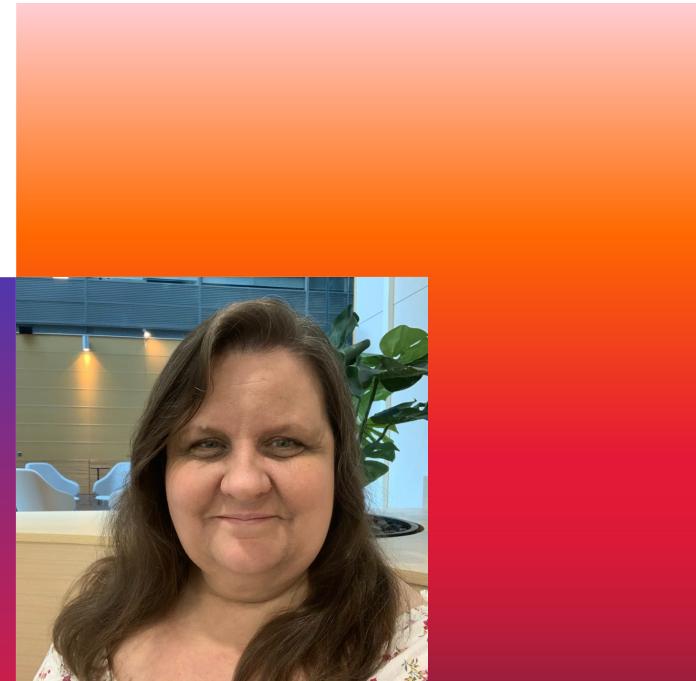


Sociotechnical Guardrails for AI-Driven Application Testing

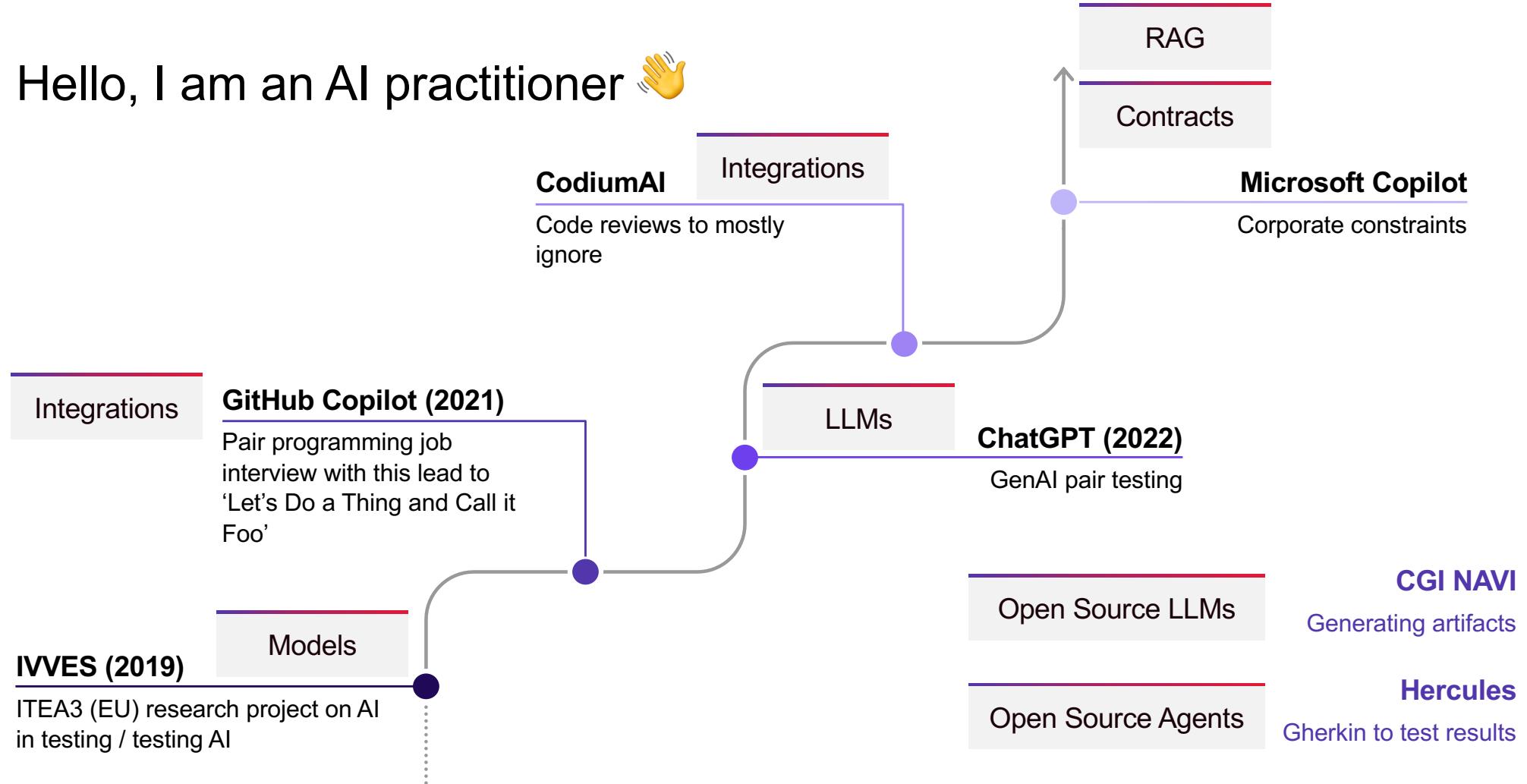
 CC BY-NC-SA 4.0

Maaret Pyhäjärvi
November 2024



CGI

Hello, I am an AI practitioner 🙌



A photograph of a snowy landscape. In the foreground, a dark asphalt road curves to the left, with a white dashed line marking its center. To the right of the road stands a red triangular warning sign with a black silhouette of a bird in flight. The background is filled with tall evergreen trees heavily laden with thick, white snow. The sky is overcast and gray.

Time used on warning about test
automation is time away from
succeeding with it.

Photo by [Filip Zrnzević](#) on [Unsplash](#)

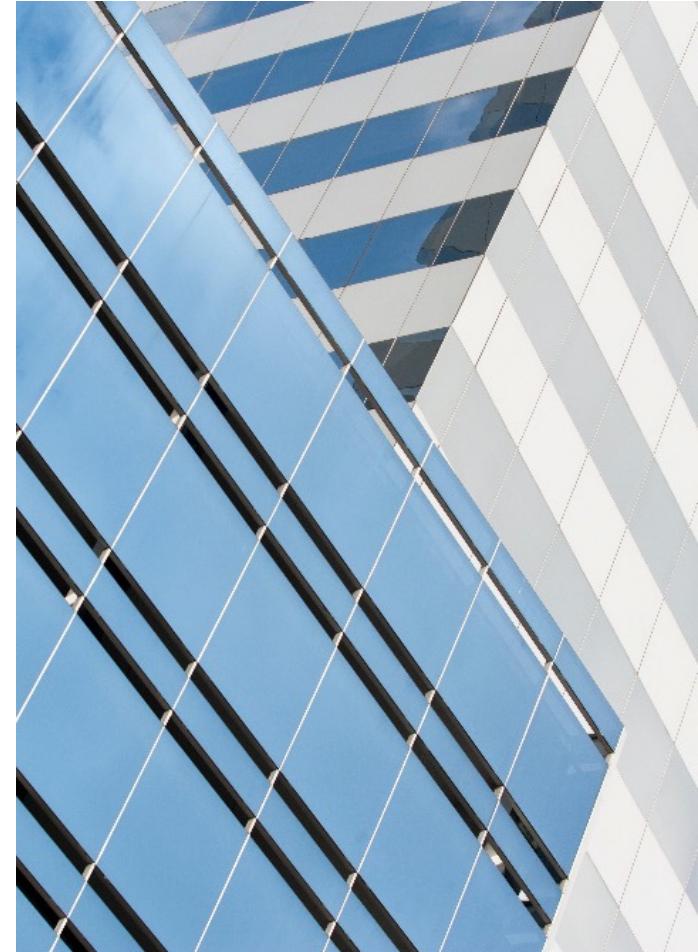
Setting expectations

01 Programmatic tests with Github Copilot

02 GenAI pair testing with general purpose LLMs

03 RAGifying with general purpose genAI

04 Sociotechnical guardrails



Programmatic tests with Github Copilot



roman.py – New Important Project



py roman.py X



py roman.py



```
1 # Integer| to Roman  
2
```

Guessing with power to accept

...



0 0

Live Share

Watch



Prettier





roman.py X



roman.py

```
1 # Integer to Roman
2 # By: Shahzeb Jadoon
3
```

...



0 0

Live Share

Watch



Prettier



Guessing with power to accept

Who Am I?



CTRL+enter for alternatives

```
important_program.py
1      #roman numeral converter
2      #by: jake
3      def |
4
```

...

 GitHub Copilot 

1 Synthesizing 10/10 solutions

3
4

```
5     def roman_numeral_converter(number):
6         if number == 1:
7             return "I"
8         elif number == 2:
9             return "II"
10        elif number == 3:
11            return "III"
12        elif number == 4:
13            return "IV"
14        elif number == 5:
15            return "V"
16        elif number == 6:
17            return "VI"
18        elif number == 7:
19            return "VII"
20        elif number == 8:
21            return "VIII"
22        elif number == 9:
23            return "IX"
24        elif number == 10:
25            return "X"
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

JUPYTER

zsh + ^ X



```
Accept Solution  
37 def int_to_roman(num):  
38     val = [  
39         1000, 900, 500, 400,  
40         100, 90, 50, 40,  
41         10, 9, 5, 4,  
42         1  
43     ]  
44     syb = [  
45         "M", "CM", "D", "CD",  
46         "C", "XC", "L", "XL",  
47         "X", "IX", "V", "IV",  
48         "I"  
49     ]  
50     roman_num = ''  
51     i = 0  
52     while num > 0:  
53         for _ in range(num // val[i]):  
54             roman_num += syb[i]  
55             num -= val[i]  
56         i += 1  
57     return roman_num  
58
```

CTRL+enter for alternatives

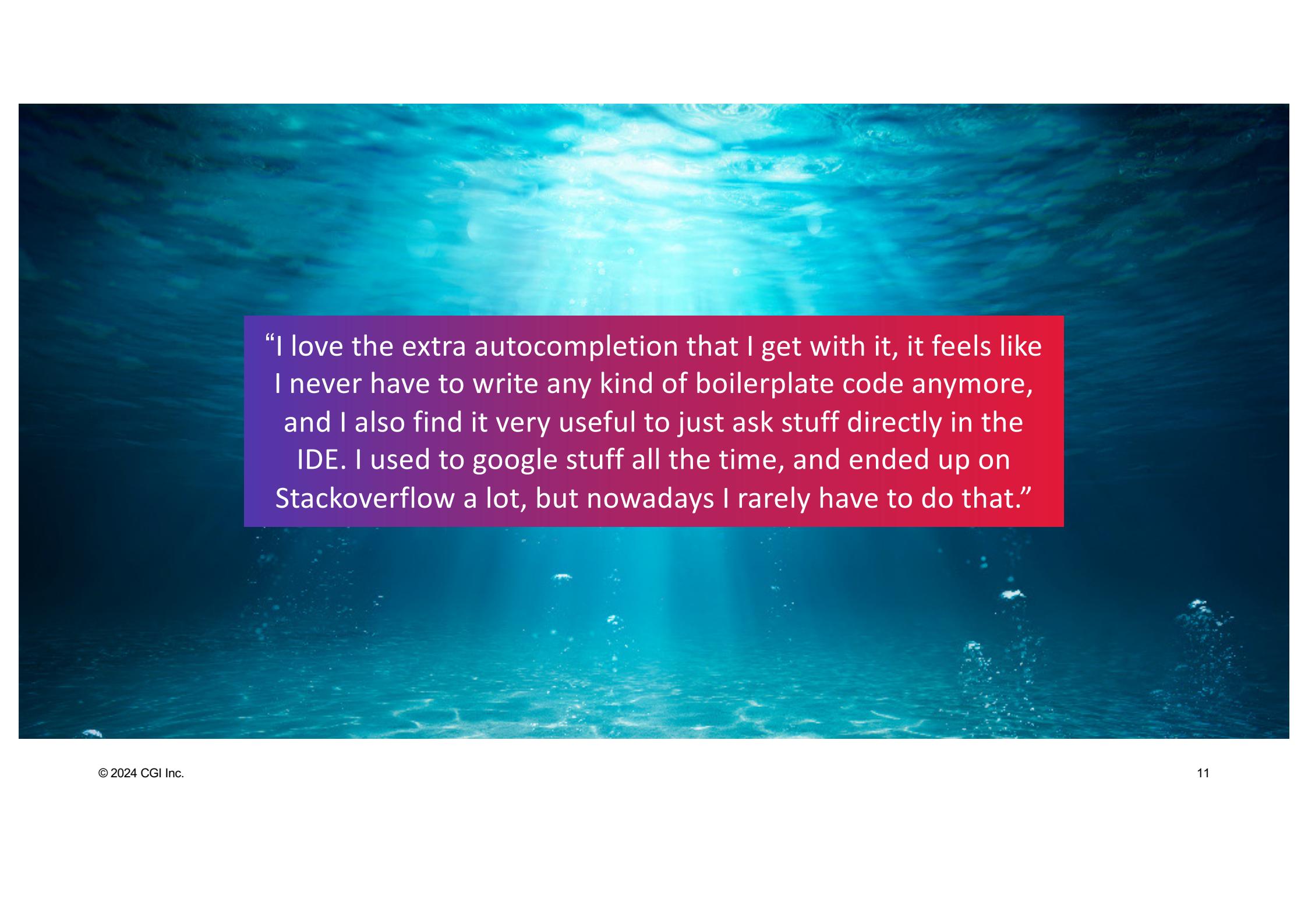
```
def decimal_to_roman(number):  
    if number < 1 or number > 3999:  
        return "Invalid number"  
    roman_numeral = ""  
    roman_numeral += "M" * (number // 1000)  
    roman_numeral += "D" * (number % 1000 // 500)  
    roman_numeral += "C" * (number % 500 // 100)  
    roman_numeral += "L" * (number % 100 // 50)  
    roman_numeral += "X" * (number % 50 // 10)  
    roman_numeral += "V" * (number % 10 // 5)  
    roman_numeral += "I" * (number % 5 // 1)  
    return roman_numeral
```

```
Accept Solution  
103 def int_to_roman(num):  
104     # ... code omitted for brevity  
105     return result  
106
```

```
30 import pytest
31 @pytest.mark.parametrize('number, expected', [(3, 'III'), (4, 'IV'),
32     (9, 'IX'), (10, 'X'), (40, 'XL'), (50, 'L'), (90, 'XC'), (100, 'C'),
33     (400, 'CD'), (500, 'D'), (900, 'CM'), (1000, 'M'), (12, 'XII'), (45, 'XLV'),
34     (1992, 'MCMXCII'), (2022, 'MMXXII'), (490, 'CDXC'), (3333, 'MMMCCCXXXIII')])]
35 def test_first(number, expected):
36     assert convert(number) == expected
37
38 @pytest.mark.parametrize('input', [(1.5), (1.6), ('moi'), (None)])
39 def test_second(input):
40     with pytest.raises(ValueError):
41         convert(input)
42
43 def numbers_list(num):
44     num_list = []
45     for i in range(1, num + 1):
46         num_list.append(i)
47     return num_list
48
49 from approvaltests.combination_approvals import verify_all_combinations
50
51 def test_all_cases():
52     verify_all_combinations(convert, [
53         numbers_list(100)])
```

Some Tests

Done?



“I love the extra autocomplete 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.”

GenAI Pair Testing with General Purpose LLMs



Photo by [Rajvir Kaur](#) on [Unsplash](#)

GenAI Pair Testing

Search boundaries: argue for different stances on assumptions

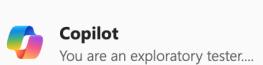
Recognize insufficiency and fix it
– creating average text is not
your goal

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

Imagine exploring from generated charters



Charter 1:

- **Target:** Login page on the web application.
- **Resources:** Desktop browser (Chrome, Firefox, Safari), various user credentials (valid, invalid, expired), network conditions (online, offline, slow connection).
- **Information:** How does the login functionality handle different types of user credentials? What error messages are displayed for invalid login attempts? How does the system behave under different network conditions?

Charter 2:

- **Target:** Login functionality on the mobile application.
- **Resources:** Mobile devices (iOS, Android), different screen sizes, biometric authentication (fingerprint, face recognition), various user credentials.
- **Information:** How does the login process perform on different mobile devices and screen sizes? Is biometric authentication working correctly? Are error messages and prompts user-friendly and clear?

Charter 3:

- **Target:** Login security features.
- **Resources:** Security testing tools, various user credentials, different attack scenarios (brute force, SQL injection, XSS).
- **Information:** How secure is the login functionality against common attack vectors? Are there any vulnerabilities that could be exploited? How effective are the security measures in place?

Charter 4:

- **Target:** User experience during login.
- **Resources:** Usability testing tools, user feedback, different user personas (new users, returning users, users with disabilities).
- **Information:** How intuitive and user-friendly is the login process? Are there any usability issues that need to be addressed? How does the login experience vary for different types of users?



Predictive text generation useful for recall not reasoning

ChatGPT ▾



How many r's are in strawberry



There are 2 "r"s in the word "strawberry."



RAGifying with general purpose genAI

Simplest form of larger context

ChatGPT ▾



How is this different from how people usually teach exploratory testing?

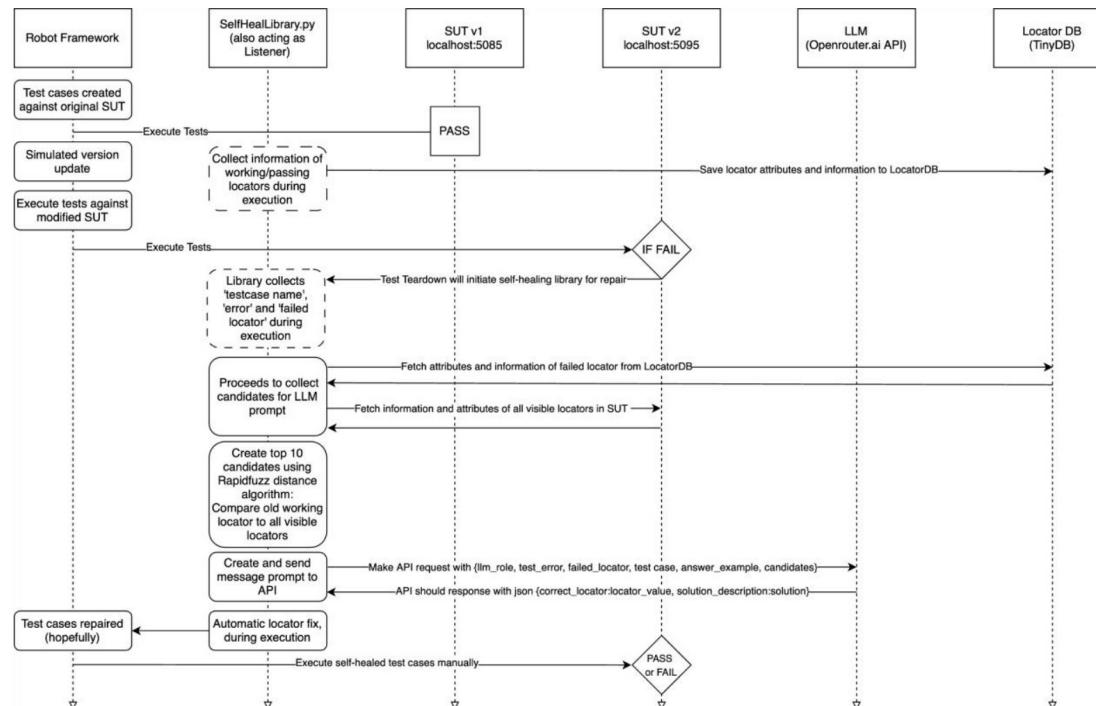


The uploaded file outlines a course on exploratory testing, emphasizing a blend of manual and automated testing. Here are the key points on how this approach differs from traditional methods of teaching exploratory testing:

1. Integration of Automation and Manual Testing:

- The course highlights the importance of using test automation as part of exploratory testing. Automation is presented not just as a means to replace manual testing, but as a tool to extend the tester's reach, enabling tests to be run continuously and serving as a basis for further exploration.

RAG + input templating applied over task breakdown



Rohamo, Paavo. Enabling Self-healing Locators for Robot Framework with Large Language Models
[\(thesis\)](#)

Figure 3.6: Self-healing mechanism process flow when using the “Strong Candidate Method”

Mapping the acronyms

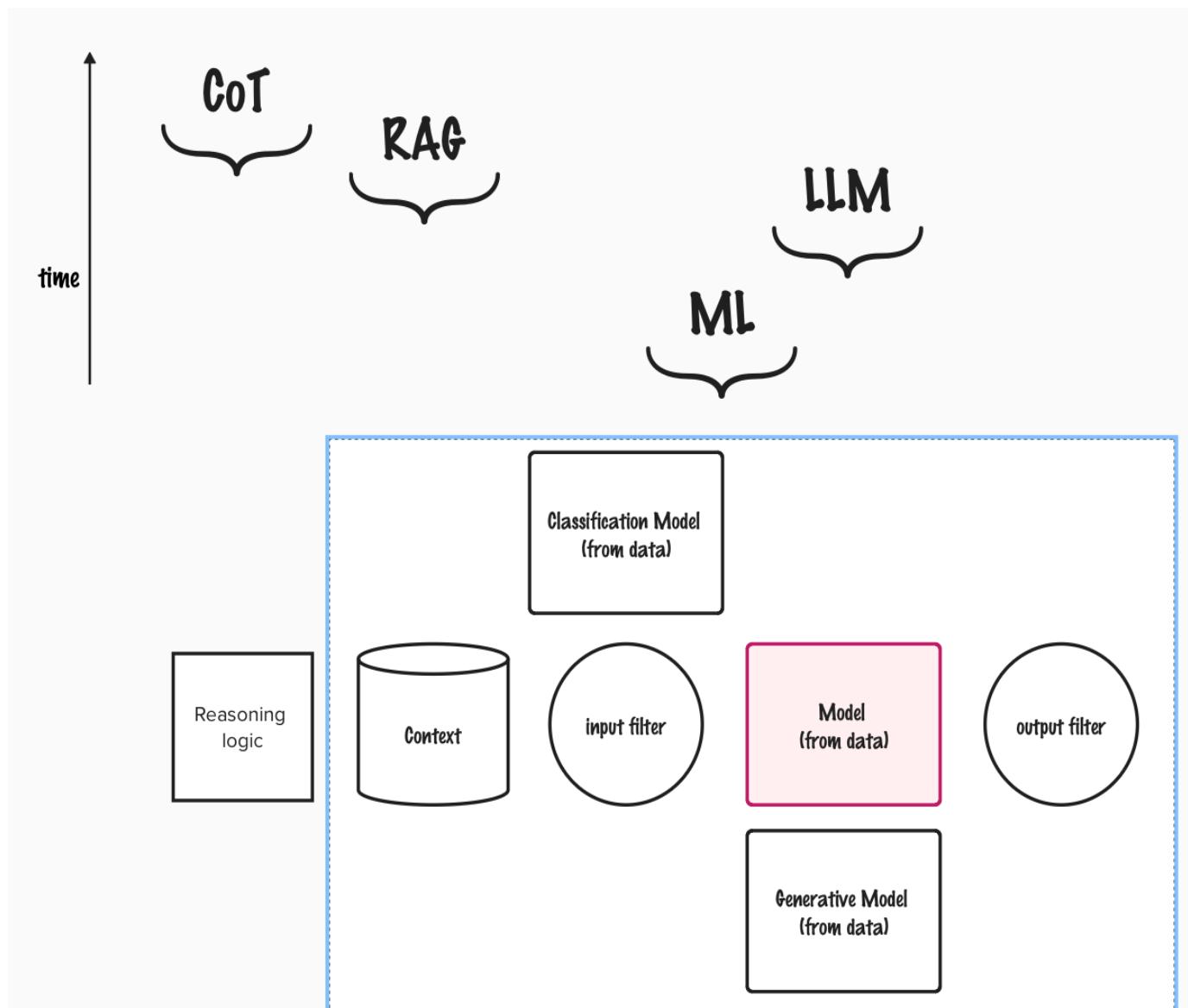
ML Machine Learning

LLM Large Language Model

RAG Retrieval Augmented Generation

CoT Chain of Thought

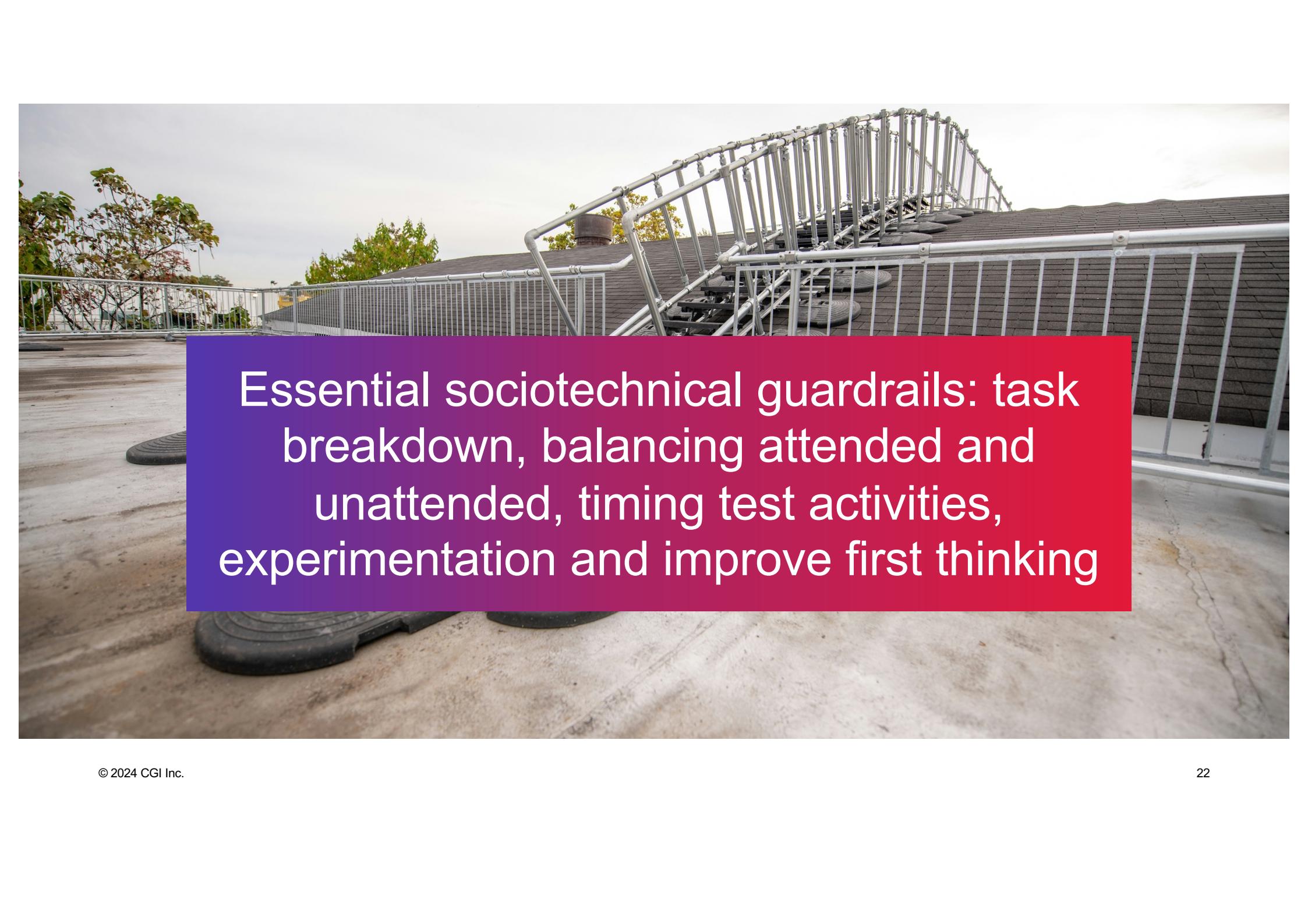
Agents Actors in Flows



Sociotechnical Guardrails



Core sociotechnical guardrail:
data ownership and
confidentiality



Essential sociotechnical guardrails: task breakdown, balancing attended and unattended, timing test activities, experimentation and improve first thinking



Boundary-seeking sociotechnical guardrails: costs, compensations, architecture

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

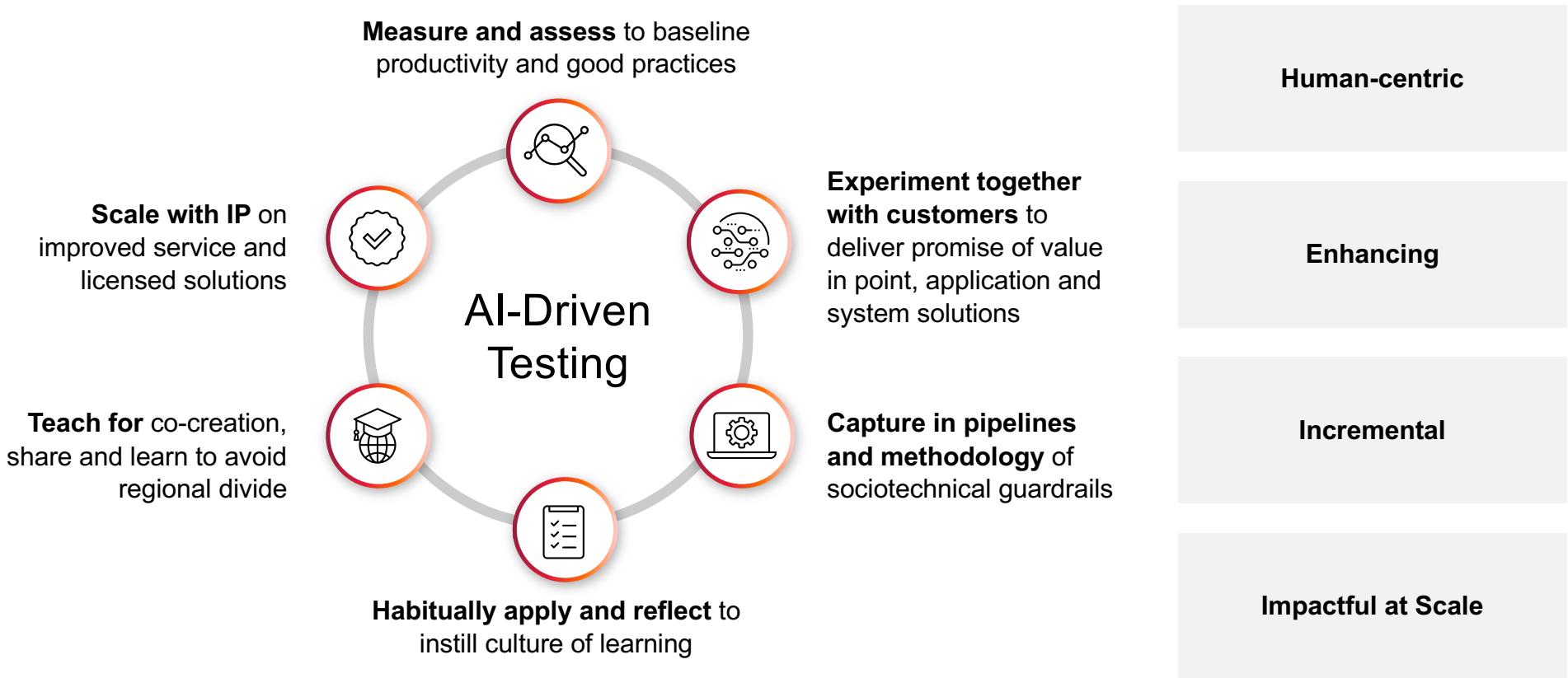
Realize that features in tools can be copied. Looking for the **one best tool makes little sense**. We need to protect our time to a partner of choice.

03

Taskwide learning

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

Shared Direction and Results that Show up in Scale



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