User-Centric Deployment of Automated Program Repair at Bloomberg

David Williams[1]
James Callan[1]
Serkan Kirbas[2]
Sergey Mechtaev[1]
Justyna Petke[1]
Thomas Prideaux-Ghee[2]
Federica Sarro[1]

[1]University College London
[2]Bloomberg

International Conference on Software Engineering (ICSE) SEIP 2024
April 17, 2024 - Lisbon, Portugal

Challenges in Deploying Automated Program Repair (APR)

Previous deployment studies:

Tool	Company	Patch Acceptance Rate
Getafix [1]	Facebook	42%
Fixie [2]	Bloomberg	48%

Previous Bloomberg deployment experience:

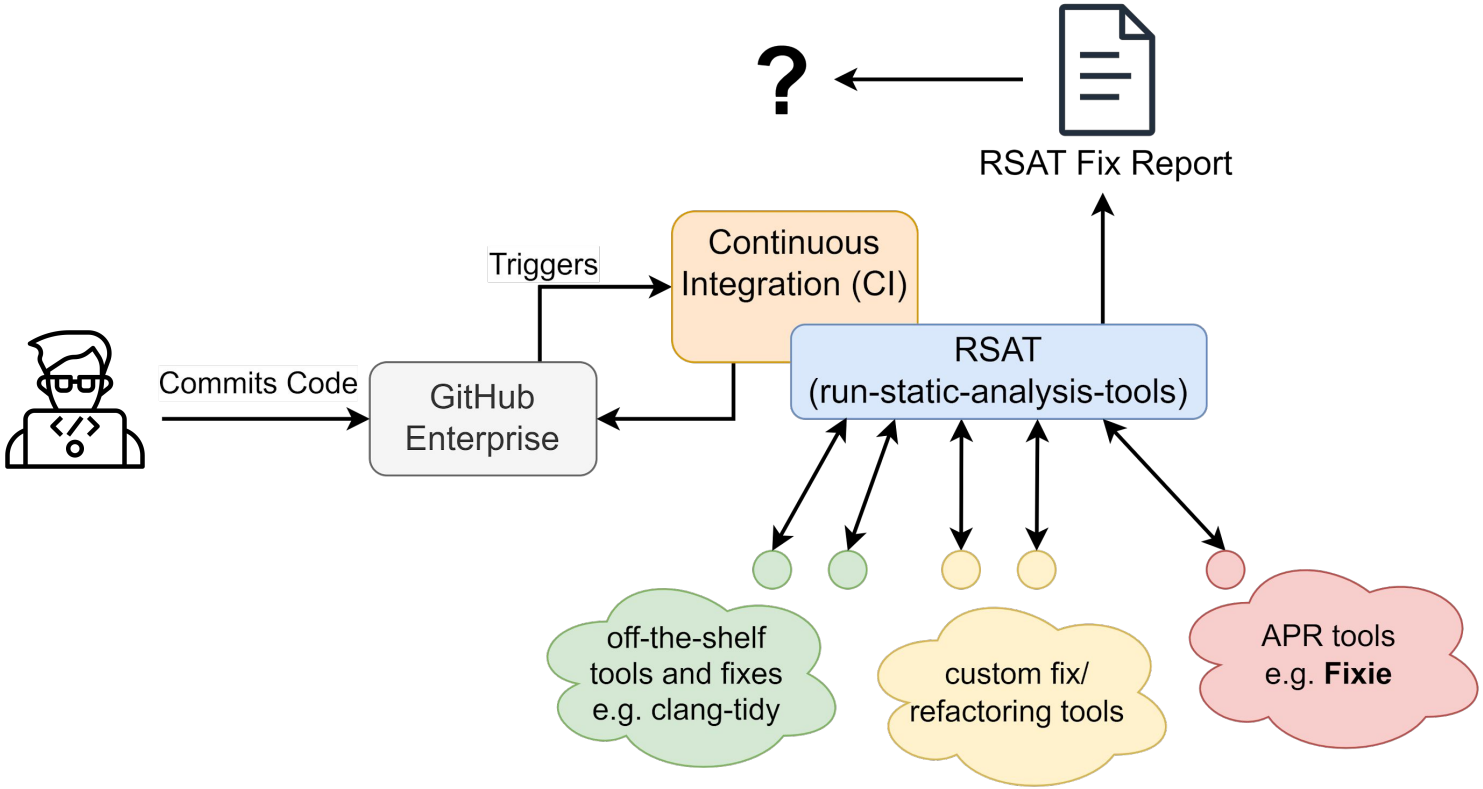
Feature Flag Removal Tool	Bloomberg	5-8%

● ● ●

How do you convince engineers to adopt your APR tool into their workflows?

[1] Bader et al.. 2019. Getafix: learning to fix bugs automatically. OOPSLA. https://doi.org/10.1145/3360585
[2] Rowan Winter et al. 2022. Towards developer-centered automatic program repair:findings from Bloomberg. ESEC/FSE. https://doi.org/10.1145/3540250.3558953

Bloomberg's Previous Workflow



Factors Influencing Patch Acceptance

Timing

Patches should be presented while engineers are working on new/existing functionality.

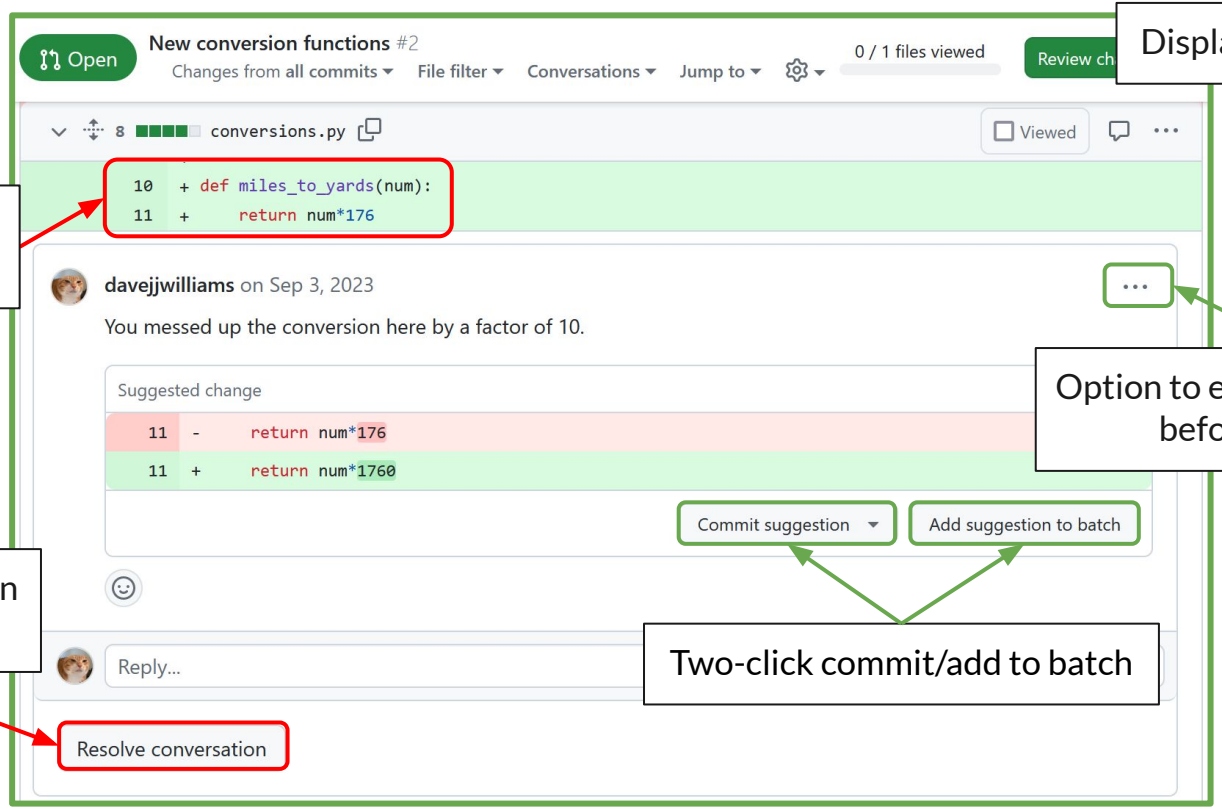
Audience

Patches should be presented to an audience familiar enough with the codebase to review them.

Context

Suggested patches should only target the code engineers are currently working on.

GitHub Suggested Changes



Displayed in the PR UI

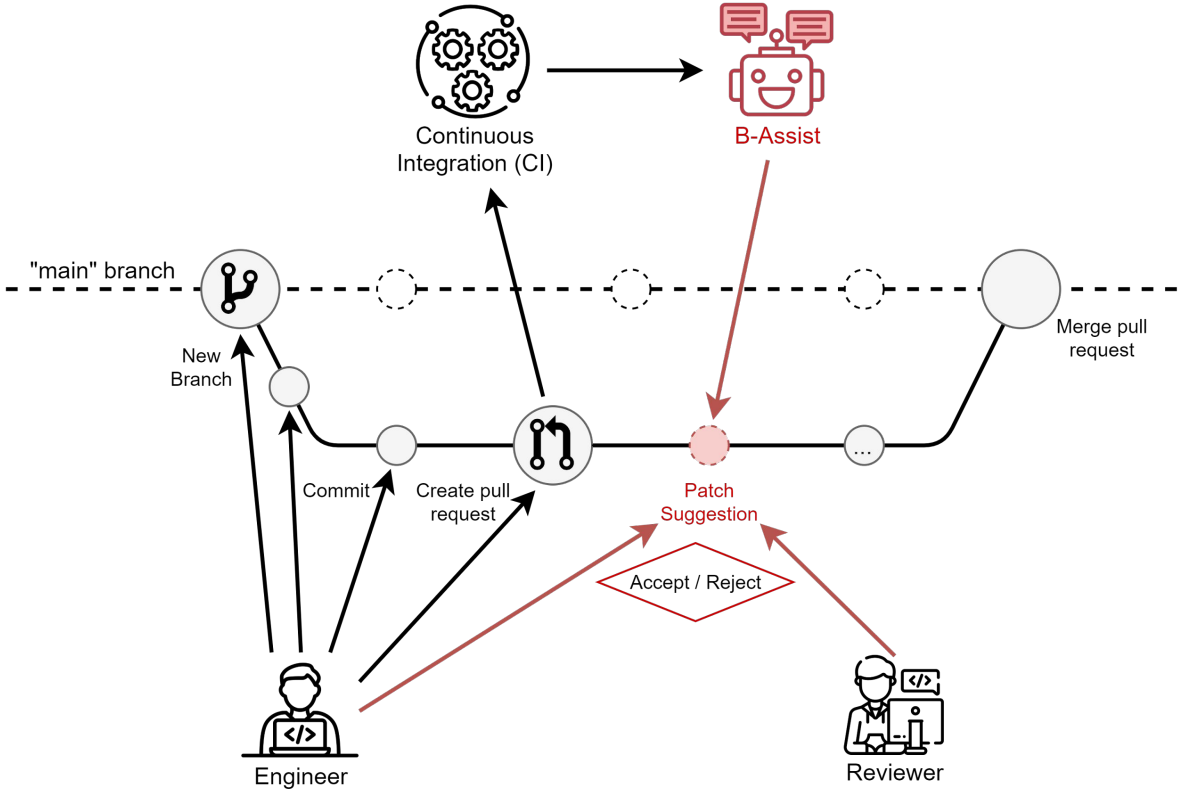
Can only be made on PR-modified code

Option to edit suggested change before committing

One-click suggestion rejection

Two-click commit/add to batch

B-Assist - Concept



Patch Acceptance Factors with B-Assist

Timing

Patches should be presented while engineers are working on new/existing functionality.

★ Suggestions are posted **when developers expect feedback** → ideal for **code review**

Audience

Patches should be presented to an audience familiar enough with the codebase to review them.

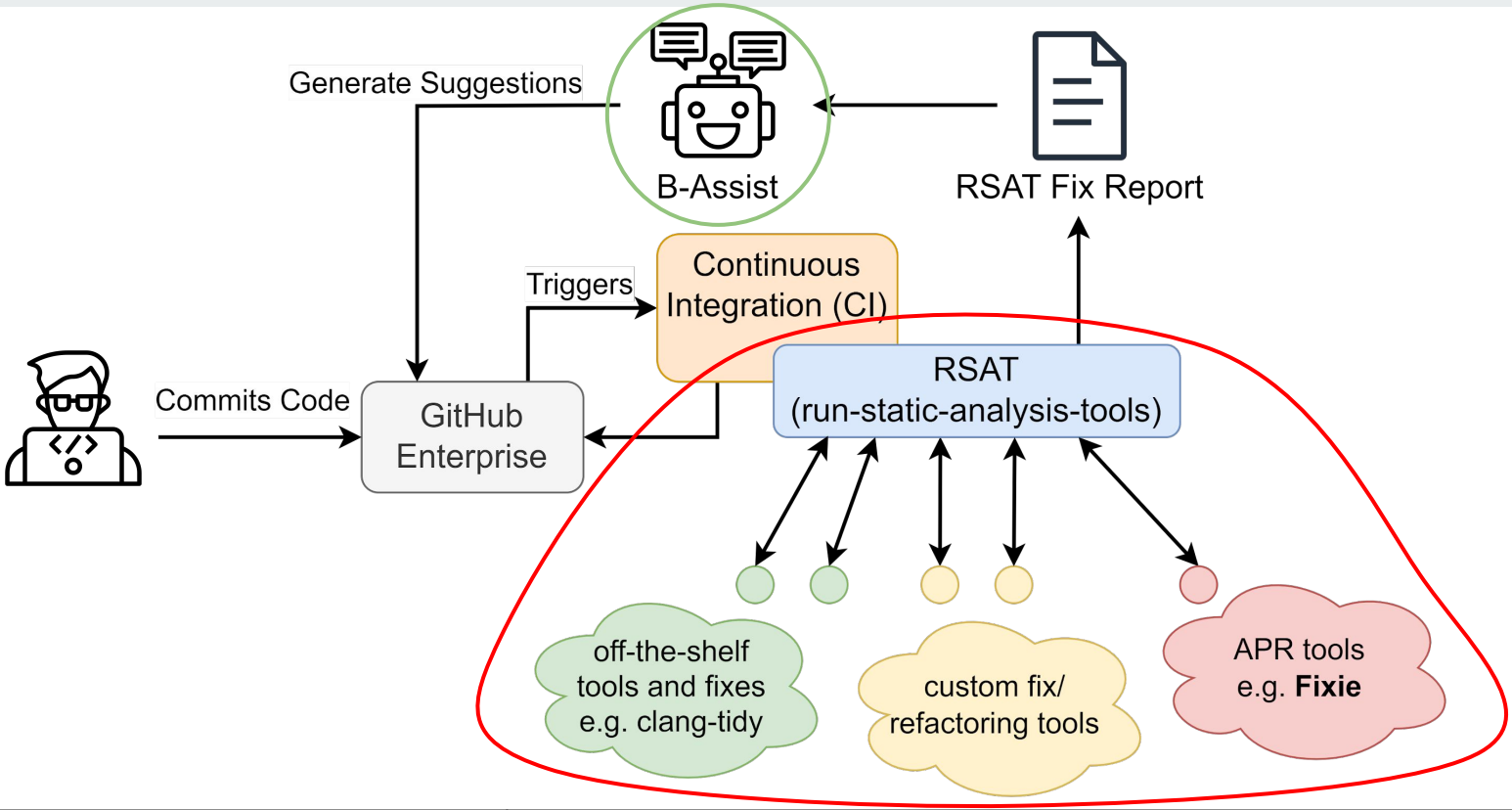
★ Suggestions are visible in the **pull request user interface** → assigned to **typical reviewers**

Context

Suggested patches should only target the code engineers are currently working on.

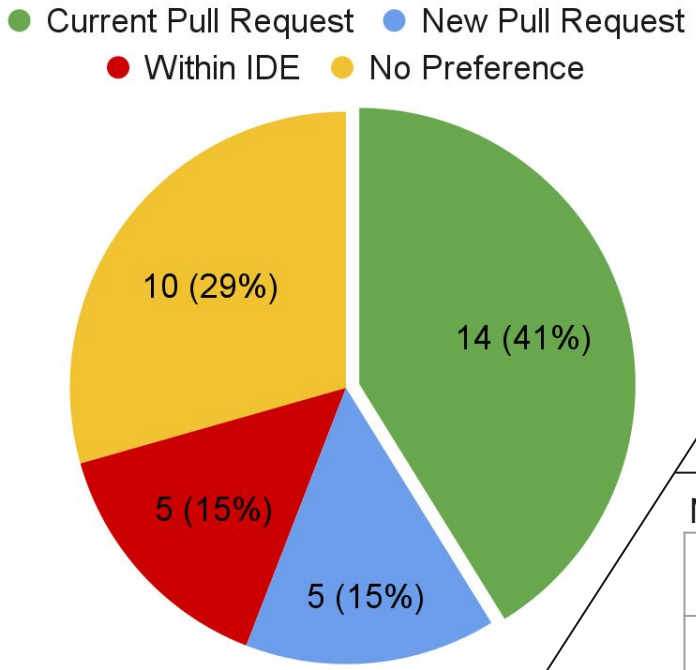
★ Suggestions are only made for **pull-request-modified code** → feedback is **relevant**

B-Assist at Bloomberg

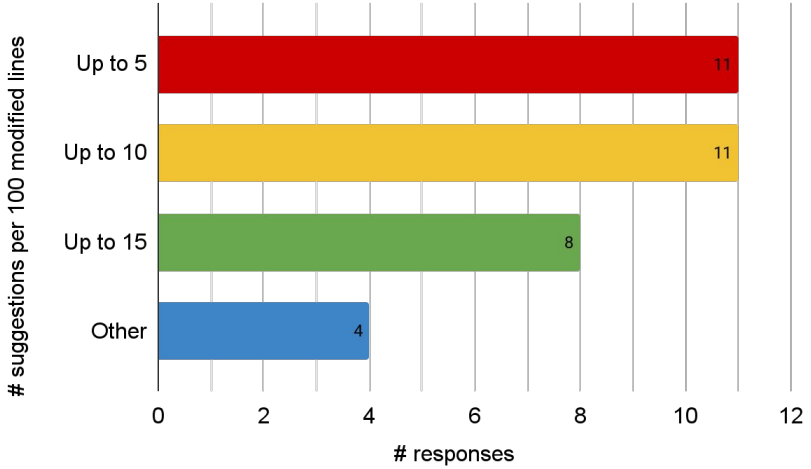


Concept Validation Survey - 34 Engineers

Where would engineers like to fix code issues during code review?



How many suggestions do engineers want to review?

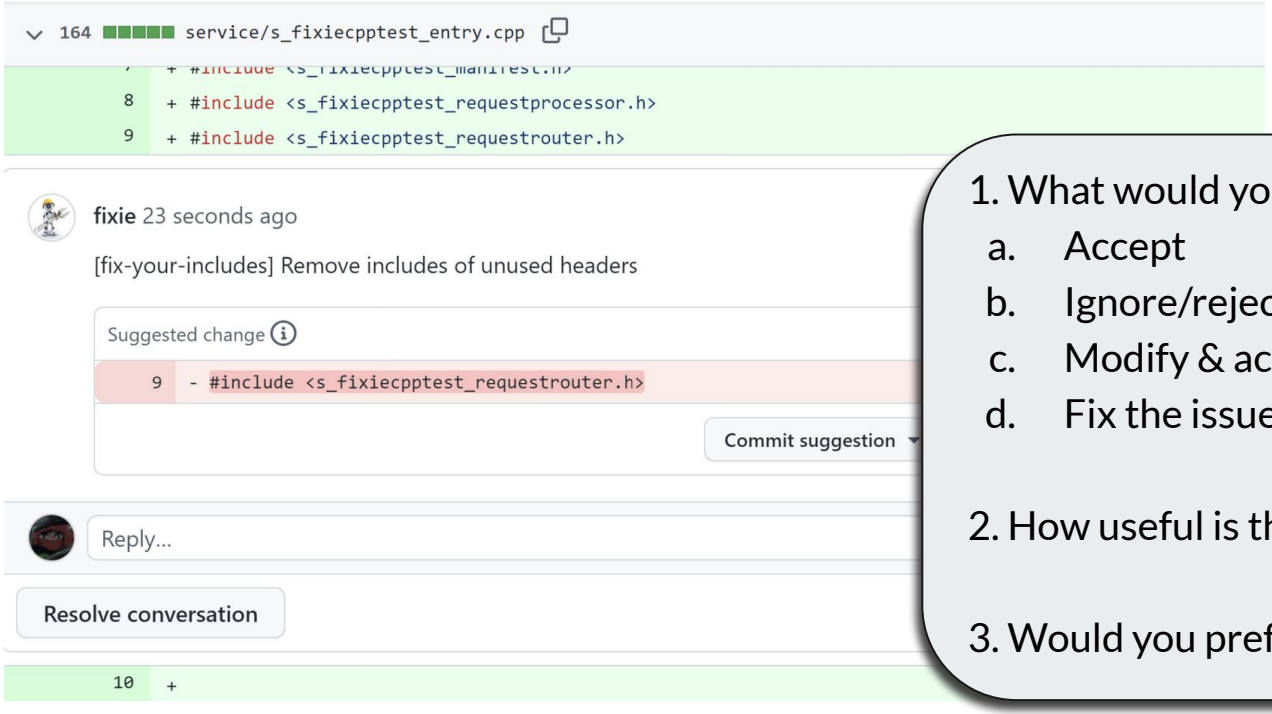


Most desirable features:

“Seamless Integration with GitHub Pull Request UI”	18 (52.9%)
“Choice to accept or reject suggestions”	13 (38.2%)

Engineer Interviews

Goal: Understand B-Assist's use cases more concretely and gauge potential developer behaviour.



1. What would you do?
   a. Accept
   b. Ignore/reject
   c. Modify & accept in GitHub user interface
   d. Fix the issue in my own way

2. How useful is this suggestion? (1-5)

3. Would you prefer a patch or a text comment?

Usage Insights - 11 Engineers

“How would you respond to these 10 example suggestions created by B-Assist?”

- **Acceptance rate:** immediately commit suggested patch or modify & commit within GitHub UI.
- **Usefulness:** average Rating of 5-point Likert scale (“Very Useful” to “Not Useful”)
- **Patch/Warning:** “Do you prefer having a suggested patch or a comment warning for this issue?”

Issue Types (# of Examples)	Acceptance Rate (%)	“Fix it my own way” (%)	Rejection Rate (%)	Usefulness (/5)	Patch/ Warning
Correctness/Best Practices (6)	57.6%	33.3%	9.1%	4.03	81.7% / 18.3%
Code Formatting (2)	100%	0%	0%	4.32	100% / 0%
Dependency Management (2)	100%	0%	0%	4.86	100% / 0%
Total (10)	**74.56%**	20%	**5.44%**	**4.26**	89% / 11%

Key Takeaways for Future APR Tools

- **Timing**, **audience** and **context** are all crucial factors to consider when presenting APR tool output.
- The GitHub pull request user interface is a natural location to present patches for code review, and **Suggested Changes** is a powerful feature for this.
- APR tool suggestions need to be consistently useful, understandable, and easy to apply.
- When presented effectively, patch suggestions can almost always be useful to engineers, regardless of whether they are immediately accepted or not.
- …Many more specific points in the full text!

ACM

ArXiv

Contact me here!
david.williams.22@ucl.ac.uk
https://davejjwilliams.github.io

References

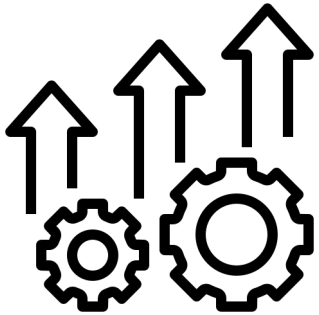
[1] Johannes Bader, Andrew Scott, Michael Pradel, and Satish Chandra. 2019. Getafix: learning to fix bugs automatically. Proc. ACM Program. Lang. 3, OOPSLA, Article 159 (October 2019), 27 pages. https://doi.org/10.1145/3360585

[2] Emily Rowan Winter, Vesna Nowack, David Bowes, Steve Counsell, Tracy Hall, Sæmundur Haraldsson, John Woodward, Serkan Kirbas, Etienne Windels, Olayori McBello, Abdurahman Atakishiyev, Kevin Kells, and Matthew Pagano. 2022. Towards developer-centered automatic program repair: findings from Bloomberg. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2022). Association for Computing Machinery, New York, NY, USA, 1578–1588. https://doi.org/10.1145/3540250.3558953

Prototype Demonstrations - 25 Engineers

What types of issues do developers want suggestions for?

1. Dependency management (missing/unused imports)
2. Code correctness (e.g. null dereferences)
3. Code formatting*



We've been needing something like this for a while. Nice Work!

Looks really cool!

Let's figure out a plan so it can get used around Bloomberg.

Any additional features?

TODO

- Patch explanations
- Source of patch suggestion
- Patch severity/urgency indicator
- Additional user configurability

Future Work

1. In-Field Evaluation
   - Collecting usage data of B-Assist as it's used in practice

2. Patch Explainability
   - Currently, there is minimal explanation provided with patches
   - Integrating patch explanation generation techniques (e.g. LLM-based)

3. Wider Applications for B-Assist
   - B-Assist is designed to be language-agnostic
   - Integrating additional languages and tools with B-Assist at Bloomberg