

# GenAI in development processes and DevTools

Bohdan Snisar



## Bohdan Snisar

- 14 years of software engineering:
  - **Different Roles:** software lead, engineer, CTO
  - **Notable Companies:** ex-Revlut, ex-Wix, etc
  - **Current:**
    - Partner Overvue: consulting for LLM/ML research and developments
    - Co-founder and CTO for redress.space: fashion search & discovery

ABOUT

LINKS

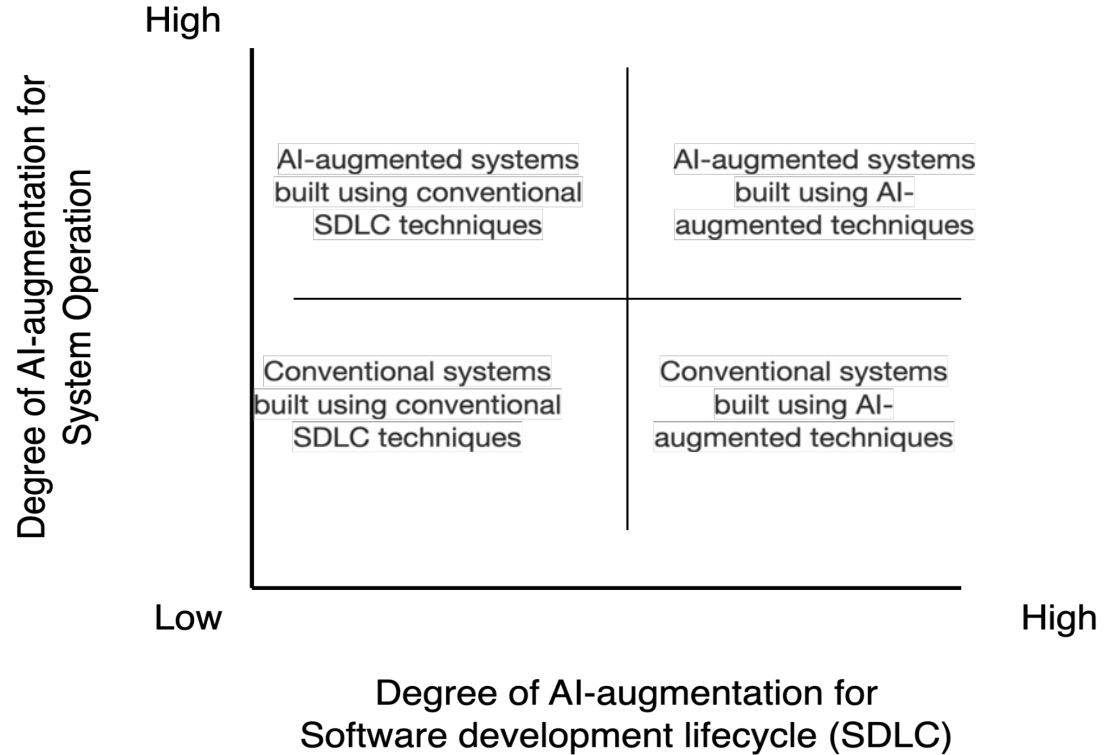


[www.overvu.solutions](http://www.overvu.solutions)

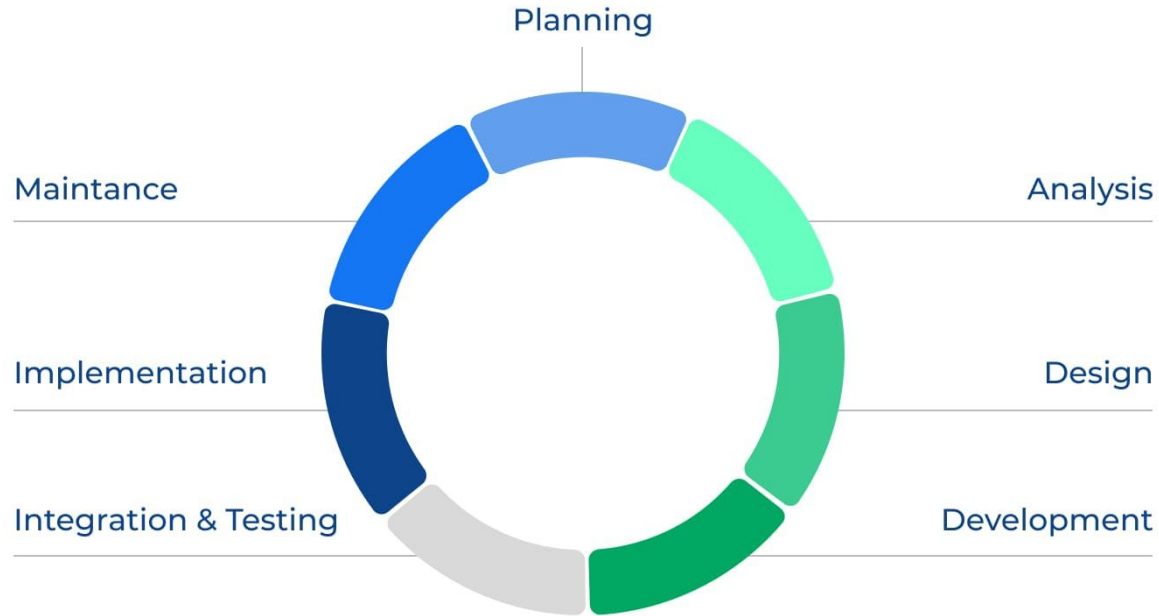
[LinkedIn](#)

X: @BSnisar

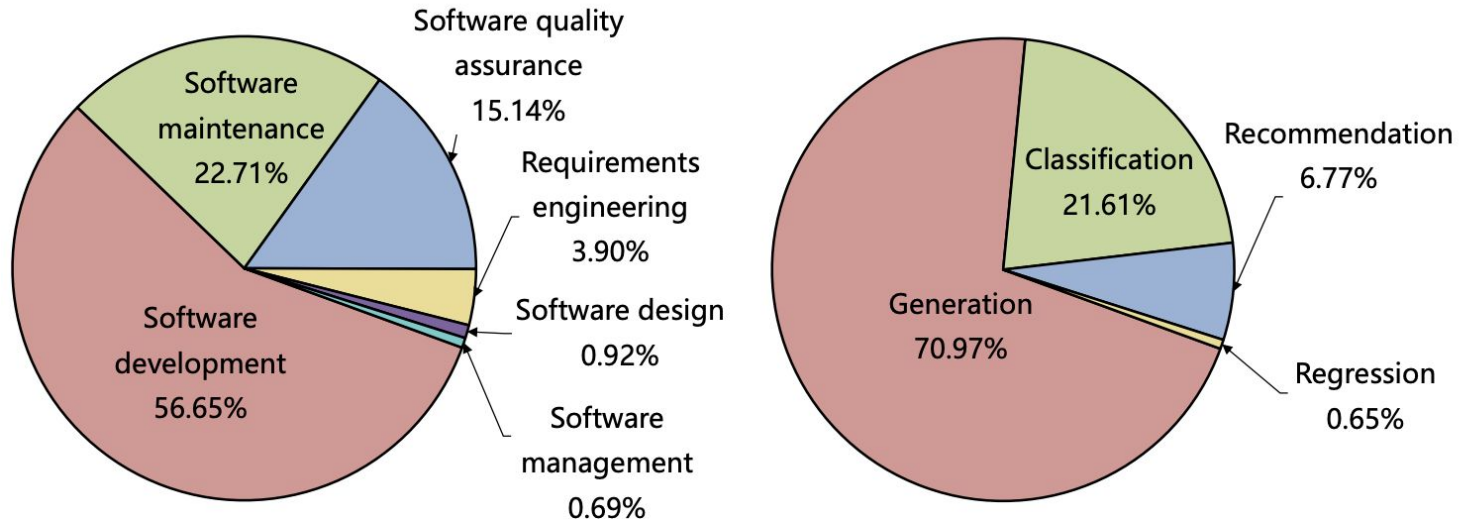
# Taxonomy of AI Augmentation for System Operations and the Software Development Lifecycle



# SDLC

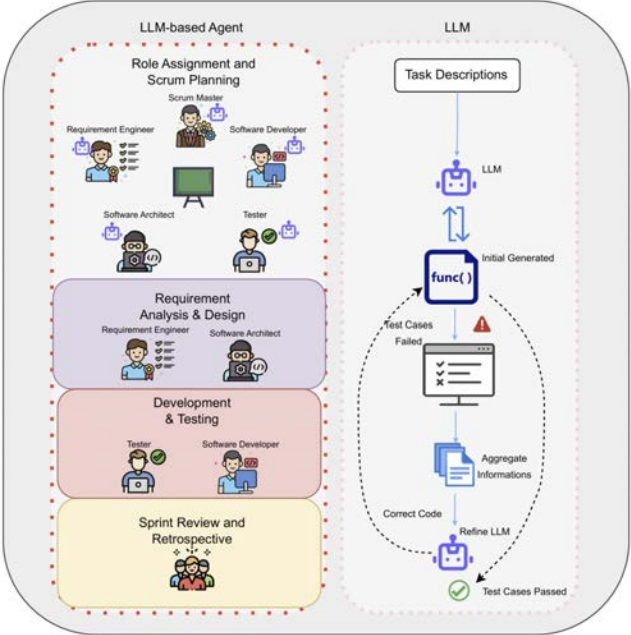


# Distribution of LLM utilization across different SE activities and problem types by white papers



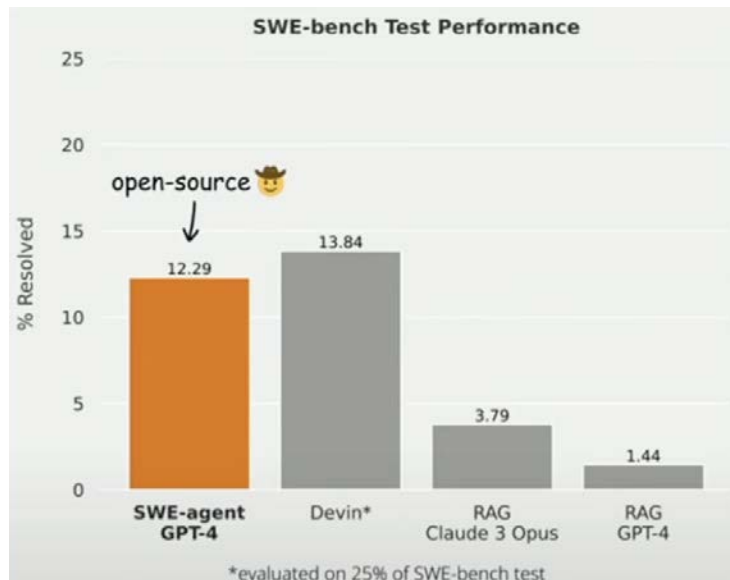
(a) Distribution of LLM usages in SE activities. (b) Problem classification based on collected studies.

# Dreams of someone about development with agents



But not right now:

- SWE-agent
- Devin
- etc..



<https://github.com/princeton-nlp/SWE-agent>

# Requirements



**@SpontaneouslyDeliberate** 10 days ago

If my job was coding solutions to problems with rigorously-defined requirements, this would be concerning.



6.9K



Reply

Ideas: <https://www.youtube.com/watch?v=xlzkmHpVrXw>



@BSnisar



[overvu.solutions](https://www.overvu.solutions)



# Requirements



@SpontaneouslyDeliberate 10 days ago

If my job was coding solutions to problems with rigorously-defined requirements, this would be concerning.



6.9K



Reply



# Requirements

- Unstructured requirements into text specs or user stories
- LLMs can help requirement engineers to highlight the unknowns in requirements documentations.
- Unknowns could be any ambiguity or uncertainty in requirements.
- Can help with the completion or suggest alternative ideas regarding the identified unknowns.
- Classification of requirements into Functional (FR) or Non-Functional Requirement (NFR).
- Identification of FRs and NFRs from user reviews on platform like play store or app store.

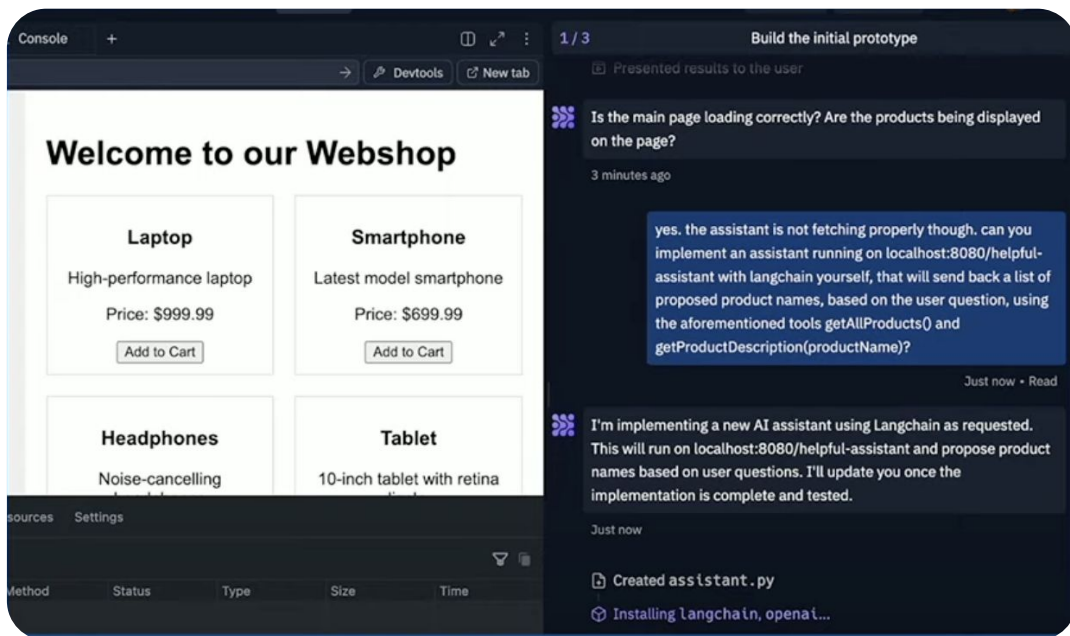
## How are LLMs used in software development?

- Code completion
- Code summarization
- Code search
- Code understanding
- Program synthesis
- API recommendation
- Method name generation
- Code recommendation
- ....

**“ There are only two hard things in  
Computer Science: cache invalidation  
and naming things”**

*Phil Karlton*

# Replit



The screenshot shows a Replit IDE interface. On the left, a web browser displays a webpage titled "Welcome to our Webshop" with four product cards: Laptop, Smartphone, Headphones, and Tablet. Each card includes a price and an "Add to Cart" button. On the right, a chat window titled "Build the initial prototype" shows a conversation with an AI assistant. The assistant asks, "Is the main page loading correctly? Are the products being displayed on the page?" and provides a response: "yes. the assistant is not fetching properly though. can you implement an assistant running on localhost:8080/helpful-assistant with langchain yourself, that will send back a list of proposed product names, based on the user question, using the aforementioned tools getAllProducts() and getProductDescription(productName)?". Below the chat, a terminal window shows the command "Created assistant.py" and "Installing langchain, openai...".

- only **python** and **js**
- nice for standard use-cases
- time saving
- crappy for specific cases

# Cursor

```
1802   onStream=async (stream) => {
1803     return await props.onReturn(
1804       (async function* () {
1805         let buffer = "";
1806         let state: "outside" | "newline" = "outside";
1807         let state: "outside" | "newline" | "inside" = "outside";
1808         for await (const datum of stream) {
1809           const newToken = datum.content;
1810           for (const char of newToken) {
1811             if (state === "inside") {
1812               buffer += char;
1813               if (char === "" && state !== "newline") {
1814                 if (state === "outside") {
1815                   state = "newline";

```

- code assistance
- completion
- naming

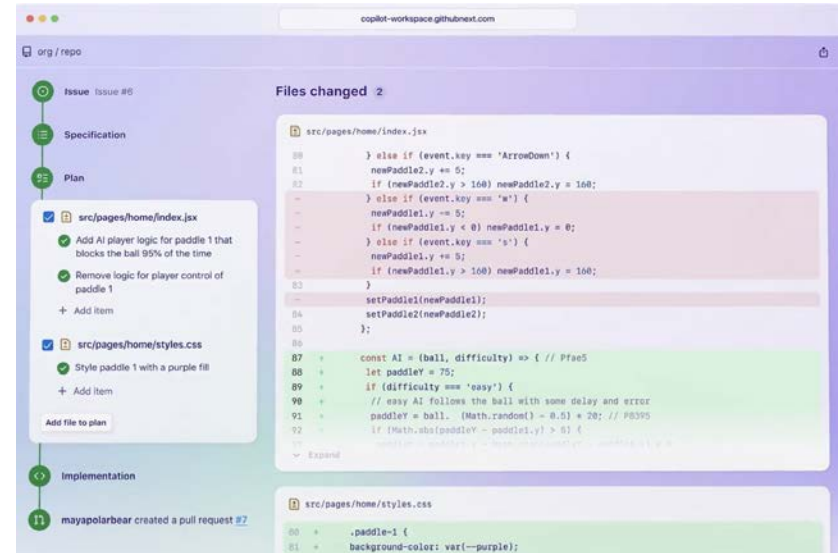
## How are LLMs used in software maintenance?

- Program repair
- Code clone detection
- Code review
- Debugging
- Bug reproduction
- Duplicate bug report detection
- Logging
- Sentiment analysis
- Bug triage
- ....

Break the information silos by allowing intelligent join search in:

- tech specs
- jira
- code
- test cases
- bug reports
- documentation
- .....

- atlassian rovo
- naboo.ai
- github copilot workspace



# Pull Requests



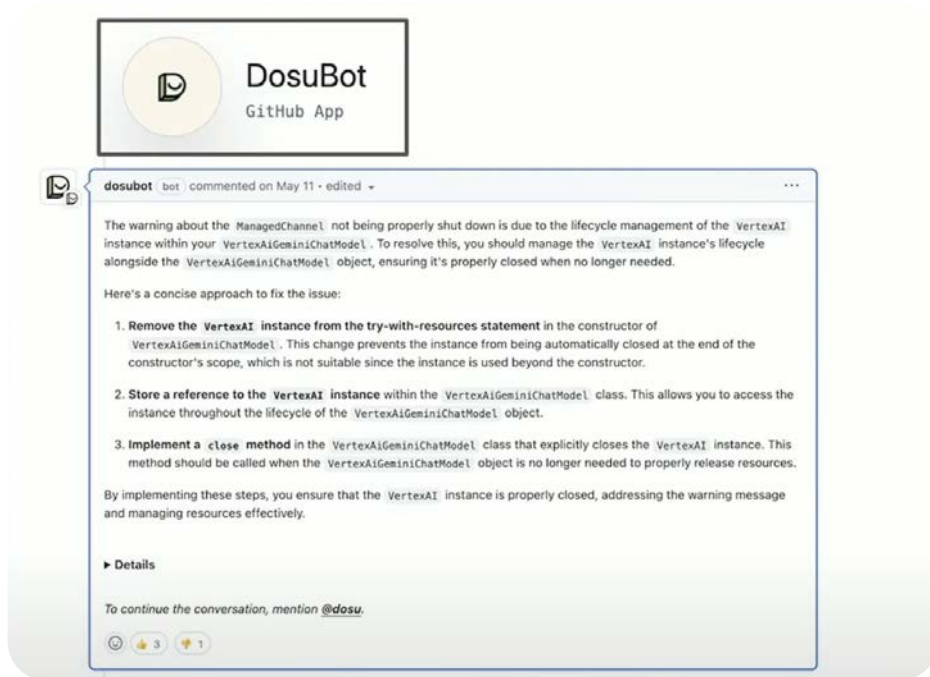
```
@@ -37,6 +37,8 @@ public OllamaClient(String baseUrl, Duration timeout) {
37 37     .connectTimeout(timeout)
38 38     .readTimeout(timeout)
39 39     .writeTimeout(timeout)
40 +     .addInterceptor(new OllamaRequestLoggingInterceptor())
HashJang marked this conversation as resolved. Show resolved
41 +     .addInterceptor(new OllamaResponseLoggingInterceptor())
HashJang marked this conversation as resolved. Hide resolved
coderabbitai (bot) on Mar 8
The addition of logging interceptors directly in the constructor does not allow for configurable logging. Consider implementing a builder pattern to make logging configurable, as suggested in previous review comments. This approach would provide more flexibility to enable or disable logging as needed.
```

- Analyze code
- Sometime it can find what I missed
- Sometime I find what AI missed

Good combo



# Issues



- Issues resolution
- Sometimes it make a good job in getting answers

## Documentation

- method names
- code comments
- readme
- ...etc.

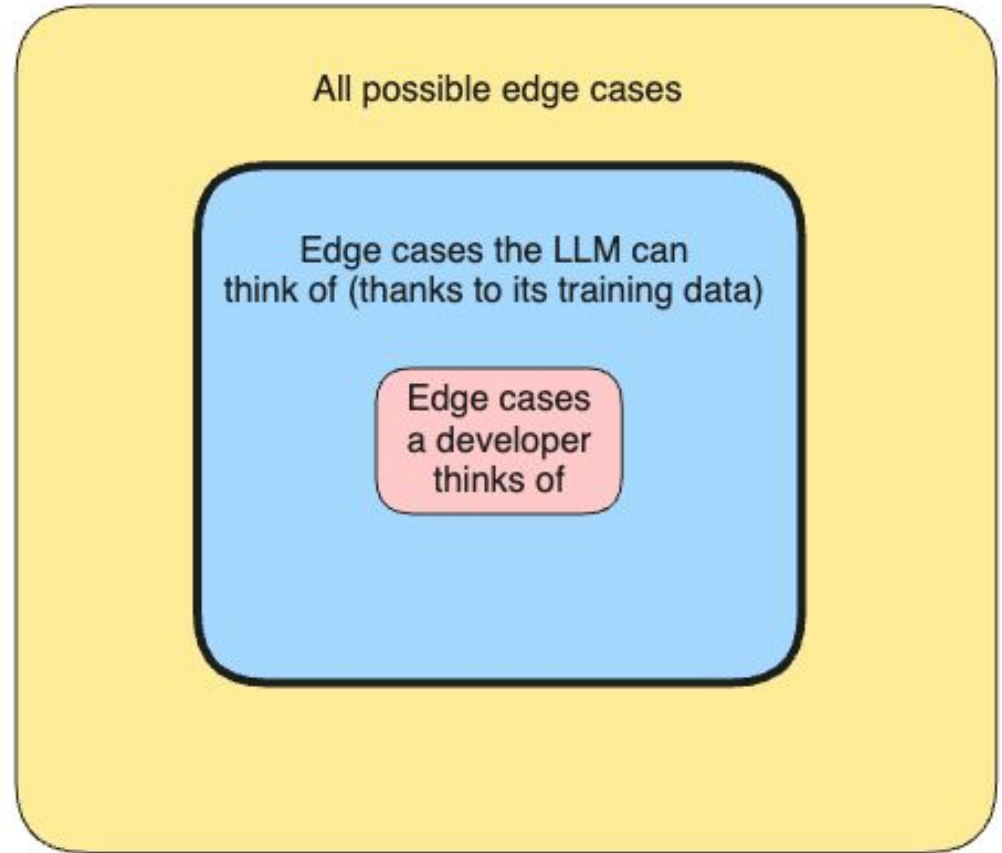
any tools?

## How are LLMs used in software quality assurance?

- Vulnerability detection
- Test generation
- Bug localization
- Verification
- Test automation

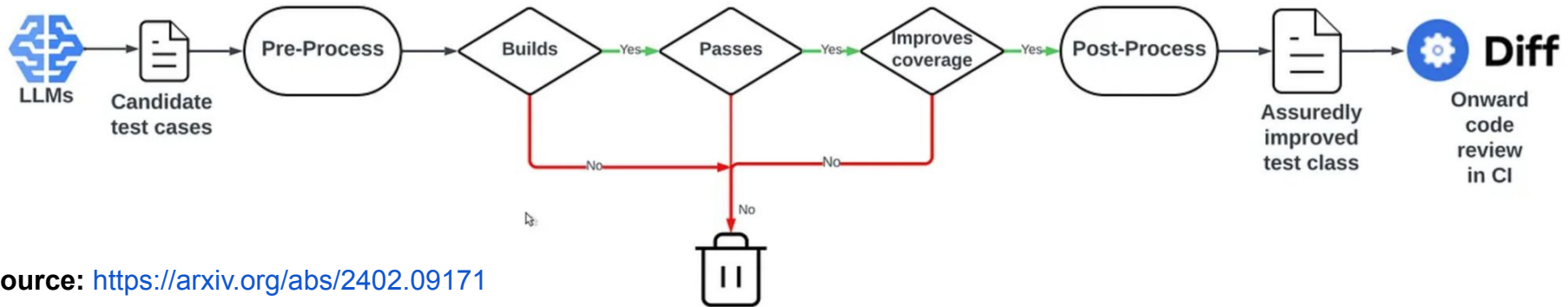
# Improved Testing

- transforming one natural language tests:
  - unit tests,
  - selenium tests,
  - synthetic test data
- proven clear value



## Example: Automated Unit Test Improvement using Large Language Models at Meta

- Candidate Test Case Generation
- Pre-Processing
- Builds
- Passes
- Post-Processing



Source: <https://arxiv.org/abs/2402.09171>

# Best applications today

- 👍 Code writing:
  - boilerplate and repetitive code
- 🔑 Changes to end-user code
  - not affecting too many classes
- 🙌 Test and test data generation
- 📖 Assistance with unfamiliar tasks
- 💪 Integrated search over data lake

## Not good 🤨:

- Changes to complex code
- Refactoring
- Generating complex code over multiple classes

# Tech debt?



Kevlin Henney

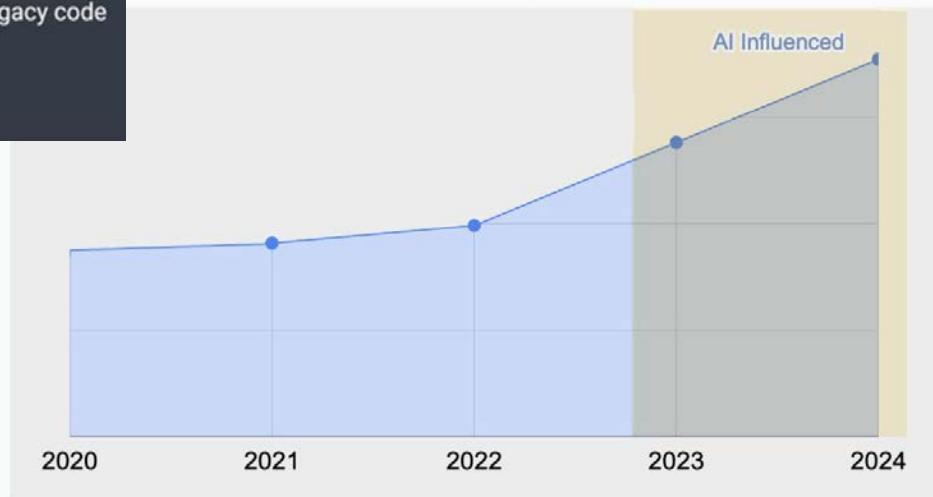
@kevin@mastodon.social

The practical consequence of using LLMs to generate code is that many developers will find they have unwittingly moved themselves into a role they were probably trying to avoid: they have automated the creation of legacy code and have redefined their job role as debugging and fixing such code.

Apr 03, 2023, 18:30 · Web · 442 · 583

mastodon.social/@kevin/110136069252875177

### Code Churn by Year



Thank you!