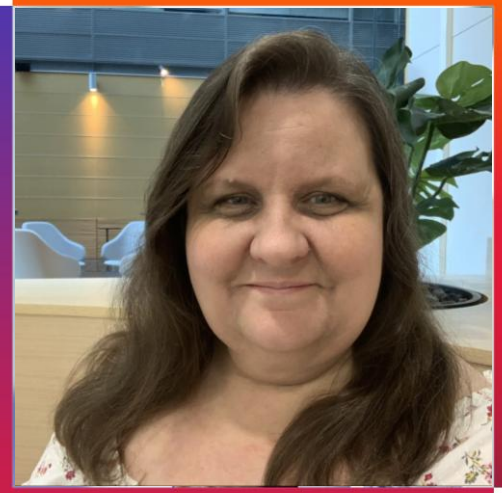
Lessons Learned from Landing a Job Offer with GenAI



CC BY-NC-SA 4.0

Maaret Pyhäjärvi

CGI

Timeline

Oct 29 ‘21 Launch of GitHub Copilot
Predictive text for code in IDE

Nov 30 ‘22 Launch of ChatGPT
Expecting chat like /fix /explain /doc /tests

01

02

03

04

Dec 10 ’21 Job Interview
Own IDE, language of choice and pairing with a developer for org without testers

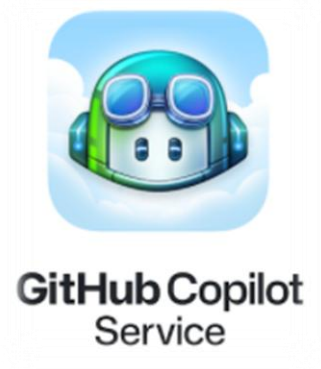
Jun 19 ’25 BCS SIGiST Summer Conference
Reflection for stage here.

© 2025 CGI Inc.

A pair programming interview

Step 1. Prepare: Roman Numerals Kata

Step 2. Surprise: Numbers to Text



ApprovalTests

© 2025 CGI Inc.

#function to convert integer to English for values from 0 to 999

```python
def int_to_text(n):
    if n < 20:
        return ["zero", "one", "two", "three", "four", "five", "six",
                "seven", "eight", "nine", "ten", "eleven", "twelve",
                "thirteen", "fourteen", "fifteen", "sixteen",
                "seventeen", "eighteen", "nineteen"][n]
    elif n < 100:
        return ["twenty", "thirty", "forty", "fifty", "sixty",
                "seventy", "eighty", "ninety"][(n // 10) - 2]
                + (["", " " + int_to_text(n % 10)][n % 10 > 0])
    elif n < 1000:
        return int_to_text(n // 100) + " hundred"
               + (["", " " + int_to_text(n % 100)][n % 100 > 0])
    else:
        return "one thousand"
```

GitHub Copilot Service

```python
def numbers_list(num):
    num_list = []
    for i in range(0, num+1):
        num_list.append(i)
    return num_list

from approvaltests.combination_approvals import verify_all_combinations

def test_all_cases():
    verify_all_combinations(int_to_english, [
        numbers_list(100)])
```

```
 999  args: (998,) => 'nine hundred ninety eight'
1000  args: (999,) => 'nine hundred ninety nine'
1001- args: (1000,) => 'one thousand'
1002
```

```
 999  args: (998,) => 'nine hundred ninety eight'
1000  args: (999,) => 'nine hundred ninety nine'
1001+ args: (1000,) => 'thousand'
1002
```

© 2025 CGI Inc.

```python
#function that turns integers into text from 0 to 999
def int_to_english(num):
    #check for edge cases
    if num == 0:
        return "zero"
    if num > 999:
        return "too large"
    #convert numbers to text
    tens = ["", "", "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", "ninety"]
    ones = ["zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine"]
    teens = ["ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", "seventeen", "eighteen", "nineteen"]
    #set variables to store the result
    result = ""
    hundreds = num // 100
    tens_place = num % 100 // 10
    ones_place = num % 10
    #convert hundreds
    if hundreds > 0:
        result += ones[hundreds] + " hundred"
    #convert tens
    if tens_place > 0:
        if tens_place == 1:
            result += " " + teens[ones_place]
        else:
            result += " " + tens[tens_place]
    #convert ones
    if ones_place > 0:
        if tens_place == 0:
            result += " " + ones[ones_place]
    return result
```

```python
def test_1becomes_one():
    assert int_to_english(1) == "one"
```

```
_____________________________________ test_1becomes_one _____________________________________

    def test_1becomes_one():
>       assert int_to_english(1) == "one"
E       AssertionError: assert ' one' == 'one'
E         - one
E         +  one
E         ? +

demo.py:33: AssertionError
========================================== short test summary info ==========================
==
FAILED demo.py::test_1becomes_one - AssertionError: assert ' one' == 'one'
=================================================== 1 failed, 1 passed in 0.15s =====
```

```
args: (0,) => 'zero'
args: (1,) => ' one'
args: (2,) => ' two'
```

```
21  args: (20,) => ' twenty'
22  args: (21,) => ' twenty'
23  args: (22,) => ' twenty'
```

```
100  args: (99,) => ' ninety'
101  args: (100,) => 'one hundred'
102  args: (101,) => 'one hundred one'
103  args: (102,) => 'one hundred two'
104  args: (103,) => 'one hundred three'
```

https://gist.github.com/maaretp/cf9b1b0c6d385578a048258394697b5b

© 2025 CGI Inc.

“I love the extra autocompletion that I get with it, it feels like I never have to write any kind of boilerplate code anymore, and I also find it very useful to just ask stuff directly in the IDE. I used to google stuff all the time, and ended up on Stackoverflow a lot, but nowadays I rarely have to do that.”

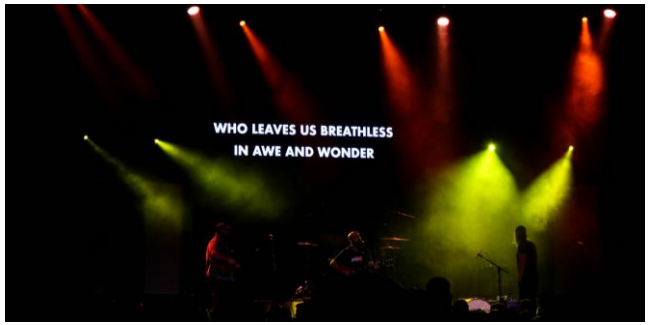
© 2025 CGI Inc.

We are accountable

1. Intent / Implementation
2. Domain for the Layman
3. Domain for the Expert
4. Reference Implementation
5. People Filtering
6. Interesting side effects

© 2025 CGI Inc.

Lessons worth sharing

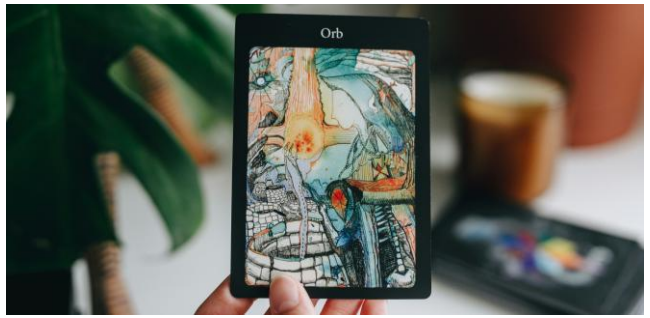


AI marketing smoke screen

Intentional and unintentional smoke screens. Parallels of surprises in interviews to surprises in tool promises.

GenAI Pair Testing

When people fail to pair, tools to pair are worthwhile.



Oracles and expectations

Good helper, insufficient guide. Being in control of your current knowledge.

© 2025 CGI Inc.

Acting Humanly for Testing

Any software is marketed as AI since it is doing something humans could do.

01 Natural Language processing

to communicate successfully in human language

- Summarizing details to insights
- Generating charters
- Creating data, instructions and oracles
- Understanding risk coverage

02 Knowledge representation

to store what it knows or hears

- Remembering features and recognizing feature relationships
- Avoiding reporting accepted problems without change in knowledge
- Remembering what was done
- Knowing what to look at

03 Automated reasoning

to answer questions and draw new conclusions

- Deciding what conversations to start
- Deciding when we can automatically revert
- Reporting with repro scripts
- Recognizing responsibility of fix

04 Machine learning

to adapt to new circumstances and to extrapolate and detect patterns

- Bug taxonomies
- Priorities
- Cross industry reuse of standard tests

05 Computer vision, speech recognition

to perceive the world

- Multilingual projects
- Sources of data
- Visual testing aids

06 Robotics

to manipulate objects and move around

- Robotic process automation as basis of testing
- Operating interfaces abstracting away technology of target

© 2025 CGI Inc.

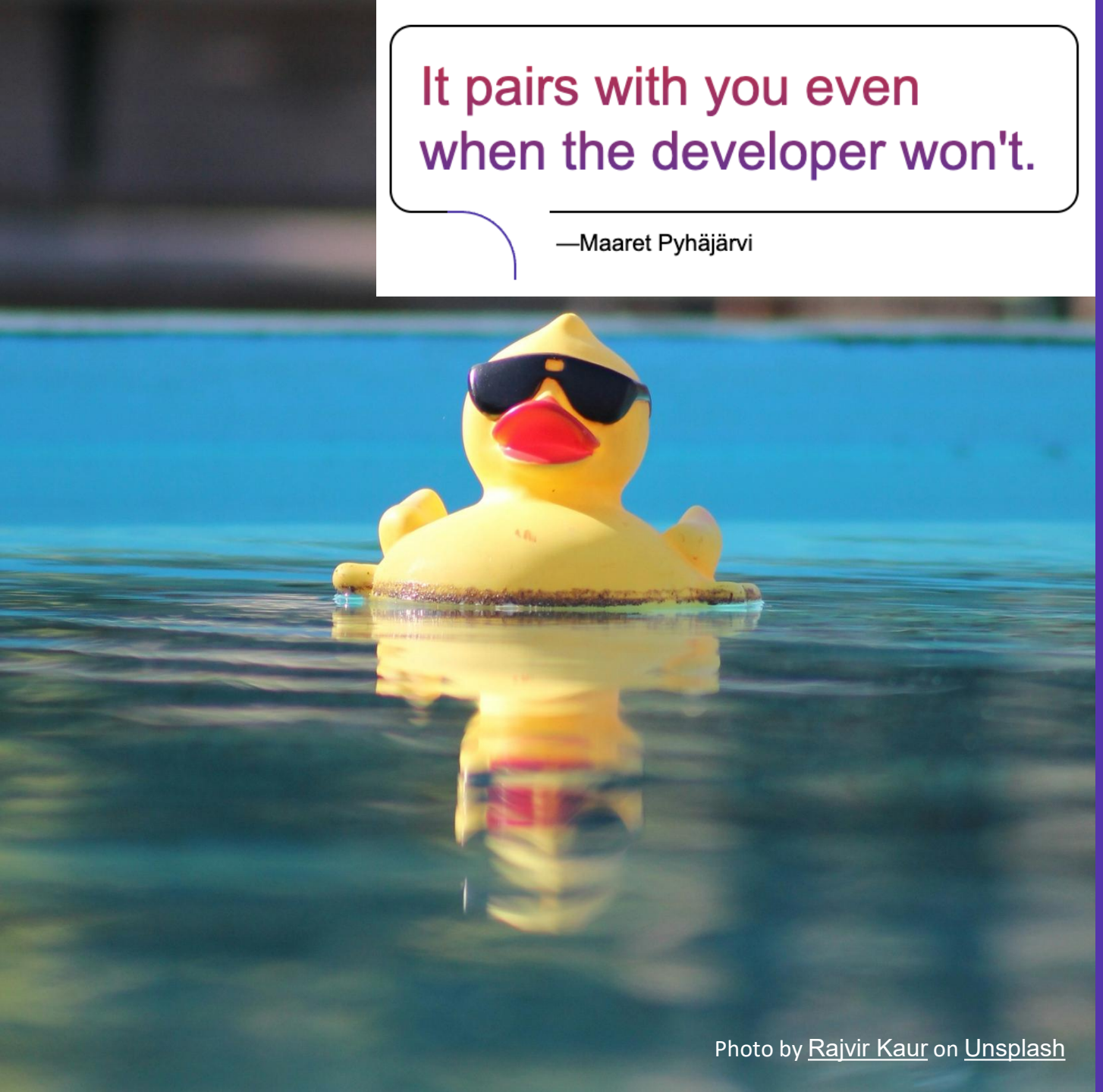


Photo by Rajvir Kaur on Unsplash

GenAI Pair Testing

Search boundaries: argue for different stances on assumptions

Recognize insufficiency and fix it – creating average text is not *your* goal

Freedom to criticize as the pair takes no offense

Dare to ask things you'd not dare to ask from a colleague

Co-piloting allows for repair

Practice-level guardrails

01

Expected values

Pay attention to the old testing wisdom of oracles and **how do we know**. Our critical thinking, built on our learning through curiosity of the world is essential.

02

Anti-toolist worldview

Realize that features in tools can be copied. Looking for the **one best tool makes little sense**. We need to protect our time to a partner of choice.

03

Taskwide learning

Not lifelong learning or life wide learning, but it's task wide learning. **Everything we do is learning activity**.

© 2025 CGI Inc.

Coding productivity levels

AI in your IDE, with agents (Cursor.com)
- LLM digs around and finds the right context
- LLM uses lint to find and fix errors while coding
- LLM creates and edits multiple files directly
- LLM can do commands like renaming folders, git commit/push, etc

AI in your IDE, with tagging (Cursor.com)
- LLM includes current file as context
- Add additional files to context by tagging files/folders
- LLM edits files directly (one at a time)

AI copy/paste
- Manually copy code from IDE to ChatGPT, and hope you didn't forget anything important
- Manually copy the resulting code from ChatGPT into your IDE, and hope you put everything in the right place.
- Error prone because you probably forgot some context, and the LLM has to guess.

Manual coding
- Write code by hand
- Google around to find examples
- Lots of manual mistakes and debugging

Future of Work by Henrik Kniberg,
at Jfocus '25
https://www.youtube.com/watch?v=_aEaq7e0LOA

What is GitHub Copilot

Code assistance. Features in the IDE and when code hosted with GitHub. Valuable use cases even when code not hosted on GitHub. In use since 2021.

Available in IDE

Code completions
Intent to code proposals for efficiency of skilled user.

Contextual help
Fixing, explaining and documenting in context of code.

Code review
Reviews of selection.

Edits and agent mode
Multifile context and background tasks with tools.

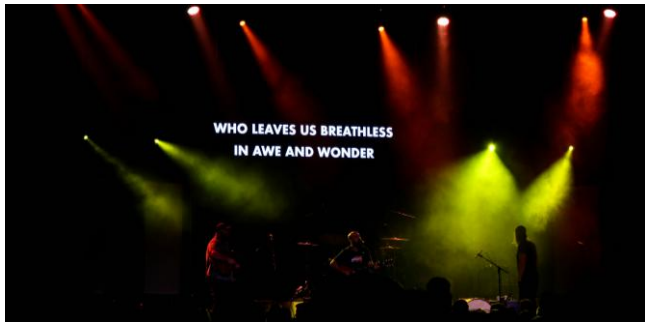
Available in GitHub

Knowledge base
Retrieval augmented generation (RAG) applied on docs close to code.

AI reviewer
Virtual team member describing and reviewing pull requests.

© 2025 CGI Inc.

Lessons worth sharing

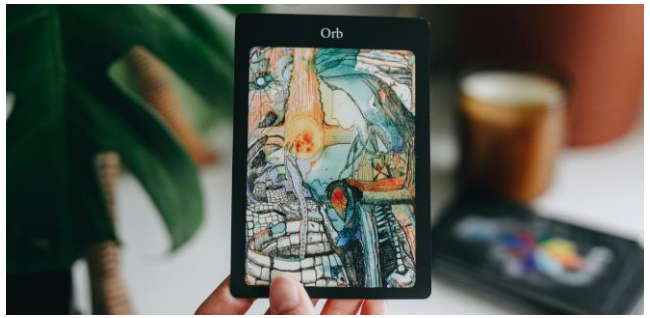


AI marketing smoke screen

Intentional and unintentional smoke screens. Parallels of surprises in interviews to surprises in tool promises.

GenAI Pair Testing

When people fail to pair, tools to pair are worthwhile.



Oracles and expectations

Good helper, insufficient guide. Being in control of your current knowledge.

© 2025 CGI Inc.

Insights you can act on

Founded in 1976, CGI is among the largest IT and business consulting services firms in the world.

We are insights-driven and outcomes-based to help accelerate returns on your investments. Across hundreds of locations worldwide, we provide comprehensive, scalable and sustainable IT and business consulting services that are informed globally and delivered locally.

cgi.com

CGI