

EXPLORATORY TESTING

10.04.2024 OSLO

YULAI DE MEER FJELD



AGENDA

why?

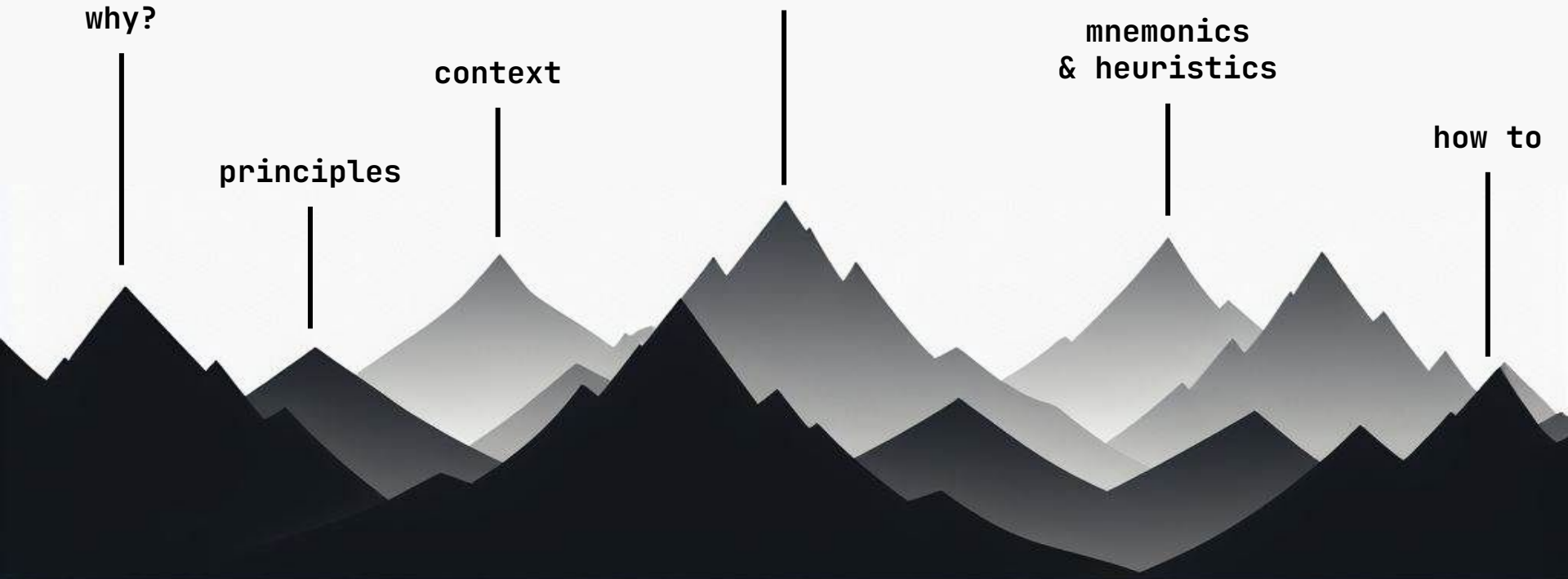
principles

context

exploratory
testing

mnemonics
& heuristics

how to

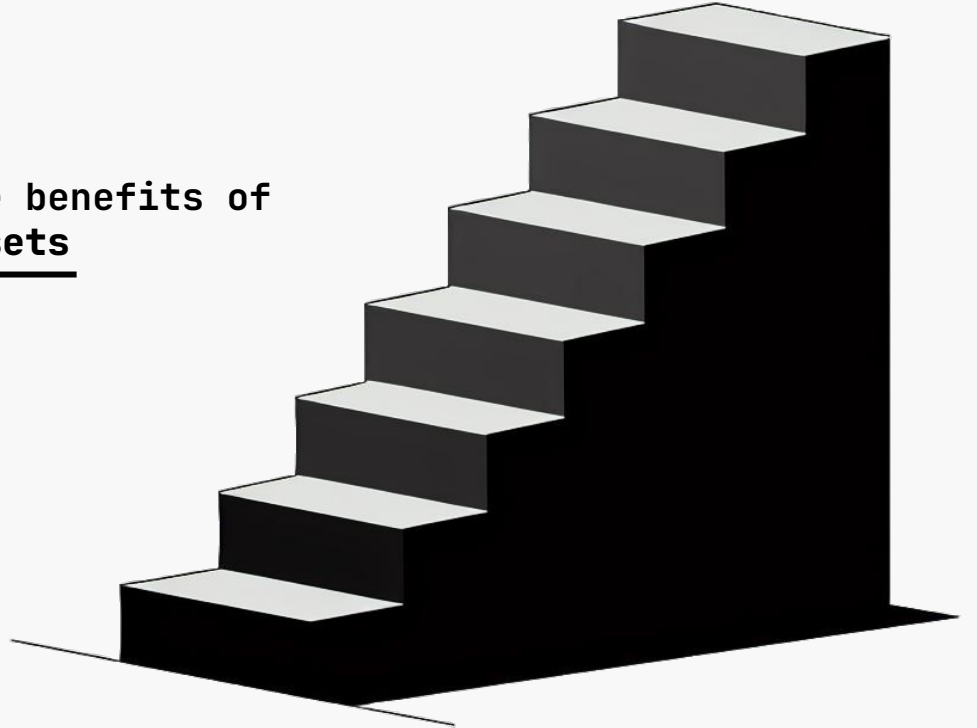


AMBITIONS

regard testing as a “thinking”
activity

acknowledge the benefits of
different mindsets

understand the fundamentals



WHY

DO WE

TEST?

make sure it works
as expected

make sure it
works



make sure it is
good

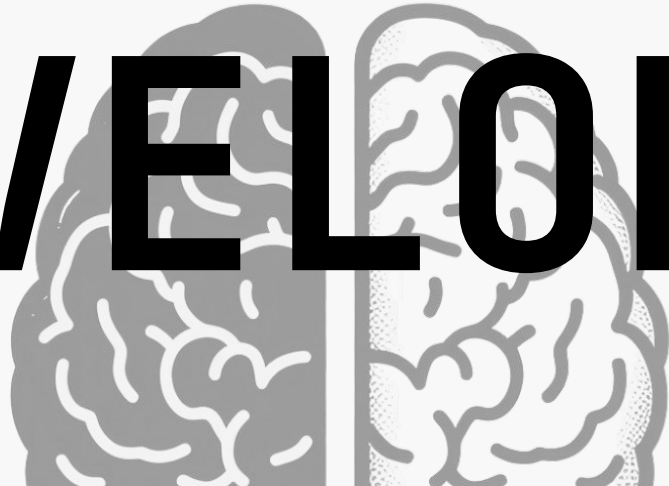
TESTING PRINCIPLES

- ▲ testing shows the presence of defects, not their absence
- ▶ exhaustive testing is impossible
- ▼ early testing (saves time and money)
- ◀ defects cluster together (the snowball effect)
- ▲ pesticide paradox
- ▶ context matters
- ▼ absence of errors fallacy

TESTER

VS.

DEVELOPER



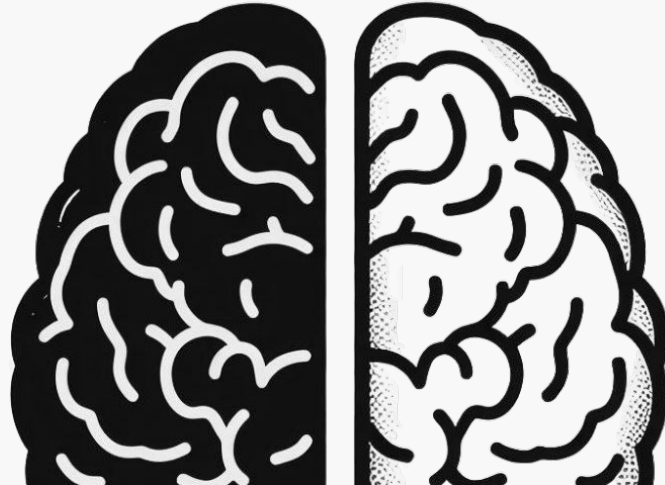
DIFFERENT MINDSETS

how can i build it?

professional optimism

how can i break it?

professional pessimism



CONTEXT MATTERS



TESTING IS CONTEXT-DEPENDENT

software systems are
not created equal

- ▲ intended for different users
- ▲ used in different ways
- ▲ pose different risks



TESTING IS CONTEXT-DEPENDENT

factors affecting the test effort

- ▲ type of **technology** (what, old, new)
- ▲ type of **project** (small, large, agile, waterfall)
- ▲ type of **product** (experimental, life-critical)



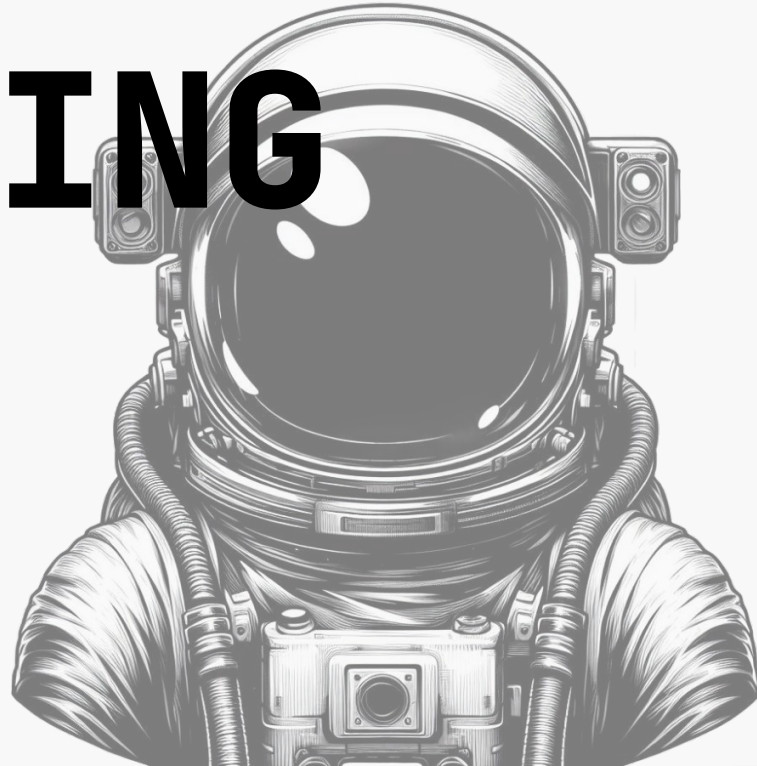
TESTING IS CONTEXT-DEPENDENT

require different
focus in test

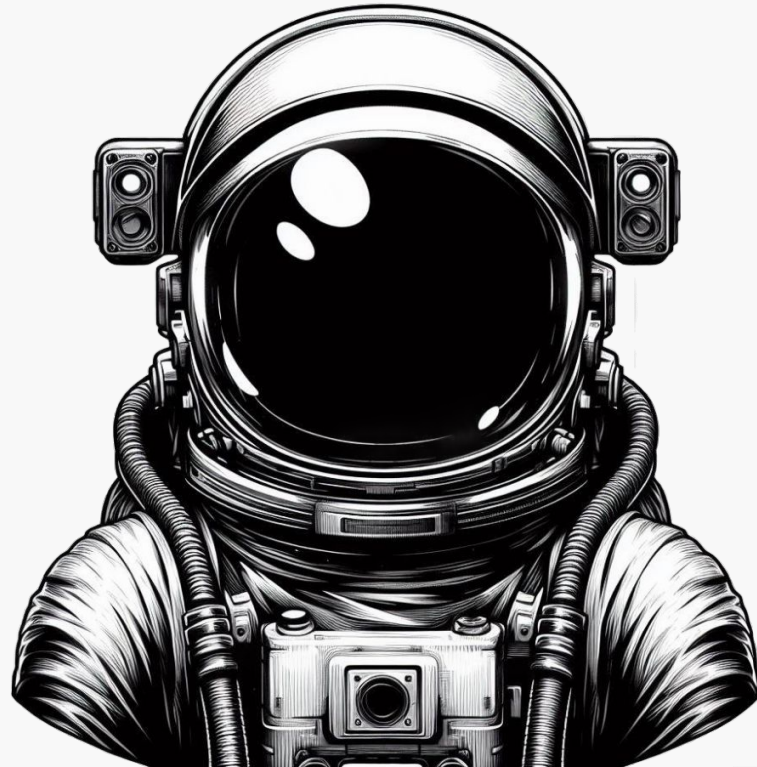
- ▲ cannot test all systems the same way
- ▲ need information about the system
- ▲ tailor test to context



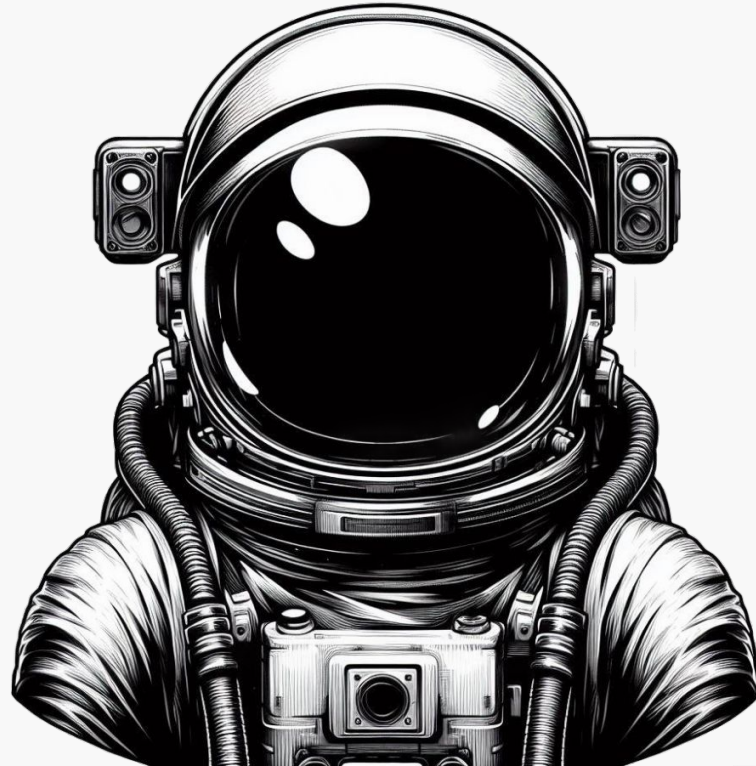
EXPLORATORY TESTING



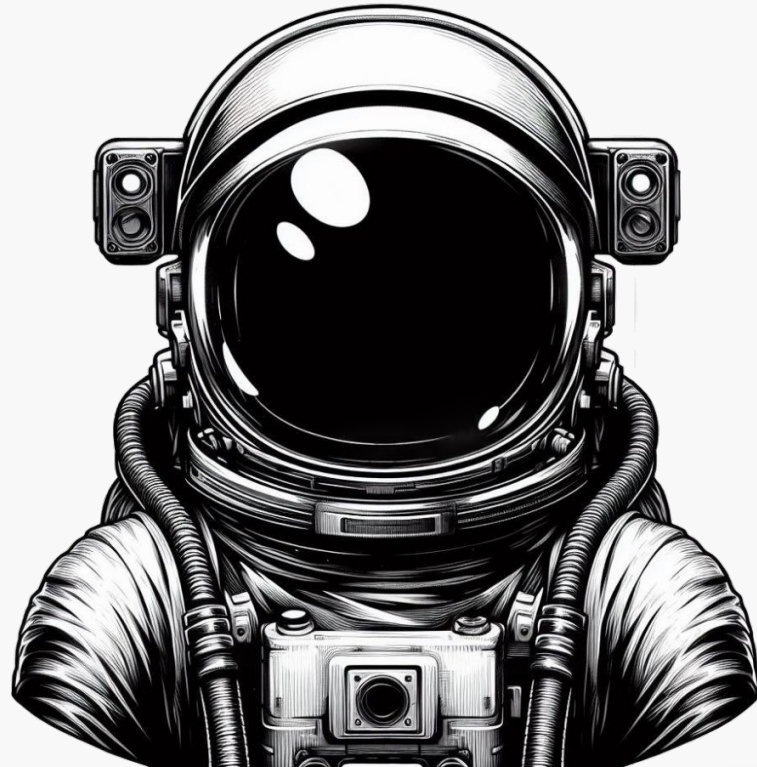
simultaneous **learning**, test **design**, and test **execution**



focuses on **discovery** and relies on the **guidance** of the individual tester




uncover defects that are not easily discovered through other approaches



EXPLORATORY TESTING

dynamic



execution of software /
components / system

focus on functionality /
expected behaviour

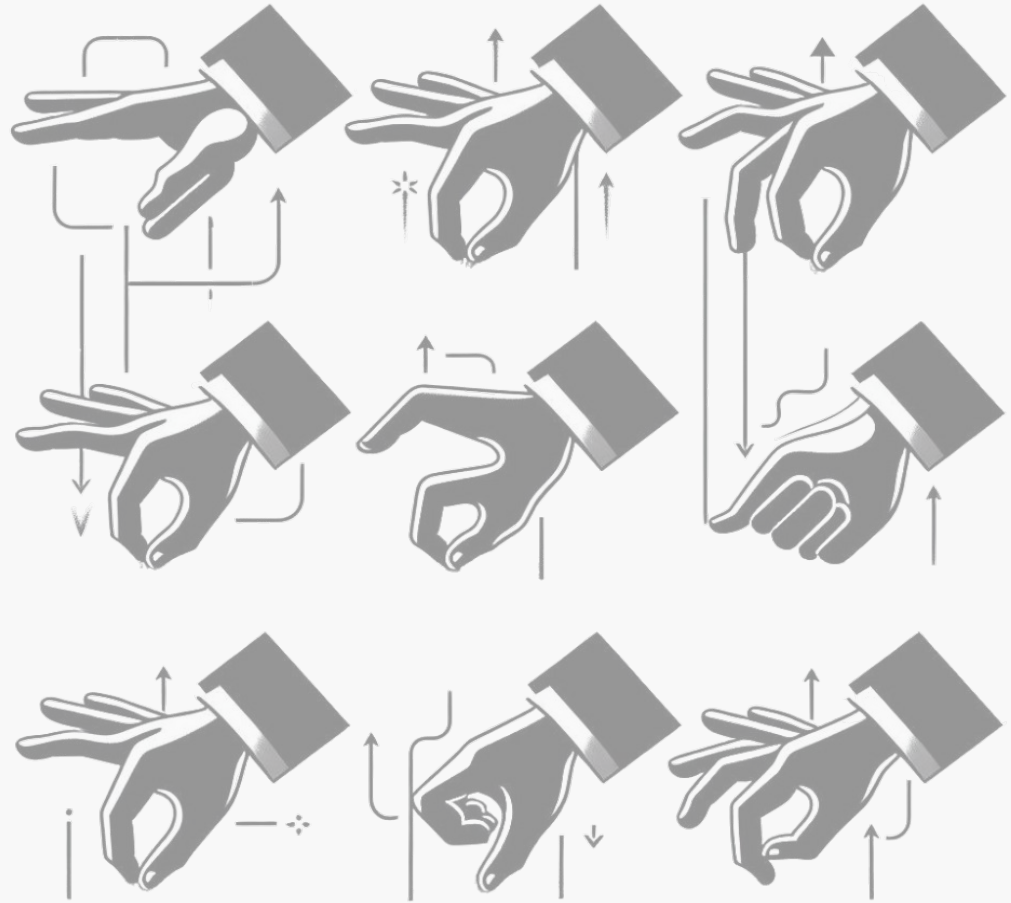
experience-based



based on previous knowledge /
intuition

often requires in-depth
domain knowledge

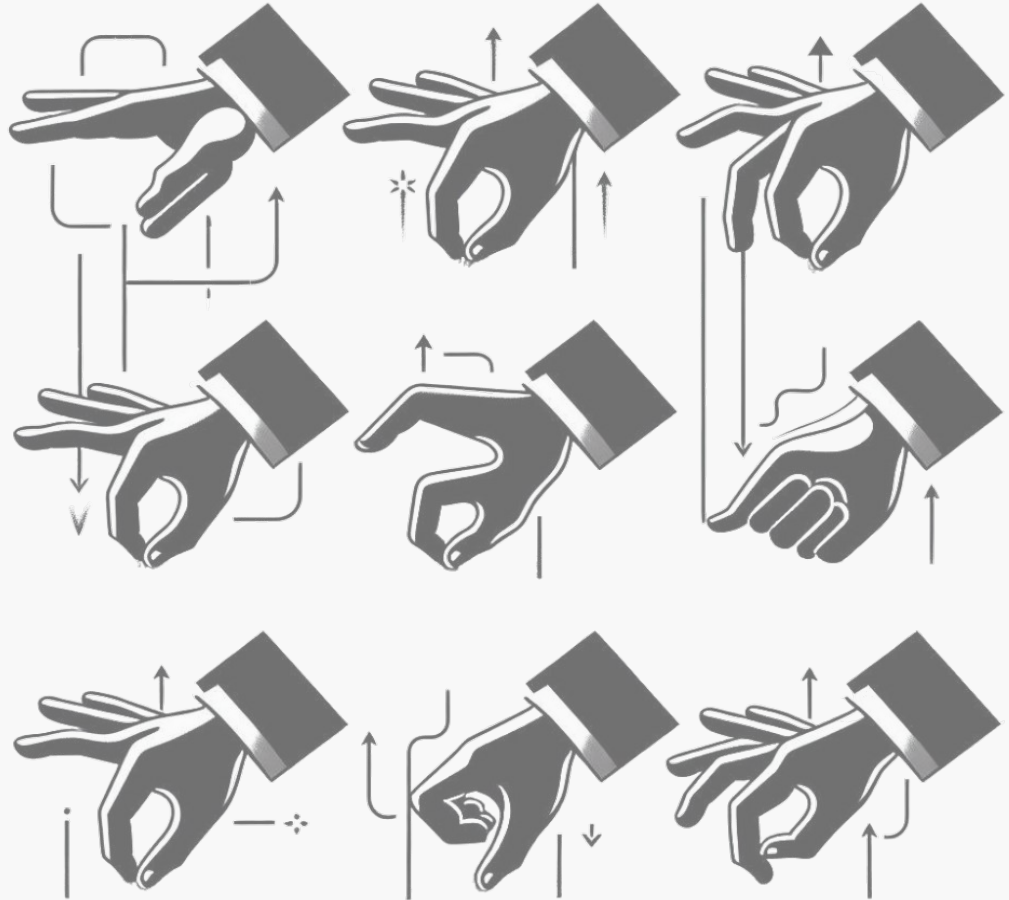
•
I before E
except
after C



how do you remember the
number of days in a month?

how do you remember which way
to set the clock?

how do you remember the
colours of the rainbow?



MNEMONICS AND HEURISTICS

mnemonic

a memory technique to aid
with retaining and retrieving
information

heuristic

a problem-solving approach
that involves using
practical, intuitive
strategies rather than
following a rigid set of
instructions

COUNT

zero, one, many. too many, too few.

- ▲ zero: search for a non-existent student
- ▲ one: search for a specific student
- ▲ many: search that returns several students
- ▲ too many / too few: searching both ends of the spectrum

GOLDILOCKS

too big, too small, just right

- ▲ too big: overload a text field
- ▲ too small: leave the text field empty
- ▲ just right: happy case input

CRUD

create, read, update, delete

- ▲ create: create new user / new user with duplicated attributes
- ▲ read: get user / get non-existent user
- ▲ update: update user / update non-existent user
- ▲ delete: delete user / delete non-existent user

RCRCRC

recent, core, risky, configuration, repaired, chronic

- ▲ recent: new features / code added
- ▲ core: key functionality that simply must work
- ▲ risky: areas relying on other services / components
- ▲ config: areas affected by config / environment settings
- ▲ repaired: code that has been changed during bug fix
- ▲ chronic: areas that frequently have issues

TRIGGER HEURISTICS

idea associated with an event or condition that **triggers** an action or **reaction**

emotions and feelings are powerful triggers

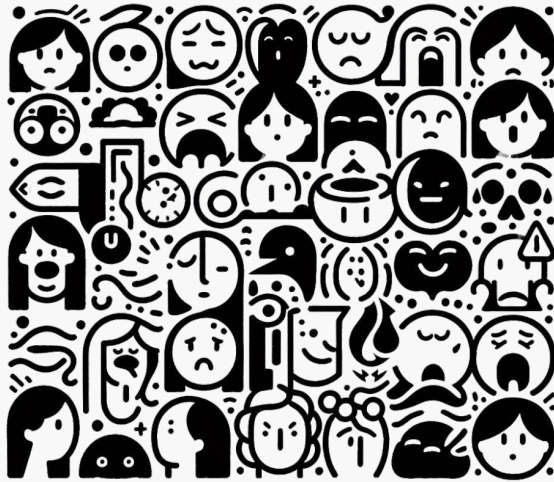


annoyance

missing feature?

impatience

delays?



surprise

inconsistency?

frustration

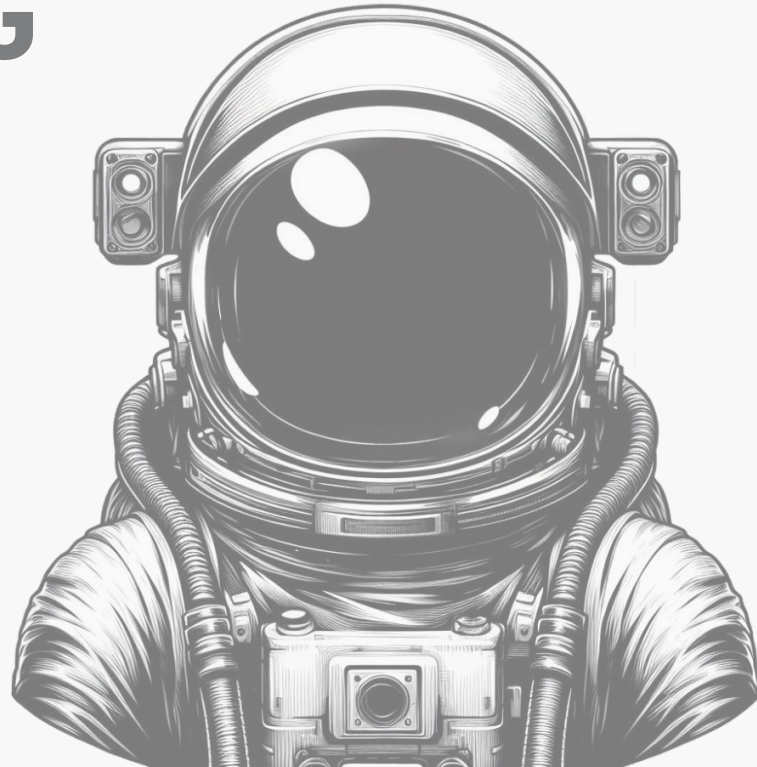
poor workflow?

confusion

counterintuitive interface?

**EXPLORATORY
TESTING**

**HOW
TO?**



STEP-BY-STEP

- I. develop a bug **classification**
- II. **understand** the system under test
- III. choose a **heuristic**
- IV. create a test **charter**
- V. continuously **assess** the findings

I. BUG CLASSIFICATION

- ▲ classify bugs found in previous projects
- ▲ analyse root causes for these bugs
- ▲ define risks (in light of typical bugs)



Input validation errors



Boundary related-errors



Calculation related errors



Security errors



Logic related errors



Usability errors



Compatibility errors



Syntax errors



Performance errors

II. UNDERSTAND THE SYSTEM UNDER TEST


- ▲ what kind of system?
- ▲ what kind of functionality?
- ▲ who are the users?

III. CHOOSE A HEURISTIC

- ▲ identify the testing heuristic that best fits the context
- ▲ multiple heuristics can be applied
- ▲ create your own heuristics over time


IV. CREATE A TEST CHARTER

self-reflection



be aware of your own testing
decisions

motivation



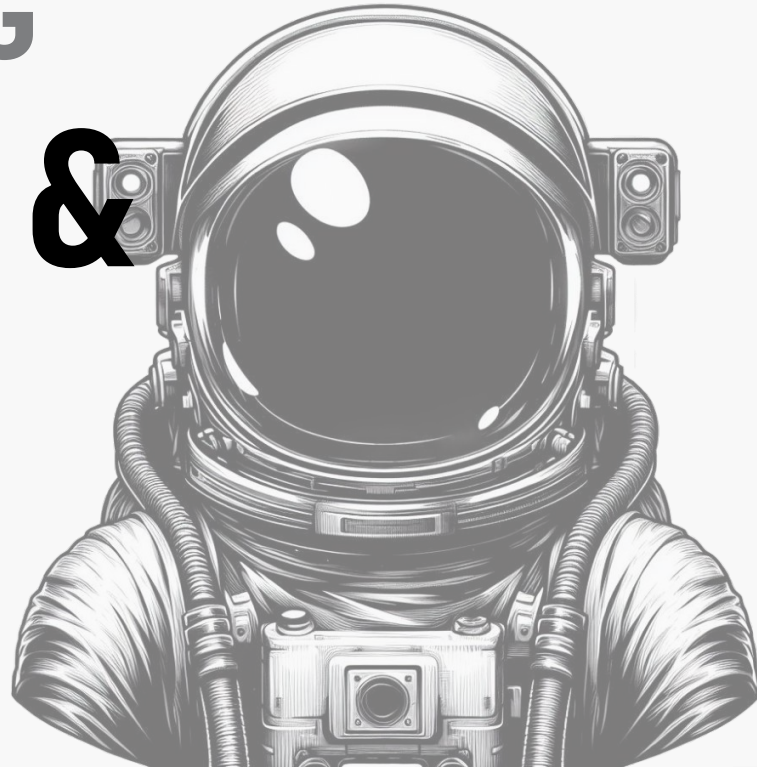
what?
how?
why?

V. ASSESS THE FINDINGS

- ▲ continuous feedback loop
- ▲ key question: what did i learn?
- ▲ use findings to guide the subsequent test efforts

EXPLORATORY
TESTING

**PROS &
CONS**



ADVANTAGES

- ▲ different kinds of bugs \Rightarrow random nature of testing
- ▲ rapid feedback \Rightarrow limited preparation necessary
- ▲ uncover new test scenarios
- ▲ suitable when there are no requirements / specifications

DISADVANTAGES

- ▲ quality of test depends on the tester's skill and experience
- ▲ requires a certain level of creativity
- ▲ difficult to reproduce failure scenarios
- ▲ the "ad hoc" nature may result in limited documentation

FINAL WORDS

- ▲ exploratory testing is essentially a mindset
- ▲ should complement other test efforts and more structured approaches

REFERENCES

- Florea, R. (2016). Exploratory testing. [online] Available at: <https://www.uio.no/studier/emner/matnat/ifi/INF3121/v15/lecture-09.pdf> [Accessed 25 Apr. 2022].
- Sam, B.H. (2019). Mindmaps and Heuristics for testing. [online] Bug Hunter Sam. Available at: <https://bughuntersam.com/mindmaps-and-heuristics-for-testing/> [Accessed 24 Apr. 2022].
- MoT. (n.d.). Software Testing Heuristics: Mind The Gap! [online] Available at: <https://www.ministryoftesting.com/dojo/lessons/software-testing-heuristics-mind-the-gap> [Accessed 24 Apr. 2022].
- Kirkham, P. (2020). More Ways to Test - Learning from Fairy Tales. [online] Atomic Spin. Available at: <https://spin.atomicobject.com/2020/01/17/software-test-goldilocks/> [Accessed 25 Apr. 2022].
- TestProject (2021). Software Testing Heuristics. [online] TestProject. Available at: <https://blog.testproject.io/2021/07/05/software-testing-heuristics/> [Accessed 24 Apr. 2022].
- Vasylyna, N. (2011). Exploratory Testing: Advantage&Disadvantage - QATestLab Blog. [online] Available at: <https://blog.qatestlab.com/2011/02/23/exploratory-testing-advantages-and-disadvantages/> [Accessed 24 Apr. 2022].
- Test Heuristics Cheat Sheet Data Type Attacks & Web Tests Paths/Files Timeouts Time Difference between Machines Crossing Time Zones Leap Days Always Invalid Days. (n.d.). [online] Available at: <https://testobsessed.com/wp-content/uploads/2011/04/testheuristicscheatsheetv1.pdf> [Accessed 25 Apr. 2022].
- Wordpress.com. (2022). [online] Available at: <https://findingdeefex.files.wordpress.com/2015/05/testingmnemonics1.jpg> [Accessed 25 Apr. 2022].