Exploratory Testing Foundations

Maaret Pyhäjärvi

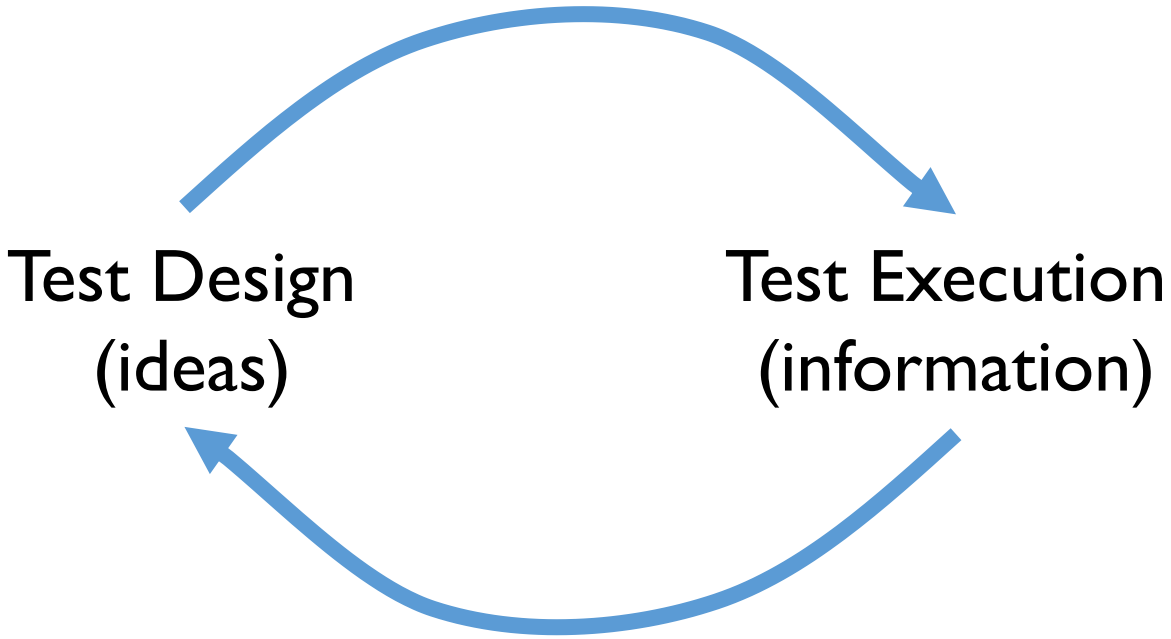
v. 2.0 (2022-02-27)

@maaretp

Optimizing the value of testing

@maaretp

Learning



@maaretp

Exploratory Testing the Verb

@maaretp

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: Closing remarks

Course Outline

Section I: Options for Exploring 1 3

Section II: Control through Choices 2

Section III: Documenting (with Automation)

4-13 Extending with Function, Data, Environment and Domain

Section IV: Use of time and coverage 14-17

Test Target and Our Options for Exploring

Chapter 1

exploratorytestingacademy.com/app/

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	Limitations	Conditions
✓ Commercial use	✗ Trademark use	ⓘ License and copyright notice
✓ Modification	✗ Liability	ⓘ State changes
✓ Distribution	✗ Warranty	
✓ Patent use		
✓ Private use		

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

@maaretp

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

@maaretp

Charters

Charter template

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

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

Choose Your Own Constraint

Deliberately excluding perspectives!
Never Be Bored!

@maaretp

Explore with Intent

INTENT

Mission	Charter
Other Charters	Details

LEARNINGS

@maaretp

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

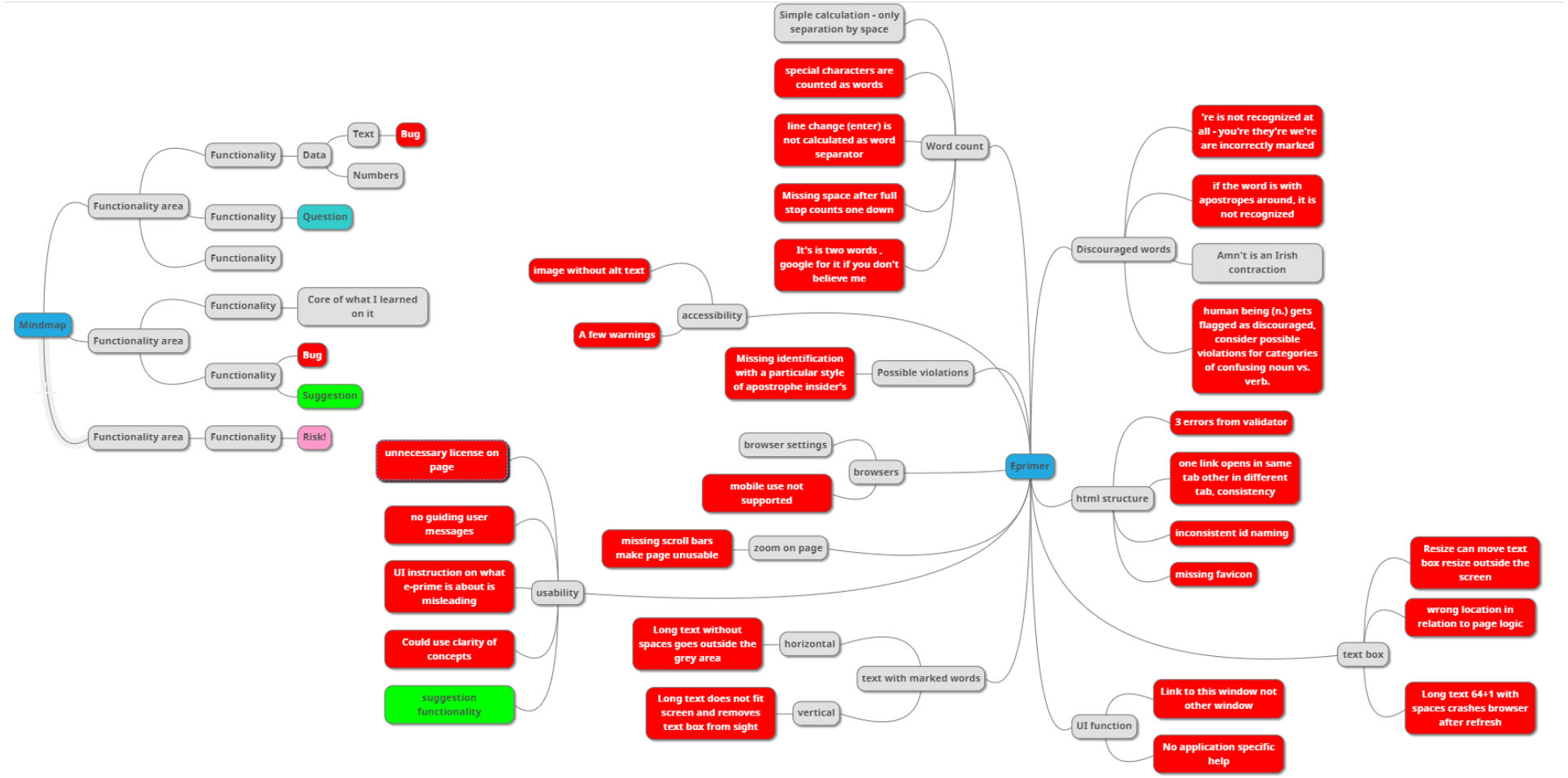
Borrow someone else's First Impression

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



@maaretp

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

@maaretp

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

@maaretp

exploratorytestingacademy.com/app/eprime.js

```
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());

}
```

@maaretp

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

@maaretp

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

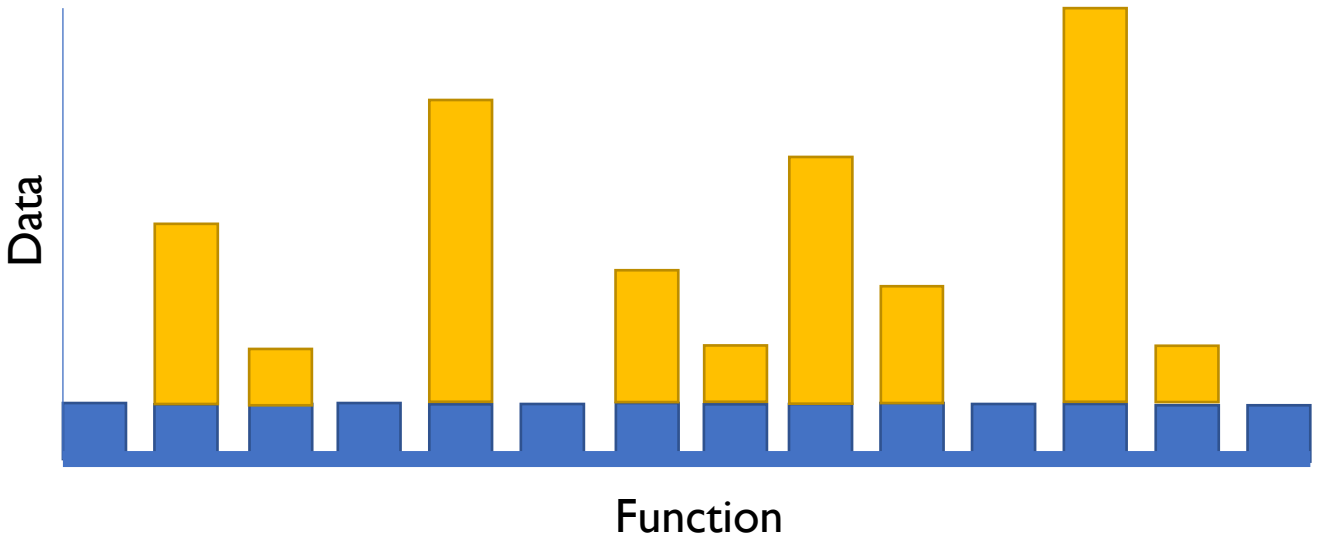
Learning of Function of E-Primer

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



@maaretp

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

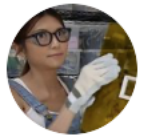
Learning of Data of E-Primer

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

@maaretp

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

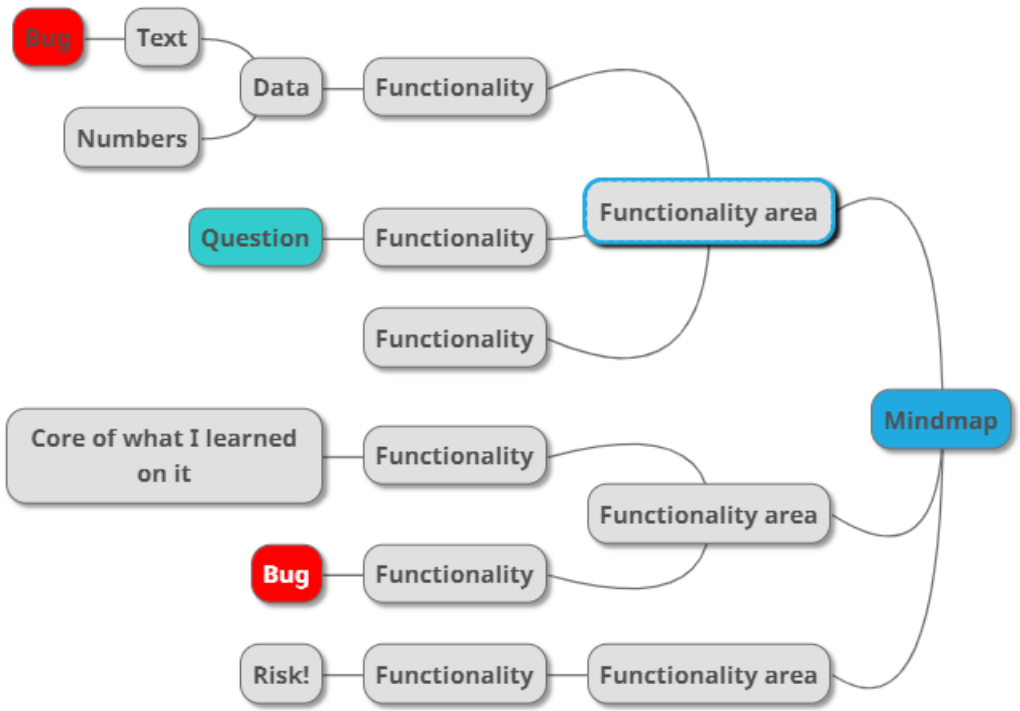
Learning of Application and Execution Environment of E-Primer

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

Documenting in a Mindmap

Chapter 8

Mindmap



@maaretp

Cem Kaner. Bug Reporting Heuristic.

Bug Reports

- **R**eplicate
- **I**solate
- **M**azimize
- **G**eneralize
- **E**xternalize
- **N**eutral tone

@maaretp

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

```python
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

@maaretp

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

```python
import pytest
from playwright.sync_api import Page

def test_example(page: Page):
    page.goto("https://www.exploratorytestingacademy.com/app/")
    page.screenshot(path="example.png")
```

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

first_test.py

```python
from playwright.sync_api import Page


def test_example(page: Page):
    page.goto("https://www.exploratorytestingacademy.com/app/")
    page.fill("#inputtext", "To be or not to be - Hamlet's dilemma")
    page.click("#CheckForEPrimeButton")
    # it should really be 8, but it's not
    assert page.inner_text("#wordCount") == "9"
    assert page.inner_text("#discouragedWordCount") == "2"
    assert page.inner_text("#possibleViolationCount") == "1"
```

Run: Python tests for first_test.test_example

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

Test Results 3 sec 622 ms

```
C:\pythonProject\eprime-bdd\venv\Scrip
Testing started at 21:56 ...
C:\Users\mapyh\AppData\Local\JetBrains
  from distutils import version
Launching pytest with arguments first_

============================= test ses
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)

eprime_tests.py

```python
import pytest
from playwright.sync_api import Page

url = "https://www.exploratorytestingacademy.com/app/"

def this_is_file(text):...

@pytest.mark.parametrize('input_text, expect_wordcount, expect_discouraged, expect_violation',
[
    ("To be or not to be - Hamlet's dilemma", 8, 2, 1),
    ("nothing", 1, 0, 0),
    ("to be or not to be", 6, 2, 0),
    ("", 0, 0, 0),
    ("be, being, been, am, is, isn't, are, aren't, was, wasn't, were, and weren't.", 17, 12, 0),
    ("\n", 0, 0, 0),
    ("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),
    ("first\nsecond", 2, 0, 0),
    ("ain't, amn't", 4, 2, 0),
    ("Hanna's Esa's Meera's Süëss-O'Reggio's or Okechukwu's", 6, 0, 5),
    ("'be'", 1, 1, 0),
    ("human being", 2, 0, 0),
    ("typewriter's apostrophe", 2, 0, 1),
    ("typesetter's apostrophe", 2, 0, 1),
    (1000 * "x", 1, 0, 0),
    (this_is_file('sample'), 508, 2, 0),
    (this_is_file('bible'), 31172, 21, 1),
    ('cat sat\n', 2, 0, 0)
],
ids=['demo', 'not eprime', 'hamlet', 'empty', 'be in forms', 'newline', 'contractions', 'newline with words', 'slang',
'possessive', 'quoted be', 'not verb', 'typewriters apostrophe', 'typesetters apostrophe', 'long word', 'file', 'bible', 'ending newline']
)

def test_parametrized_test(page: Page, input_text, expect_wordcount, expect_discouraged, expect_violation):
    page.goto(url)
    page.fill("#inputtext", input_text)
    page.click("#CheckForEPrimeButton")
    assert page.inner_text("#wordCount") == str(expect_wordcount)
    assert page.inner_text("#discouragedWordCount") == str(expect_discouraged)
    assert page.inner_text("#possibleViolationCount") == str(expect_violation)
```

18 tests,
3 browsers

Run: Python tests for eprime_tests.test_parametrized_test

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

Test Results	8 sec 470 ms
eprime_tests	8 sec 470 ms
test_parametrized_test	8 sec 470 ms
(chromium-demo)	1 sec 126 ms
(chromium-not eprime)	391 ms
(chromium-hamlet)	496 ms
(chromium-empty)	406 ms
(chromium-be in forms)	417 ms
(chromium-newline)	495 ms
(chromium-contractions)	347 ms
(chromium-newline with words)	306 ms
(chromium-slang)	576 ms
(chromium-possessive)	337 ms
(chromium-quoted be)	393 ms
(chromium-not verb)	394 ms
(chromium-typewriters apostrophe)	434 ms
(chromium-typesetters apostrophe)	348 ms
(chromium-long word)	528 ms
(chromium-file)	318 ms
(chromium-bible)	618 ms
(chromium-ending newline)	540 ms

```
C:\pythonProject\eprime-bdd\venv\Scripts\
Testing started at 22:29 ...
C:\Users\mapyh\AppData\Local\JetBrains\Py
  from distutils import version
Launching pytest with arguments eprime_te

============================= test sessio
collecting ... collected 18 items

eprime_tests.py::test_parametrized_test[c
eprime_tests.py:10 (test_parametrized_tes
9 != 8

Expected :8
Actual   :9
<Click to see difference>

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

10 bugs

eprime.feature

```
Feature: Eprime text analysis
    As a user,
    I want to verify my text for violations of eprime,
    So I learn to write proper English

    Scenario Outline: Eprime samples are correctly analyzed
        Given the eprime page is displayed
        When user analyses sentence <sentence>
        Then user learns sentence has <count_certain> be-verbs, <count_possible> possible be-verbs and total <count_total> words

        Examples:
        | sentence                                | count_certain | count_possible | count_total |
        | to be or not to be - Hamlet's dilemma   | 2             | 1              | 8           |
        | cat is hat                              | 1             | 0              | 3           |
        | nothing                                 | 0             | 0              | 1           |
        | be, being, been, am, is, isn't          | 6             | 0              | 7           |
        | are, aren't, was, wasn't, were, weren't | 6             | 0              | 9           |
        | I'm, you're, we're, they're, he's, she's | 6             | 0              | 12          |
        | it's, there's, here's, where's, how's   | 5             | 0              | 10          |
        | what's, who's, aint's, that's           | 4             | 0              | 8           |
```

Requires pytest-bdd + refactoring to step-methods

Documenting as Executable Test Automation

Throwaway 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

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

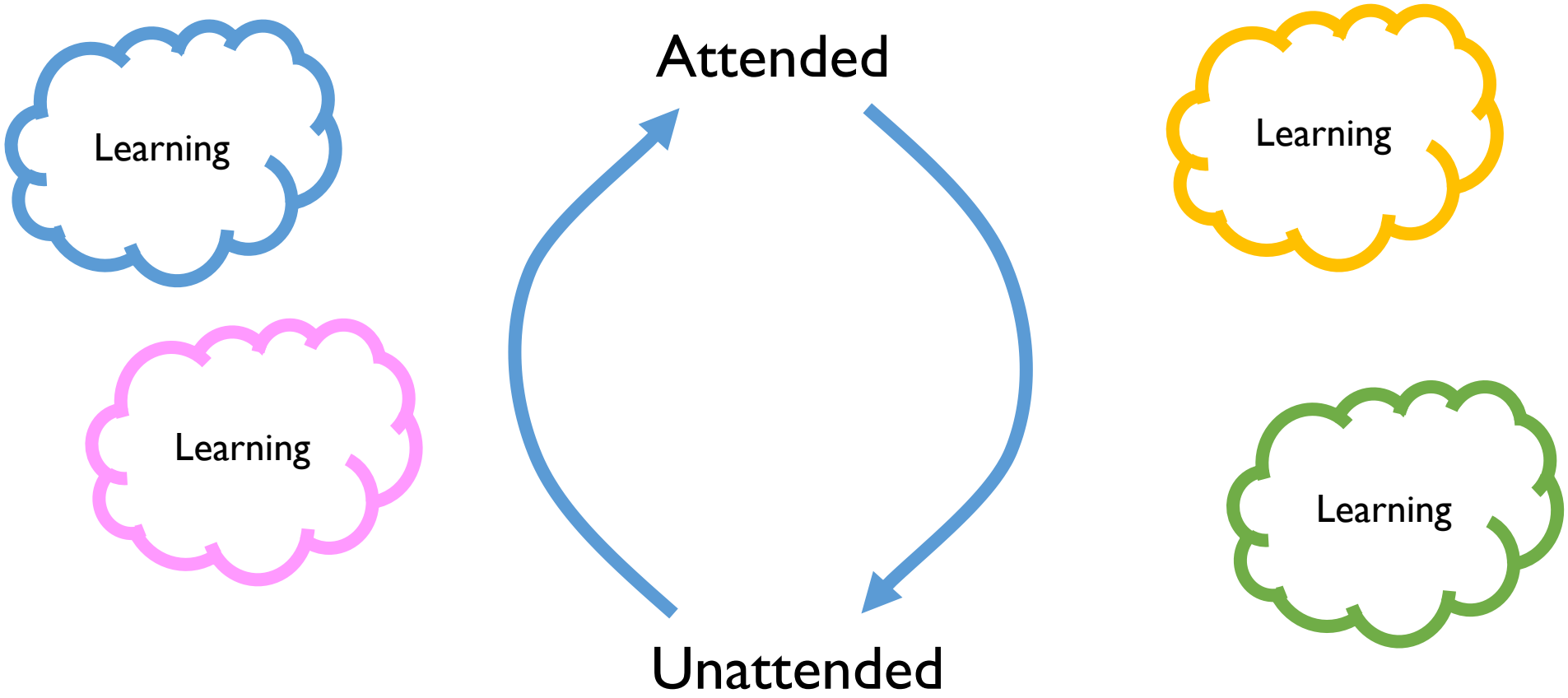
@maaretp

Automation in Frame of Exploratory Testing

✍️ Documenting
🎣 Extending reach
🚨 Alerting to attend
🔎 Guiding to detail

@maaretp

Moving Focus



@maaretp

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

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

@maaretp

Stop-and-Think: Time and Traps

Where did your time go on testing of the application?

Coverage

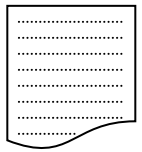
Chapter 15

Setting the Stage for Testing

WHAT
WHEN
WHO
HOW
WHY

We target these…

…to find these…



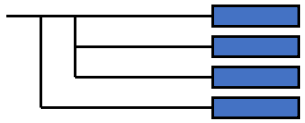
Test ideas

Serious

- B business
- PM project (time)
- T testing (time)
- U user

Coverage?

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



Coverage

REQUIREMENTS
RISKS (of relevant bugs)
CODE
ENVIRONMENTS

@maaretp

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

@maaretp

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

Closing Remarks

Chapter 17

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: Closing remarks

Maaret Pyhäjärvi (from Finland)

EuroSTAR
TESTING EXCELLENCE AWARD

2020

AGILE TESTING DAYS

MIATPP

Most Influential Agile Testing Professional Person

2016

tivi
100 ICT-influencers in Finland

2019, 2020, 2021

EXPLORATORY Testing

https://exploratorytestingacademy.com

Ohjelmistotestaus ry

TECH VOICES

https://techvoices.org

Email: maaret@iki.fi
Twitter: @maaretp
Web: maaretp.com
Blog: visible-quality.blogspot.fi

(please connect with me through Twitter or LinkedIn)

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

@maaretp