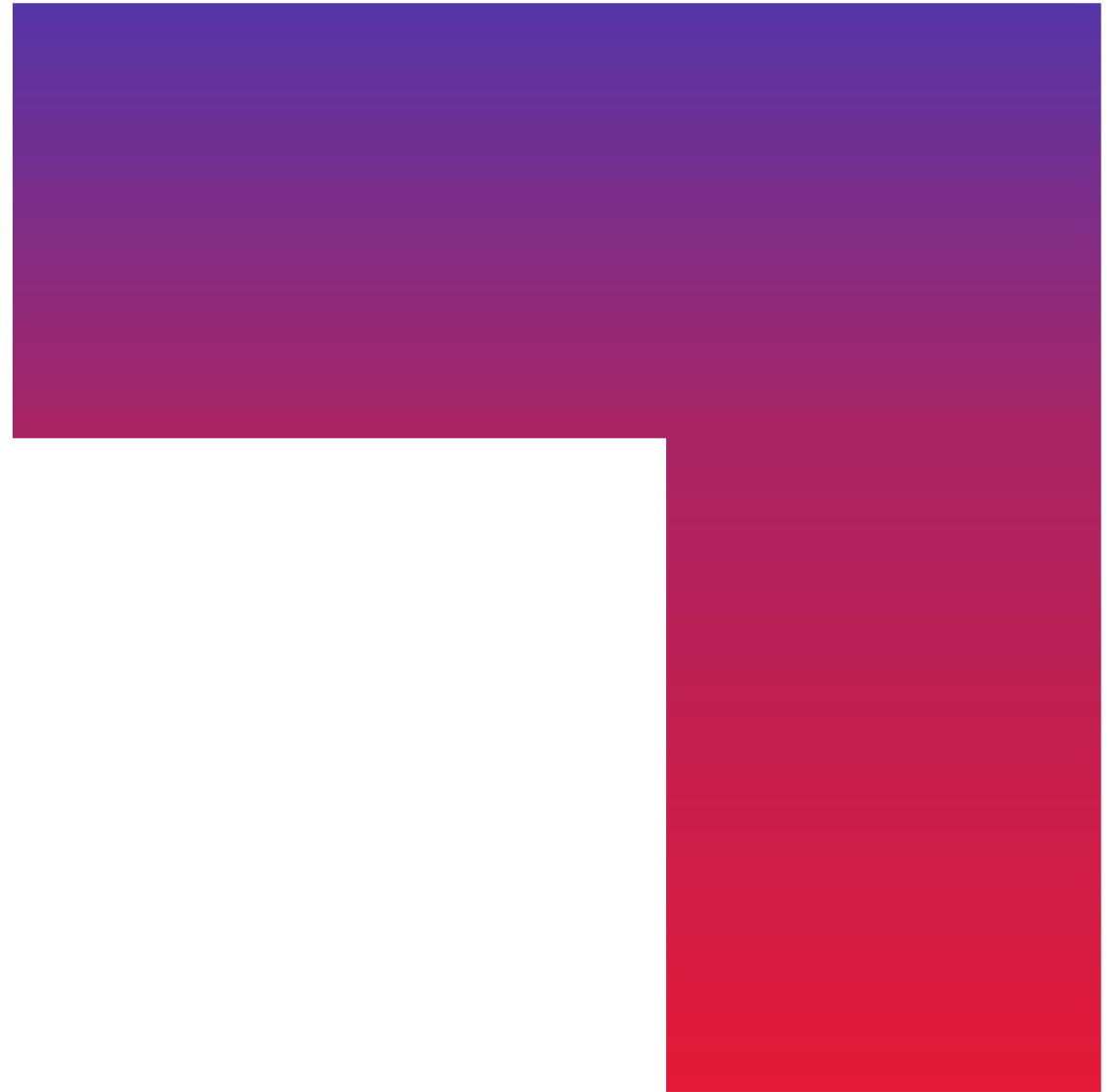


Contemporary exploratory testing foundations

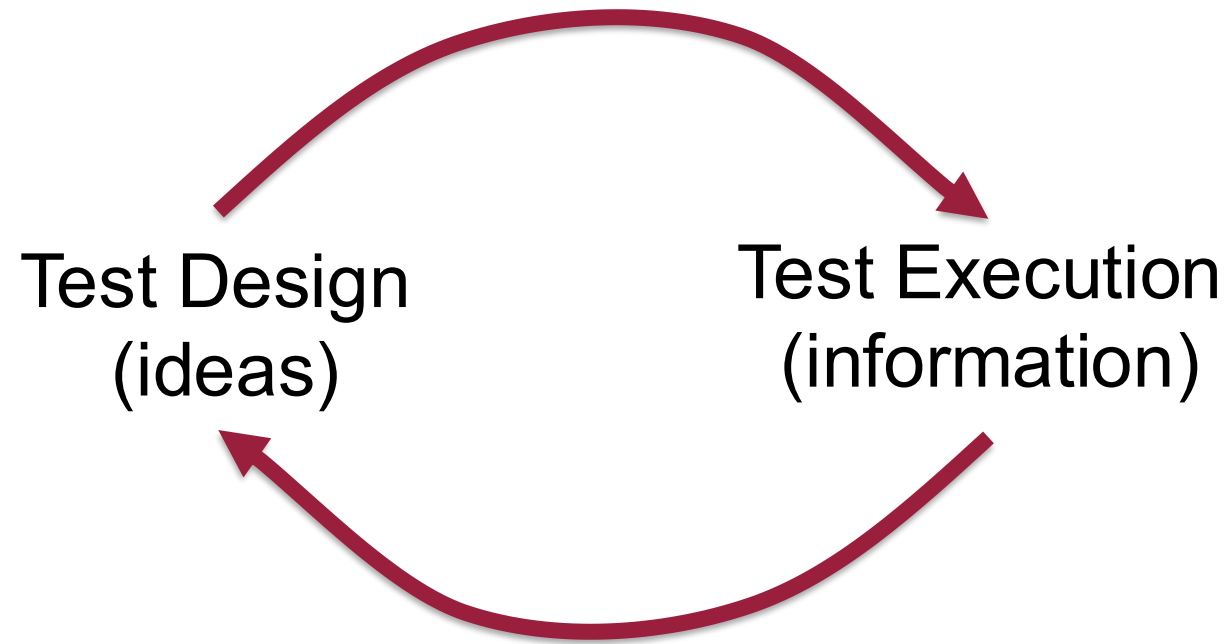
 CC BY-NC-SA 4.0

CGI / Maaret Pyhäjärvi
v. 3.0 (2026-04-22)

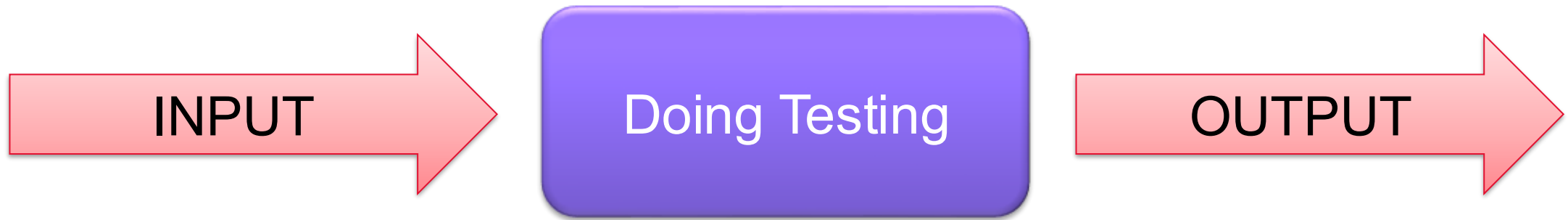


Optimizing the value of testing

Learning



Exploratory testing the verb





Input

Tester

Domain knowledge

Requirements and specifications

Testing knowledge

Miscellaneous knowledge



Output

Better tester
Coverage

Information incl. defects and change requests

Documentation: Strategy

Documentation: Tests

Course outline

- Chapter 1: Test target and our options for exploring
- Chapter 2: Self-management basics on setting yourself constraints
- Chapter 3: The moment of first impression
- Chapter 4: Recognizing and learning a domain
- Chapter 5: Recognizing functionality
- Chapter 6: Recognizing data
- Chapter 7: Recognizing application and execution environment
- Chapter 8: Documenting in a mindmap
- Chapter 9: pytest the very basics
- Chapter 10: Documenting as skeleton test automation
- Chapter 11: Playwright library and CSS selectors on web pages
- Chapter 12: Documenting as executable test automation
- Chapter 13: Why this is not about any specific tool
- Chapter 14: Use of time
- Chapter 15: Coverage
- Chapter 16: Test Strategy
- Chapter 17: Full results and reproducing from customer feedback
- Chapter 18: Closing remarks

The Self-Study
Edition

Course outline

Section 1:	Options for Exploring	
Section II:	Control through Choices	1 3
Section III:	Documenting (with Automation)	
	Extending with Function, Data, Environment and Domain	2
Section IV:	Use of time and coverage	14-17

The Classroom
Edition

Test target and our options for exploring

Chapter 1



eviltester/TestingApp is licensed under the
Apache License 2.0

A permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions

- ✓ Commercial use
- ✓ Modification
- ✓ Distribution
- ✓ Patent use
- ✓ Private use

Limitations

- ✗ Trademark use
- ✗ Liability
- ✗ Warranty

Conditions

- ④ License and copyright notice
- ④ State changes

This test target is from collections of [Alan Richardson, eviltester](#), a brilliant exploratory tester.

E-Primer an e-prime checking tool

Do you want to write without using the verb "to be"?

Do you want to master [e-prime](#)?

Use our online tool to check your writing.

- Word Count:
- Discouraged Words:
- Possible Violations:

Text:

Check For E-Prime

Stop-and-think: options for exploring

What would you do first, and soon after you get started?

List all things that come to your mind about how you could test this. What would you start from? What you would not do?

Options for exploring

Research the Domain

Use test target with a constraint

Self-management basics on setting yourself constraints

Chapter 2

Charters

Charter template

*Elizabeth Zagroba's concise template
adapted from Elizabeth Hendrickson's
template*

- *target*: where you're exploring
- *resources*: what you're using/how you're exploring
- *information*: what question you want to answer

Choose your own constraint

Deliberately excluding perspectives!

Never Be Bored!

Explore with intent

INTENT

Mission

Charter

Other
Charters

Details

LEARNINGS

Stop-and-think: charters, constraints, intent

You're approaching the moment of first impression. How do you want to frame your moment of first impression?

The moment of first impression

Chapter 3

Options expire

Capture First Impression

Borrow someone else's First Impression

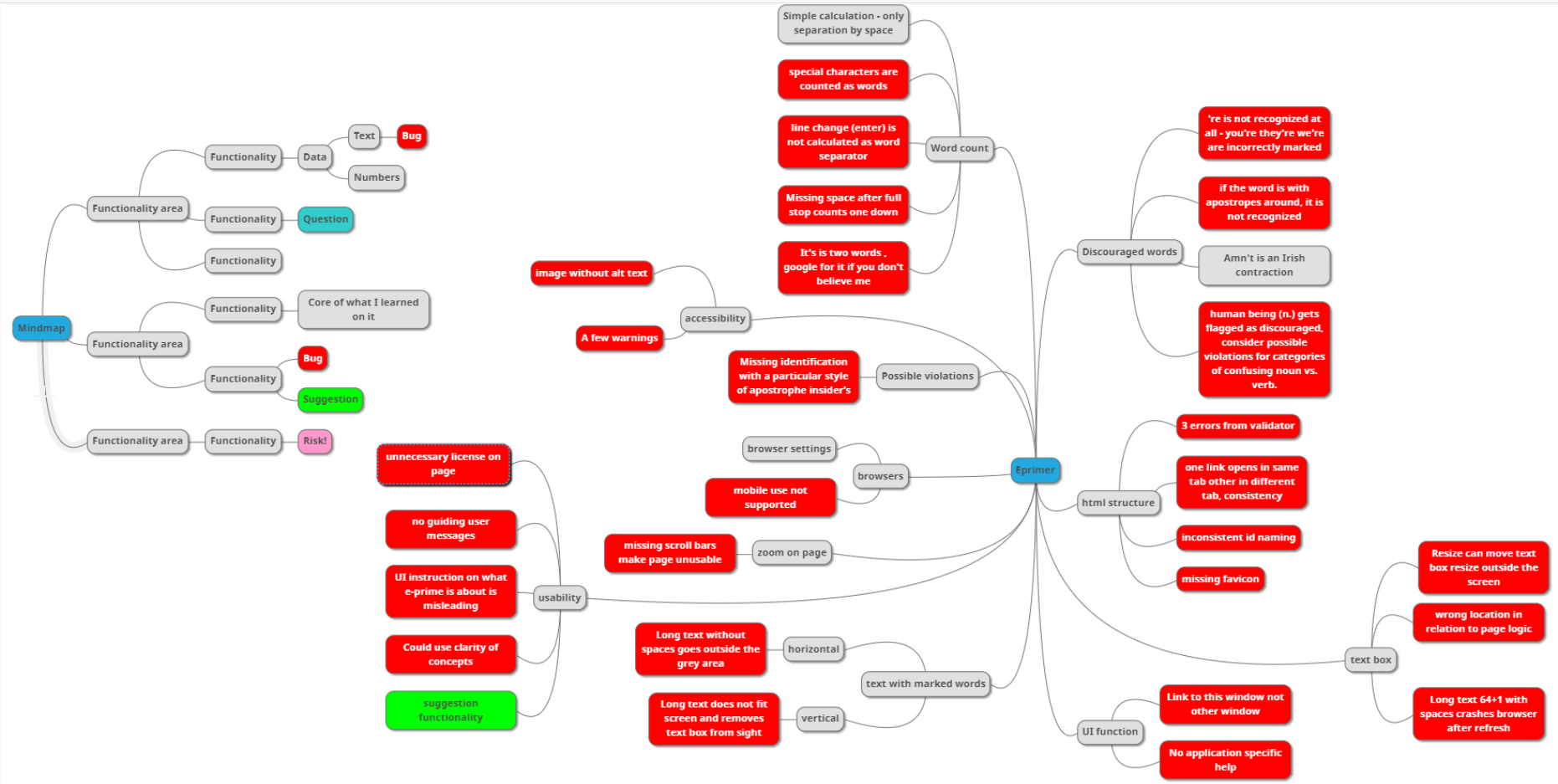
Timing of feedback changes reaction to it!

Let's test

<https://www.exploratorytestingacademy.com/app/>

<https://eviltester.github.io/TestingApp/apps/eprimer/eprimer.html>

Example: test results, red is bug



Bugs are conversation starters

Bug is anything that might bug a user.
You start conversations about defects and change requests

Recognizing and learning a domain

Chapter 4

Conference reference inference

 eviltester/TestingApp is licensed under the Apache License 2.0 A permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.	Permissions ✓ Commercial use ✓ Modification ✓ Distribution ✓ Patent use ✓ Private use	Limitations ✗ Trademark use ✗ Liability ✗ Warranty	Conditions ① License and copyright notice ② State changes
--	---	--	--

This test target is from collections of [Alan Richardson, eviltester](#), a brilliant exploratory tester.

E-Primer an e-prime checking tool

Do you want to write without using the verb "to be"?

Do you want to master [e-prime](#)?

Use our online tool to check your writing.

- Word Count: 9
- Discouraged Words: 3
- Possible Violations: 1

To **be** or not to **be is** Hamlet's dilemma

Text:

To be or not to be is Hamlet's dilemma

Check For E-Prime

```
function inEPrimeOutputFormat(aWord){
    return '<span class="ep_violation">' + aWord + "</span>";
}

function inPossibleEPrimeOutputFormat(aWord){
    return '<span class="ep_warning">' + aWord + "</span>";
}

function isDiscouragedWord(aWord){

    var discouragedWords = new Array();
    discouragedWords['be'] = 'be';
    discouragedWords['being'] = 'being';
    discouragedWords['been'] = 'been';
    discouragedWords['am'] = 'am';
    discouragedWords["isn't"] = "isn't";
    discouragedWords["are"] = "are";
    discouragedWords["aren't"] = "aren't";
    discouragedWords["was"] = "was";
    discouragedWords["wasn't"] = "wasn't";
    discouragedWords["were"] = "were";
    discouragedWords["weren't"] = "weren't";
    discouragedWords["is"] = "is";
    discouragedWords["ain't"] = "ain't";
    discouragedWords["i'm"] = "i'm";
    discouragedWords["amn't"] = "amn't";

    return (discouragedWords[aWord.toLowerCase()]==aWord.toLowerCase());

}
```

Let's test

<https://www.exploratorytestingacademy.com/app/>

<https://eviltester.github.io/TestingApp/apps/eprimer/eprimer.html>

Learning of domain of e- primer

Core Idea	Writing English language avoiding verb “be” in all its forms
Why?	Someone claims it had benefits, intellectual challenge
Examples	Used in sentences Listed examples
Sample texts	The Bible!

Recognizing functionality

Chapter 5

Naming of function

Functions in Code
Expected Features
Visible Features

Let's test

<https://www.exploratorytestingacademy.com/app/>

<https://eviltester.github.io/TestingApp/apps/eprimer/eprimer.html>

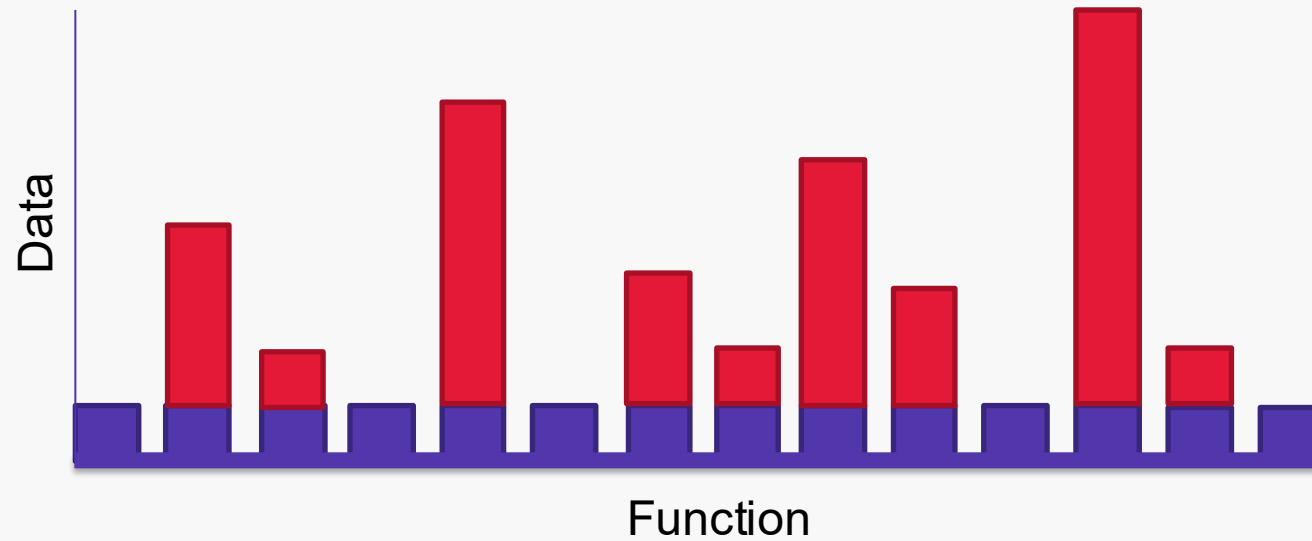
Learning of function of e- primer

Input	Text field and button
Output	Three numbers, text area
Containers	Resizable text field, resizable browser window, page
Presentation	Fonts, text and element sizes, order of functions
Browser	Settings, zoom
Algorithm	Recognizing eprime

Recognizing data

Chapter 6

Data or variables



Versatile data

Lifecycle of Data: Create, Read, Update, Delete
Known problematic inputs: GitHub Naughty Strings

<https://github.com/minimaxir/big-list-of-naughty-strings/blob/master/blns.txt>

Let's test

<https://www.exploratorytestingacademy.com/app/>

<https://eviltester.github.io/TestingApp/apps/eprimer/eprimer.html>

Learning of data of e- primer

Word delimiter	Space, wordcount breaks with characters and line change
Types of apostrophes	Typesetter / typewriter
Long text	Copied / tool generated
Valid eprime	Recognizing right as right
Eprime violations	Recognizing wrong as wrong

Recognizing application and execution environment

Chapter 7

What you coded is a bad constraint



Naomi Wu 机械妖姬
@RealSexyCyborg



You can't say "Signal is secure, it's the OS that's not" if Signal cannot operate without an OS. They are a system—can only be used as a system, they need to be evaluated as a system, and their effectiveness as a system disclosed to customers.

3:58 AM · Jan 16, 2021 · Twitter Web App

Execution environment

Different browsers: web and mobile

Browser functionality and add-ons

HTML standard compatibility

Accessibility standard compatibility

Let's test

<https://www.exploratorytestingacademy.com/app/>

<https://eviltester.github.io/TestingApp/apps/eprimer/eprimer.html>

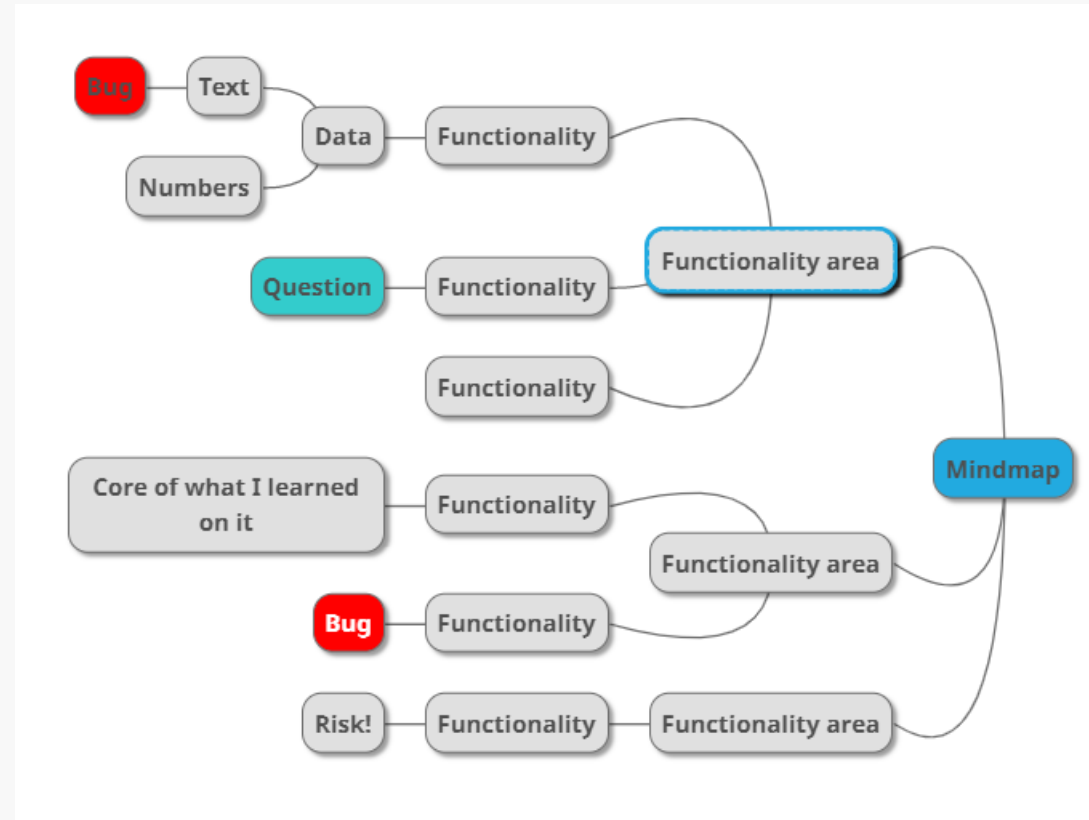
Learning of application and execution environment of e-primer

Browser	Chrome, Brave, ...
Screen size	Web, Mobile
Browser Settings	Zoom, Security, ...
Add-ons	BugMagnet
Validators	HTML, Accessibility, ...

Documenting in a mindmap

Chapter 8

Mindmap



Bug Reports

R eplicate

I solate

M azimize

G eneralize

E xternalize

N eutral tone

Let's test

<https://www.exploratorytestingacademy.com/app/>

<https://eviltester.github.io/TestingApp/apps/eprimer/eprimer.html>

Mindmapping as future reference

Notetaking in the moment
Restructure as you learn
Documentation for the future
General purpose mindmaps

Pytest

the very basics

Chapter 9

Pytest

Popular test runner for python
language

Used both for unit tests and
orchestrated tests

Integrates with libraries in python
ecosystem



Documenting as skeleton test automation

Chapter 10

Document in context of code

```
def this_is_a_test():  
    #first thing to do  
    #second thing to do  
    #third thing to do  
    pass
```

```
===== test session starts =====  
platform win32 -- Python 3.10.2, pytest-7.0.0, pluggy-1.0.0  
rootdir: C:\avimet\avimet-bdd, configfile: pytest.ini  
plugins: anyio-3.5.0, approvaltests-0.2.3, base-url-1.4.2, bdd-5.0.0, playwright-0.2.2  
collected 1 item  
  
test.py . [100%]  
  
===== 1 passed in 0.02s =====  
PS C:\avimet\avimet-bdd\tests>
```

Scenario: **Eprime analysis**

Given the eprime page is displayed

When user analyses sentence to be or not to be

Then user learns sentence has 2 be-verbs, 0 possible be-verbs and total 6 words

* Runnable with pytest-bdd

Let's test

<https://www.exploratorytestingacademy.com/app/>

<https://eviltester.github.io/TestingApp/apps/eprimer/eprimer.html>

Skeleton test automation

Stepwise Test Cases as
Automation Placeholders

Like test cases but version
controlled as code

Handoff to a task that is
decomposing testing differently



Playwright library and css selectors on web page

Chapter 11

Playwright library

Browser driver by Microsoft
Speed – Reliability – Visibility
Automatic waits

```
from playwright.sync_api import Page
```

Library / methods

```
1 import pytest
2 from playwright.sync_api import Page
3
4 def test_example(page: Page):
5     page.goto("https://www.exploratorytestingacademy.com/app/")
6     page.screenshot(path="example.png")
7
```

<https://playwright.dev/python/docs/intro>

css selectors

css=

#id

.class

tag

[attribute='value']

[part_of_attribute_value_contains*='value']

Documenting as executable test automation

Chapter 12

Let's test

<https://www.exploratorytestingacademy.com/app/>

<https://eviltester.github.io/TestingApp/apps/eprimer/eprimer.html>

```
1 from playwright.sync_api import Page
2
3
4 ▶ def test_example(page: Page):
5     page.goto("https://www.exploratorytestingacademy.com/app/")
6     ⚡ page.fill("#inputtext", "To be or not to be - Hamlet's dilemma")
7     page.click("#CheckForEPrimeButton")
8     # it should really be 8, but it's not
9     assert page.inner_text("#wordCount") == "9"
10    assert page.inner_text("#discouragedWordCount") == "2"
11    assert page.inner_text("#possibleViolationCount") == "1"
12
```



Run: Python tests for first_test.test_example x



✓ Tests passed: 1 of 1 test – 3 sec 622 ms

✓ Test Results 3 sec 622 ms

```
C:\pythonProject\eprime-bdd\venv\Scripts
Testing started at 21:56 ...
C:\Users\mapyh\AppData\Local\JetBrains
from distutils import version
Launching pytest with arguments first_
===== test session start =====
collecting ... collected 1 item

first_test.py::test_example[chromium]
```

Tests passed: 1

Run TODO Problems Terminal Python Packages Python Console Services

Tests passed: 1 (a minute ago)

```

1  import pytest
2  from playwright.sync_api import Page
3
4  url = "https://www.exploratorytestingacademy.com/app/"
5
6  def this_is_file(text):...
10
11  @pytest.mark.parametrize('input_text, expect_wordcount, expect_discouraged, expect_violation',
12  [
13      ("To be or not to be - Hamlet's dilemma", 8, 2, 1),
14      ("nothing", 1, 0, 0),
15      ("to be or not to be", 6, 2, 0),
16      ("", 0, 0, 0),
17      ("be, being, been, am, is, isn't, are, aren't, was, wasn't, were, and weren't.", 17, 12, 0),
18      ("\n", 0, 0, 0),
19      ("I'm, you're, we're, they're, he's, she's, it's, there's, here's, where's, how's, what's, who's, aint's, that's.", 30, 4, 11),
20      ("first\nsecond", 2, 0, 0),
21      ("ain't, amn't", 4, 2, 0),
22      ("Hanna's Esa's Meera's Süß's-0'Reggio's or Okechukwu's", 6, 0, 5),
23      ("be'", 1, 1, 0),
24      ("human being", 2, 0, 0),
25      ("typewriter's apostrophe", 2, 0, 1),
26      ("typesetter's apostrophe", 2, 0, 1),
27      (1000 * "x", 1, 0, 0),
28      (this_is_file('sample'), 508, 2, 0),
29      (this_is_file('bible'), 31172, 21, 1),
30      ('cat sat\n', 2, 0, 0)
31  ],
32  ids=['demo', 'not eprime', 'hamlet', 'empty', 'be in forms', 'newline', 'contractions', 'newline with words', 'slang',
33  'possessive', 'quoted be', 'not verb', 'typewriters apostrophe', 'typesetters apostrophe', 'long word', 'file', 'bible', 'ending newline']
34  )
35
36  def test_parametrized_test(page: Page, input_text, expect_wordcount, expect_discouraged, expect_violation):
37      page.goto(url)
38      page.fill("#inputtext", input_text)
39      page.click("#CheckForEPrimeButton")
40      assert page.inner_text("#wordCount") == str(expect_wordcount)
41      assert page.inner_text("#discouragedWordCount") == str(expect_discouraged)
42      assert page.inner_text("#possibleViolationCount") == str(expect_violation)

```

18 tests,
3 browsers




















✖ Tests failed: 10, passed: 8 of 18 tests – 8 sec 470 ms

```
C:\pythonProject\epime-bdd\venv\Scripts\
Testing started at 22:29 ...
```

```
C:\Users\mapyh\AppData\Local\JetBrains\Py
from distutils import version
```

Launching pytest with arguments eprime_t

```
===== test session =====
collecting ... collected 18 items
```

```
eprime_tests.py::test_parametrized_test[d
eprime_tests.py:10] (test_parametrized_tes
9 != 8
```

Expected :8

Actual : 9

```
page = <Page url='https://www.exploratory  
input_text = "To be or not to be - Hamlet  
expect_discouraged = 2, expect_violation
```

```
@pytest.mark.parametrize('input_text',
```

("To be or not to be - Hamlet's d

```
("nothing", 1, 0, 0),
```

10 bugs

Requires pytest-bdd +
refactoring to step-
methods

```
eprime.feature x
1 >> Feature: Eprime text analysis
2     As a user,
3     I want to verify my text for violations of eprime,
4     So I learn to write proper English
5
6 >> Scenario Outline: Eprime samples are correctly analyzed
7     Given the eprime page is displayed
8     When user analyses sentence <sentence>
9     Then user learns sentence has <count_certain> be-verbs, <count_possible> possible be-verbs and total <count_total> words
10
11 Examples:
12 | sentence | count_certain | count_possible | count_total |
13 | to be or not to be - Hamlet's dilemma | 2 | 1 | 8 |
14 | cat is hat | 1 | 0 | 3 |
15 | nothing | 0 | 0 | 1 |
16 | be, being, been, am, is, isn't | 6 | 0 | 7 |
17 | are, aren't, was, wasn't, were, weren't | 6 | 0 | 9 |
18 | I'm, you're, we're, they're, he's, she's | 6 | 0 | 12 |
19 | it's, there's, here's, where's, how's | 5 | 0 | 10 |
20 | what's, who's, aint's, that's | 4 | 0 | 8 |
21
```



```
tests\step_defs\test_eprime.py ..F..FFFFF
```

```
===== short test summary info =====
FAILED tests/step_defs/test_eprime.py::test_eprime_samples_correctly_analyzed[to be or not to be - Hamlet's dilemma-2-1-8] - AssertionError: assert '9' == '8'
FAILED tests/step_defs/test_eprime.py::test_eprime_samples_correctly_analyzed[be, being, been, am, is, isn't-6-0-7] - AssertionError: assert '6' == '7'
FAILED tests/step_defs/test_eprime.py::test_eprime_samples_correctly_analyzed[are, aren't, was, wasn't, were, weren't-6-0-9] - AssertionError: assert '6' == '9'
FAILED tests/step_defs/test_eprime.py::test_eprime_samples_correctly_analyzed[I'm, you're, we're, they're, he's, she's-6-0-12] - AssertionError: assert '6' == '12'
FAILED tests/step_defs/test_eprime.py::test_eprime_samples_correctly_analyzed[it's, there's, here's, where's, how's-5-0-10] - AssertionError: assert '5' == '10'
FAILED tests/step_defs/test_eprime.py::test_eprime_samples_correctly_analyzed[what's, who's, aint's, that's-4-0-8] - AssertionError: assert '4' == '8'
===== 6 failed, 4 passed in 9.43s =====
```

Documenting as executable test automation

Single line

- See it fail
- First test
- Same test but variables
- Same test but templates
- **Failing test with a bug**
- Spec to tests
- Guess the values that are likely to fail
- Multiple browsers
- Runs in CI



Throwaway
automation?

Why this is not about any specific tool

Chapter 13

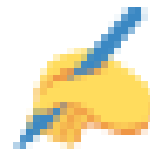
Documentation as a constraint

A Balancing Act between Now
and Future

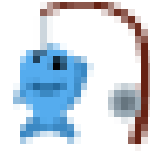
Never be bored is not possible
without automation



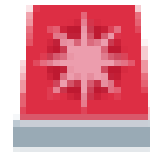
Automation in frame of exploratory testing



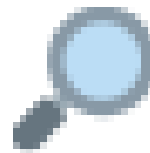
Documenting



Extending reach

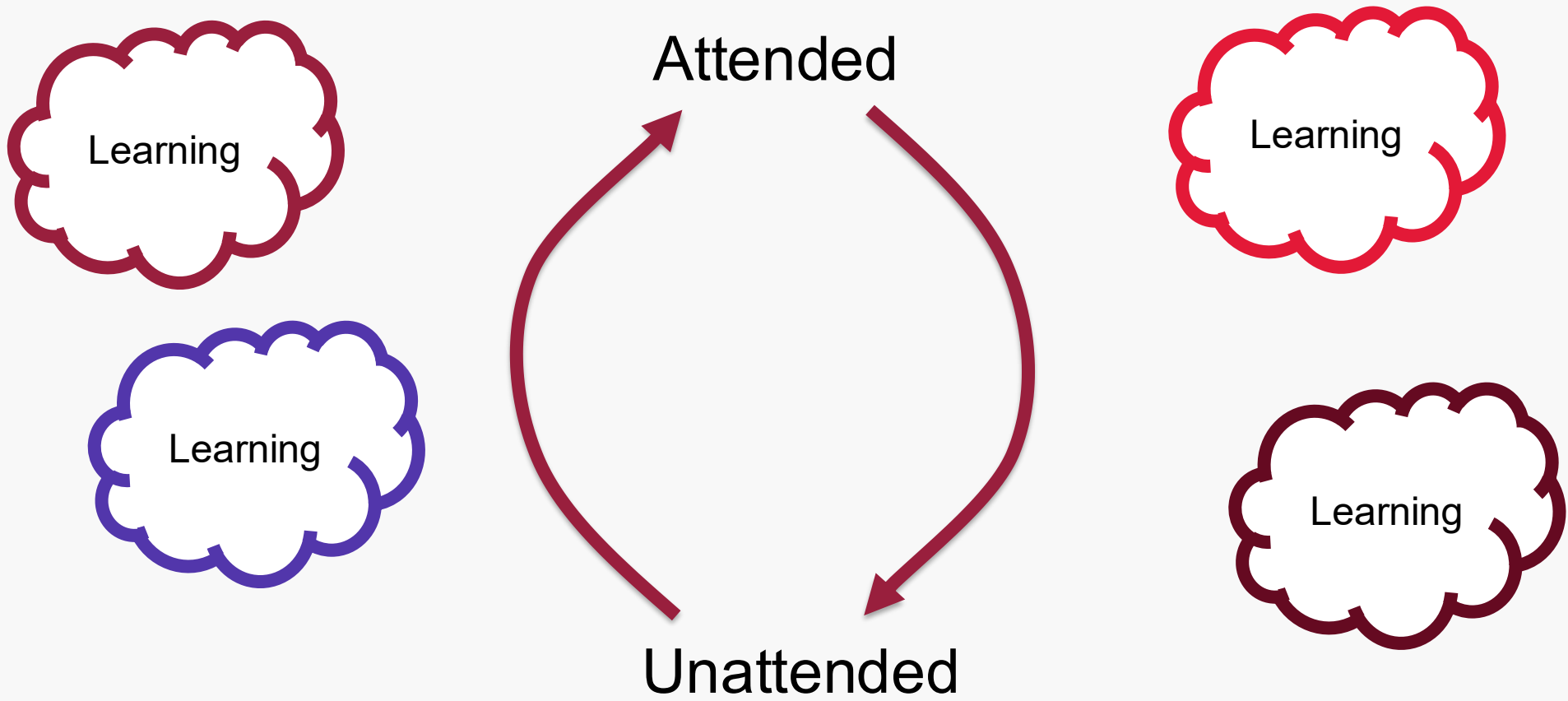


Alerting to attend



Guiding to detail

Moving focus



Stop-and-think: test automation as documentation

How would the testing you did
before this have been different
if you were to start with this?

Use of time

Chapter 14

Test, bug, setup

Software with little bugs is faster to test

Setup is configuring, learning and
documenting

Test grows coverage



eviltester/TestingApp is licensed under the
Apache License 2.0

A permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions

- ✓ Commercial use
- ✓ Modification
- ✓ Distribution
- ✓ Patent use
- ✓ Private use

Limitations

- ✗ Trademark use
- ✗ Liability
- ✗ Warranty

Conditions

- ⓘ License and copyright notice
- ⓘ State changes

This test target is from collections of [Alan Richardson, eviltester](#), a brilliant exploratory tester.

E-Primer an e-prime checking tool

Do you want to write without using the verb "to be"?

Do you want to master [e-prime](#)?

Use our online tool to check your writing.

- Word Count: 9
- Discouraged Words: 2
- Possible Violations: 1

to **be** or not to **be** - **hamlet's** dilemma

Text:

to be or not to be - hamlet's dilemma

Check For E-Prime



Test Cases
trap



Bug trap



Algorithm
trap



Data trap

Stop-and-think: time and traps

Where did your time go on testing of the
application?

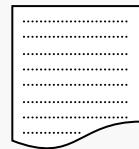
Coverage

Chapter 15

Setting the stage for testing

WHAT
WHEN
WHO
HOW
WHY

We target
these...



Test ideas

...to find
these...



Serious



business



project
(time)



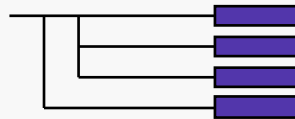
testing (time)



user

Coverage?

...to tell if there's
more and what
level we know
things.



Coverage

REQUIREMENTS
RISKS (of relevant bugs)
CODE
ENVIRONMENTS

Risk coverage

Coverage of relevant bugs
Effectiveness – results of overall strategy
facilitate experience of quality for
stakeholders

Stop-and-think: coverage of today's testing

Would the testing you thought of have
missed any of the bugs we have seen?
What did we not test?

Test strategy

Chapter 16

Ideas that guide test design

Specific to Application Under Test
Risks to ways of testing for them



Let's test

<https://www.exploratorytestingacademy.com/app/>

<https://eviltester.github.io/TestingApp/apps/eprimer/eprimer.html>

Test strategy for e-primer

What is the product?

- E-Primer is an English text validator that checks text against specific rules around avoiding the verb 'to be'. It identifies rule breaking in two categories: one that can be checked by a rule, and another that needs human assessment (for now).

What are the key potential risks?

- It suggest the wrong corrections and misses corrections in realistic text samples
- It miscounts words in a way that leads us to underappreciate the scale of processing.
- It looks wrong on some browsers and data samples
- It requires too much effort to learn in relation to the value of proofreading it provides

How could we test the product so as to evaluate the *actual* risks associated with it?

- Understand the rules of e-prime through research
- Collect data samples (short and long ones) that represent both e-prime text and text that violates rules of e-prime and run them through the program.
- Verify common forms of 'to be' are systematically recognized across the samples
- Document specification as automation that shows the rules of e-prime and enables running subset of all tests across browsers.
- Try fooling word count to count less words or more words by specific data samples
- Run the web page through a set of html-validators
- Visually verify the page with realistic e-prime text samples
- Read the code of the application for inspiration focusing on names of functions rather than understanding implementation
- Summarize learning obstacles for user and value of the application as comparison sheet

Full results and reproducing from customer feedback

Chapter 17

Invisible ink

Customers will tell you of some of the things
you missed

Observing customers (incl. telemetry) will tell
you of some of the things you missed

Limited reporting capability

BUGS:

- ☐ Css validator gives errors
- ☐ Special character as start of line forces an extra line change in grey display box
- ☐ Firefox text field does not clear with ctrl+R
- ☐ html validator identifies 3 errors
- ☐ You're / we're / They're contractions not recognised as violations of e-prime
- ☐ Two words separated by line feed are counted as one
- ☐ Human being is noun but recognised as violation
- ☐ Space is considered only separator for words and special characters are counted as words
- ☐ Long text moves button outside user's access as vertical scroll is disabled
- ☐ id naming is inconsistent, some are camel case, others not
- ☐ Long texts without spaces go outside the grey area reserved for displaying the texts
- ☐ Red/blue on grey has bad contrast
- ☐ Zoom or resize of browser renders page unusable due to missing scroll bars
- ☐ Contractions for word count (I'm) count as two words as per general searchable rules of how word counting works
- ☐ The possible violation's category takes possessives and leaves for human assessment and would probably be expected to be something to create programmatic rules on
- ☐ Possible violations does not handle typesetter's apostrophe, only typewriter's apostrophe in calculation
- ☐ Two part words (like people's last names) in possessive form are not recognised as possible violations
- ☐ Images missing alt text necessary for accessibility
- ☐ Accessibility warnings on contrast
- ☐ Mobile use not supported, styles very non-responsive
- ☐ UI instructions for user are unclear
- ☐ if word is in single quotes, it is not properly recognised as e-prime.
- ☐ text box location in UI is not where user would expect it to be as per the logic of how web pages are usually operating
- ☐ Site is missing favicon and security.txt - both common conventions for web applications
- ☐ Resizing the input text field can move it outside view so that it cannot be resized back
- ☐ Choosing which links are to overload this app and which open new browser window are inconsistent

word + space + enter creates extra vertical space in grey area

'is are' both are not recognised

zoom with and horizontal bar has issues, seen on edge and chrome, on win and Mac with touchpad

Wave accessibility tool does a forced refresh on close **TOOL?**

On Chrome when settings panel scroll does not work. It works on Firefox. **TOOL?**

Enter in end of line moves last word to different line on grey box

Firefox does not come back from mobile use simulation without forced refresh with this site **TOOL?**



34 bugs

Let's test

<https://www.exploratorytestingacademy.com/app/>

<https://eviltester.github.io/TestingApp/apps/eprimer/eprimer.html>

Finding (more of some) relevant conversation starters

You will never find all bugs.
Target finding only things that matter. Learn
what matters.
Seeing and not reporting is better than not
seeing problems.

Closing remarks

Chapter 18

Course outline

- Chapter 1: Test target and our options for exploring
- Chapter 2: Self-management basics on setting yourself constraints
- Chapter 3: The moment of first impression
- Chapter 4: Recognizing and learning a domain
- Chapter 5: Recognizing functionality
- Chapter 6: Recognizing data
- Chapter 7: Recognizing application and execution environment
- Chapter 8: Documenting in a mindmap
- Chapter 9: pytest the very basics
- Chapter 10: Documenting as skeleton test automation
- Chapter 11: Playwright library and CSS selectors on web pages
- Chapter 12: Documenting as executable test automation
- Chapter 13: Why this is not about any specific tool
- Chapter 14: Use of time
- Chapter 15: Coverage
- Chapter 16: Test Strategy
- Chapter 17: Full results and reproducing from customer feedback
- Chapter 18: Closing remarks

Maaret Pyhäjärvi (from Finland)



Email: maaret@iki.fi
Mastodon: [@maaretp@mas.to](https://mas.to/@maaretp)
Web: maaretp.com
Blog: visible-quality.blogspot.fi
(please connect with me through
Mastodon or LinkedIn)



#PayToSpeak #TechVoices
#EnsembleTesting #EnsembleProgramming #StrongStylePairing
#ExploratoryTesting #TestAutomation
#ModernAgile
#ContemporaryExploratoryTesting



<https://exploratorytestingacademy.com>



Ohjelmistotestaus ry



<https://techvoices.org>

