

Digital Nudges for Encouraging Developer Behaviors

Chris Brown

dcbrow10@ncsu.edu

April 6, 2021

Committee:

Dr. Chris Parnin (Chair)

Dr. Anne McLaughlin (PSY, GSR)

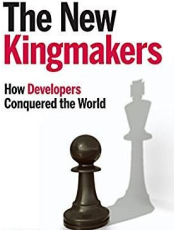


Dr. Sarah Heckman

Dr. Kathryn Stolee

Final Oral Exam (Defense)
Department of Computer Science

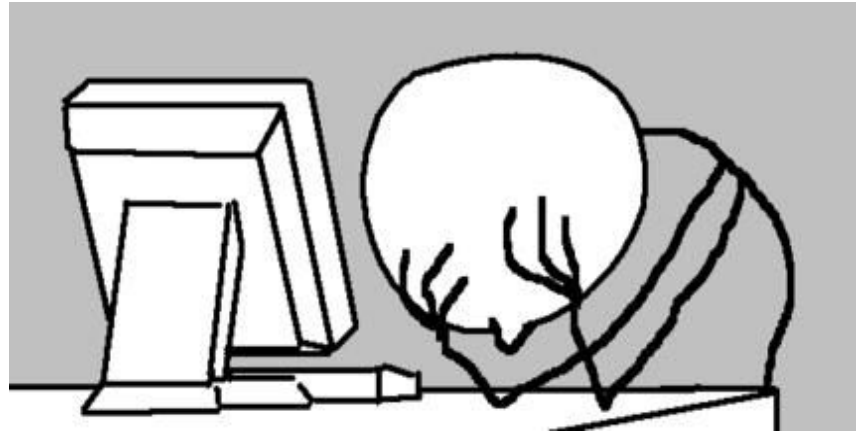
NC STATE
UNIVERSITY

Decision-Making in 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 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] 
- “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 often make bad decisions!



[Johnson, 2013]



[Rahman, 2019]



[Runeson, 2006]



[McNamara, 2011]

Impact

PEOPLE AFFECTED (AT LEAST)
3,683,212,665
1,715,430,778,504
LOSSES FROM SOFTWARE FAILURES (USD)

[Tricentis, 2017]

PAIRAGRAPH

Is Technology Actually Making Things Better?

John K. Davis
Philosophy, Cal State Fullerton



Jason Crawford
Author, The Roots of Progress



“We face a growing array of problems that involve technology directly or indirectly... The growing gap between our technological power and our wisdom is the ultimate cause of all these problems.”

[Crawford, 2020]



Greg Wilson

@gvwilson

Follow



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 ...

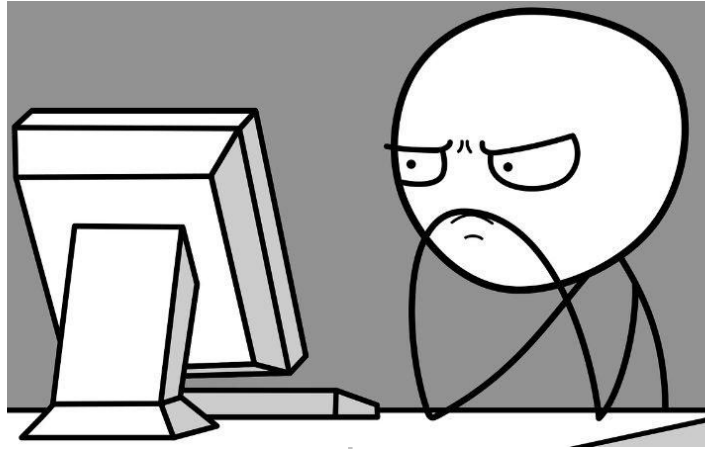
bmcpsychology.biomedcentral.com

6:38 PM - 21 Jun 2019

7 Retweets 16 Likes



How can we encourage software engineers to adopt developer behaviors in their work?



Bad Decision



Good Decision



Outline

Background

Thesis Statement

Exploring Effective Developer Recommendations

Developing a Conceptual Framework

Analyzing Existing Recommendation Systems

Designing New Recommender Bots

Future Work and Conclusion

Background: Human Behavior

Tools and guidelines informed by science can encourage humans to adopt beneficial behaviors and make better decisions.

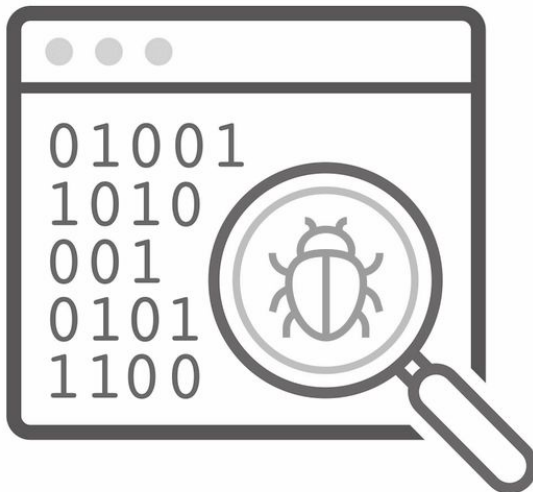
However, people often ignore these recommendations.



Background: *Developer Behavior*



Tools and practices designed by researchers to help software engineers complete programming tasks. However, developers ignore these behaviors.



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

[Johnson, 2013]

***Developer Behavior
Adoption Problem***

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.

Research Contributions

1. *A set of experiments* to explore recommendations to developers and motivate the need for a new approach. ☐
2. *A conceptual framework* to design effective developer recommendations. ☐
3. *A set of experiments* to provide evidence supporting the conceptual framework. ☐
4. *An automated recommender system* that incorporates the framework to nudge programmers toward developer behaviors. ☐

Thesis: *Effective Recommendations*



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.



Peer Interactions

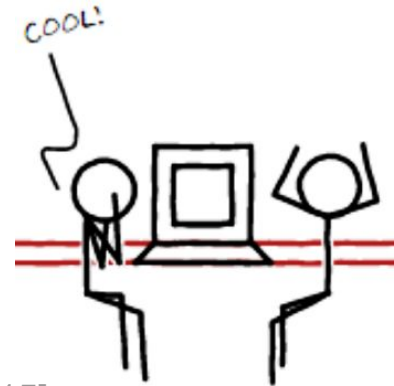
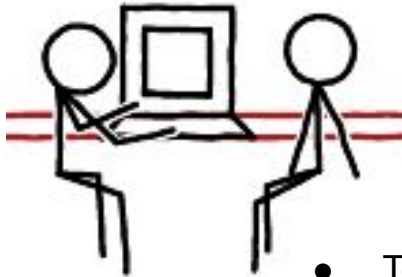


Naive *Telemarketer Design*

Peer Interactions



The process of discovering tools from colleagues during normal work activities.



- Tool Discovery [Murphy-Hill, 2015]
- Code Comprehension [Maalej, 2014]
- Security Tool Adoption [Xiao, 2014]
- Pair Programming [Cockburn, 2001]
- Code Reviews [Cohen, 2006] ...

[Murphy-Hill, 2011]

Methodology



Study Design

- 13 pairs of participants
- Data Analysis Tasks
- Tool Recommendations

■ **Characteristics**

■ **Effectiveness**

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

Effective User tries recommended tool

Ineffective User ignores recommended tool

Unknown No opportunity to use tool



Results



	Effective	Ineffective	Unknown	Total
<i>n</i>	71	35	36	142

1. Politeness
2. Persuasiveness
- 3. *Receptiveness****
4. Time Pressure
5. Tool Observability

Demonstrate Desire

Familiarity

** (Wilcoxon, $p = 0.0002$, $OR = 0.2840$)*

[Fogg, 2009]

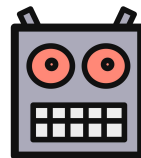
Baseline Automated Approach



Naive telemarketer design: A simple approach for designing automated recommendations to software engineers.

- *Static Recommendations*
- *Generic Messages*
- *Socially Inept*

tool-recommender-bot



Error Prone Static Analysis Tool #82



Open

cass-green wants to merge 1 commit into [apache:master](#) from [cass-green:master](#)



Conversation 0



Commits 1



Checks 0



Files changed 1



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.

Methodology



Study Design

- Error Prone
- 52 GitHub repositories
 - Java 8+
 - Maven

- Effectiveness
- Feedback

Effective

Merged

Ineffective

Closed

Open



Vest commented on Feb 7, 2019

This PR failed automatic checks, I think it should be closed.



gastaldi commented on Jan 29, 2019

Thanks for the contribution, but given the number of errors, I think it would cause more harm than good ;)



bendem commented on Jan 28, 2019

Contributor

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.



gunnarmorling commented on Jan 31, 2019

Member

So I'm going to close this one, as it's not mergeable as is. If you'd like to re-open it, please do so by not altering large parts of the POM. Also it'd be great to see how ErrorProne would actually help us, e.g. you could attach a report with actual findings in our code base instead of just some generic example. Also it'd be good to analyse the impact in terms of build time.

Results



	<i>n</i>	Percent
Merged	2	4%
Closed	10	19%
No Response	40	77%

Social Context

```
88      88      </plugin>
89 + <plugin>
90 + <groupId>org.apache.maven.plugins</groupId>
91 + <artifactId>maven-compiler-plugin</artifactId>
92 + <version>3.5.1</version>
```

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

Developer Workflow



All checks have failed

1 errored check



continuous-integration/travis-ci/pr — The Travis CI build

Recap



- Peer interactions are effective because of their ability to foster *desirable* and *familiar* suggestions, but infrequent among developers.
- Simple automated approaches are ineffective because they lack *social context* and interrupt *developer workflow*.

Developer Recommendation Preconditions

Demonstrate Desire

Familiarity

Social Context

Developer Workflow

Can automated recommendations be improved to better encourage developer behaviors?

Thesis: *Conceptual Framework*

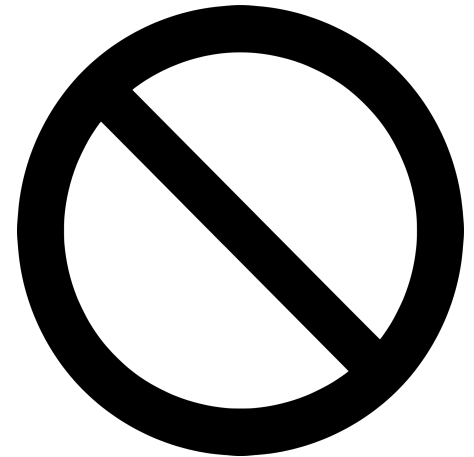
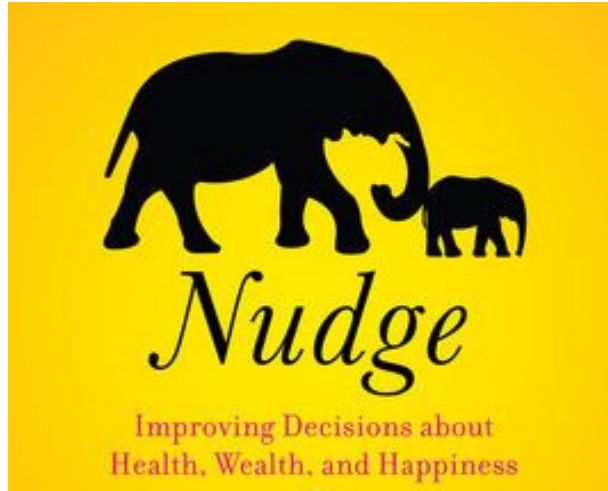
By incorporating ***developer recommendation choice architectures*** into recommendations for software engineers, we can ***nudge*** developers to adopt behaviors useful for improving code quality and developer productivity.



Developer Recommendation Choice Architectures

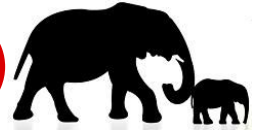
Background: Nudge Theory

A behavioral science framework to improve human behavior without providing incentives or banning alternatives.

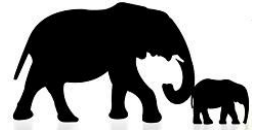


[Thaler and Sunstein, 2009]

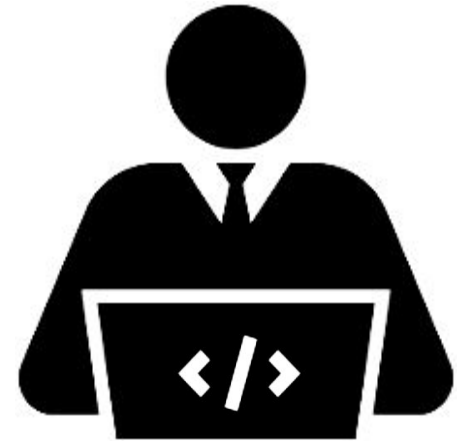
Background: Nudge Theory (cont.)



Background: Digital Nudges

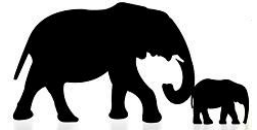


The use of nudges to guide user behavior in digital choice environments.



[Weinmann, 2016]

Background: Choice Architecture

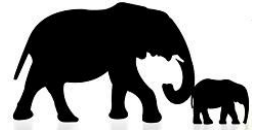


The framing and presentation of choices to decision-makers



[Thaler, 2009]

Tools for Choice Architectures



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

Developer Recommendation Choice Architectures

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

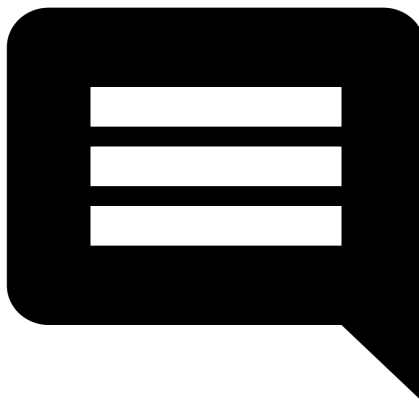
Actionability

The ease with which developers can adopt recommendations



Feedback

Information provided in recommendations



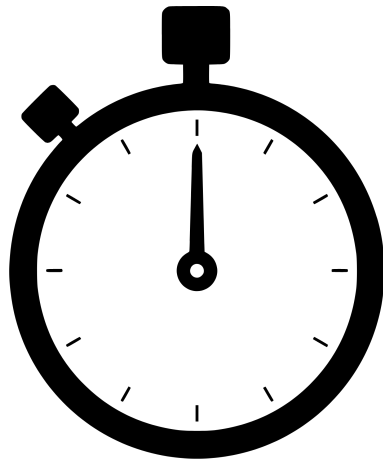
Locality: *Spatial*

The setting (placement) of recommendations



Locality: *Temporal*

The setting (timing) of recommendations



Actionable Recommendations



```
9 + if status is True:
10 +     print 'passed'
```



cass-green now



Hi, the latest version of Python changes `print` to a built-in function instead of a statement, leading to a PEP 3105 warning here [1]. We recommend changing this line to:

```
print('passed')
```

This change
longer support
project to Py

code. Additionally, Python is officially no
Please consider upgrading the code for your

[1] <https://www.python.org/dev/peps/pep-3105/>
[2] <https://www.python.org/doc/sunset-python-2/>



Reply...

```
9 + if status is True:
10 +     print 'passed'
```



cass-green 33 seconds ago



Hi, the latest version of Python changes `print` to a built-in function instead of a statement, leading to a PEP 3105 warning here [1]. We recommend changing this line to

Suggested change ⓘ

```
10 - print 'passed'
10 + print('passed')
```

Commit suggestion ▼

Add suggestion to batch

This change will not impact the functionality of your code. Additionally, Python is officially no longer supporting Python 2 as of Jan. 1, 2020 [2]. Please consider upgrading the code for your project to Python 3. Thanks!

[1] <https://www.python.org/dev/peps/pep-3105/>
[2] <https://www.python.org/doc/sunset-python-2/>



Reply...

0%

100%

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.



GitHub Suggested Changes

Recommendation Styles
Developer Impact

9 + int c;



chbrown13 26 days ago

Author

Owner



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

 Open

Suggested change ⓘ

9 - int c;

9 + int count;



Commit suggestion ▼

Add suggestion to batch

Update src/main/java/ShortSet.java

Actionability

Spatial Locality

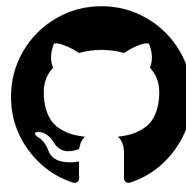
Feedback

Temporal Locality



Commit changes

Recommendation Styles

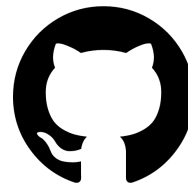


Automated approaches to convey recommendations to developers.

- Static Analysis Tool Adoption

1. Email
2. GitHub Issue
3. GitHub Pull Request
4. GitHub Suggested Changes

Methodology



Study Design

- 14 professional developers
- Tool Recommendations



- Think-aloud study
- Semi-structured interview

- **Likelihood of Adoption**
- **Open-ended**



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 ⓘ

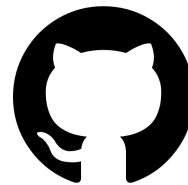
146	-	<code>if applied != None:</code>
146	+	<code>if applied is not None:</code>

Commit suggestion ▼

Add suggestion to batch

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.

Results



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

“Very good detail...clear” (P10) “Pretty neat integration” (P7) ...

“[It’s] pretty suspicious” (P4) “I’d immediately delete it” (P6) ...

* Kruskal-Wallis ($H = 16.7527$, $p = .00079$, $\alpha = .05$)

Integration

Popularity

Reliability

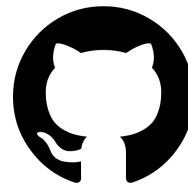
Examples

Marketing

Content

Design

Observations



Content

Design



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 ⓘ

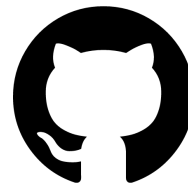
146	-	<code>if applied != None:</code>
146	+	<code>if applied is not None:</code>

Commit suggestion ▼

Add suggestion to batch

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.

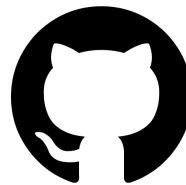
Developer Impact



***An Empirical
Study on
GitHub
Suggested
Changes***

***Developer
Feedback on
Suggested
Changes***

Detecting Suggested Changes



 **chbrown13** on Nov 6, 2019 Author Owner 😊 ⋮

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

Suggested change ⓘ

9	-	<code>int c;</code>
9	+	<code>int count;</code>

Commit suggestion ▼ Add suggestion to batch

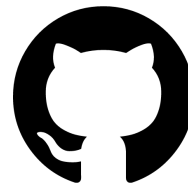
Please don't use single character variable names...

```suggestion

int count;

```

Methodology: Phase 1



An Empirical Study on GitHub Suggested Changes

Study Design

RQ1. What types of recommendations do developers make with suggested changes?

- Recently updated repositories
- 100 suggested changes



(IRR = 71%, Cohen's κ = 0.5942)

RQ2. How effective are recommendation systems on pull requests?

- Top-forked repositories (51,250 PRs)
- 17,712 suggested changes
- 134,318 review comments



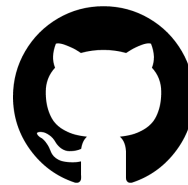
RQ3. What impact do suggested changes have on pull requests?

- Top-forked repositories
- 4,319 PRs with suggestions
- 46,931 PRs without



[Gousios, 2014]

Results: Phase 1 (RQ1)



RQ1. Most suggested changes are *non-functional*.

	<i>n</i>	Percentage
Non-Functional	36	36%
Improvement	34	34%
Corrective	16	16%
Formatting	14	14%

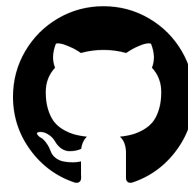
(a) Non-Functional:

Suggested change ⓘ

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

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

Results: Phase 1 (RQ2)

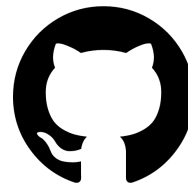


RQ2. Suggested changes are ***accepted**** more often than review comments with code.

	<i>n</i>	Percentage
<i>Suggested Changes (SC)*</i>	17,712	59.6%
Review Comments (RC)	6,937	0.9%

* ($\chi^2 = 6961.3765, p < 0.00001, \alpha = .05$)

Results: Phase 1 (RQ2 cont.)



RQ2. Pull requests with suggested changes have longer ***review time****, but developers can ***make**** and ***respond**** to recommendations faster.

Recommendation Time (days)*

	Mean	Median
SC	10.5	0.7
RC	14.6	1.9

* ($W = 49186174$, $p < 0.00001$, $\alpha = .05$)

* Statistically significant

Time (days)*

	Mean	Median
SC	16.4	5.0
RC	6.4	1.1

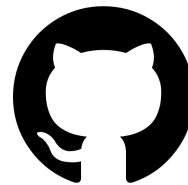
* (Wilcoxon, $W = 87857043$, $p < 0.00001$, $\alpha = .05$)

Acceptance Time (days)*


	Mean	Median
SC	5.4	0.3
RC	8.0	0.7


* ($W = 256013$, $p = 0.0001$, $\alpha = .05$)


Results: Phase 1 (RQ3)



RQ3. PRs with suggested changes have *longer reviews** but more *coding activity** and *collaboration between developers**.

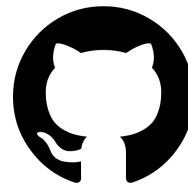
	Mean (diff)	p
<i>lifetime_minutes*</i>	+15,805.78	$p < 0.00001$
<i>mergetime_minutes*</i>	+13,266.33	$p < 0.00001$

	Mean (diff)	p
<i>num_commits*</i>	+1.9	$p < 0.00001$
<i>src_churn*</i>	+2,345.78	$p < 0.00001$
<i>files_changed</i>	+3.83	$p = 0.5051$

	Mean (diff)	p
<i>commit_comments*</i>	+11.56	$p < 0.00001$
<i>issue_comments*</i>	+3.6	$p < 0.00001$
<i>num_participants*</i>	+0.88	$p < 0.00001$

* Statistically significant (Wilcoxon, $\alpha = .05$)

Methodology: Phase 2



Developer Feedback on Suggested Changes

Study Design

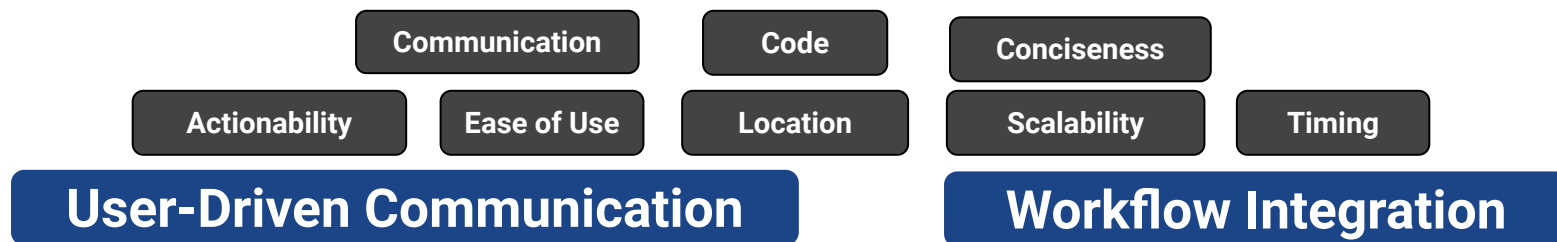
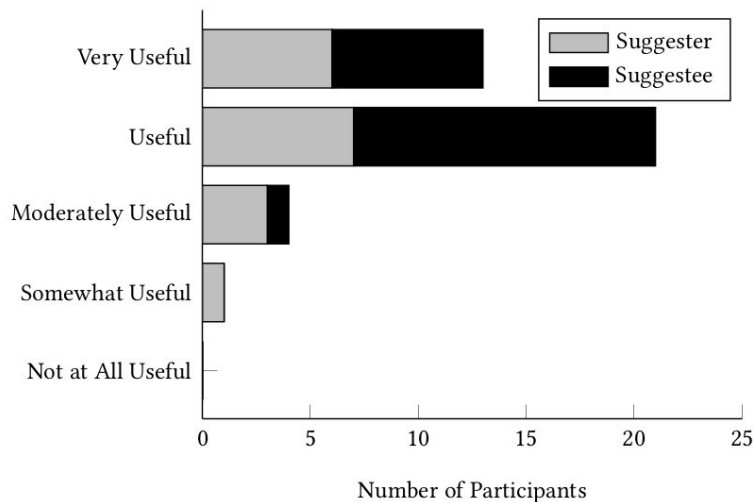
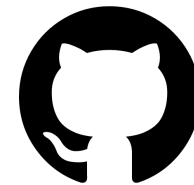
RQ4. How useful are suggested changes for recommendations between developers?

- Suggesters and Suggestees
- 39 responses
- Usefulness

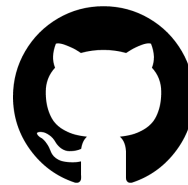


(IRR = 72%, Cohen's κ = 0.682)

Results: Phase 2 (RQ4)



Observations



User-Driven Communication

“It is very convenient that the reviewer can write what they suggest to change in code instead of formulating it in words (which will often be longer)” (R6)

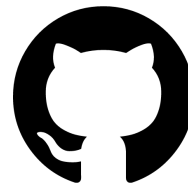
```
Suggested change ⓘ  
9 - int c;  
9 + int count;
```

Workflow Integration

“Small changes can be applied immediately, and the fact that they can be described by the reviewer in a way that a button fixes it instead of going to your code.” (C3)

Commit suggestion ▼

Contribution



3. *A set of experiments to provide evidence supporting the conceptual framework.*



- Systems incorporating *developer recommendation choice architectures* are:



- preferred by developers

Content

Design

- effective for improving development practices

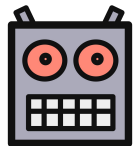
User-Driven Communication

Workflow Integration

Thesis: *Developing New Tools*

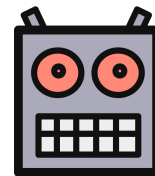


By incorporating ***developer recommendation choice architectures*** into recommendations for software engineers, we can ***nudge*** developers to adopt behaviors useful for improving code quality and developer productivity.



class-bot

Background: Student Behavior



- Poor programming behaviors lead to high attrition in CS [Beaubouef, 2005]
- Effective and Ineffective Behaviors of Students Impact Performance [Edwards, 2009]
- Translates to newly hired CS graduates in industry [Charrette, 2005]

Can developer recommendation choice architectures improve behavior on programming assignments?



class-bot commented on Jul 16, 2020 • edited by dcbrow10



Hi! This bot provides daily updates tracking your progress on the software process requirements for this assignment. If you have any questions or problems, feel free to leave a comment below or email the instructor.



ProjectSpecification.pdf: ✖

Updated README: ✖



Compilation and execution steps: ✖



Project structure: ✖



Added source code: ✖



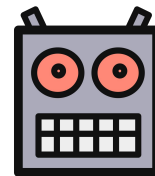
[class-bot] Comprehensive Exercise Software Process #1

Open

class-bot opened this issue on Jul 16, 2020 · 1 comment



Methodology



Study Design

- Introduction to Java (CSC116)
- 35 Students; 151 repositories
 - Projects 3-5 (Control)
 - Project 6 and CE (Treatment)
- Software Engineering Process

[Beaubouef, 2005] [Boehm, 1984]

Software Process Phases

1. Requirements 
2. Design 
3. Implementation 
4. Testing  
5. Deployment 

RQ1. How do class-bot nudges impact the quality of student projects?

- Project Grade
- Points Deducted



RQ2. How do class-bot nudges influence student productivity?

- Commits
- Code Churn
- Commit Timing

Initial commit



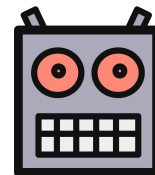
dcbrow10 committed on Jul 16, 2020

f37eec3 on Jul 23, 2020



39 commits

Results



RQ1. Quality

	class-bot	Mean	Median
Grade*	Without	74.29	87.66
	With	76.89	95
Points Deducted	Without	-20.71	-5
	With	-9.43	0

* (Wilcoxon, $p < 0.0097$, $\alpha = .05$)

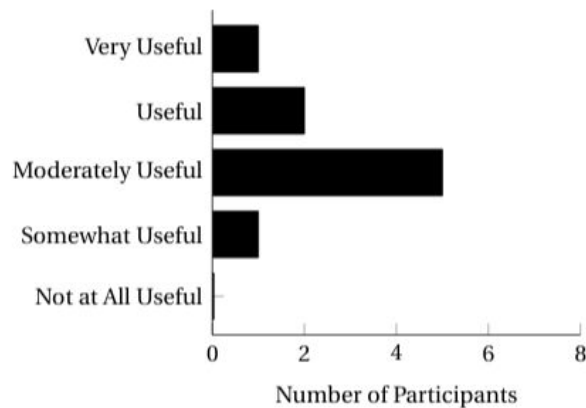
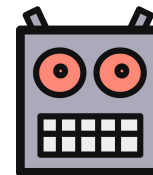
RQ2. Productivity

	class-bot	Mean	Median
Commits	Without	9.84	7
	With	12.64	9
Code Churn*	Without	205.03	4
	With	1101.57	11
First Commit (days)**	Without	8.32	7.41
	With	1.99	5.94
Last Commit (hours)	Without	-21.72	-1.60
	With	-9.67	-2.47

* (Wilcoxon, $p = 0.0348$, $\alpha = .05$)

** (Wilcoxon, $p < 0.0001$, $\alpha = .05$)

Feedback

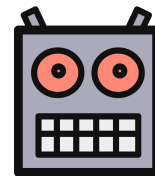


"The class bot didn't update frequently enough" (P4)



"I checked it once at the end to make sure everything was correct but thats it" (P7)

Contribution



4. An *automated recommender system* that incorporates the framework to nudge programmers toward developer behaviors.

- Automated recommendations from `class-bot` were useful for encouraging students to adhere to software engineering processes.
 - Higher grades
 - Increased code churn
 - Prevented procrastination



Revisiting Research Contributions

1. *A set of experiments* to explore recommendations to developers and motivate the need for a new approach. ☐



2. *A conceptual framework* to design effective developer recommendations. ☐



3. *A set of experiments* to provide evidence supporting the conceptual framework. ☐



4. *An automated recommender system* that incorporates the framework to nudge programmers toward developer behaviors. ☐

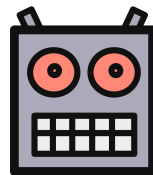
Conclusion and Future Work

Future Work

Future directions of this research can continue exploring ways to improve the productivity, behavior, and decision-making of programmers.

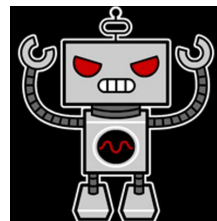


Analyze behavior



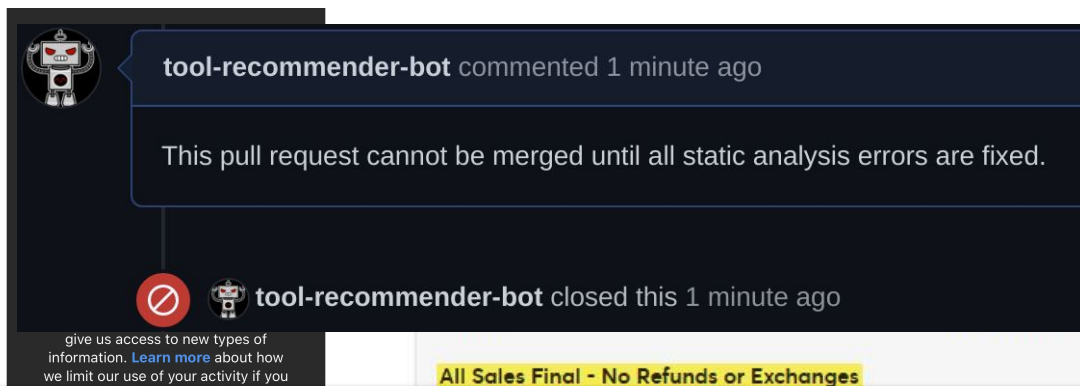
Develop new tools

Interdisciplinary Methods



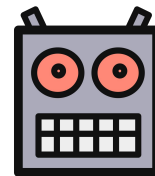
Dark Patterns: deceptive user interface design created to influence the decision-making and behavior of users online.

[Gray, 2018]



Can dark patterns be used to observe programmer behavior and improve the decision-making of developers?

Nudge-Bots



Develop systems that utilize various interventions in different online programming communities to nudge developers toward adopting better behaviors.

1. Cannot provide incentives
2. Must allow alternative behaviors

Behaviors



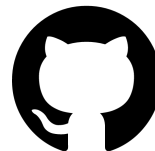
1. Requirements
2. Design
3. Implementation
4. Testing
5. Deployment



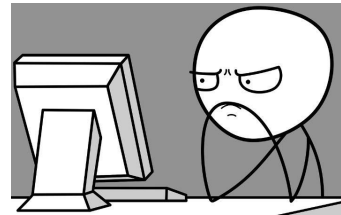
Interventions



Communities



Summary



Decision-making is a vital part of software engineering.

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.



Effective Developer Recommendations



Developer Recommendation Choice Architectures

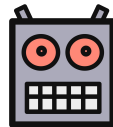


Analyzing Existing Systems

Developing New Tools



Analyze behavior



Develop new tools

Publication List

1. **Chris Brown**, Emerson Murphy-Hill, Justin Middleton, and Esha Sharma. "How Software Users Recommend Tools to Each Other". In the *Visual Languages and Human Centric Computing* (VL/HCC 2017).
2. Justin Smith, **Chris Brown**, and Emerson Murphy-Hill. "Flower: Navigating Program Flow in the IDE". In the *Visual Languages and Human Centric Computing* (VL/HCC 2017).
3. **Chris Brown** and Chris Parnin. "Sorry to Bother You: Designing Bots for Effective Recommendations". In the *International Workshop on Bots in Software Engineering* in conjunction with ICSE (BotSE 2019).
4. **Chris Brown**. "Digital nudges for encouraging developer actions". In the Proceedings of the *International Conference on Software Engineering* (ICSE 2019) doctoral symposium.
5. Peng Sun, **Chris Brown**, Ivan Beschastnikh, and Kathryn T. Stolee. "Mining specifications from documentation using a crowd". In the *International Conference on Software Analysis, Evolution and Reengineering* (SANER 2019).
6. **Chris Brown** and Chris Parnin. "Sorry to Bother You Again: Developer Recommendation Choice Architectures for Designing Effective Bots". In the *International Workshop on Bots in Software Engineering* in conjunction with ICSE (BotSE 2020).



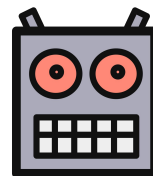
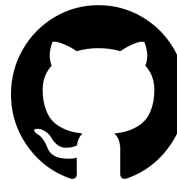
Publication List (cont.)

7. **Chris Brown** and Chris Parnin. "Comparing Different Developer Behavior Recommendation Styles". In the *International Workshop on Cooperative and Human Aspects of Software Engineering* in conjunction with ICSE (CHASE 2020). 
8. Peipei Wang, **Chris Brown**, Jamie A. Jennings, and Kathryn T. Stolee. "An Empirical Study on Regular Expression Bugs". In the *International Conference on Mining Software Repositories* (MSR 2020).
9. **Chris Brown** and Chris Parnin. "Understanding the Impact of GitHub Suggested Changes on Recommendations Between Developers". In the *Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (ESEC/FSE 2020). 
10. **Chris Brown** and Chris Parnin. "Nudging Students Toward Better Software Engineering Behaviors". To appear in the *International Workshop on Bots in Software Engineering* in conjunction with ICSE (BotSE 2021). 
11. **Chris Brown** and Chris Parnin. "Dark Patterns for Influencing Developer Behavior". To appear as a *Position Paper* at the Workshop "What Can CHI Do About Dark Patterns?" at the CHI Conference on Human Factors in Computing Systems (Dark Patterns Workshop 2021). 

Acknowledgements

- Dr. Chris Parnin
- Thesis Advisory Committee
- alt-code
- EB2 3228 & 3229
- Friends and Family
- NSF #1714538

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.

Developer Recommendation Choice Architectures

 **Actionability**

 **Feedback**

 **Locality**

Chris Brown



 dcbrow10@ncsu.edu

 <https://chbrown13.github.io>

 <https://github.com/chbrown13>

 [@d_chrisbrown2](https://twitter.com/d_chrisbrown2)

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/>
- Cockburn, A. & Williams, L. "Extreme Programming Examined". Ed. by Succi, G. & Marchesi, M. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001. Chap. The Costs and Benefits of Pair Programming, pp. 223–243.
- Cohen, J. et al. Best kept secrets of peer code review. Smart Bear Somerville, 2006.
- 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)

Citations (cont.)

- Evans, D. and Larochelle, D., 2002. Improving security using extensible lightweight static analysis. *IEEE software*, 19(1), pp.42-51.
- 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)
- Gray, Colin M., Yubo Kou, Bryan Battles, Joseph Hoggatt, and Austin L. Toombs. "The dark (patterns) side of UX design." In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, pp. 1-14. 2018.
- 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)
- Herbsleb, J. D. "Global Software Engineering: The Future of Socio-technical Coordination". Future of Software Engineering (FOSE '07). IEEE, 2007, pp. 188–198.
- 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)
- Maalej, W. et al. "On the Comprehension of Program Comprehension". *ACM Trans. Softw. Eng. Methodol.* 23.4 (2014).
- 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/>

Citations (cont.)

- 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/>
- Turkle, S. "Alone together: Why we expect more from technology and less from each other". Hachette UK, 2017
- Weinmann, M., Schneider, C., vom Brocke, J.: Digital nudging. Business & Information Systems Engineering 58(6), 433–436 (2016)
- Whitley, B. E., and Kite, M. E.. "Principles of research in behavioral science." *Psychology Press* (2013).
- 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

Developer Recommendation Choice Architectures

DRCA	Tool [16]	Definition
Actionability	Technology and decision aids	Introducing technology to aid decision makers in choice tasks
	Use defaults	The way decision makers initially encounter choice tasks
Feedback	Reduce number of alternatives	Limiting the number of choice options presented to decision makers
	Focus on satisficing	Helping users consider outcomes that lead to higher choice satisfaction
	Attribute parsimony and labeling	Limiting the number of characteristics presented with options
	Translate and rescale for better evaluability	Presenting attributes to increase impact and clarity
	Customized information	Personalization to account for individual differences between decision-makers
	Focus on experience	Considering the background and knowledge of decision-makers
Locality	Limited time windows	Providing time restrictions for users to make decisions
	Partitioning of options	Groups or categories of options or attributes
	Decision staging	Dividing decisions into multiple stages

Actionability

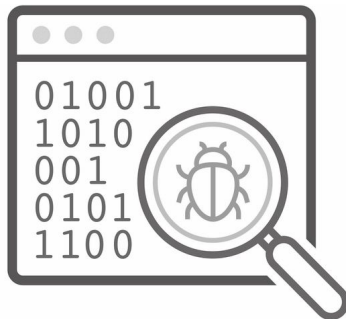


The ease with which users can act on recommendations

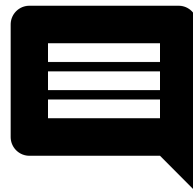
Default Rule
Automatic Enrollment
[Madrian, 2001]



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 symbol: variable varnam 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]



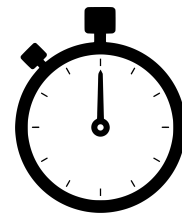
Flower

In situ navigation
[Smith, 2017]



```
SQLFileCache.java  ✖
createTables createProcedures executeSQLFile dropTables
26/** @param fileName String path to SQL file*/
27public List<String> getQueries(String fileName)
28    throws Exception {
29    List<String> queries = cache.get(fileName);
```

Locality: *Temporal*

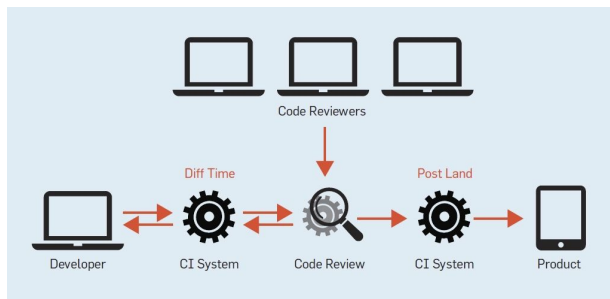


The setting of recommendations to improve user behavior

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



Scaling Static Analyses at Facebook
“diff time”
[Distefano, 2019]



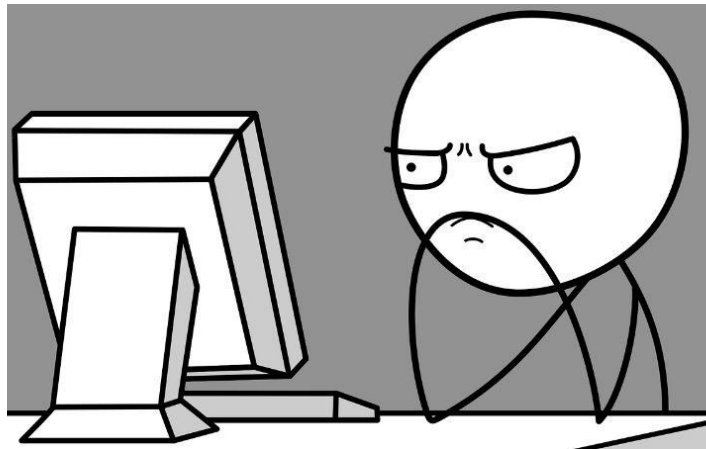
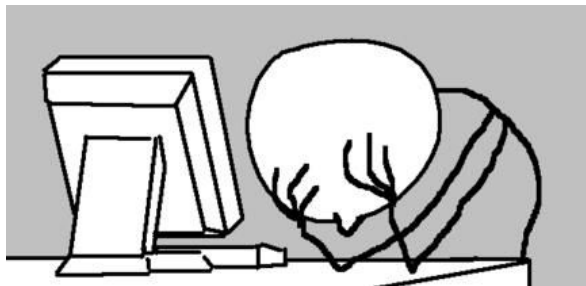
Future Work

Limitations



To improve developer decision-making, I plan to explore developing customized nudges and making individualized recommendations.

Limitations



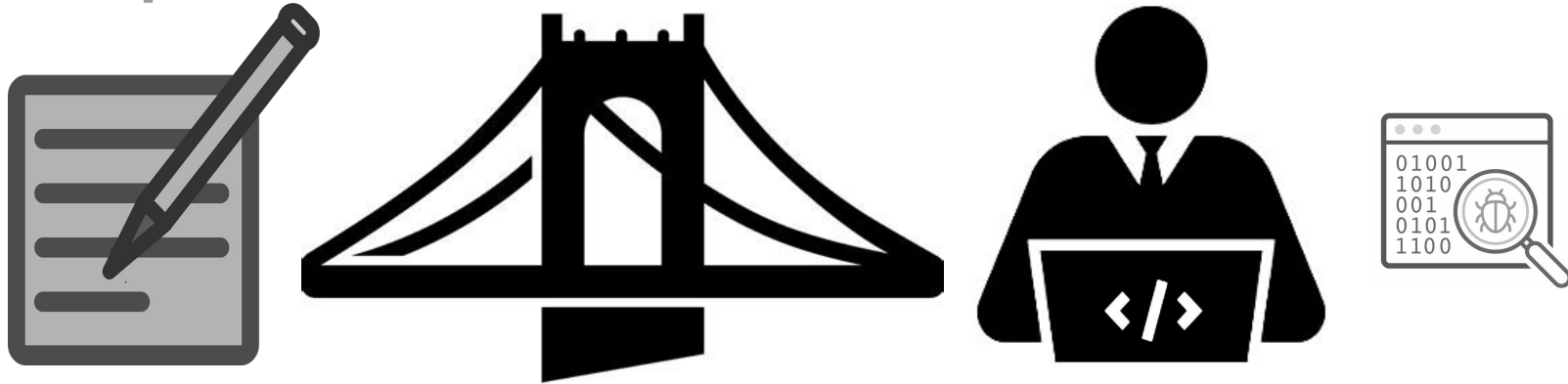
To improve developer decision-making, I plan to explore using machine learning techniques to predict poor developer behaviors and generate proactive recommendations for useful tools and practices.

Research-Practice Gap



Tools and practices developed by software engineering researchers often don't fit the needs of software practitioners

[Norman, 2010]



Increase the awareness of software engineering research tools, techniques, and findings in industry

Background: Developer Behavior Adoption Problem



Developer Inertia



[Murphy-Hill, 2015]

Managers



Companies



Researchers



[Norman, 2010]

Decision Fatigue



[Makabee, 2011]

Physical Isolation



[Turkle, 2011]

Global SE



[Herbsleb, 2007]

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.

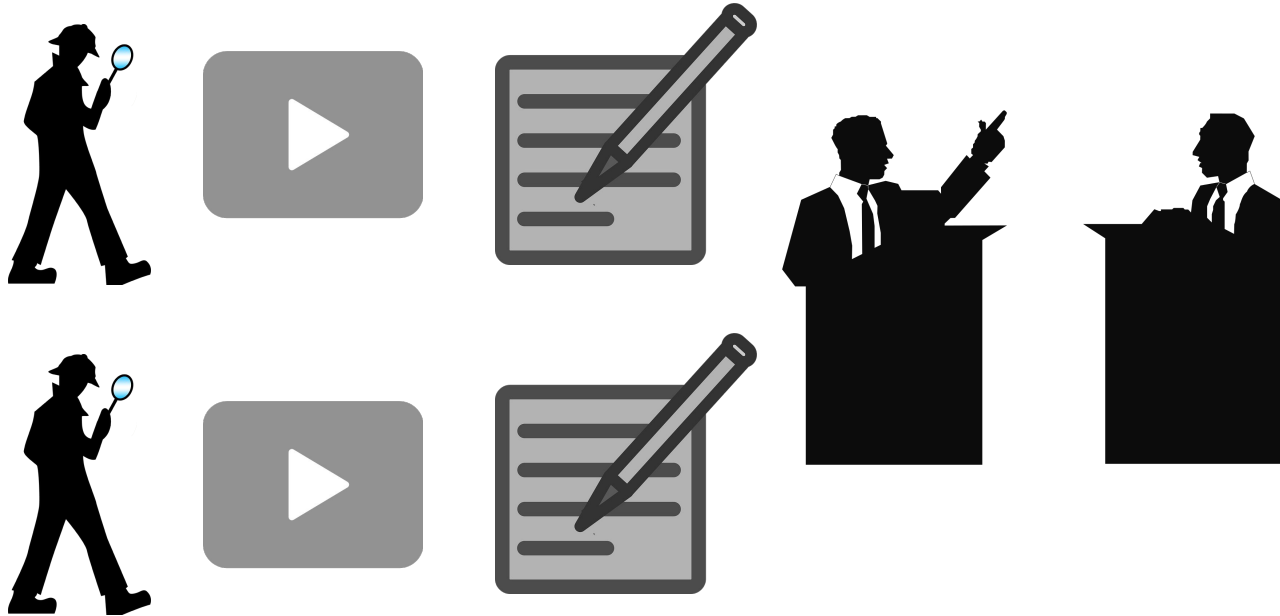
3. Dialogue

- *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

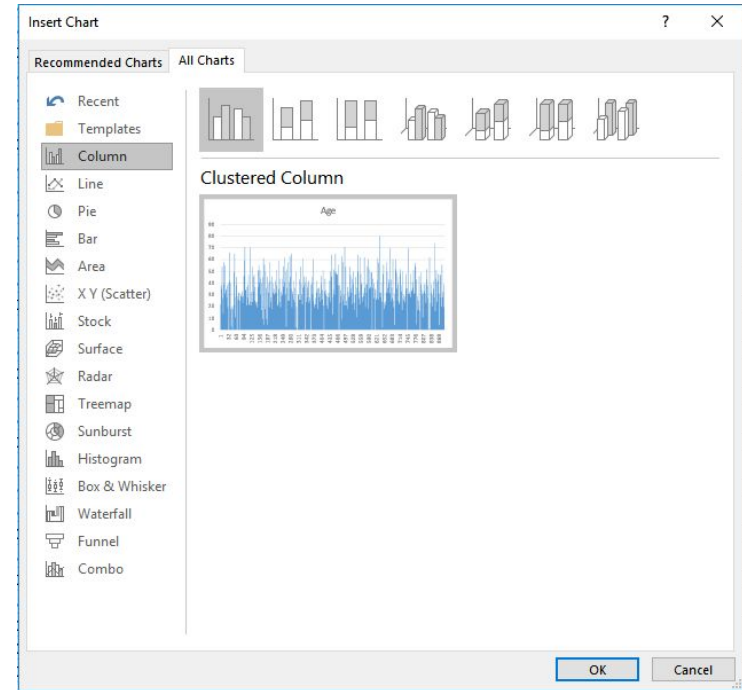
Criteria	Definition
Time Pressure	Driver or navigator makes a statement about time before, during, or after a recommendation

[Andrews, 1996]

Types of Tools

1. Observable

2. Non-Observable



[Murphy-Hill, 2015]

Methodology: Scoring

Types of Tools Politeness, Persuasiveness, Receptiveness

Observable: Proposed tool has user interface
Yes: Recommender always specifies intended tool

Non-Observable: Proposed tool does not have a user
No: Recommender only provides tool name

the study articulated specific criteria

1 Recommender mostly ignores or never uses
recommended tool

Results: *Interaction Characteristics*

	Polite	Neutral	Impolite
<i>n</i>	27	104	11

$(p = 0.4936)^W$

	Persuasive	Unpersuasive
<i>n</i>	14	128

$(p = 0.4556)^W$

	Receptive	Neutral	Unreceptive
<i>n</i>	64	56	22

$(p = 0.0002)^*^W$

Time Pressure?	Yes	No
<i>n</i>	19	123

$(p = 0.1470)^C$

Results: *Tool Observability*

	Observable	Non-Observable
<i>n</i>	115	27

$(p = 0.4928)^{\text{C}}$

**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

Suggested Changes

Recommendation Style

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 deserializeth

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 ⓘ

-	<code>`(function(){__BUILD_MANIFEST = JSON.parse('\${clientManifest</code>
+	<code>`(function(){self.__BUILD_MANIFEST = JSON.parse('\${clientManifest</code>

Types of Suggested Changes

Improvement: changes to refactor or optimize code.

(c) Improvement:

Suggested change ⓘ

```
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



To

Cc Bcc

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!



Send



User Study Issue

Add static analysis tool to check for errors #2

[Edit](#)[New issue](#)

Open 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

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

 Conversation **0**

 Commits **1**

 Checks **0**

 Files changed **1**



tool-recommender-bot commented on Jul 15

First-time contributor



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 ⓘ

146 - `if applied != None:`

146 + `if applied is not None:`

Commit suggestion ▼

Add suggestion to batch

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.