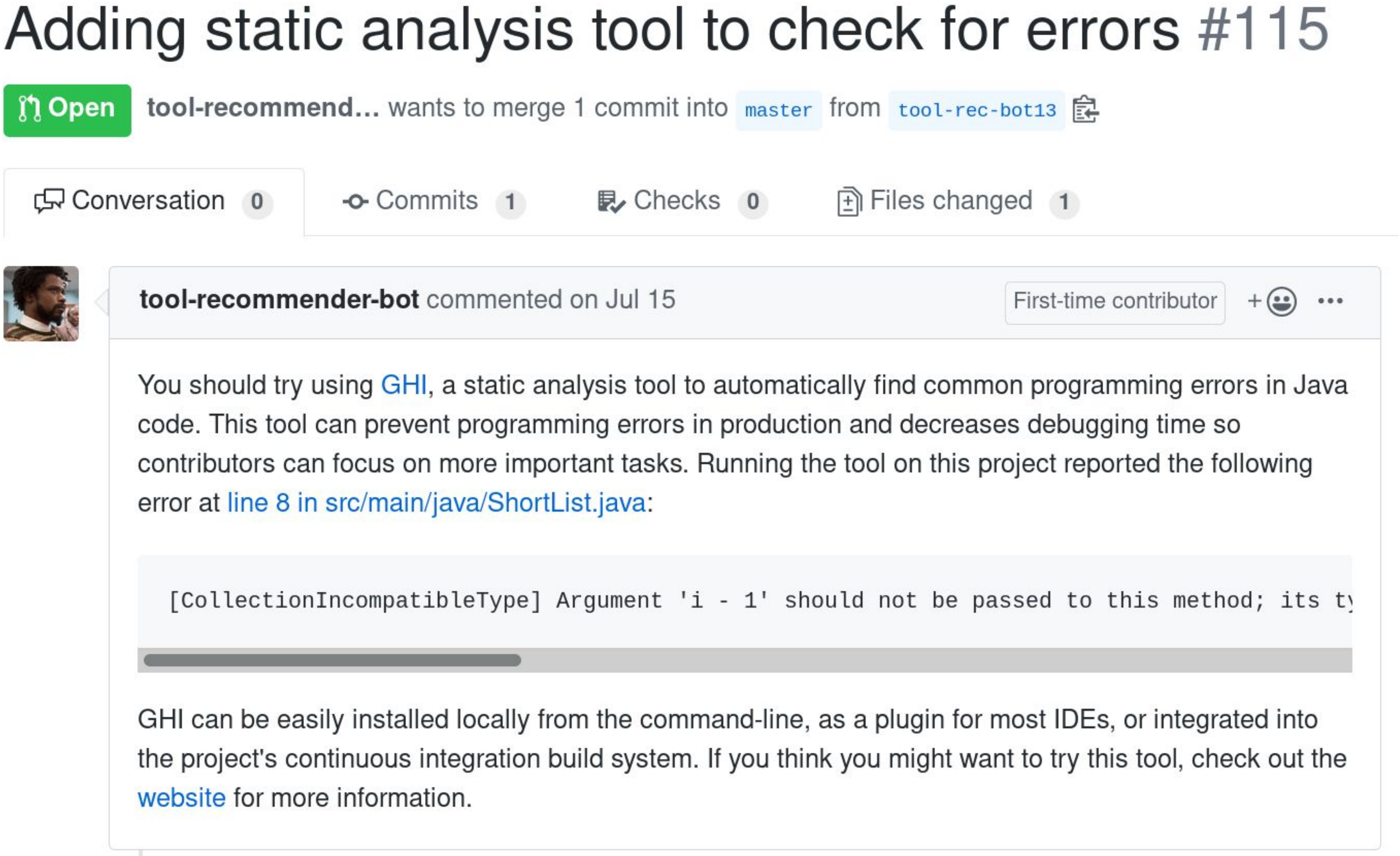
This project should try using DEF, a static analysis tool to automatically find common programming errors in Python code. This tool can prevent programming errors in production and decreases debugging time so developers can focus on more important tasks. Running the tool on this project currently reports ***56*** errors for this repository.

DEF can be easily installed locally from the command-line, as a plugin for most IDEs, or integrated into the continuous integration build system for this project. If you think you might want to try this tool, check out the website for more information.

Assignees: No one—assign yourself

Labels: enhancement

Projects: None yet



# Adding static analysis tool to check for errors #115

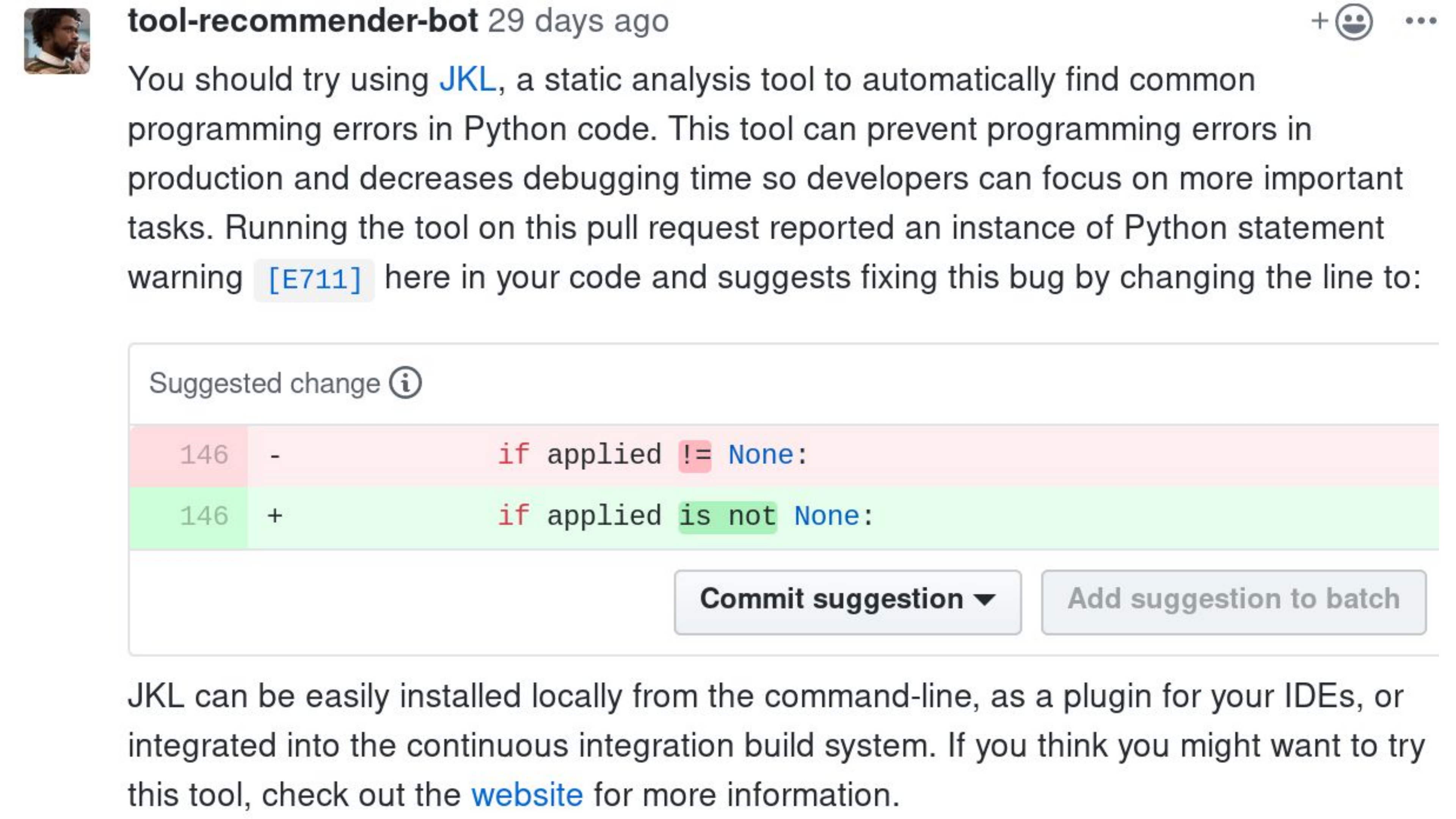
tool-recommend... wants to merge 1 commit into `master` from `tool-rec-bot13`

**tool-recommender-bot** commented on Jul 15

You should try using GHI, a static analysis tool to automatically find common programming errors in Java code. This tool can prevent programming errors in production and decreases debugging time so contributors can focus on more important tasks. Running the tool on this project reported the following error at line 8 in src/main/java/ShortList.java:

```
[CollectionIncompatibleType] Argument 'i - 1' should not be passed to this method; its ty
```

GHI can be easily installed locally from the command-line, as a plugin for most IDEs, or integrated into the project's continuous integration build system. If you think you might want to try this tool, check out the website for more information.



**tool-recommender-bot** 29 days ago

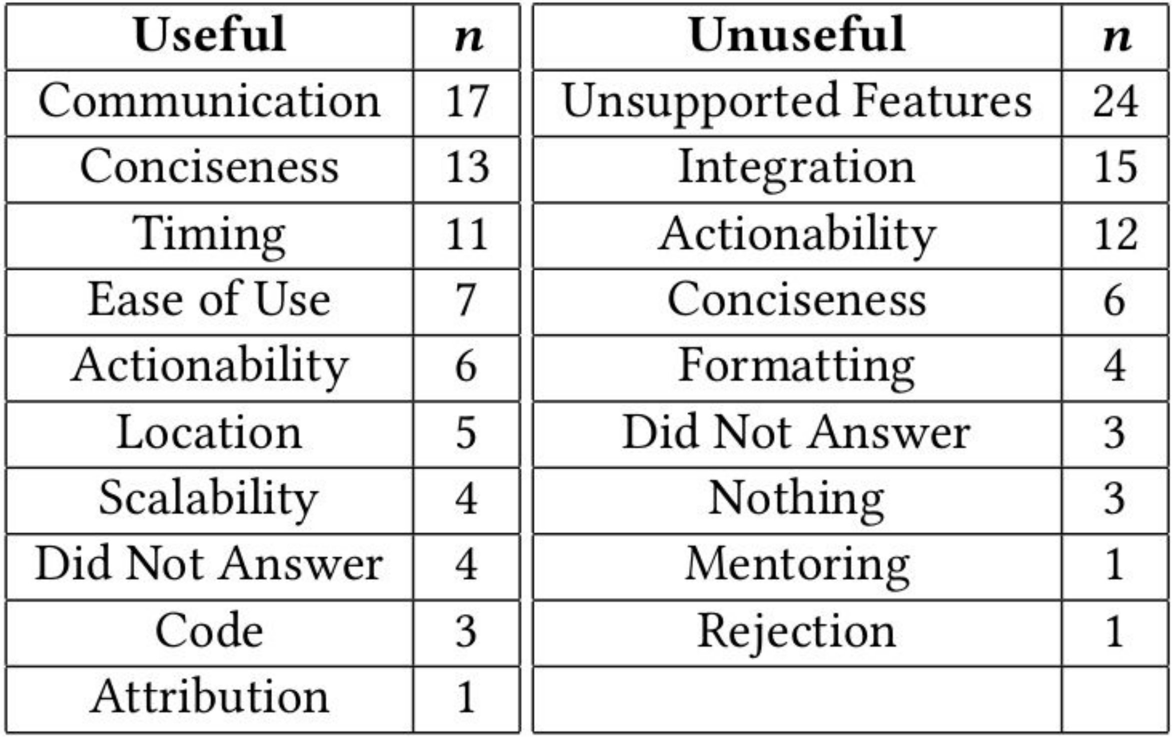
You should try using JKL, a static analysis tool to automatically find common programming errors in Python code. This tool can prevent programming errors in production and decreases debugging time so developers can focus on more important tasks. Running the tool on this pull request reported an instance of Python statement warning `[E711]` here in your code and suggests fixing this bug by changing the line to:

Suggested change

```
-            if applied != None:
+            if applied is not None:
```

JKL can be easily installed locally from the command-line, as a plugin for your IDEs, or integrated into the continuous integration build system. If you think you might want to try this tool, check out the website for more information.

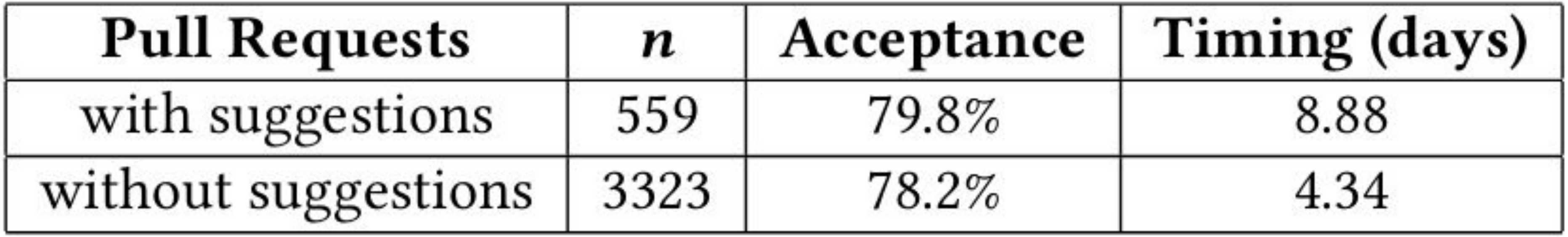
# **Suggestions:** Usefulness



| Useful | *n* |
|---|---|
| Communication | 17 |
| Conciseness | 13 |
| Timing | 11 |
| Ease of Use | 7 |
| Actionability | 6 |
| Location | 5 |
| Scalability | 4 |
| Did Not Answer | 4 |
| Code | 3 |
| Attribution | 1 |

| Unuseful | *n* |
|---|---|
| Unsupported Features | 24 |
| Integration | 15 |
| Actionability | 12 |
| Conciseness | 6 |
| Formatting | 4 |
| Did Not Answer | 3 |
| Nothing | 3 |
| Mentoring | 1 |
| Rejection | 1 |
| | |

# **Suggestions:** Pull Requests



| **Pull Requests** | ***n*** | **Acceptance** | **Timing (days)** |
|---|---|---|---|
| with suggestions | 559 | 79.8% | 8.88 |
| without suggestions | 3323 | 78.2% | 4.34 |

# Recommendations on GitHub

## Pull Requests

## Issues