



Open Universiteit



**UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA**

The future of automating regression testing at the Graphical User Interface (GUI)

Tanja Vos





Open Universiteit



**UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA**

The future of automating regression testing at the Graphical User Interface (GUI)

**(or how to stop flushing money away doing
it the way we currently do)**

Tanja Vos



5 billion euros

annual cost of
maintaining broken
GUI test scripts

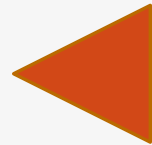


Estimation based on EU automation testing market \approx €10–11 B (2024-25) \times 30–50 % maintenance share (sources: Market Data Forecast, Fortune Business Insights, Capgemini World Quality Report 2023-24, IT Convergence 2023.)

Testing at the GUI a user story to “apply for a loan”

Step 1

Open Bank Application



Step 2

Login with a valid user account

Step 3

Click on Request Loan

Step 4

Type a valid amount of money

Step 5

Click to apply now the loan



Testing at the GUI a user story to "apply for a loan"

Step 1
Open Bank Application

Step 2
Login with a valid user account

Step 3
Click on Request Loan

Step 4
Type a valid amount of money

Step 5
Click to apply now the loan

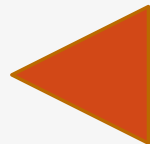


Testing at the GUI a user story to "apply for a loan"

Step 1
Open Bank Application

Step 2
Login with a valid user account

Step 3
Click on Request Loan



Step 4
Type a valid amount of money

Step 5
Click to apply now the loan

PARA BANK
Experiences by difference

Welcome to Account Services

Account Services

- Open New Account
- Accounts Overview
- Transfer Funds
- Bill Pay
- Card Transactions
- Request Loan**

Accounts Overview

Account	Balance*	Available Amount
11215	-12500.00	\$0.00
11216	\$12.43	\$12.43
11217	\$100.00	\$100.00
11218	-1198.00	\$0.00
11219	\$100.00	\$100.00
11220	\$0.00	\$0.00
11221	\$100.00	\$100.00
11222	\$1100.00	\$1100.00
11223	\$100.00	\$100.00
11224	\$1231.18	\$1231.18
55321	\$1331.12	\$1331.12
Total	\$1932.67	
Total	\$1932.67	

*Balance includes Accounts that may be subject to holds

Testing at the GUI a user story to “apply for a loan”

Step 1

Open Bank Application

Step 2

Login with a valid user account

Step 3

Click on Request Loan

Step 4

Type a valid amount of money

Step 5

Click to apply now the loan

The screenshot displays the PARA BANK website interface. At the top, the logo and tagline 'Experience the difference' are visible. A navigation menu on the left includes links for Solutions, About Us, Services, Products, Locations, and Admin Page. The main content area features a 'Welcome to Account Services' banner with three icons. Below the banner, a 'Welcome John Smith' message is followed by an 'Account Services' section with a list of links: Open New Account, Accounts Overview, Transfer Funds, Bill Pay, Find Transactions, Update Contact Info, Request Loan, and Log Out. The 'Apply for a Loan' form is highlighted, showing input fields for 'Loan Amount: \$' and 'Down Payment: \$', a dropdown menu for 'From account #: 12557', and an 'APPLY NOW' button. A green box highlights the input fields, and an orange arrow points to the 'Request Loan' link in the navigation menu.

Testing at the GUI a user story to “apply for a loan”

Step 1

Open Bank Application

Step 2

Login with a valid user account

Step 3

Click on Request Loan

Step 4

Type a valid amount of money

Step 5

Click to apply now the loan



The screenshot displays the PARA BANK website interface. At the top, the logo and tagline 'Experience the difference' are visible. A navigation menu on the left includes links for Solutions, About Us, Services, Products, Locations, and Admin Page. A banner area features a 'Welcome to Account Services' message with three icons. Below this, a 'Welcome John Smith' message is shown. The 'Account Services' section lists various options, with 'Request Loan' highlighted. The 'Apply for a Loan' form contains three input fields: 'Loan Amount: \$ 999', 'Down Payment: \$ 111', and 'From account #: 12567'. An 'APPLY NOW' button is highlighted with a red box.

Testing at the GUI a user story to "apply for a loan"

After last step:

- The user story succeeds
- The loan is approved

After each step:

- No failure occurs
- No error message appears

The screenshot displays the PARA BANK website interface. At the top, the logo and tagline "PARA BANK Experience the difference" are visible. A navigation menu on the left includes links for Solutions, About Us, Services, Products, Locations, and Admin Page. The main content area features a "Welcome to Account Services" banner with three icons representing home, account, and mail. Below the banner, a "Loan Request Processed" notification box is shown, containing the following information:

- Loan Provider: Wealth Securities Dynamic Loans (WSDL)
- Date: 11-4-2024
- Status: Approved

Below the notification box, a message reads: "Congratulations, your loan has been approved" and "Your new account number: 123456".

**We need to repeat this every time
the software changes!**



Manual regression testing

- Expensive
- Boring
- Error-Prone

Needs to be automated

State of automation practice: make scripts

Step 1

Open Bank Application

Step 2

Login with a valid user account

Step 3

Click on Request Loan

Step 4

Type a valid amount of money

Step 5

Click to apply now the loan

```
1 from playwright.sync_api import sync_playwright
2
3 with sync_playwright() as p:
4     # Launch browser
5     browser = p.chromium.launch(headless=False)
6     page = browser.new_page()
7
8     # Step 1: Open bank application
9     page.goto("https://parabank.parasoft.com/parabank/index.htm")
10
11     # Step 2: Login with a valid user account
12     page.fill("input[name='username']", "john")
13     page.fill("input[name='password']", "demo")
14     page.click("input[value='Log In']")
15
16     # Step 3: Click on 'Request Loan'
17     page.click("a[href='requestloan']")
18
19     # Step 4: Type a valid amount of money
20     page.fill("input[name='amount']", "999")
21     page.fill("input[name='downPayment']", "123")
22
23     # Step 5: Click to apply for the loan
24     page.click("input[value='Apply Now']")
25
26     # Optional: check that loan was approved
27     if "Loan Request Processed" in page.content() or "approved" in page.content().lower():
28         print("Loan approved successfully.")
29     else:
30         print("Loan approval failed or not displayed.")
31
32     # Close browser
33     browser.close()
```

When the GUI changes.....

Step 1

Open Bank Application

Step 2

Login with a valid user account

Step 3

Click on Request Loan

Step 4

Type a valid amount of money

Step 5

Click to apply now the loan

```
from playwright.sync_api import sync_playwright
with sync_playwright() as p:
    # Launch browser
    browser = p.chromium.launch(headless=False)
    page = browser.new_page()

    # Step 1: Open bank application
    page.goto("https://parabank.parasort")

    # Step 2: Login with a valid user account
    page.fill("input[name='username'", "john")
    page.fill("input[name='password'", "demo")
    page.click("input[value='Log In']")

    # Step 3: Click on 'Request Loan'
    page.click("a(href='requestloan')")

    # Optional: check that loan was approved
    if 'Loan Request Processed' in page.content() or "or" "approved":
        print("Loan approved successfully.")
    else:
        print("Loan approval failed or not displayed.")

    # Close browser
    browser.close()
```

ERROR

Writing scripts for
automated test
execution is over in
the era of AI.



from SCRIPTED
to SCRIPTLESS

SCRIPTLESS

≠

low-code, no code, codeless, code less

from SCRIPTED to SCRIPTLESS

How did we get there?



What this means for:



testers



companies



research



AI

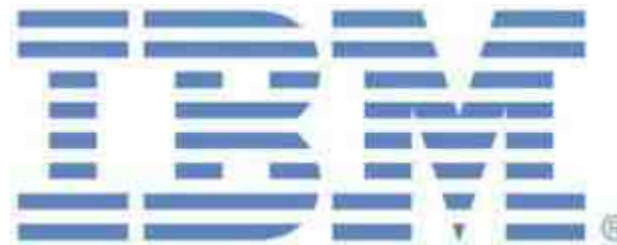


2010 - 2013



EU funding: 5.845.000 euros

It all started in 2010.....



Universiteit Utrecht



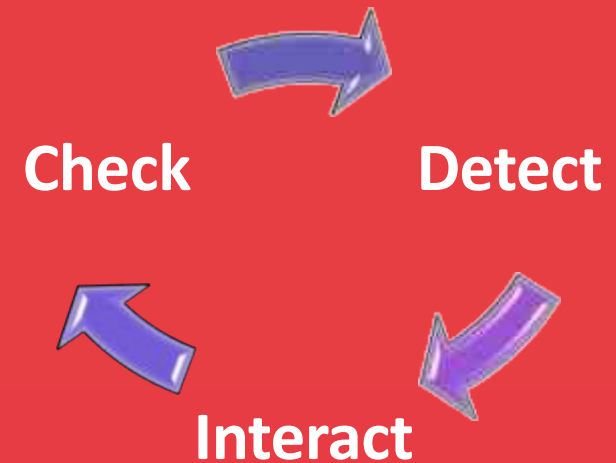
Conditions for doing the research

- ✗ Use testers' time
- ✗ Access the source code
- ✗ Access the bug database
- ✗ Publish how many bugs we found
- ✓ Access to the GUI of the system in production

GUIs

- **Simple**
 - always the same building blocks
 - buttons, menus, text fields, etc.
- **Universal**
 - every app (web, desktop, mobile) has them
- **Accessible**
 - browsers and OS APIs reveal GUI state
 - **GUI ripping** (Atif Memon - GUITAR)
 - we can perform actions automatically
 - **GUI crawling** (Ali Mesbah – CRAWLJAX)

If you have access to the GUI,
you already know the user's
possible actions.



Maandag 20 September 2010

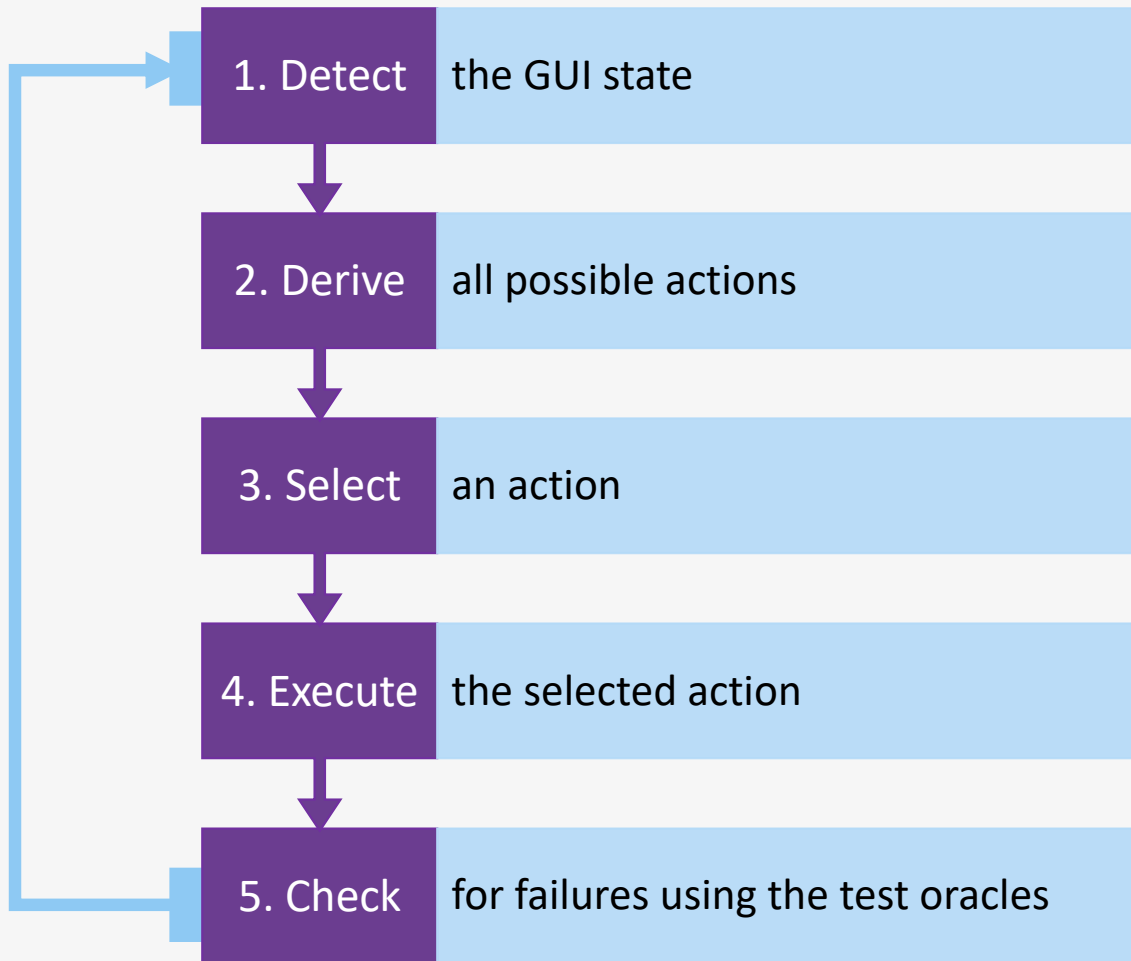
omdenken.nl





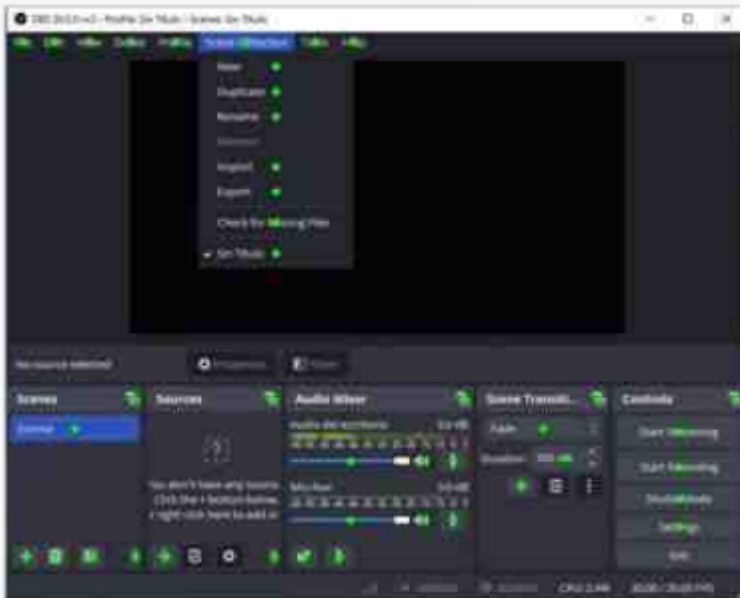
**Then we can just start testing
at the GUI level
no need for scripts!**

How does scriptless testing work?



1. Get current state of the SUT

- Accessibility API of Windows (UIAutomation)
- Webdriver for web applications
- Appium for mobile
- iv4XR for XR games



- The state consists of all widgets + their properties



2. Derive actions

The screenshot displays the PARA BANK website interface. At the top left, there is a navigation menu with items: Solutions, About Us, Services, Products, Locations, and Admin Page. The main header features the PARA BANK logo and the tagline 'Experience the difference.' Below this is a banner with the text 'Welcome to Account Services' and three icons representing home, services, and contact. The main content area is titled 'Apply for a Loan' and contains a form with the following fields:

- Loan Amount: \$
- Down Payment: \$
- From account #:

Below the form is an 'APPLY NOW' button. On the left side of the page, there is a 'Welcome John Smith' message and a list of 'Account Services' including: Open New Account, Accounts Overview, Transfer Funds, Bill Pay, Find Transactions, Update Contact Info, Request Loan, and Log Out.

- Click (Home)
- Click (About Us)
- Click (Contact Us)
- Type (text, 'Loan Amount'-field)
- Type (text, 'Down Payment'-field)
- Click (From Account)
- Click (Apply Now)
- Click (Services)
- Click (Open New Account)
- Click (Accounts Overview)
- Click (Transfer Funds)
- Click (Bill Pay)
- More actions...

3. Select action



The screenshot shows the PARA BANK website interface. At the top left, there is a navigation menu with items: Solutions, About Us, Services, Products, Locations, and Admin Page. Below this is a welcome message for John Smith and a list of 'Account Services' including Open New Account, Accounts Overview, Transfer Funds, Bill Pay, Find Transactions, Update Contact Info, Request Loan, and Log Out. The main content area features a banner for 'Welcome to Account Services' with three icons (Home, Services, Contact Us). Below the banner is the 'Apply for a Loan' form, which includes fields for Loan Amount (\$), Down Payment (\$), and From account # (12567). The 'Apply Now' button is highlighted in yellow. A small cartoon monkey wearing glasses is sitting at a laptop in the bottom right corner of the screenshot.

- Click (Home)
- Click (About Us)
- Click (Contact Us)
- Type (text, 'Loan Amount'-field)
- Type (text, 'Down Payment'-field)
- Click (From Account)
- **Click (Apply Now)**
- Click (Services)
- Click (Open New Account)
- Click (Accounts Overview)
- Click (Transfer Funds)
- Click (Bill Pay)
- More actions...

4. Execute the action, go to new state



Click (Apply Now)



Derive actions again

5. Check oracles



- Oracle = Mechanism to determine whether we found a failure
- In TESTAR: Oracles give verdicts about each state of the GUI
- Per default: general system failures:
 - crashes
 - hangs
 - suspicious outputs

It worked! We found new bugs!

Evaluating Rogue User Testing in Industry: an Experience Report

Sebastian Hauerfeld
 Center for ITDS
 Universidad Politécnica de Valencia
 Valencia, Spain
 Email: shauerfeld@upv.es

Roberto de Rojas
 Clave Informatica SL
 Madrid, Spain
 Email: roberto@clave.es

Tamas E. J. Nils
 Center for ITDS
 Universidad Politécnica de Valencia
 Valencia, Spain
 Email: tamnils@upv.es

Abstract—Testing applications with a standard and mature GUI is no exception. Usually challenging and time-consuming task. The task of the test is to simulate an user explore and verify tools which may identify the recording and execution of input requests but do not support the tester in finding fault-sensitive test cases and leads to a great overhead on maintenance of the test cases when the GUI changes. In earlier works we presented the Rogue User Testing Tool, an automated approach to finding specifications of the GUI level whose objective is to solve part of the maintenance problem by automatically generating test cases based on a structure that is automatically derived from the GUI. In this paper we report on our experience obtained when employing the Rogue User Testing Tool with the Spanish software leader Clave who decided to apply the tool to derive test a comparison of one of their GUI applications. This case put us in discovery potential problems that arise during the work of the Rogue User testing tool. While exploring and test tools, we discovered critical and potential sensitive bugs in the application under test.

1. Introduction

Testing software applications at the Graphical User Interface (GUI) level is a very important testing phase in modern mobile and desktop GUI applications. A central objective in the application under test from where all the functionality is accessed. It is highly to work in an industry with where development are constant in software GUI testing, because operating the application in a vehicle (i.e. the specific components are hard to communicate their bugs or to use with developer).

The previous work we have presented an approach to derive at the GUI level [1], [2] whose objective is to solve part of the maintenance problem by automatically generating test cases based on a structure that is automatically derived from the GUI. The main goal of the tool is to solve the problem of maintaining and updating GUI test cases and their properties and enable professional developers to derive test cases and automatically derive and execute appropriate test cases to cover the GUI and manually verify it. The tool uses a structure learning algorithm called Grammar-based Grammar (GBG) to generate test cases.



	TESTAR
Preparation	26 hour
Testing	91 hour
Post testing	1,5 hour
Critical faults	10



In this application, a critical fault was found without requiring any manual test cases. The goal of this research is to discover and find new bugs in the application under test.

Scriptless Testing at the GUI Level in an Industrial Setting

Hatim Chahin¹, Mehmet Duran², Tanja E. J. Maas^{3,4,5,6}, Pekka Ahu³, and Nelly Condori Fernandez³

¹ ProRail, Utrecht, The Netherlands
² Capgemini, Utrecht, The Netherlands
³ Open Universiteit, Heerlen, The Netherlands
⁴ Universidad Politécnica de Valencia, Valencia, Spain
⁵ vcondori@open.nl
⁶ University of A Coruña/Vrije Universiteit, Amsterdam, The Netherlands

Abstract. TESTAR is a universal based and scriptless tool for test automation at the Graphical User Interface (GUI) level. It is different from existing test approaches because no test cases need to be defined before testing. Instead, the tests are generated during the execution, on-the-fly. This paper presents an empirical case study in a realistic industrial context where we compare TESTAR to a manual test approach of a web-based application in the rail sector. Both qualitative and quantitative research methods are used to investigate learnability, effectiveness, efficiency, and satisfaction. The results show that TESTAR was able to detect more faults and higher functional test coverage than the used manual test approach. As far as efficiency is concerned, the preparation time of both test approaches is identical, but TESTAR can realize test execution without the use of human resources. Finally, TESTAR turns out to be a learnable test approach. As a result of the study described in this paper, TESTAR technology was successfully transferred and the company will use both test approaches in a complementary way in the future.

Keywords: GUI test automation tools · TESTAR · Compare test approaches · Industrial case study · Technology transfer · Railway sector

Introduction

Testing software at the Graphical User Interface (GUI) level is an important part of realistic testing [1]. The GUI represents a central juncture to the Software Under Test (SUT) from where all the functionality is accessed. Consequently, testing at the GUI

	TESTAR	Manual
Preparation	44 hour	43 hour
Testing	51 hour	6 hour
Post testing	5 hour	2 hour
Critical faults	3	0



Finding bugs is one thing...

How can we test user stories?

Well, with random.. you can't!



user stories



non-user stories

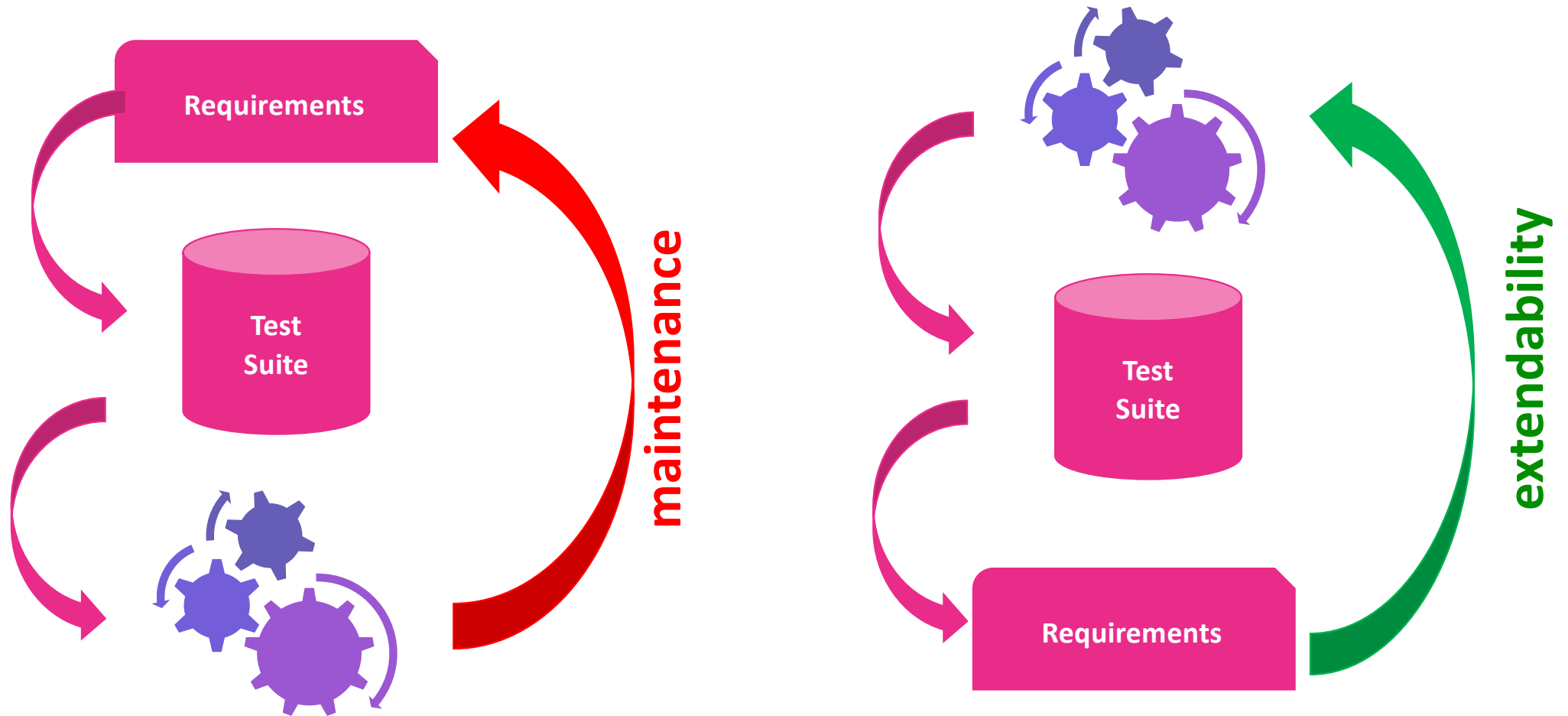


Complementary approaches

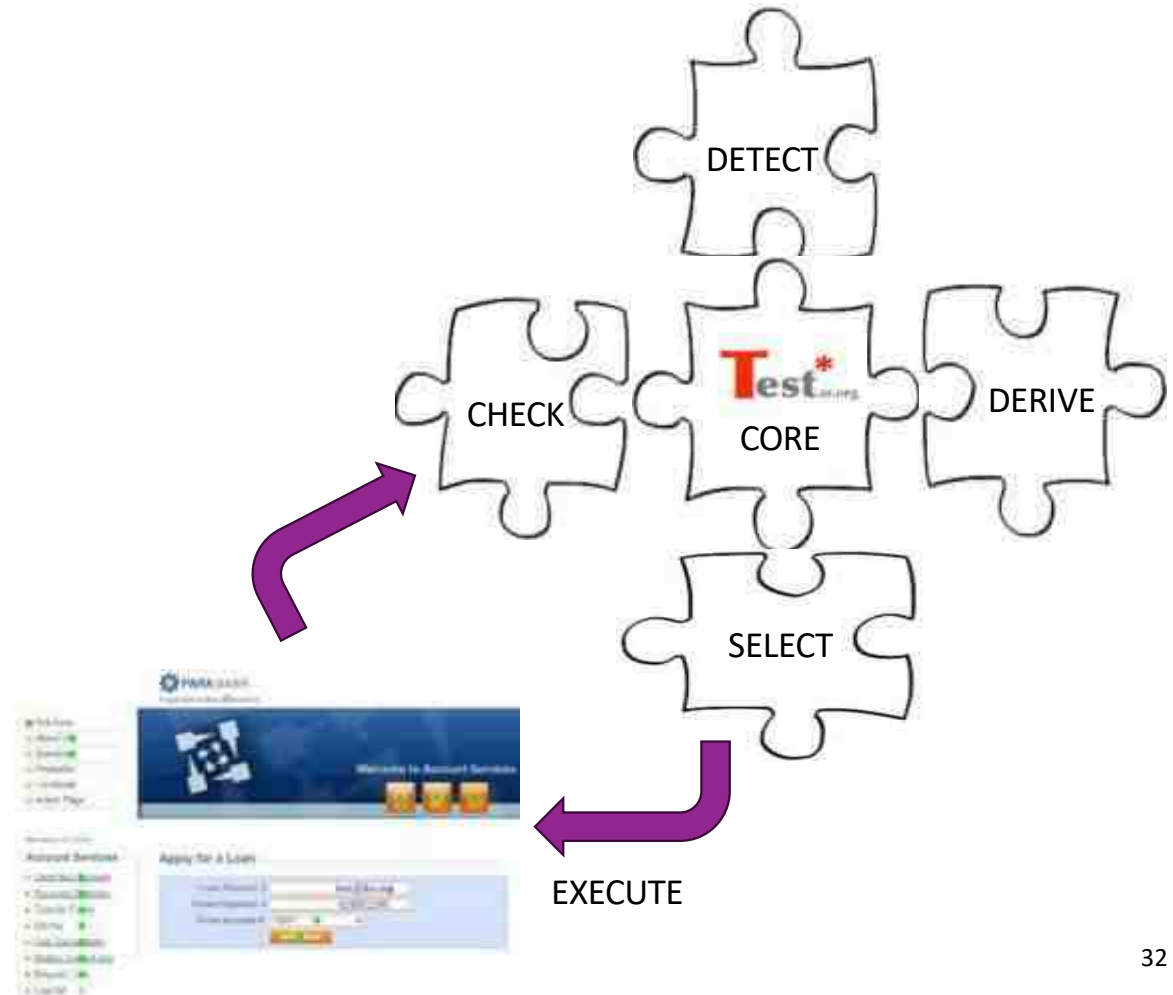
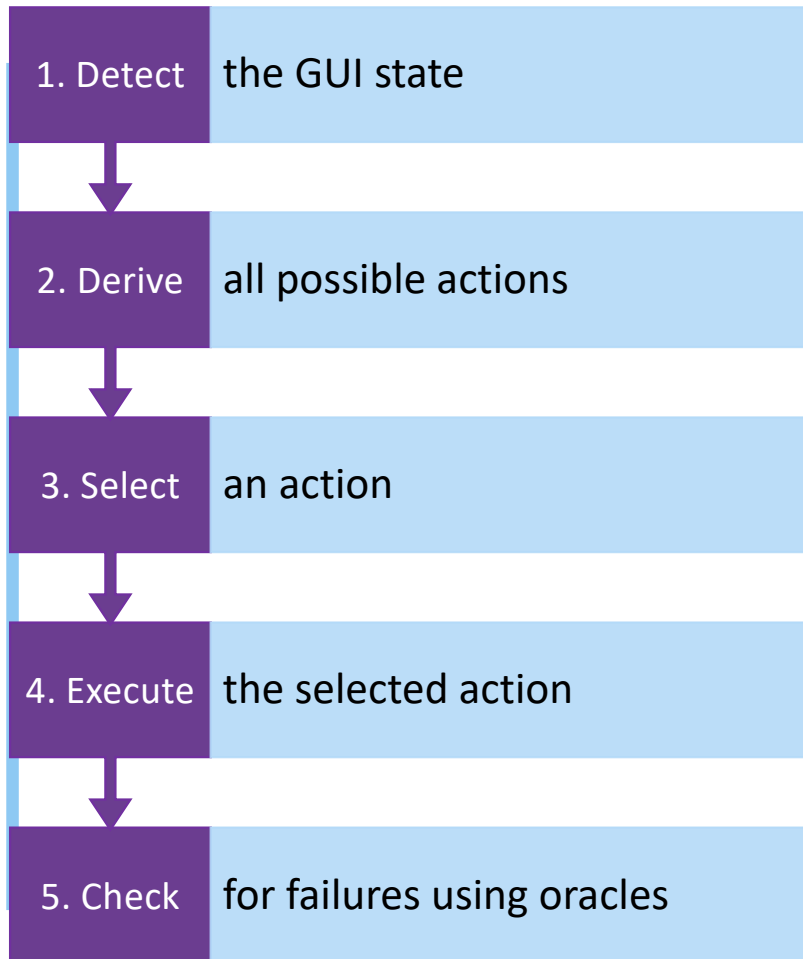
A photograph of a sunset over a body of water. The sun is a bright yellow orb on the horizon, casting a golden glow across the sky and reflecting on the water. The sky transitions from a deep orange near the horizon to a vibrant blue at the top, with scattered white and pinkish clouds. The water in the foreground is a deep blue.

Research dot on the horizon: get rid of scripts!

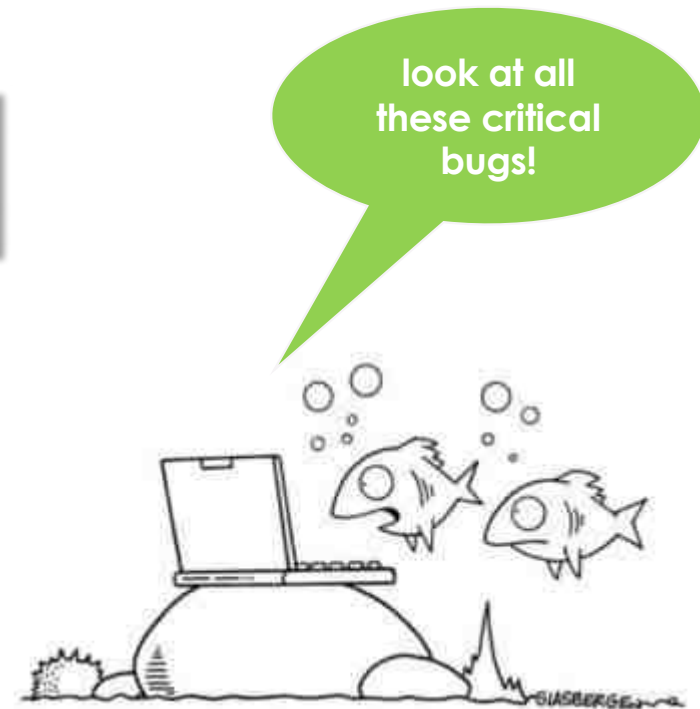
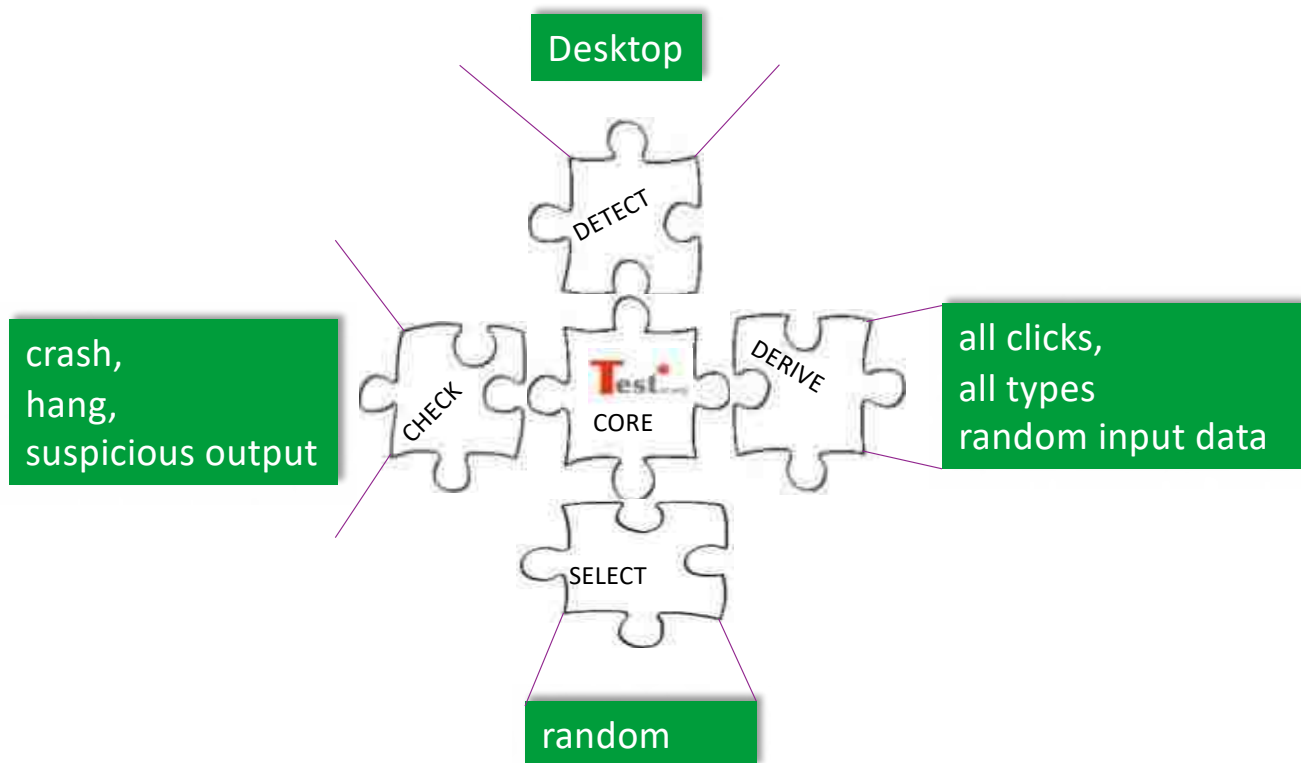
Flip test automation



Modular plug-in based implementation



In 2010....



2010 - 2022

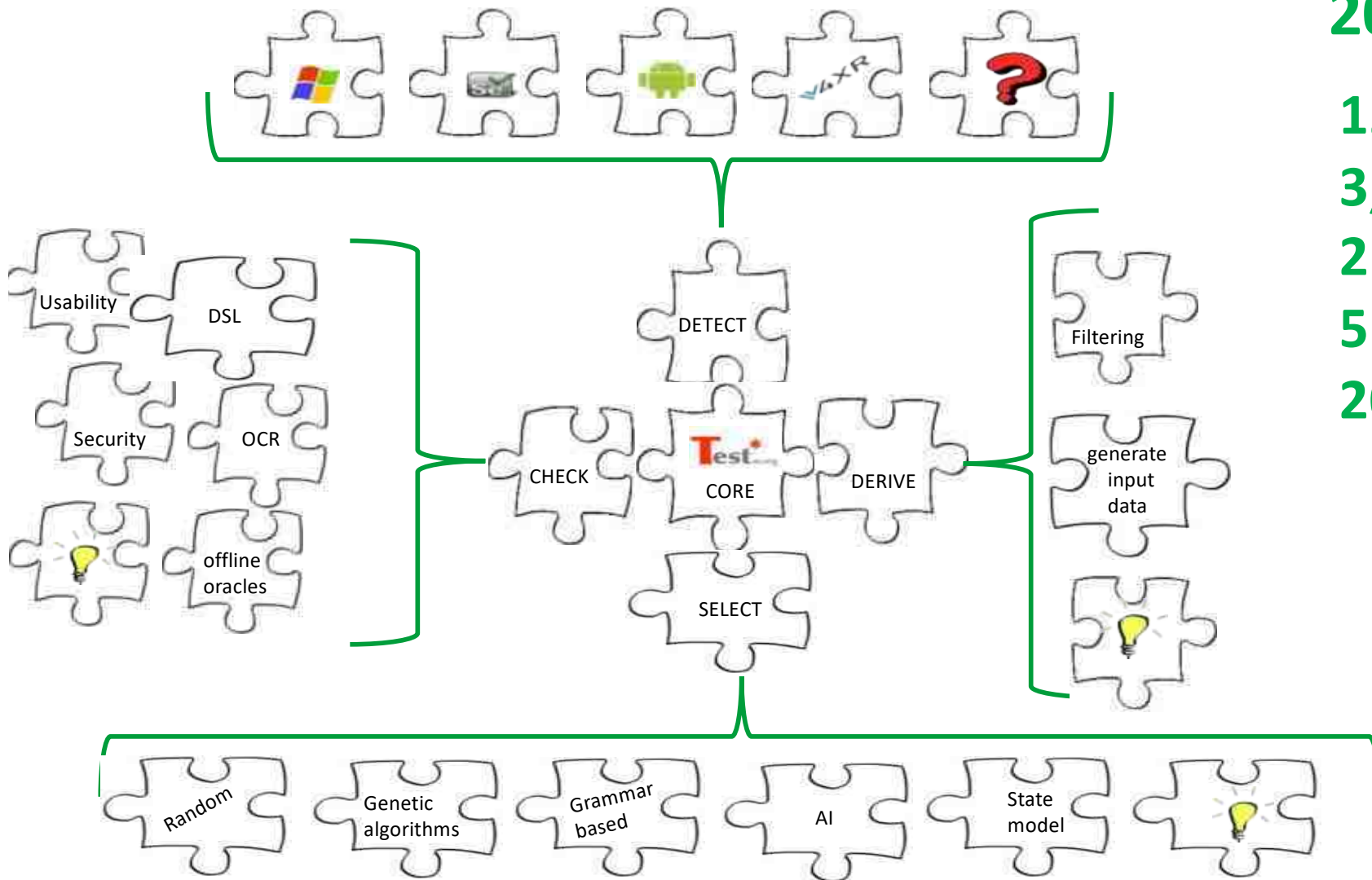
120k LOC

3,7 PhD thesis

21 Ba/MSc thesis

5 EU projects

20 companies



Infer state-models



Layout: State

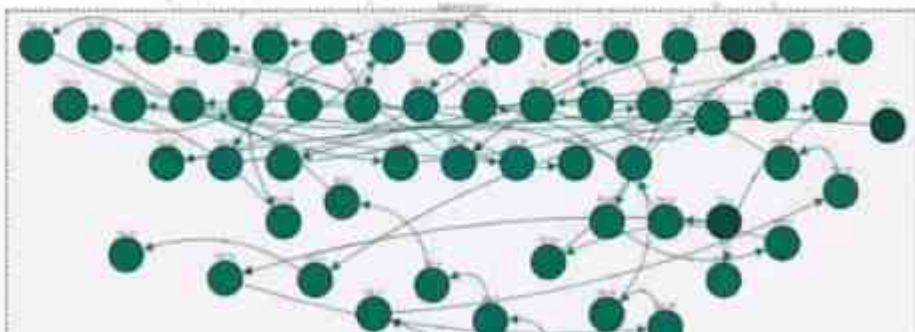
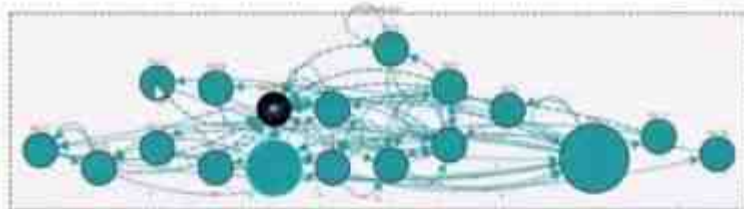
Show labels

Show all nodes

- Show abstract layer
- Show concrete layer
- Show sequence layer
- Show inter-layer edges

Legend:

- Abstract state
- Concrete State
- Sequence Node



Highlight Make readable Close

4	4
2	2

Filter attribute values

Element data:

Attribute name	Attribute value
concreteStateId	{5C1a532c8a42a51845477, 5C1y2a5e48a11556021001}
counter	1
customLabel	AS-E
id	ad13_0
initial	false
modelIdentifier	xalpra.251727495448
parent	AbstractLayer
stateId	5A1hc77da5983100820019
wid	ba9k62a2112085113

Test*

Layout: Flow

Show labels

Show all nodes

Show abstract layer

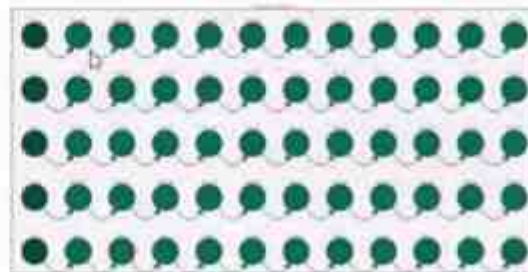
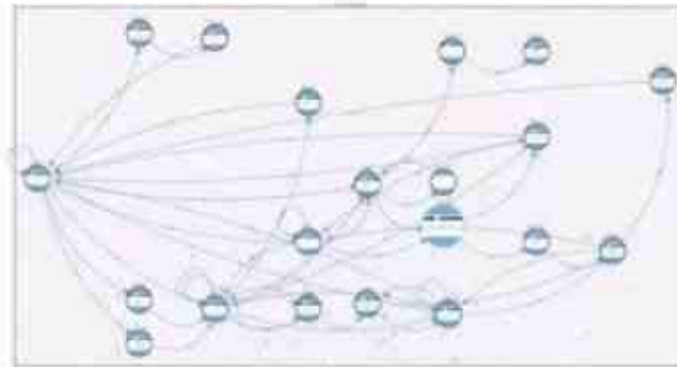
Show concrete layer

Show sequence layer

Show inter-layer edges

Legend:

 Abstract state
 Concrete State
 Sequence Node



View Full Highlight Make visible Inspect element data Close



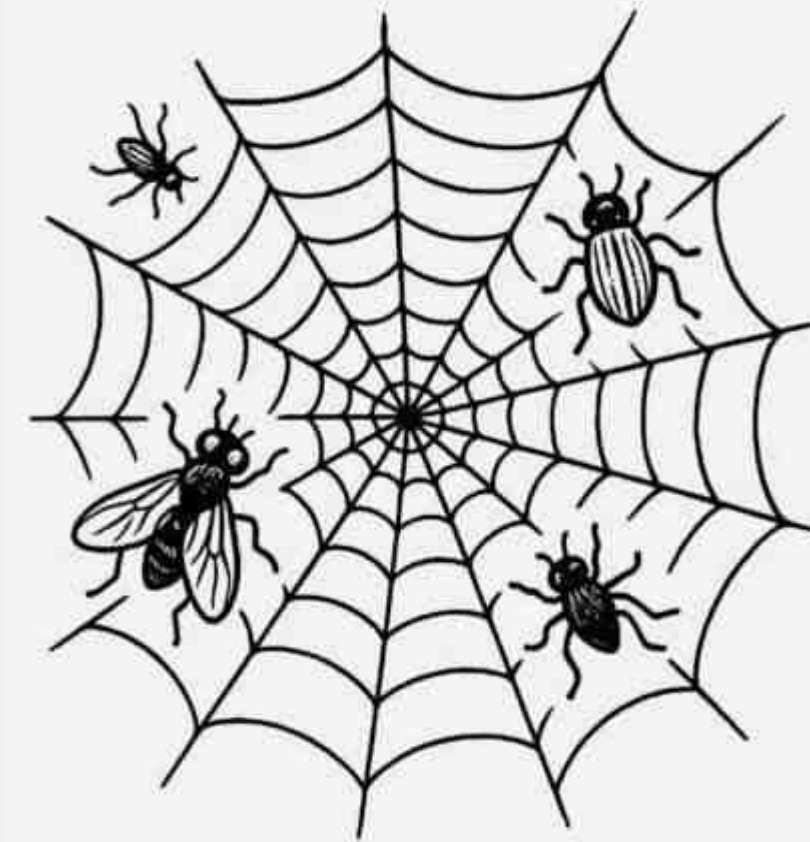
Element data:

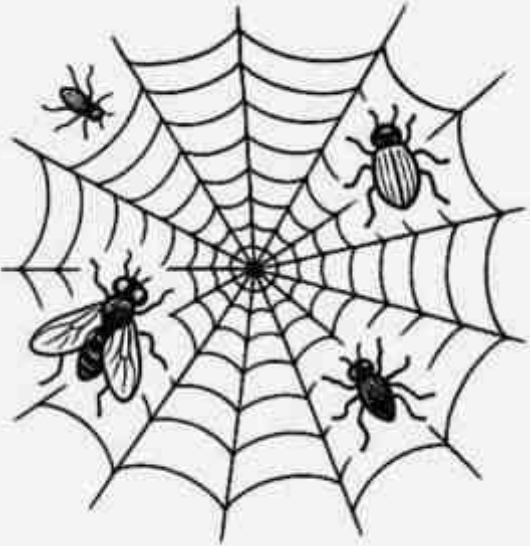
Attribute name	Attribute value
Abstr(R)ID	582aa07c7cde13064191173
Abstr(R_k_T)ID	5T2gph3t093606229513
Abstr(R_k_T_k_P)ID	5P1qiz7501979918752
AbstractID	5Aag0j253b3105190963
Blocked	false

Failure Detection Potential of scriptless oracles

Industrial Bug database
N=2500

 **DigiOffice**



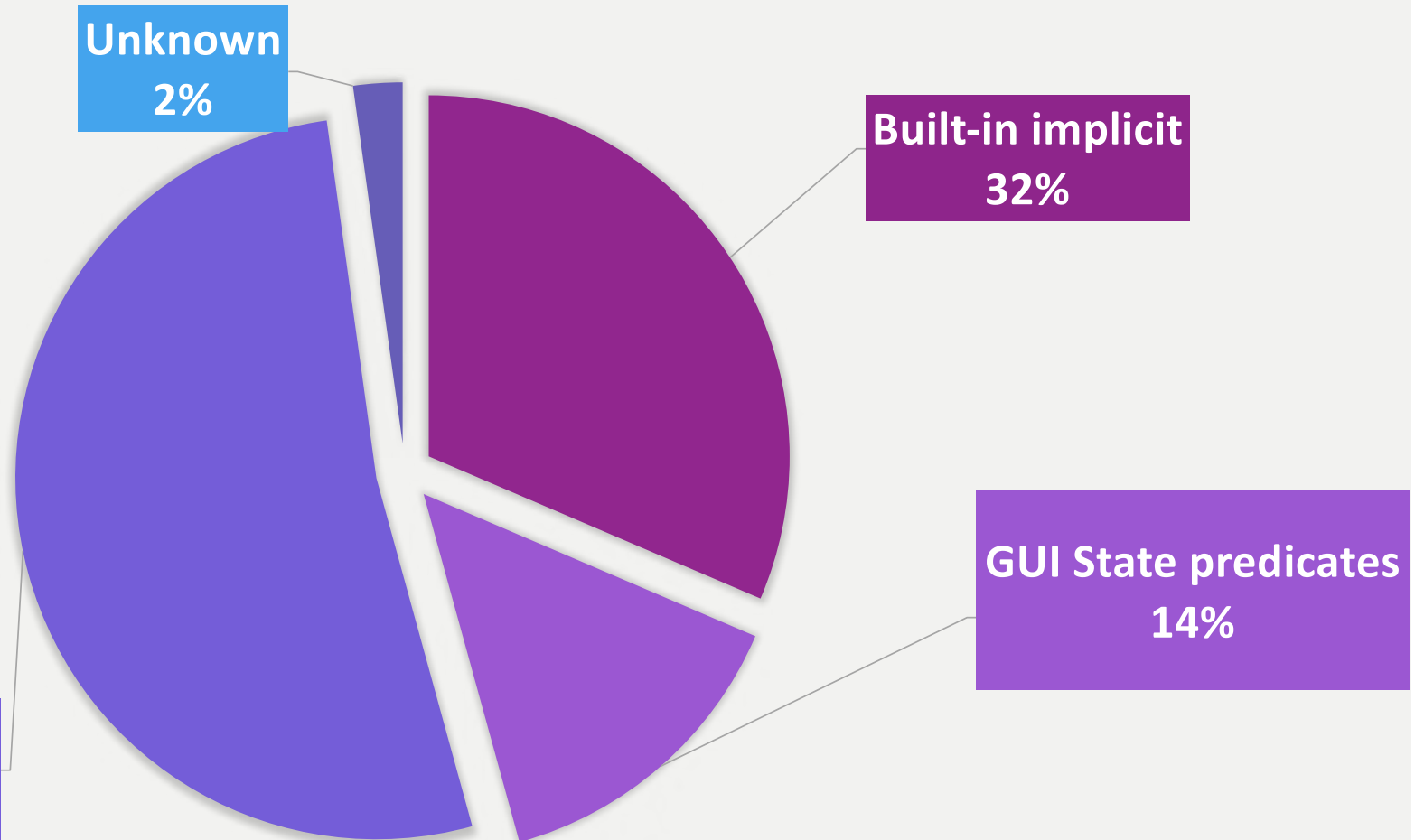


Industrial Bug database
N=2500

 **DigiOffice**

External and domain
52%

Failure Detection Potential of scriptless oracles



Adding oracles like GUI state predicates

```
1 public Verdict widgetImageWithoutAlternativeText(State state) {
2     // Loop through all the widgets of the state
3     for(Widget widget : state) {
4         // Check whether the widget is an image (<img>)
5         // and if it lacks alternative text
6         if(widget.get(Tags.Role, Roles.Widget).equals(WdRoles.WdImg))
7             &&
8             (widget.get(WdTags.WebAlt, null) == null
9             ||
10            widget.get(WdTags.get(WdTags.WebAlt, "").isBlank())) {
11
12            String verdictMsg = String.format("Detected web image
13                ↪ widgets '%s' without alternative text!",
14                ↪ widget.get(WdTags.WebTitle, ""));
15
16            return new
17                ↪ Verdict(Verdict.Severity.WARNING_ACCESSIBILITY,
18                ↪ verdictMsg);
19        }
20    }
21    return Verdict.OK;
22 }
```

Oracle language

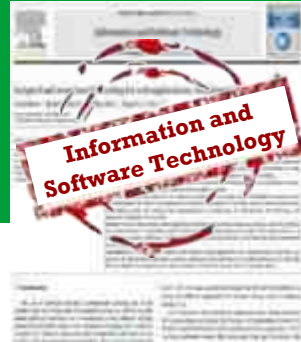


R a s c a l

```
1 public Verdict widgetImageWithoutAlternativeText(State state) {
2   // Loop through all the widgets of the state
3   for(Widget widget : state) {
4     // Check whether the widget is an image (<img>)
5     // and if it lacks alternative text
6     if(widget.get(Tags.Role, Roles.Widget).equals(WdRoles.WdImg))
7       &&
8       (widget.get(WdTags.WebAlt, null) == null
9        ||
10       widget.get(WdTags.get(WdTags.WebAlt, "").isBlank())) {
11
12       String verdictMsg = String.format("Detected web image
13         ↳ widgets '%s' without alternative text!",
14         ↳ widget.get(WdTags.WebTitle, ""));
15
16       return new
17         ↳ Verdict(Verdict.Severity.WARNING_ACCESSIBILITY,
18         ↳ verdictMsg);
19     }
20   }
21   return Verdict.OK;
22 }
```

```
assert image "img" has nonempty label
| | "Image Without Alternative Text".
```

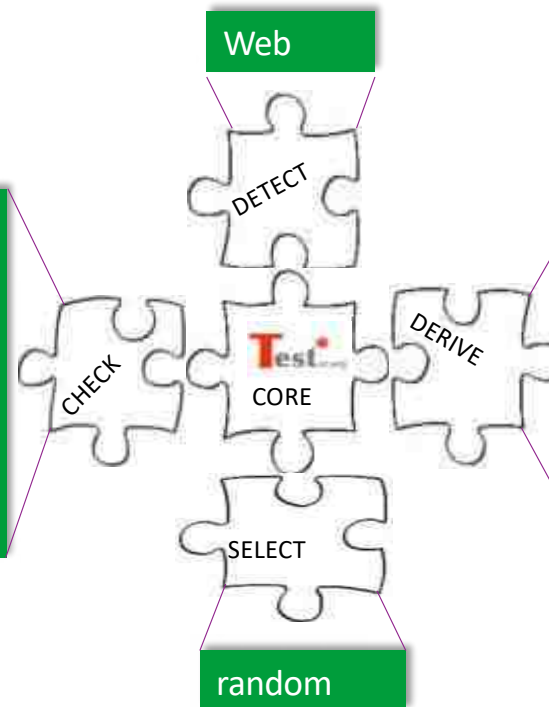
Academia/Industry studies: E-Dynamics + Yoobi Web app



Tool	New bugs	Known bugs
Selenium	2 high	1 high
TESTAR	2 high 4 low	-

added 3 SUT specific oracles for:

- *errors on browser console*
- *specific class selector*
- *fallback text for missing translations*



added rules for:

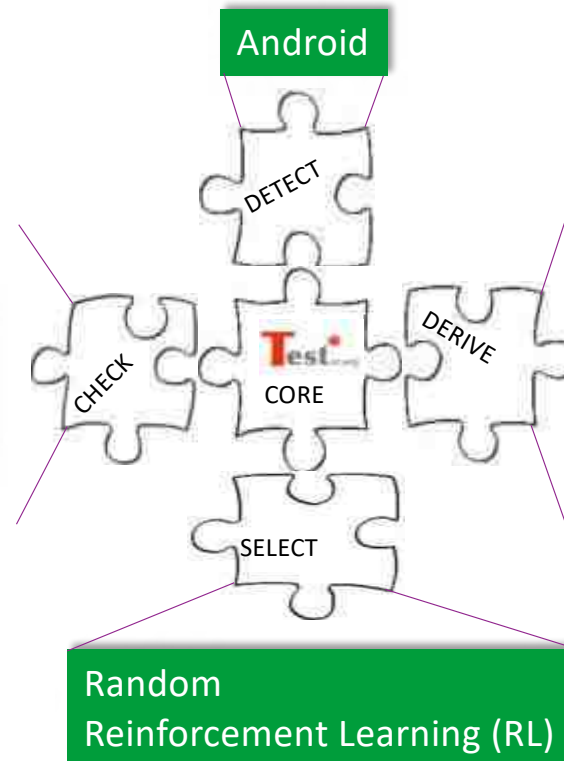
- only SUT allowed domains
- the cookie consent
- the login
- the elements that can be ignored.

Academia/Industry studies: ING + Android app



Tool	% IC	% LC	% MC
Espresso	43,9	43,4	45,9
TESTAR	41,0	40,7	40,8
combined	52,3	52,1	52,3

crash,
hang,
suspicious output



2010 – 2022, still.....

Scripted

Scriptless

Test*

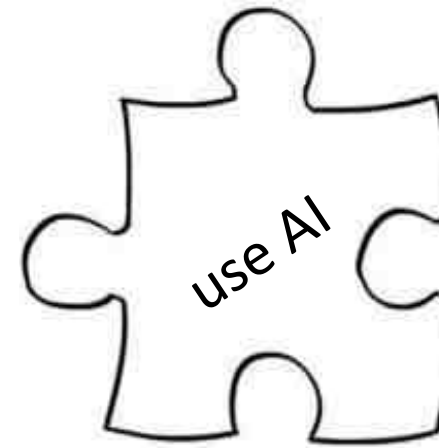
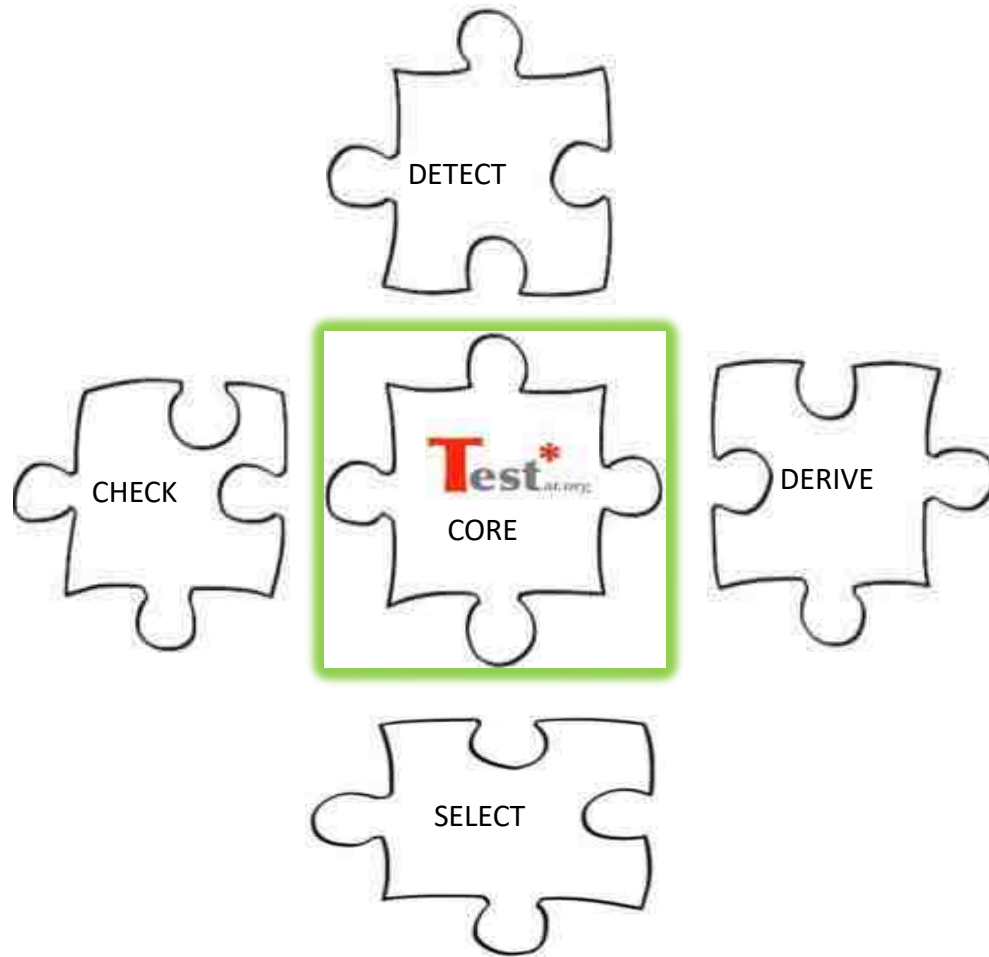


Not for long....

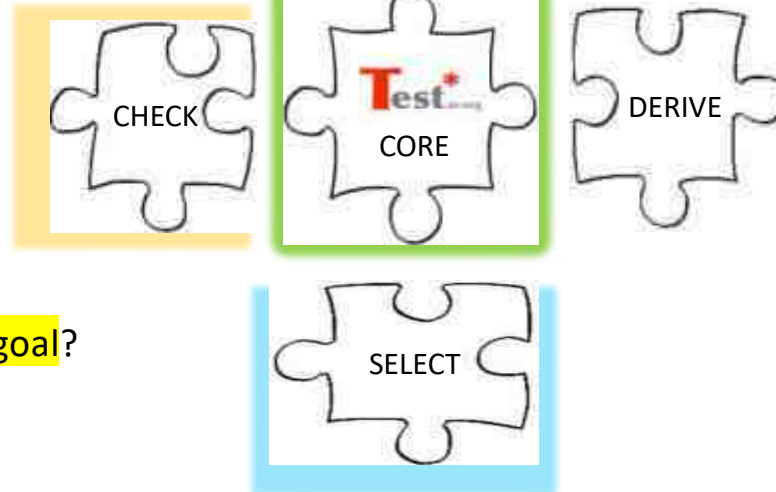
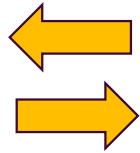
- November 2022
 - launch chatGPT and genAI to general public
 - everything started to become AI-powered
- Just when we started a new project: AUTOLINK
- Funded by NWO
- LINKing requirements and testing



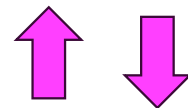
AI-powered scriptless testing



AI-powered scriptless testing



Did we reach the test goal?



What is the best next action to reach the test goal?



The test goals (can be anything!)

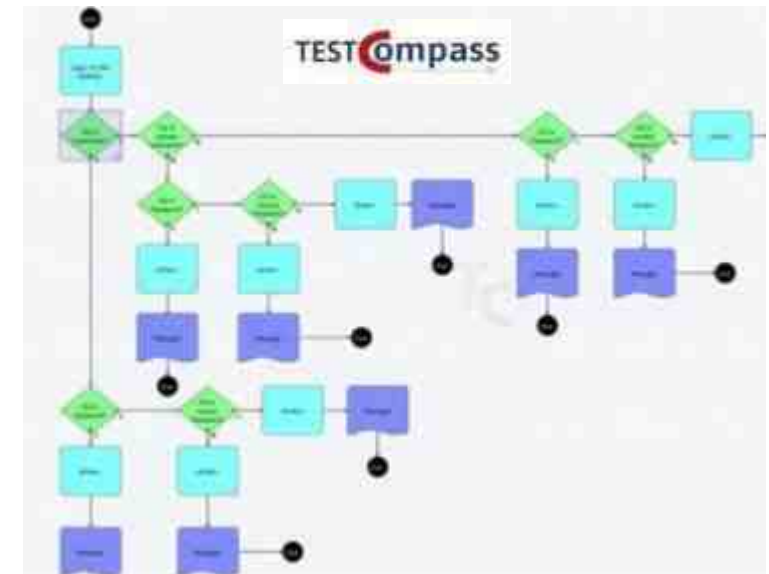
- Natural language
- GWT (Given-When-Then), like Gherkin
- User stories (As a < WHO >, I want < WHAT > so that < WHY >)
- Use cases (Actors, Preconditions, Flow)
- Models

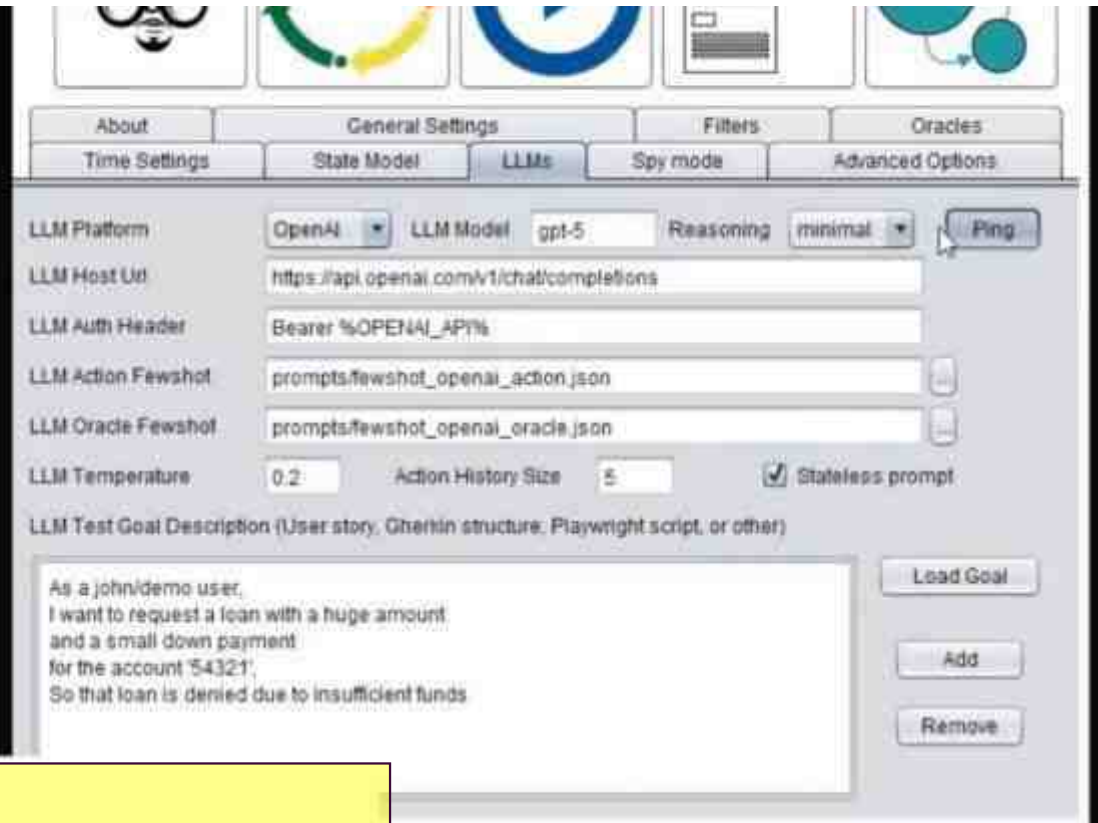
Or re-use your old (broken) scripts!

- Playwright
- Selenium
- Robot's keywords
- etc..

```
with app playwright() as a:  
  a.visit(browser)  
  browser.url = path(browser, launch/headless/100)  
  page = browser.new_page()  
  
  # Step 1: Sign in as user  
  # playwright: fill in form with values  
  page.fill("#username", "john")  
  page.fill("#password", "demo")  
  page.click("#login")  
  
  # Step 2: Click on "Request Loan"  
  page.click("#requestLoan")  
  
  # Step 3: Fill in form with values  
  page.fill("#amount", "999")  
  page.fill("#downPayment", "123")  
  page.click("#applyNow")  
  
  # Step 4: Check if loan is approved  
  page.click("#applyNow")  
  
  # Step 5: Print the result  
  print("Loan approved: ", page.click("#applyNow"))  
  
  # Step 6: Close browser  
  browser.close()
```

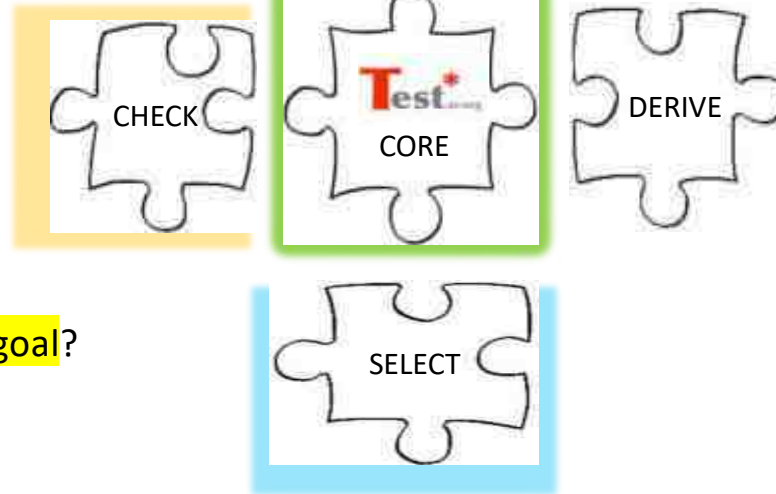
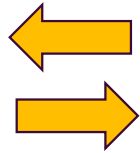
```
Scenario Outline: Request Loan Parabank  
Given I want to log in  
And fill in username 'john'  
And fill in password 'demo'  
When login  
Then Request Loan link is shown  
  
When fill in amount '999'  
And fill in downPayment '123'  
And Apply Now  
Then the loan has been approved.
```



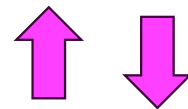


As a john/demo user,
I want to request a loan with a huge amount
and a small down payment
for the account '54321',
So that loan is denied due to insufficient funds

AI-powered scriptless testing



Did we reach the test goal?



What is the best next action to reach the test goal?



```
Users\Fernando\TESTAR\Documents\Github\TESTAR_dev\testar\target\install\testar\bin\testar
starting for transport dt_socket at address: 5005
started java version is: C:\Program Files\Java\jdk-11.0.10
STAR version is: v2.7.15 (3-Nov-2021)
st_settings is: C:\settings\03_wbdriver_11e_parabank\test.settings
```


Test goal in natural language





getting closer

Research dot on the horizon: get rid of scripts!



**What began as random action
selection in 2010 is now goal-driven
intelligent action selection.**

T
*
estar.org

What changes for the tester?

- No more fixing broken scripts
- Know how to write good test goals, not scripts!
- Maintain test goals as the software requirements evolve
- Keep tests relevant, not executable

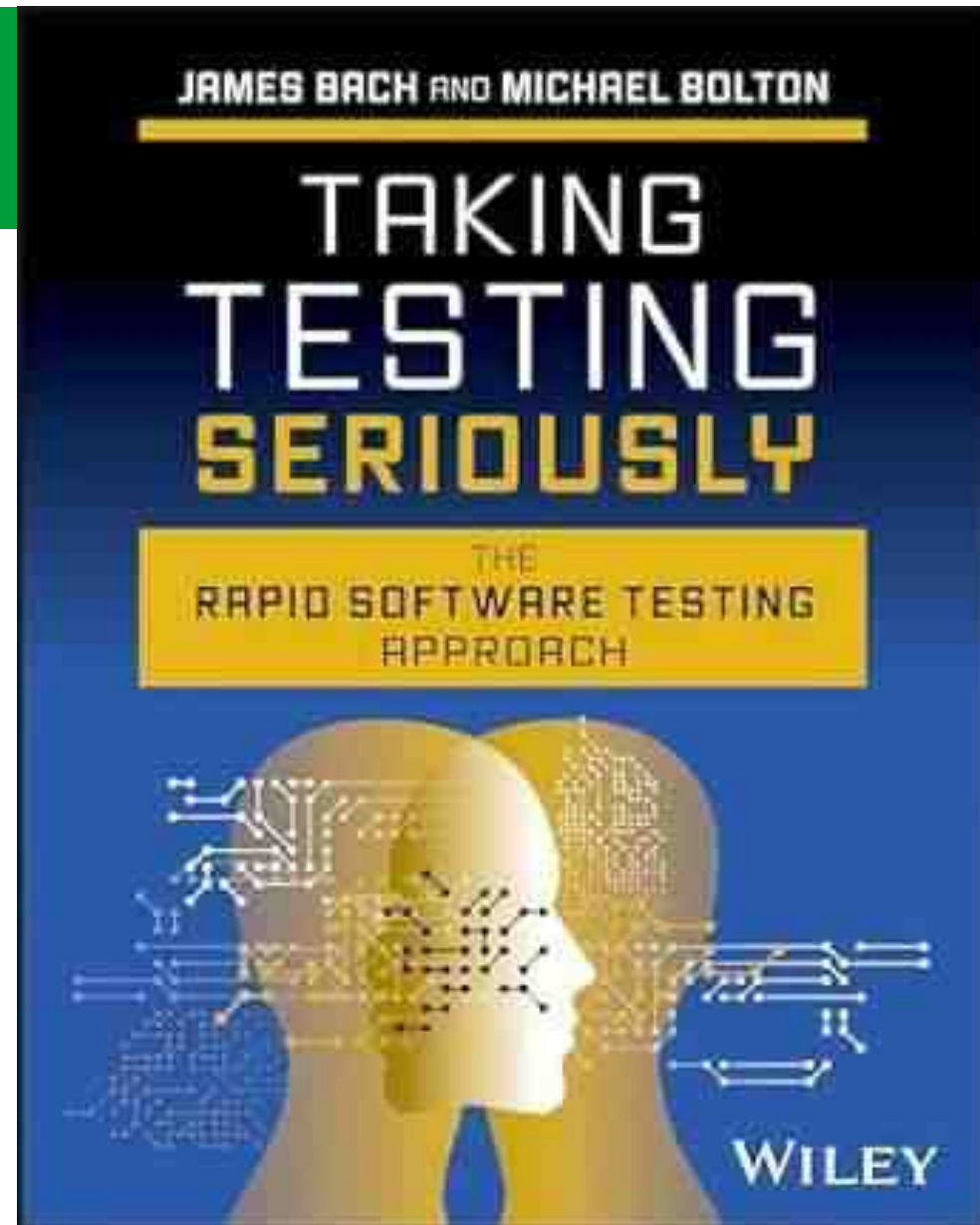
From keeping scripts in sync with a changing GUI
to keeping testing in sync with changing requirements.

(as it should be!)

Finally, people will start.....

The authors of this book, which came out this month, have been saying it for years:

test scripts are not testing!



What changes for companies?

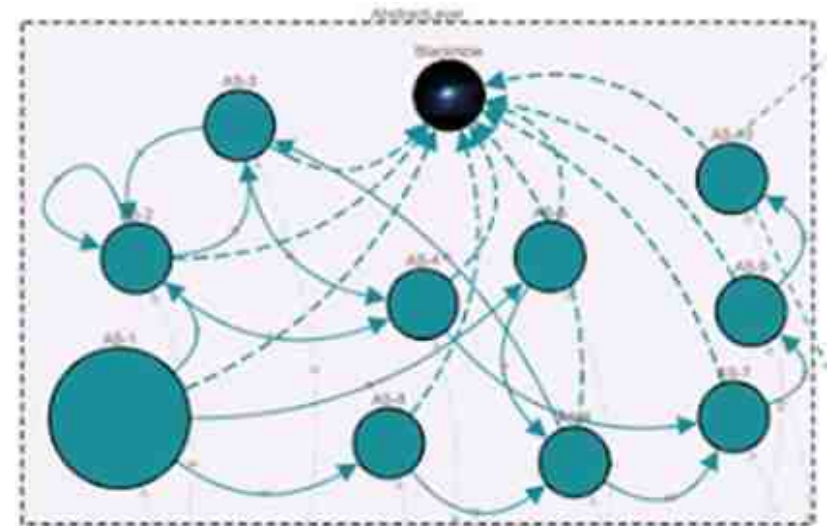
- Stop losing money on maintaining old scripts
- Put testers' time where it adds value: improving quality, not fixing automation.



From maintenance cost to product value!

What changes for our research?

- Agents explore stochastically: every run is different
- Product owners (still) want deterministic evidence for acceptance tests
- Traceability will need to link each test sequence to its goals and requirements



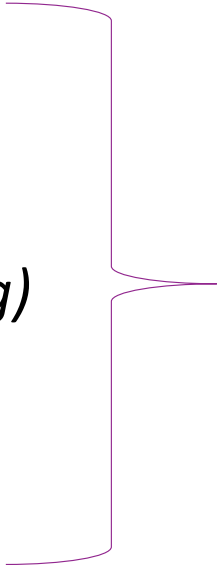
From exploration to explanation



(AUTOLINK)

What changes for the AI we use?

1. **Give it better context** (*lightweight, no retraining*)
2. **Teach it new skills** (*medium effort, partial retraining*)
3. **Build it for your domain (SUT)** (*heavy, specialized*)



About testing,
your SUT,
your users,
your test

From general purpose to domain-specific

TESTAR

No more scripts.
Just goals and serious testing.

Contact:

+34 690 917 971

info@testar.org

<https://testar.org/>

https://github.com/TESTARtool/TESTAR_dev



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Open Universiteit
www.ou.nl