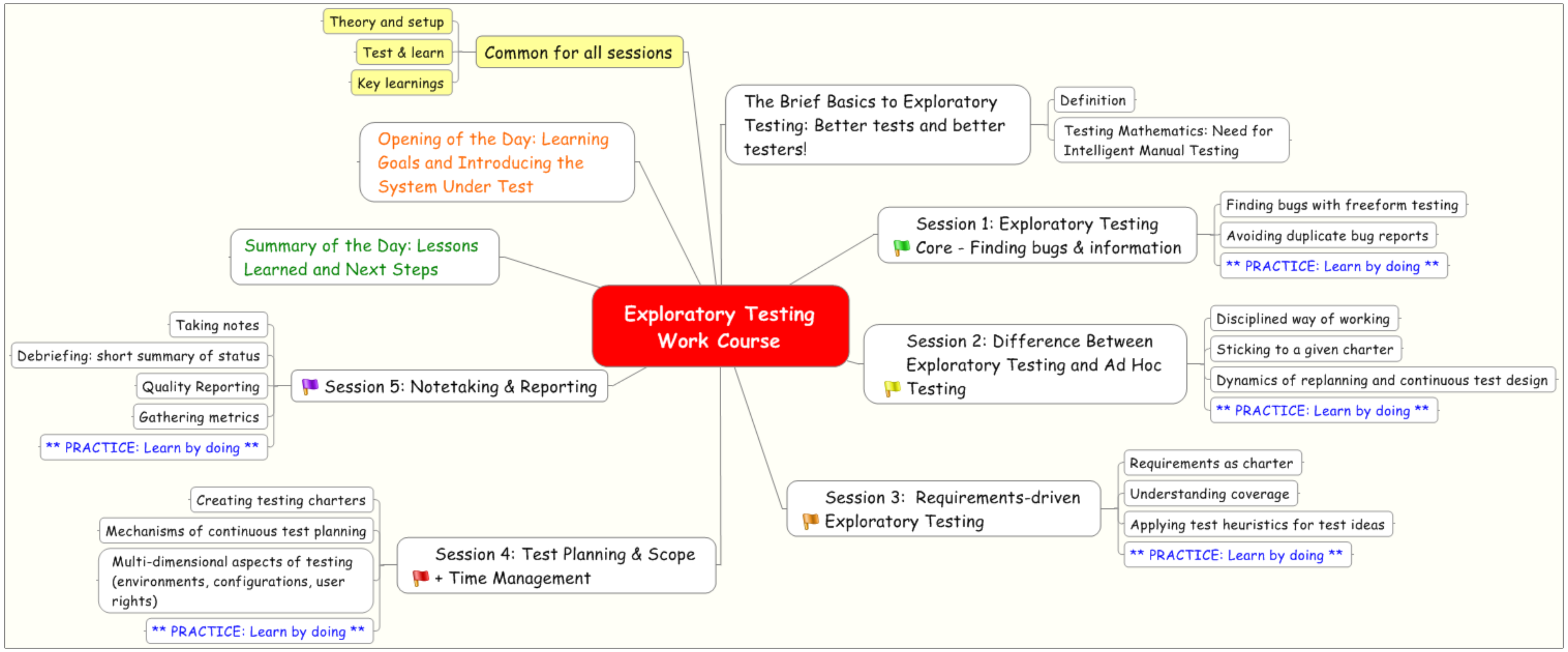
Exploratory Testing Work Course

Maaret Pyhäjärvi

November 2025

CGI

Course outline

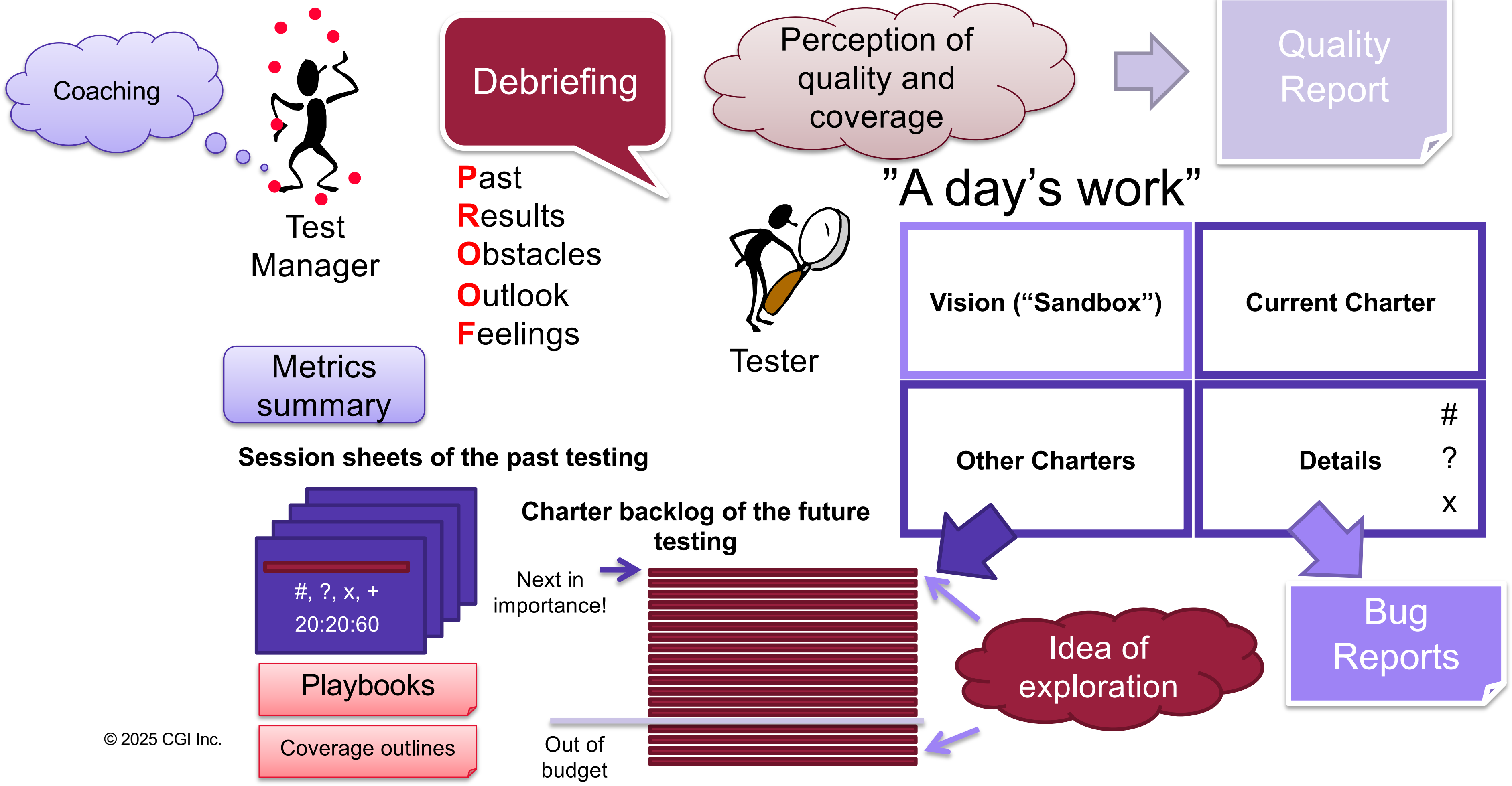

Exploratory Testing Work Course
Common for all sessions
Theory and setup
Test & learn
Key learnings
Opening of the Day: Learning Goals and Introducing the System Under Test
Summary of the Day: Lessons Learned and Next Steps
The Brief Basics to Exploratory Testing: Better tests and better testers!
Definition
Testing Mathematics: Need for Intelligent Manual Testing
Session 1: Exploratory Testing Core - Finding bugs & information
Finding bugs with freeform testing
Avoiding duplicate bug reports
** PRACTICE: Learn by doing **
Session 2: Difference Between Exploratory Testing and Ad Hoc Testing
Disciplined way of working
Sticking to a given charter
Dynamics of replanning and continuous test design
** PRACTICE: Learn by doing **
Session 3: Requirements-driven Exploratory Testing
Requirements as charter
Understanding coverage
Applying test heuristics for test ideas
** PRACTICE: Learn by doing **
Session 4: Test Planning & Scope + Time Management
Creating testing charters
Mechanisms of continuous test planning
Multi-dimensional aspects of testing (environments, configurations, user rights)
** PRACTICE: Learn by doing **
Session 5: Notetaking & Reporting
Taking notes
Debriefing: short summary of status
Quality Reporting
Gathering metrics
** PRACTICE: Learn by doing **

Target of Testing

Open Office Impress

https://www.openoffice.org/download/

Exploratory Testing: Frame of Management



The Pieces in Management Framework

A disciplined tester replanning on various levels

Session with charter that provides a report

Classification of information created as metrics

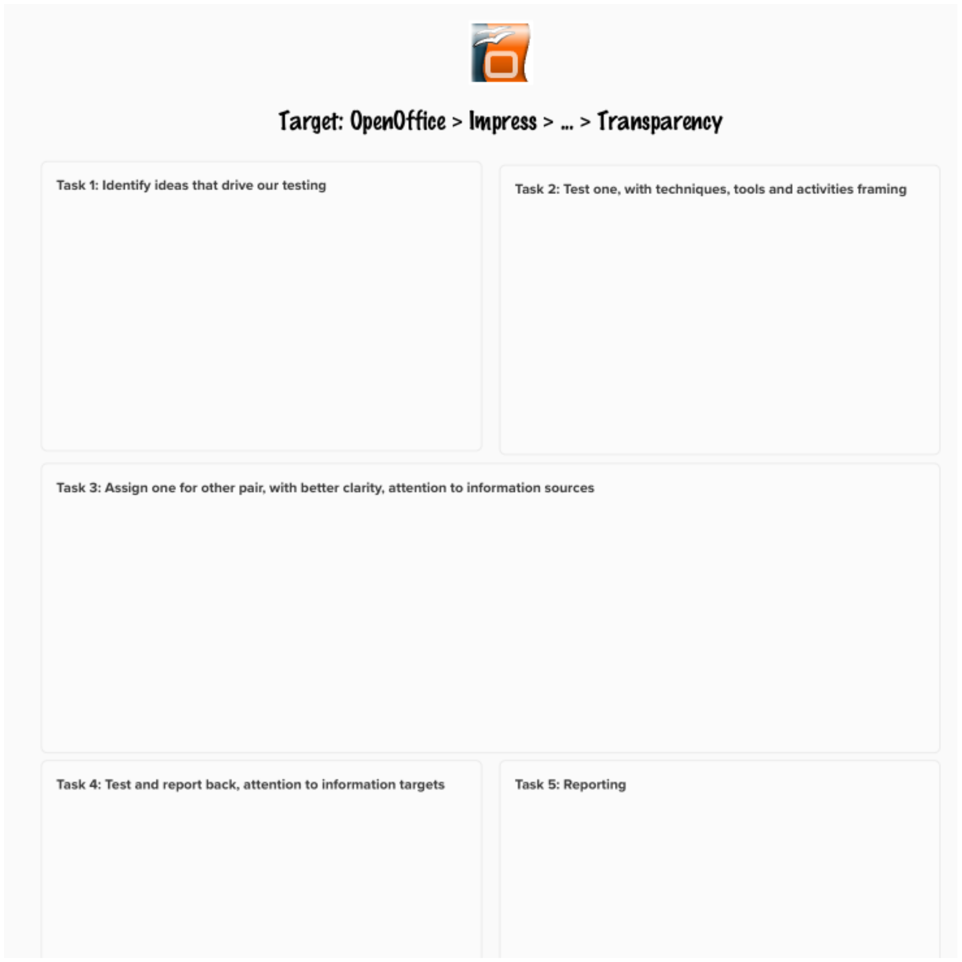
Prioritizing of what test idea comes next

Supporting reporting by debriefing

Supporting skills development by coaching or pairing

Creating a combined judgement of quality by quality reporting

Work course framing ideas



Target: OpenOffice > Impress > ... > Transparency

Task 1: Identify ideas that drive our testing

Task 2: Test one, with techniques, tools and activities framing

Task 3: Assign one for other pair, with better clarity, attention to information sources

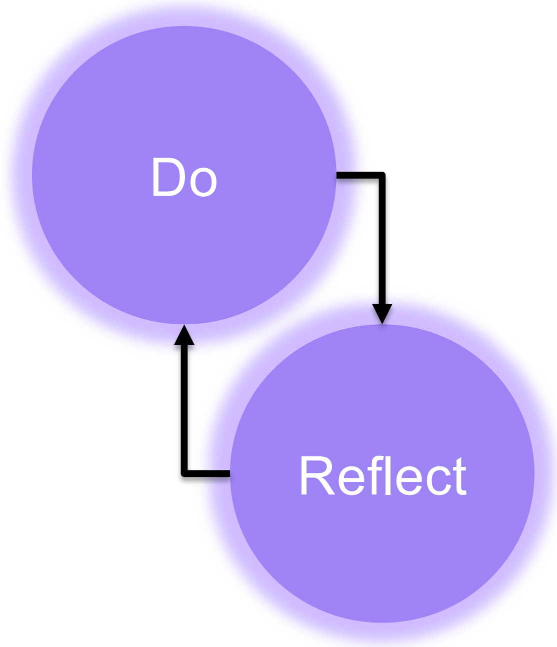
Task 4: Test and report back, attention to information targets

Task 5: Reporting

https://bit.ly/3XHVolf

We Learn to Test by Testing

TestThisBox
Gilded Rose
Roman Numerals
E-Primer
DFEditor
Freemind
Zippopotamus
React-Weather
Odoo



Exploratory testing:
Better tests, better testers!

Approach to testing

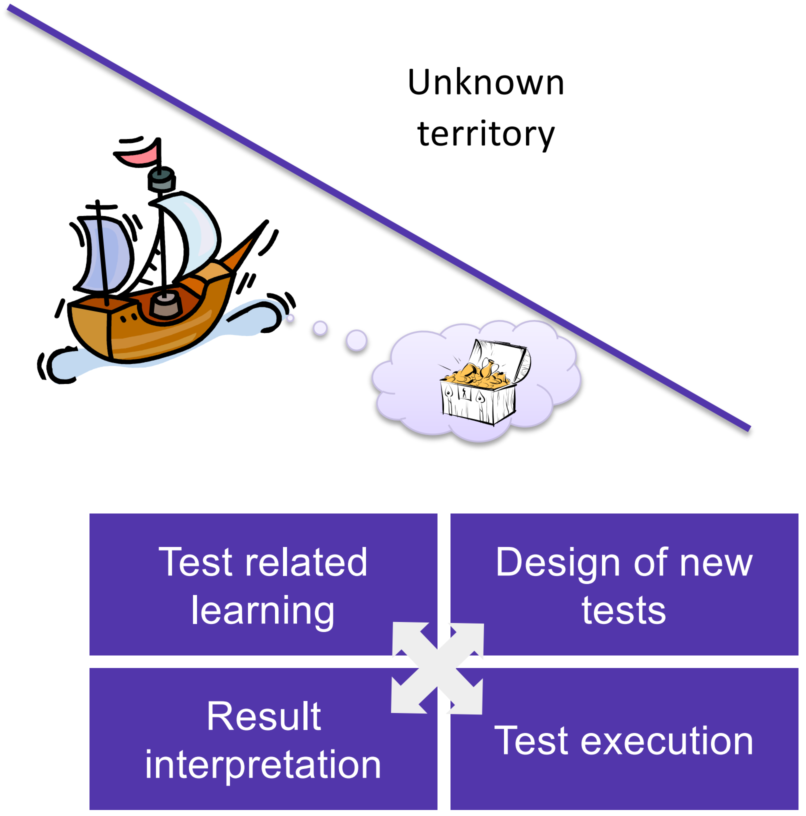
- Emphasizes individual’s freedom and responsibility on a process where continuous value delivery optimization is needed

Not a technique, but a way of thinking about testing: “Any testing process that involves simultaneous learning, test design, and execution.”

Disciplined, planned and controlled way to test that emphasizes learning

Finnish research Itkonen et al. 2007

- Comparable results with preplanned testcases and exploratory testing
- More false alarms with test cases
- More effort going into the test case approach



Two Sides of Exploratory Testing

For the tester

Fun

Freedom

Flexibility

Professionalism

Respect

For the manager

Value

Controllability

Reliability

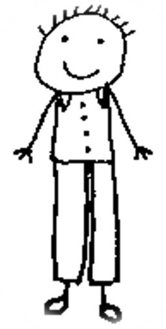
Visibility

Results

@maaretp

First Experience to Exploratory Testing?

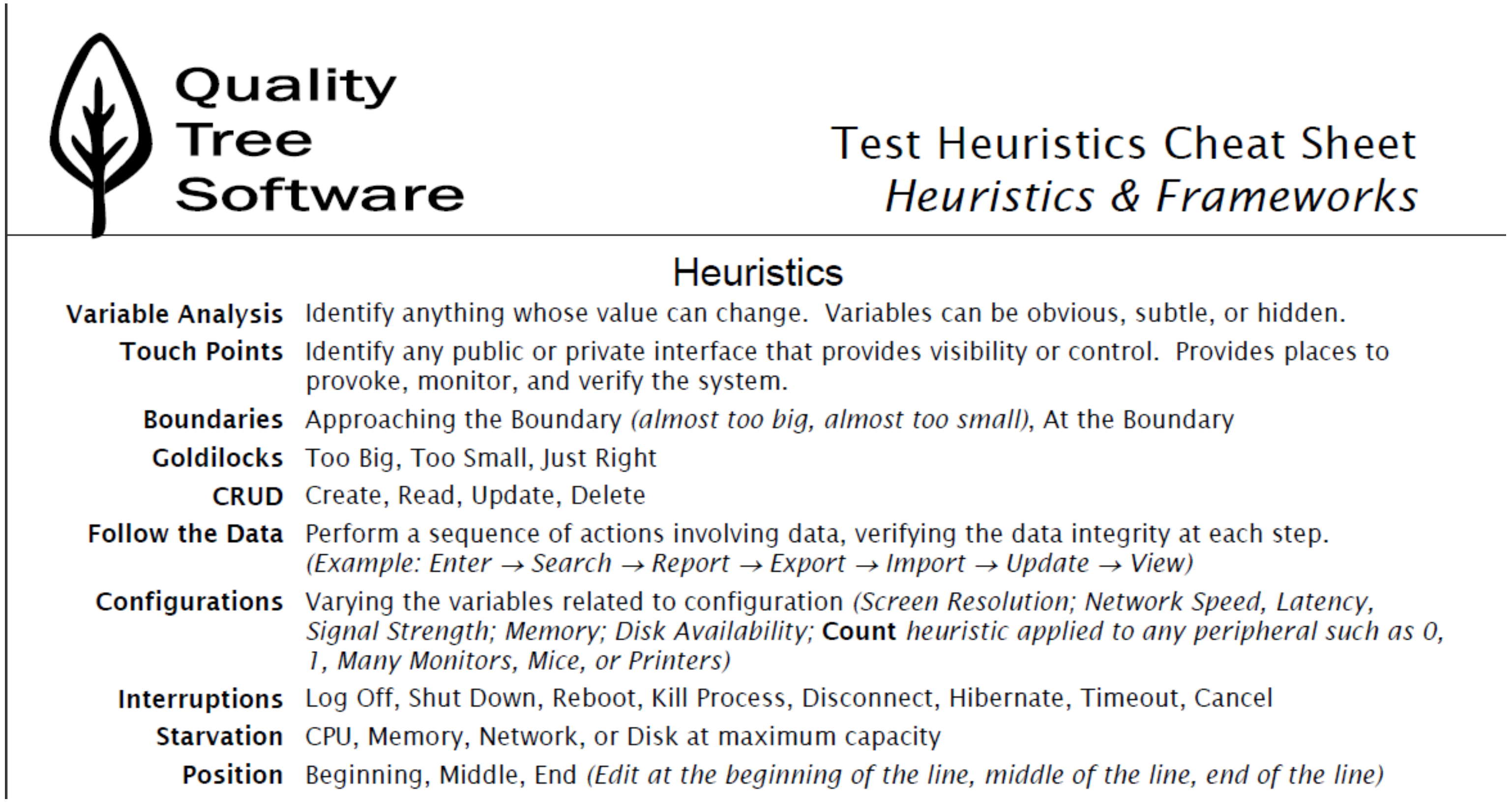
Spec => Tests
+ a Session to Find all Bugs



Repeat list of test cases
+ a conference awakening

Some tests, budget of time, QA on results https://www.ministryoftesting.com/dojo/lessons/test-heuristics-cheat-sheet

Test Ideas / Quicktests



Quality Tree Software

Test Heuristics Cheat Sheet
Heuristics & Frameworks

Heuristics

Variable Analysis	Identify anything whose value can change. Variables can be obvious, subtle, or hidden.
Touch Points	Identify any public or private interface that provides visibility or control. Provides places to provoke, monitor, and verify the system.
Boundaries	Approaching the Boundary *(almost too big, almost too small)*, At the Boundary
Goldilocks	Too Big, Too Small, Just Right
CRUD	Create, Read, Update, Delete
Follow the Data	Perform a sequence of actions involving data, verifying the data integrity at each step. *(Example: Enter → Search → Report → Export → Import → Update → View)*
Configurations	Varying the variables related to configuration *(Screen Resolution; Network Speed, Latency, Signal Strength; Memory; Disk Availability;* **Count** *heuristic applied to any peripheral such as 0, 1, Many Monitors, Mice, or Printers)*
Interruptions	Log Off, Shut Down, Reboot, Kill Process, Disconnect, Hibernate, Timeout, Cancel
Starvation	CPU, Memory, Network, or Disk at maximum capacity
Position	Beginning, Middle, End *(Edit at the beginning of the line, middle of the line, end of the line)*

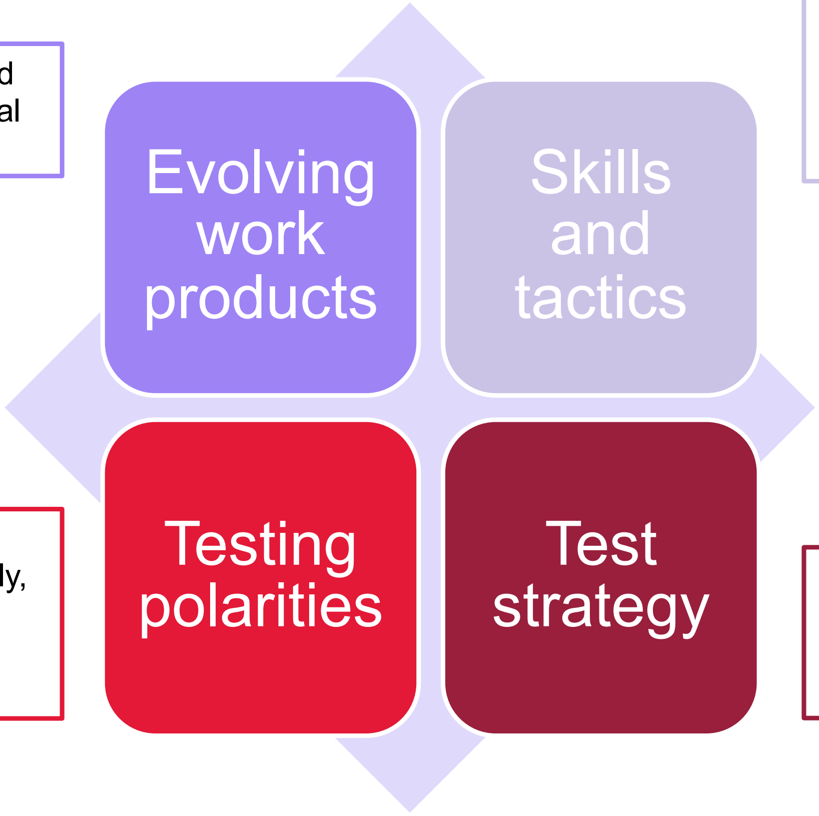
Download the full 2-page Cheat Sheet with ideas from Elisabeth Hendrickson, James Lyndsay, and Dale Emery

(Exploratory) Testing Dynamics

Source: Adapted from James Bach, Jon Bach, Michael Bolton. Exploratory Testing Dynamics. v.2.2. 2009

”Exploratory testing spirals upward toward a complete and professional set of test artifacts”

” …skills that comprise professional and cost effective exploration of technology. Each is distinctly observable and learnable, and each is necessary to exploratory work.”

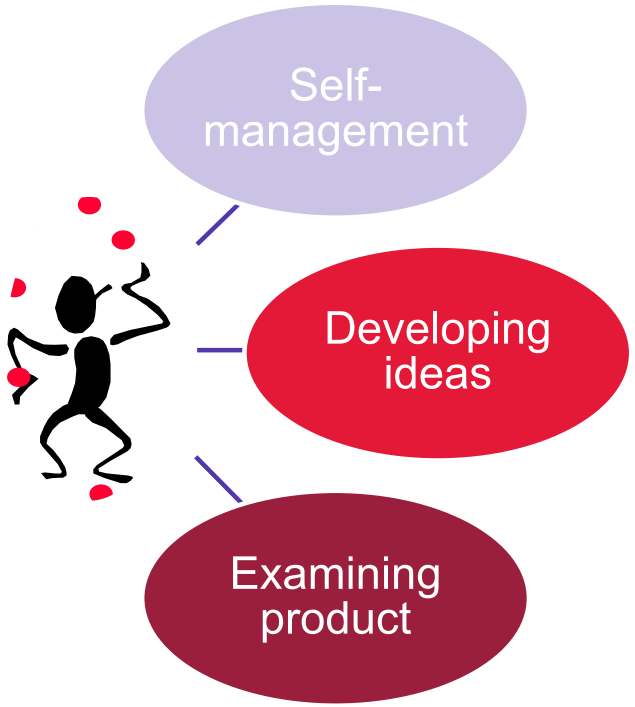


” To develop ideas or search a complex space quickly yet thoroughly, not only must you look at the world from many points of view and perform many kinds of activities”

”A set of considerations designed to help you test robustly or evaluate someone else’s testing.”

Exploration Skills

Adapted from James Bach, Jon Bach, Michael Bolton. Exploratory Testing Dynamics. v.2.2. 2009



Done
To Do
Issues
Coverage

All sources available
Best use of time – effective and efficient work

Making models
Tool support – creative solutions
Risk-based testing – scientific approach
Keeping one’s eyes open

Realizations on Nature of Testing

20

16

1639

5±2

Realizations on Nature of Testing

20	DYNAMICALLY ADAPT FOR LIMITED BUDGET OPPORTUNITY COST
16	EXPECT THE UNEXPECTED
1639	ROUTES ARE RELEVANT NOT ALL BUGS ARE EQUAL
5±2	TAKE NOTES CREATE CHECKLISTS

Some Examples of Testing Practices That Take Skill to Apply

Seeing problems without someone telling you specifically what to look for

Building understanding of coverage level in sessions

Stopping testing voluntarily without exhausting ideas of things to test

Continuous but controlled replanning and redesign in testing

Testing in various depths from basic functionalities to reliability in use

Getting the message across as intended when reporting a bug

Skipping bug reporting based on value of information

Flavors of Test Activity

Intake - negotiating and accepting your mission, understand what you're here to do; systematic intake, find out context with project environment questions / satisfice test strategy model

Survey - learn mental model of the product (used to call this recon)

Analysis - getting organized with your test ideas, test strategy (list = analysis, learning = survey). leads to an artifact.

Setup - get ready for particular type of test

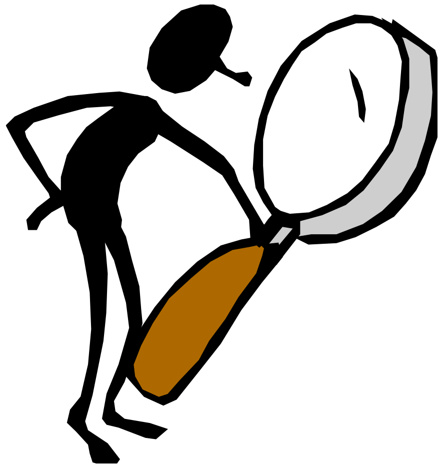
Deep coverage - testing to find non-obvious yet serious problems; surprised if you find more there, and learn of surprise.

Closure – getting ready to ship, ready to report. Documentation for next tester, regression testing, bug fix testing. Stuff done towards the end.

Session 1: Apply a Test Technique

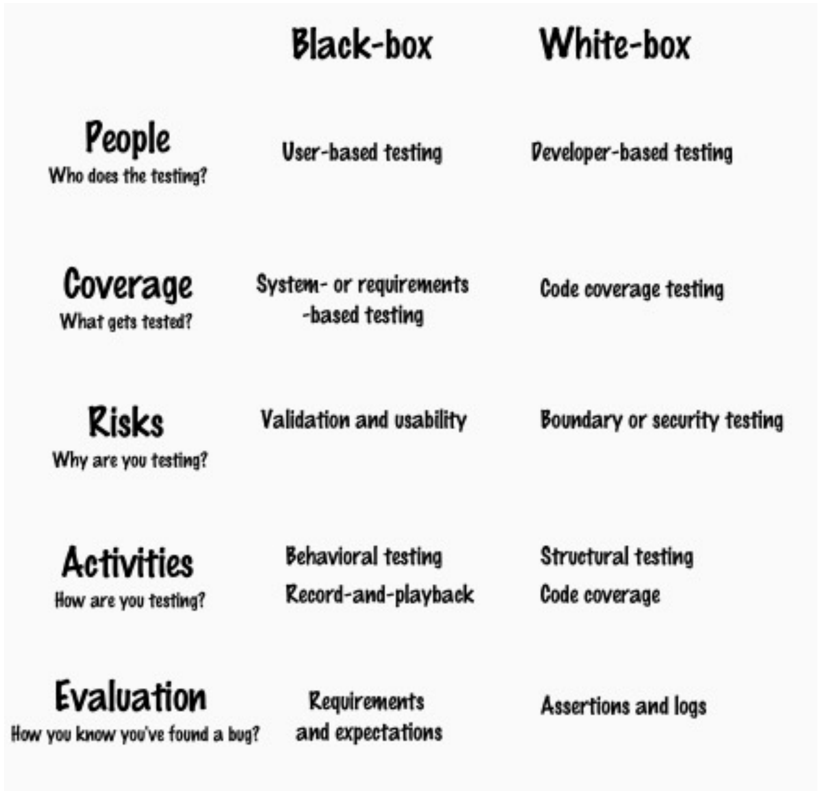
Risk-based domain testing: variables and variable relationships

Bug reporting to avoid duplicates

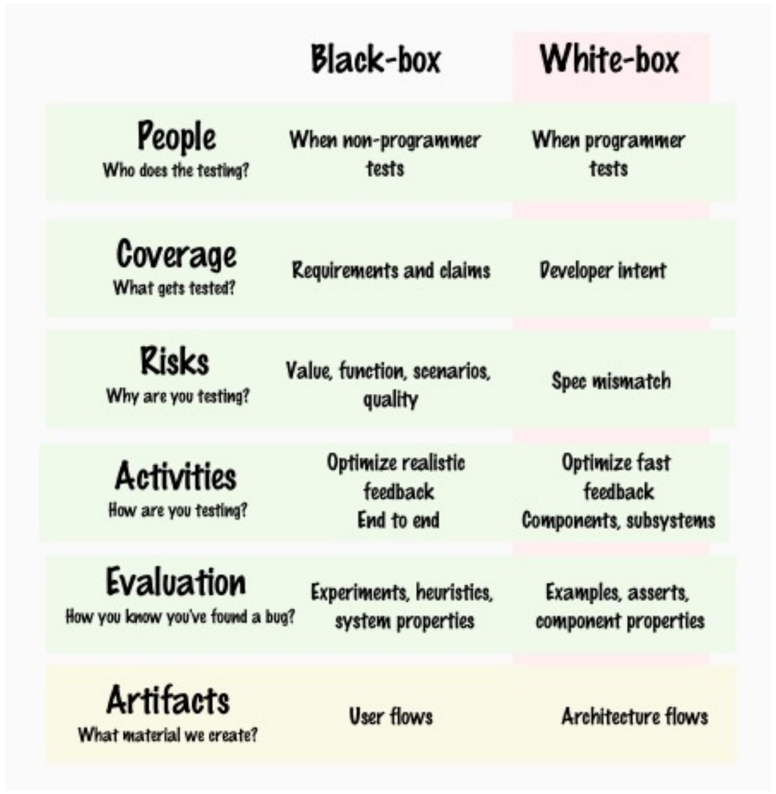


Kaner's five-fold test techniques classification system

Kaner et al. 2001. Lessons Learned in Software Testing

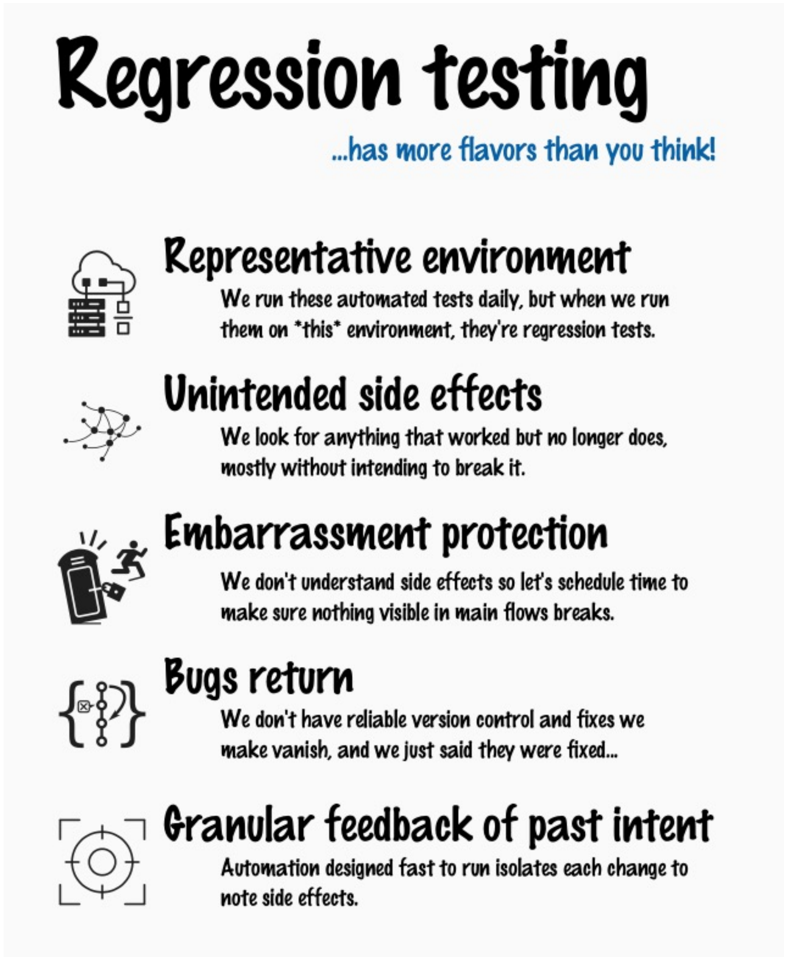


	Black-box	White-box
People Who does the testing?	User-based testing	Developer-based testing
Coverage What gets tested?	System- or requirements -based testing	Code coverage testing
Risks Why are you testing?	Validation and usability	Boundary or security testing
Activities How are you testing?	Behavioral testing Record-and-playback	Structural testing Code coverage
Evaluation How you know you've found a bug?	Requirements and expectations	Assertions and logs



	Black-box	White-box
People Who does the testing?	When non-programmer tests	When programmer tests
Coverage What gets tested?	Requirements and claims	Developer intent
Risks Why are you testing?	Value, function, scenarios, quality	Spec mismatch
Activities How are you testing?	Optimize realistic feedback End to end	Optimize fast feedback Components, subsystems
Evaluation How you know you've found a bug?	Experiments, heuristics, system properties	Examples, asserts, component properties
Artifacts What material we create?	User flows	Architecture flows

Framing regression testing



Regression testing

...has more flavors than you think!

Representative environment
We run these automated tests daily, but when we run them on *this* environment, they're regression tests.

Unintended side effects
We look for anything that worked but no longer does, mostly without intending to break it.

Embarrassment protection
We don't understand side effects so let's schedule time to make sure nothing visible in main flows breaks.

Bugs return
We don't have reliable version control and fixes we make vanish, and we just said they were fixed...

Granular feedback of past intent
Automation designed fast to run isolates each change to note side effects.

Exploring a feature

Transparency
Single variable for test ideas

Variable: shape area transparency in Impress

Basics of the variable:

input: a value between 0 (opaque) and 100 (transparent)

- connects with toolbars and dialogs displaying and changing the value
- is set in relation to its previous value
- is saved an attribute of a shape
- has a default that can be set separately (graphics style: default)

output: a shape in different degrees of transparency

- connects with presentation modes showing the shape
- printing includes possibility to turn transparency off
- matches concepts for transparency used in other presentation software

Research: Googled “why is transparency hard” https://shaderfun.com/2020/09/20/why-transparency-is-hard/

Assignment

List risks: what it is you worry about with the feature

e.g. it accepts values that are not numbers

Risks

it does not accept values that are integers between 0-100, particularly the boundary values

it accepts values that are too small or too large

it accepts values that are not numbers

it incorrectly rounds fractions of numbers with commas/full stops as fraction separator

it forgets earlier values given values that are too small or too large

it does not correctly render the transparency of image on screen in normal editing views

it does not correctly render the transparency in presentation or printing

it displays increasing / decreasing transparency illogically for some colors

it displays transparency illogically when transparent shape is overlaying another shape in front/back of it

it does not follow higher level turning off instructions, such as printing setting requiring forced opaqueness

it does not show transparency for some types of shapes such as images

it does not get saved with the shapes

it gets saved when the shape itself is not saved

it displays incorrect values on one of its many places of presenting it on the UI

it does not return to its previous value with undo

its set value holds in combinations of settings it is not present in, such as gradients

its set value is lost in changing temporarily to other combinations of transparency its not part of

its set default style isn’t applied on all shapes

it cannot be adjusted by using the increment arrows

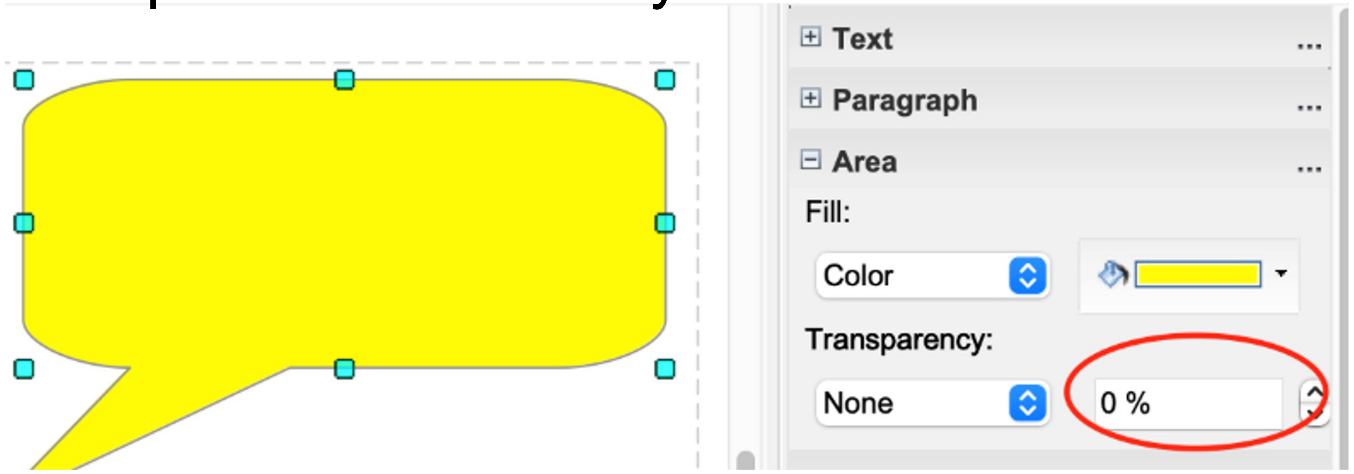
it does not render correctly with regards to animations

it increases disk space / memory consumption significantly when used

it does not render transparency in 3D objects well in relation to 2D objects

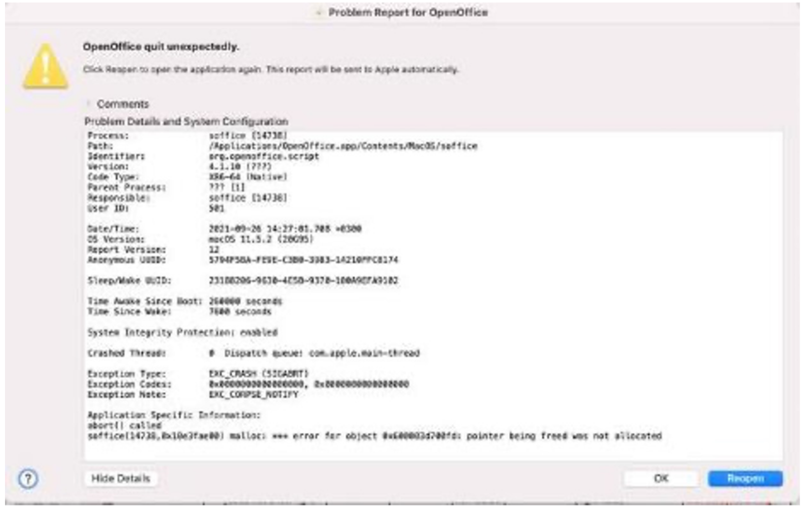
it shows different combined color of two transparent objects changes depending on which color is in front

Experience what they miss



Fill: Color

Transparency: None 0 %



OpenOffice quit unexpectedly.

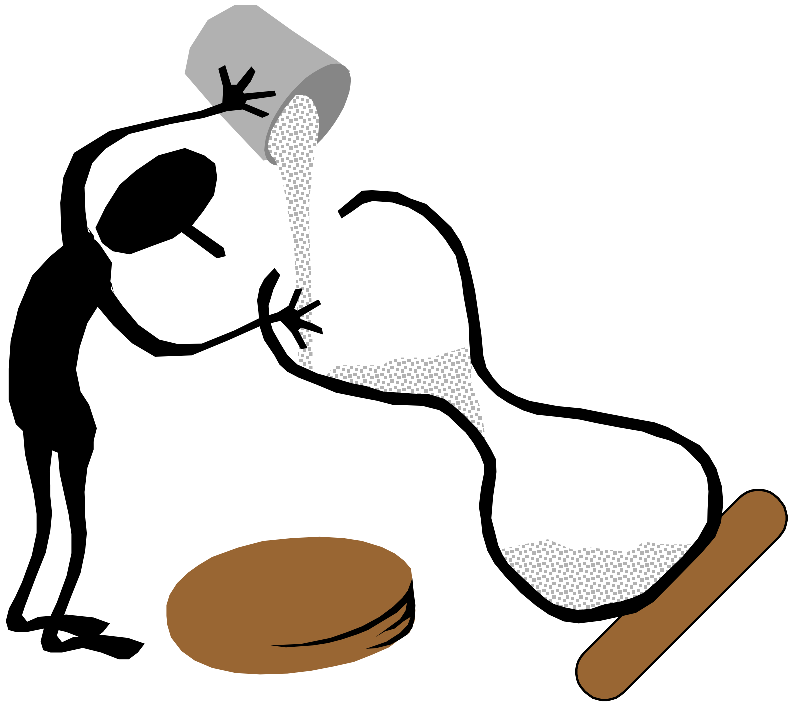
Click Reopen to open the application again. This report will be sent to Apple automatically.

Problem Details and System Configuration

```
Process:               soffice [14738]
Path:                  /Applications/OpenOffice.app/Contents/MacOS/soffice
Identifier:            org.openoffice.script
Version:               4.1.10 (???)
Code Type:             X86-64 (Native)
Parent Process:        ??? [1]
Responsible:           soffice [14738]
User ID:               501

Date/Time:             2021-09-26 14:27:01.708 +0300
OS Version:            macOS 11.5.2 (20G95)
Report Version:        12
Anonymous UUID:        5794F58A-FE9E-C3B0-3983-14210FFC8174

Sleep/Wake UUID:       23108206-9630-4E5D-9378-100A9EFA9182

Time Awake Since Boot: 260000 seconds
Time Since Wake:       7600 seconds

System Integrity Protection: enabled

Crashed Thread:        0  Dispatch queue: com.apple.main-thread

Exception Type:        EXC_CRASH (SIGABRT)
Exception Codes:       0x0000000000000000, 0x0000000000000000
Exception Note:        EXC_CORPSE_NOTIFY

Application Specific Information:
abort() called
soffice(14738,0x10e3fae00) malloc: *** error for object 0x600003d700fd: pointer being freed was not allocated
```

it does not accept values that are integers between 0-100, particularly the boundary values
it accepts values that are too small or too large when written in field
it cannot be adjusted by using the increment arrows
it incorrectly rounds fractions of numbers with full stops as fraction separator
it incorrectly rounds fractions of numbers with commas as fraction separator
accepts large number with more than three decimals
it accepts arbitrarily large number
decimal values that round down incorrectly rounded
multiple rounding decimals lead to wrong values on UI
moving focus away from variable field does not apply value
enter in variable field does not apply value
invisible shape cannot be selected
it does not return to its previous value with undo
multiple values in sequence crash the program
selecting a transparent shape does not show right value
it accepts values that are not numbers
it accepts values that are mix of numbers and letters
it forgets earlier values given values that are too small or too large
it distorts transparency of shape when shape has text
it does not correctly render the transparency of image on screen in normal editing views
it does not correctly render the transparency in presentation or printing
it displays increasing / decreasing transparency illogically for some colors
it displays transparency illogically when transparent shape is overlaying another shape in front/back of it
it does not follow higher level turning off instructions, such as printing setting requiring forced opaqueness
it does not show transparency for some types of shapes such as images
it does not get saved with the shapes
it gets saved when the shape itself is not saved
it displays incorrect values on one of its many places of presenting it on the UI
its set value holds in combinations of settings it is not present in, such as gradients
its set default style isn't applied on all shapes
it does not render correctly with regards to animations
it increases disk space / memory consumption significantly when used
it shows different combined color of two transparent objects changes depending on which color is in front
…

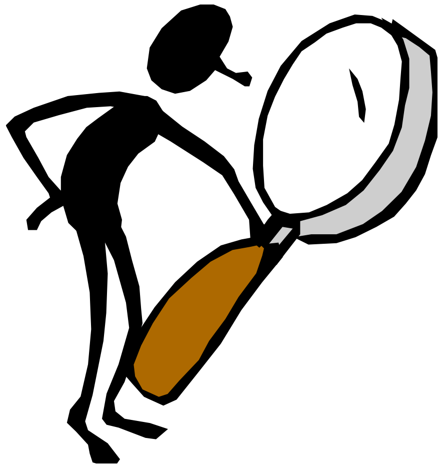
Testing-Time



Session 1: What Did We Learn?

Domain testing: variables and variable relationships

Bug reporting to avoid duplicates

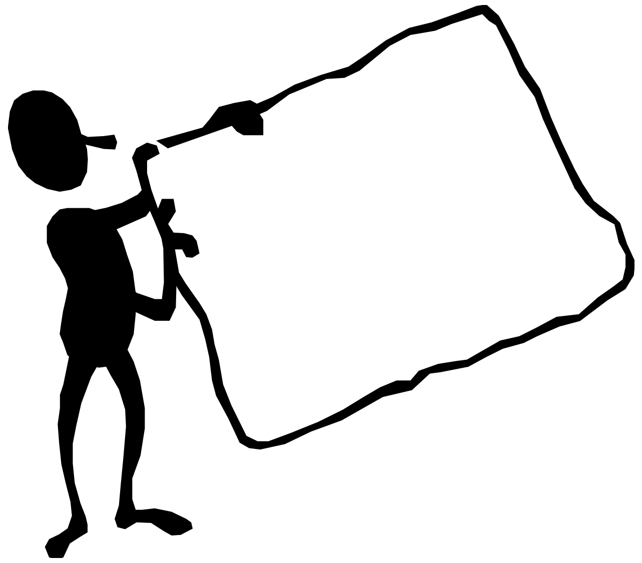


Session 2: Testing On Charter

Using a given charter

Being disciplined

Coverage: knowing when you're done

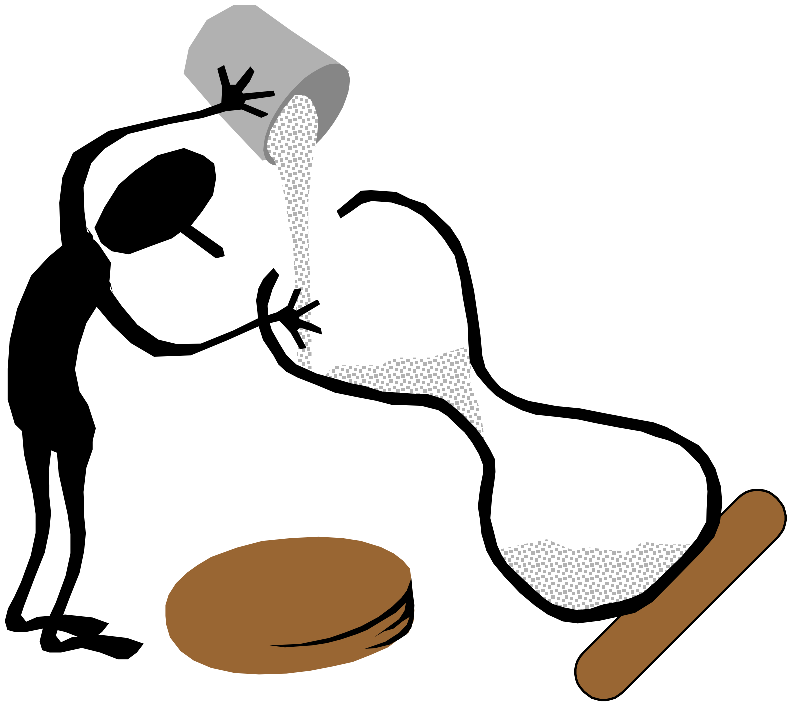


Architecting the Charters – Test Planning

Brief and flexible plans:

- Punch line of the test: one-liner description
- What and why, how, what to look for
- Tools to use, specific tactics or viewpoints, risks and concerns, documents to use, results to deliver with testing

Testing-Time

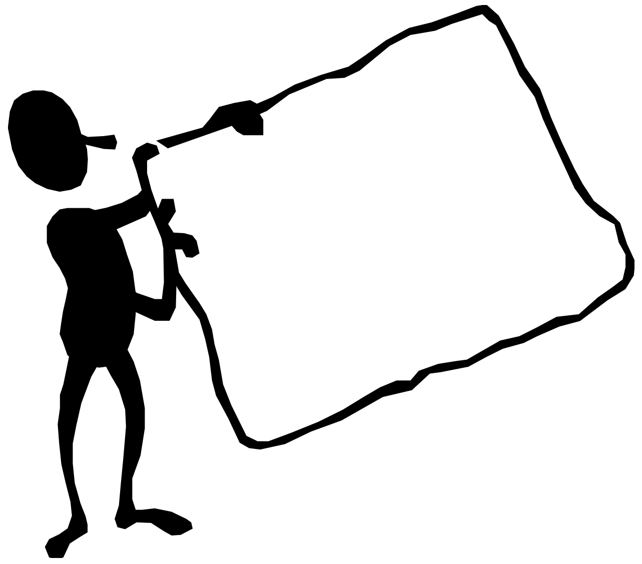


Session 2: What Did We Learn?

Using a given charter

Being disciplined

Coverage: knowing when you're done



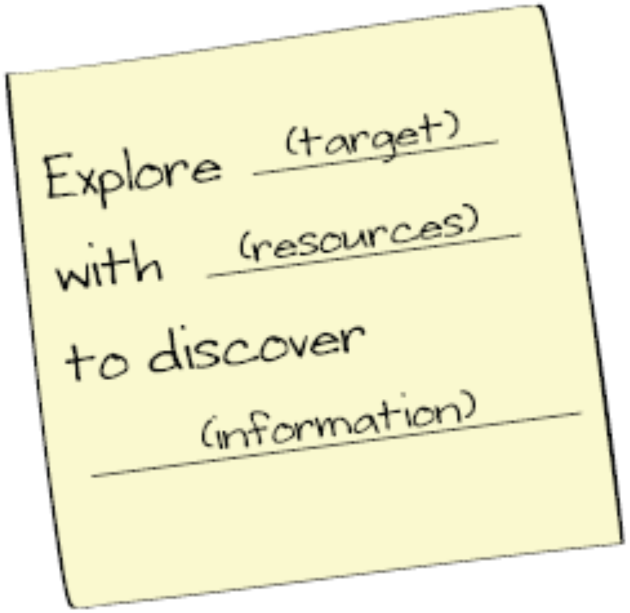
Session 3: Continuous Test Planning

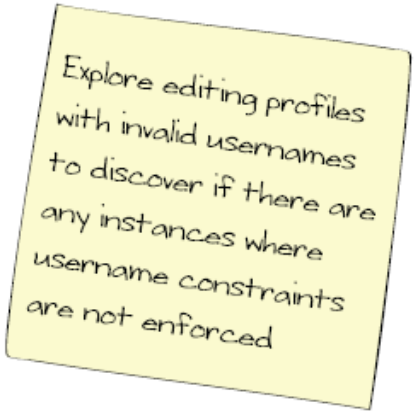
Charter creation

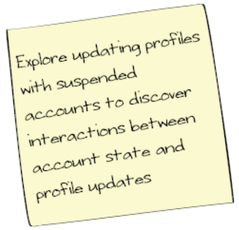
Requirements in exploratory testing

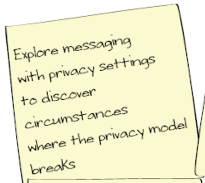
Depths of testing

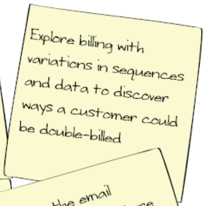
Examples from Elisabeth Hendrickson's book Explore It

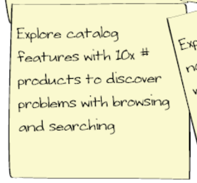

Explore (target)
with (resources)
to discover (information)

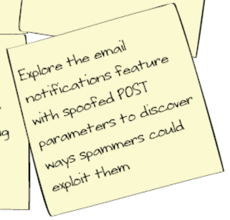

Explore editing profiles with invalid usernames to discover if there are any instances where username constraints are not enforced


Explore updating profiles with suspended accounts to discover interactions between account state and profile updates

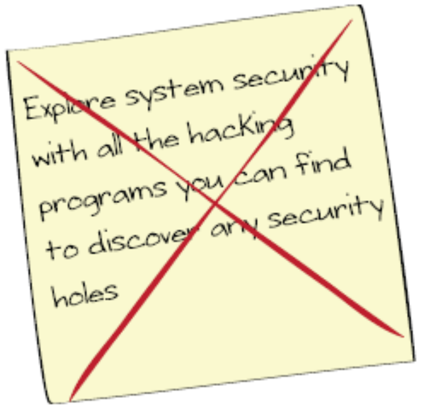

Explore messaging with privacy settings to discover circumstances where the privacy model breaks


Explore billing with variations in sequences and data to discover ways a customer could be double-billed

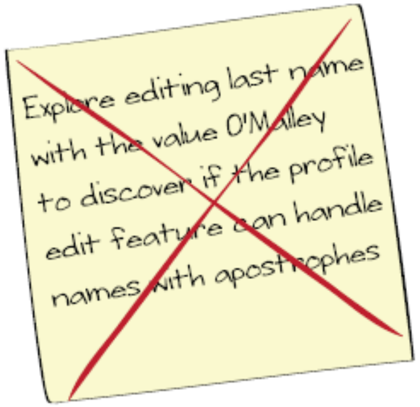

Explore catalog features with 10x # products to discover problems with browsing and searching


Explore the email notifications feature with spoofed POST parameters to discover ways spammers could exploit them

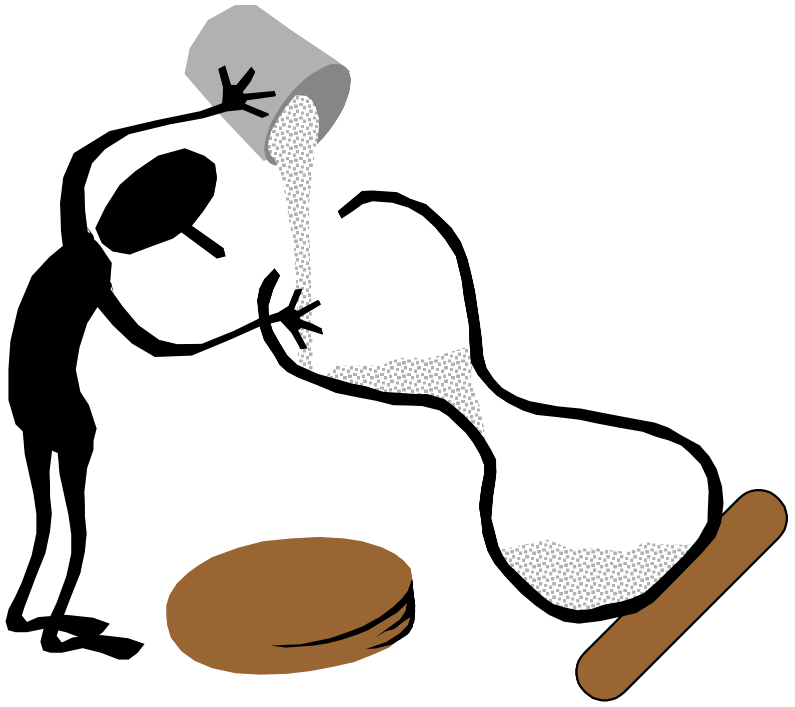
Too Broad!
(You can never explore enough to fulfill this mission.)


~~Explore system security with all the hacking programs you can find to discover any security holes~~

Too Specific!
(There isn't much to explore. Plus, it's like a weirdly worded test case.)


~~Explore editing last name with the value O'Malley to discover if the profile edit feature can handle names with apostrophes~~

Testing-Time



Session 3: What Did We Learn?

Charter creation

Requirements in exploratory testing

Depths of testing

Session 4: Exploratory Team Work

Making notes of details

Debriefing

Quality reporting

Documentation

Charter collection

- ”High-level test cases”
- You may or may not want to use them again

Data collection

- Data files – input data used for testing
- Setting up data for the testing needs – with reusability considerations

Tools collection

- Little tools that are useful – also for sharing with others

Lessons learned collection

- What was difficult for me, what others might benefit from. Not every day, but regular reflection.
- Questions and answers -collection

Most essential:
Bug reports

Secondary:
Useful for others

Rarely:
Detailed proof of testing

Debriefing: Reviewing Testing Work

Source: Jon Bach

Past	How did you spend your time?
Results	What did you find?
Obstacles	Do you need some help or tools?
Outlook	Do you think there's more to do?
Feelings	Was this task / charter reasonable?

Use PROOF to anticipate scrutiny: Testing ourselves is just as important as testing software. It has won testers more credibility, autonomy, and respect.

Example: Quality Reporting

Quality Report

	Release-prep. Builds	Incremental builds	Pace of Change
FOR EACH AREA			
Functionality area 1	🟩 1	🟩 1	↑
Functionality area 2	🟩 3	🟥 1	↑
Functionality area 3	🟨 3	🟨 2	→
Functionality area 4	🟥 2	🟥 2	↑
Functionality area 5	🟨 2	🟨 2	↓
Functionality area 6	🟨 1	🟨 2	↓
FOR THE SYSTEM			
F - functionality	🟥 2	🟥 2	↑
U - usability	🟨 1	🟨 1	↓
R - reliability	🟥 1	🟥 1	↑
P - performance	🟥 0	🟥 0	↑
S - security	🟨 1	🟨 1	↓
S - supportability	🟩 0	🟩 0	↓

Interpretation:

Numbers - coverage of testing; tester's subjective assessment as group

- 0 - we have not tested
- 1 - touched a little, but professional tester would be hesitant to call that testing
- 2 - basic coverage, normal use and misuse cases covered
- 3 - tested as per criteria of a group of professional testers

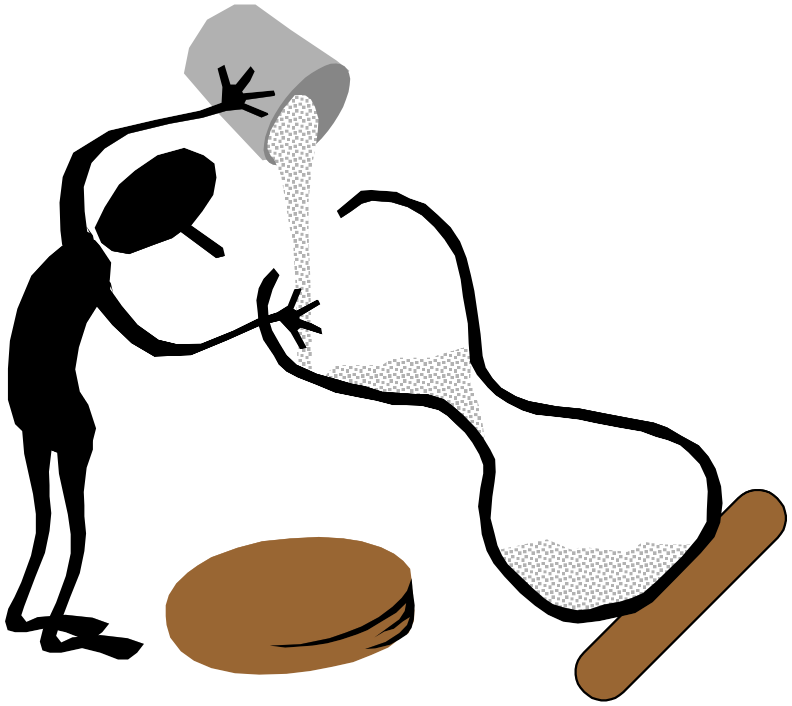
Colors - quality of the software; tersters subjective assessment as a group

- green - works, no known major problems
- yellow - some problems that cause concerns, keeps user safe and stakeholders could live with it
- red - significant problems, does not keep user safe, never show to relevant stakeholders

Arrows - amount of changes in program; joint assessment with developers

- arrow up - lots of changes, nullifies what we used to know about quality and coverage, needs to be tested again
- arrow side - some changes, controlled approach to testing still brings significant reusable information
- arrow down - stabilizing area, only small changes, testing progresses to new areas instead of repeating basics

Testing-Time



Session 4: What Did We Learn?

Making notes of details

Debriefing

Quality reporting

Session 5: Dynamics in Real Testing

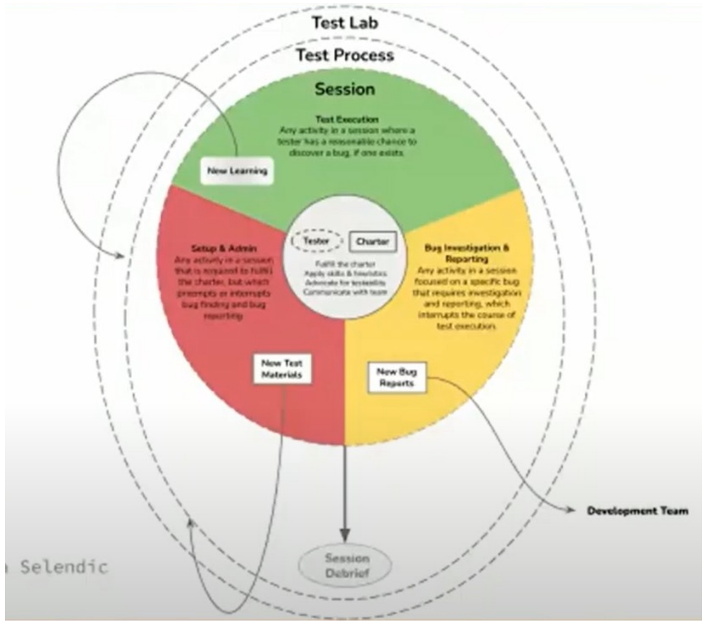
Continuing work after another tester

Intentional unrepeatability of tests

Managing Testing in Sessions

https://www.youtube.com/watch?v=tAoqScOGD-M

Session-based Test Management
See Djuka Selendic et al.



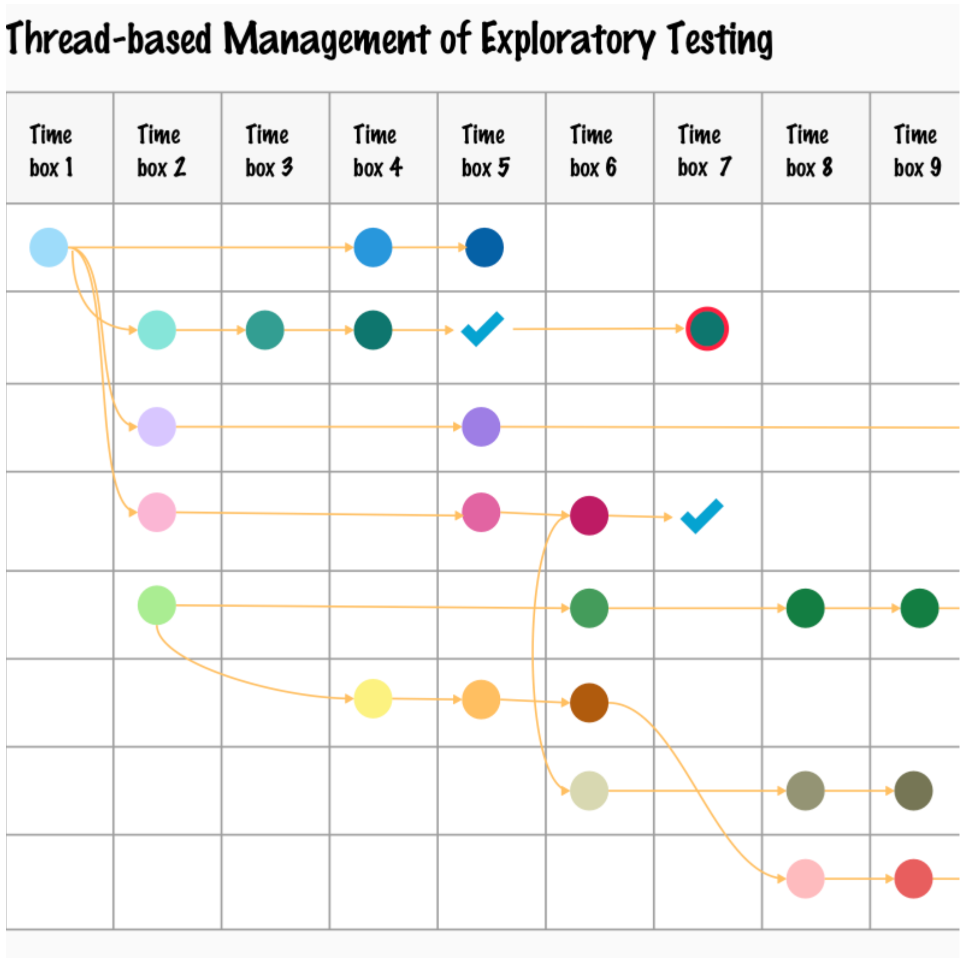
Test Lab
Test Process
Session
Test Execution
Any activity in a session where a tester has a reasonable chance to discover a bug, if one exists.
New Learning
Setup & Admin
Any activity in a session that is required to fulfil the charter, but which preempts or interrupts bug finding and bug reporting
Tester
Charter
Fulfil the charter
Apply skills & heuristics
Advocate for testability
Communicate with team
Bug Investigation & Reporting
Any activity in a session focused on a specific bug that requires investigation and reporting, which interrupts the course of test execution.
New Test Materials
New Bug Reports
Development Team
Session Debrief
Selendic

Contemporary Exploratory Testing
my practice

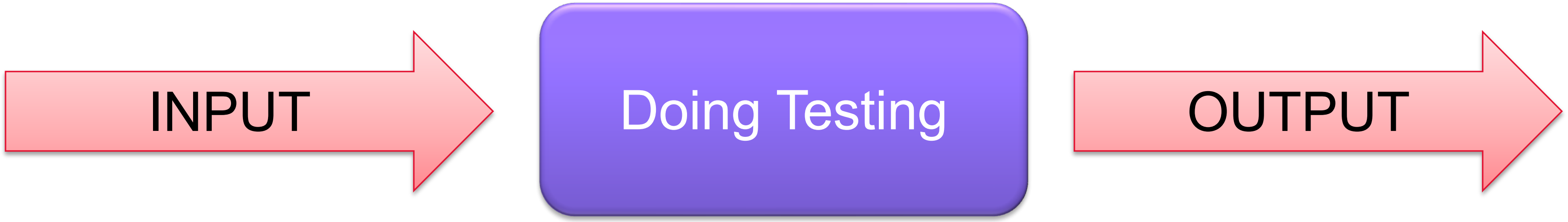
Days of work
Changes / Stories / Epics / Charters
Focus on input:
- Making space for testing
- Enabling new people

Focus on output:
- Docs to make future testing easier
- Test automation as documentation

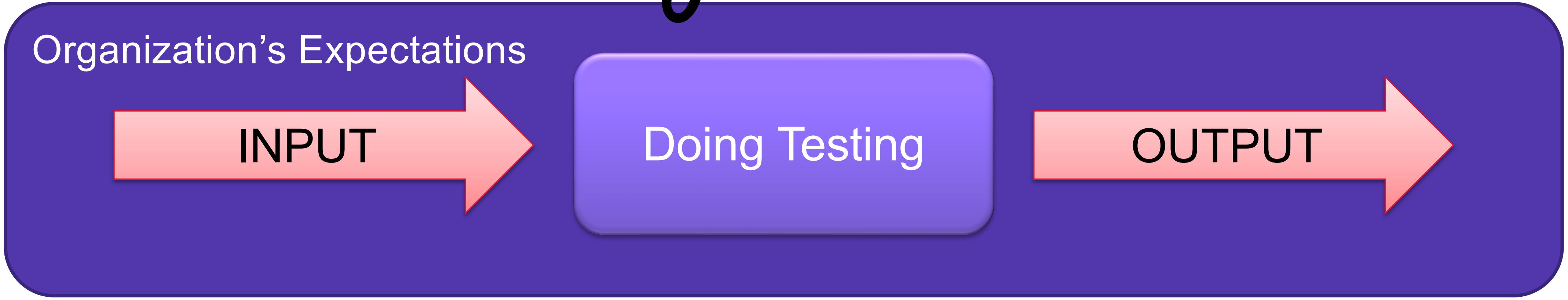
Thread-based Management of Exploratory Testing



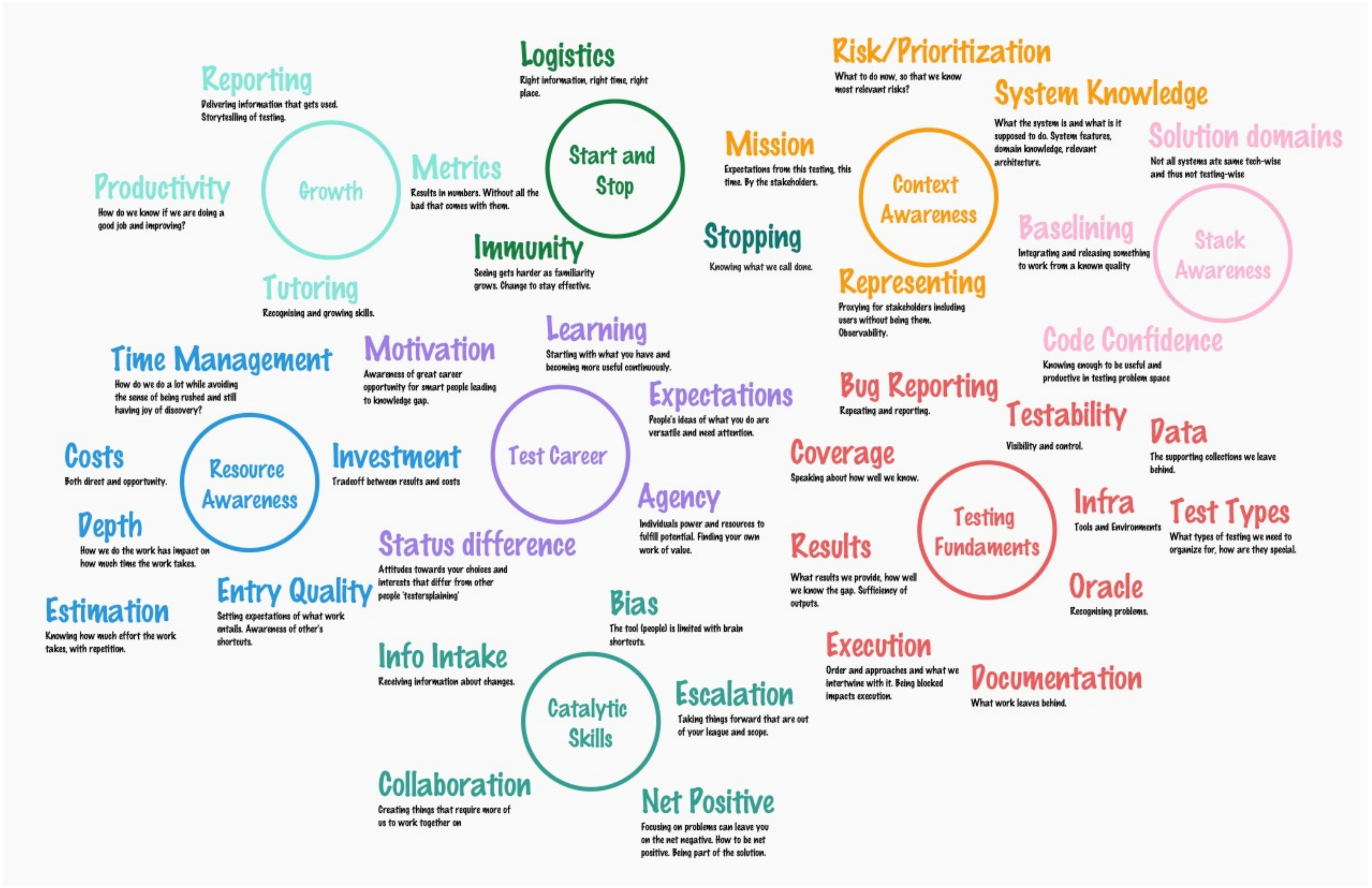
Exploratory Testing the Verb



Exploratory Testing *the Noun*



Testing Challenges



Reporting
Delivering information that gets used. Storytelling of testing.

Productivity
How do we know if we are doing a good job and improving?

Tutoring
Recognising and growing skills.

Metrics
Results in numbers. Without all the bad that comes with them.

Logistics
Right information, right time, right place.

Immunity
Seeing gets harder as familiarity grows. Change to stay effective.

Mission
Expectations from this testing, this time. By the stakeholders.

Stopping
Knowing what we call done.

Risk/Prioritization
What to do now, so that we know most relevant risks?

Representing
Proxying for stakeholders including users without being them. Observability.

System Knowledge
What the system is and what is it supposed to do. System features, domain knowledge, relevant architecture.

Solution domains
Not all systems ate same tech-wise and thus not testing-wise

Baselining
Integrating and releasing something to work from a known quality

Code Confidence
Knowing enough to be useful and productive in testing problem space

Time Management
How do we do a lot while avoiding the sense of being rushed and still having joy of discovery?

Costs
Both direct and opportunity.

Investment
Tradeoff between results and costs

Depth
How we do the work has impact on how much time the work takes.

Estimation
Knowing how much effort the work takes, with repetition.

Entry Quality
Setting expectations of what work entails. Awareness of other's shortcuts.

Motivation
Awareness of great career opportunity for smart people leading to knowledge gap.

Learning
Starting with what you have and becoming more useful continuously.

Expectations
People's ideas of what you do are versatile and need attention.

Agency
Individuals power and resources to fulfill potential. Finding your own work of value.

Status difference
Attitudes towards your choices and interests that differ from other people 'testersplaining'

Bias
The tool (people) is limited with brain shortcuts.

Info Intake
Receiving information about changes.

Escalation
Taking things forward that are out of your league and scope.

Collaboration
Creating things that require more of us to work together on

Net Positive
Focusing on problems can leave you on the net negative. How to be net positive. Being part of the solution.

Bug Reporting
Repeating and reporting.

Coverage
Speaking about how well we know.

Testability
Visibility and control.

Data
The supporting collections we leave behind.

Results
What results we provide, how well we know the gap. Sufficiency of outputs.

Infra
Tools and Environments

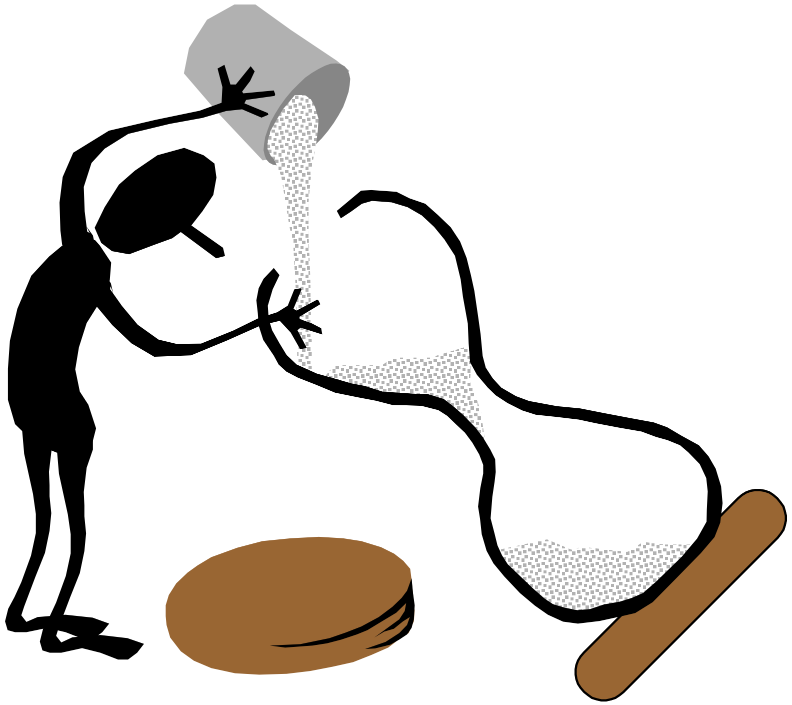
Test Types
What types of testing we need to organize for, how are they special.

Oracle
Recognising problems.

Execution
Order and approaches and what we intertwine with it. Being blocked impacts execution.

Documentation
What work leaves behind.

Testing-Time

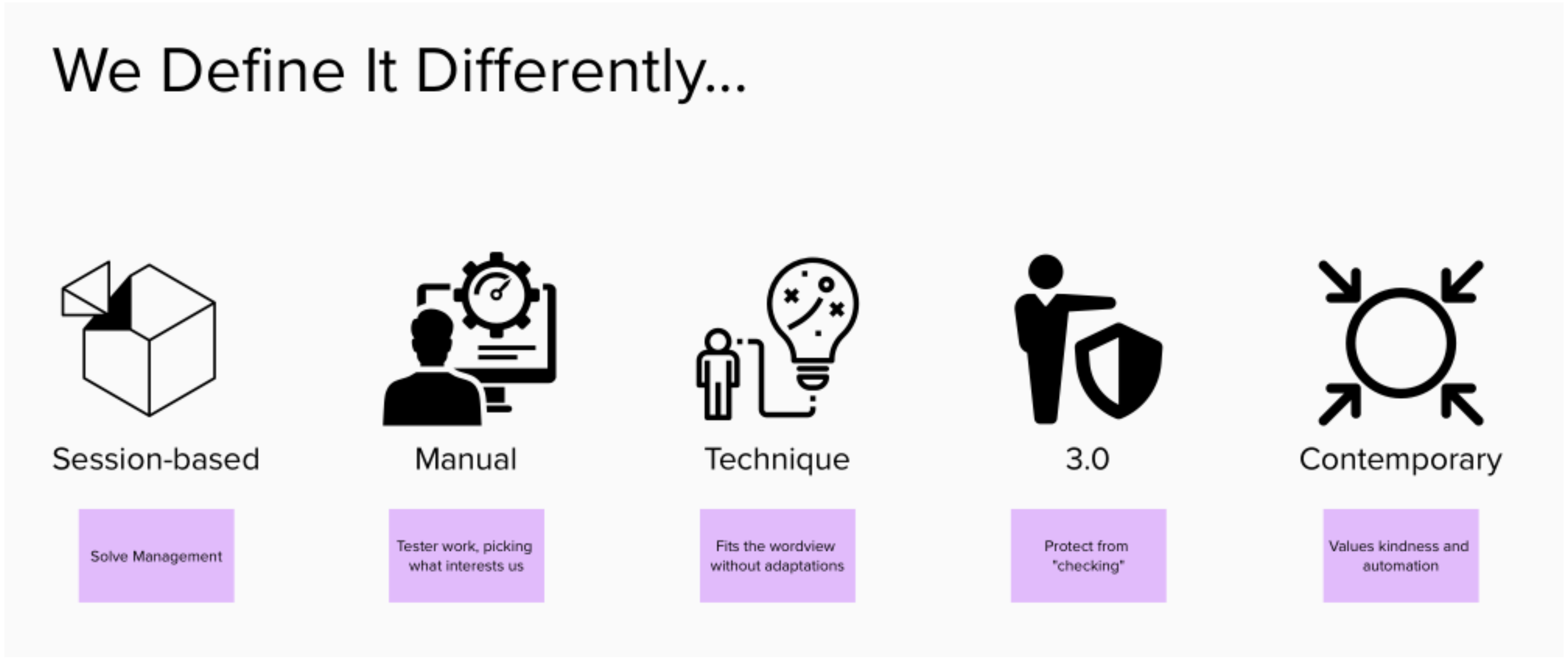


Session 5: What Did We Learn?

Continuing work after another tester

Intentional unrepeatability of tests

What is exploratory testing



We Define It Differently...

Session-based

Solve Management

Manual

Tester work, picking what interests us

Technique

Fits the wordview without adaptations

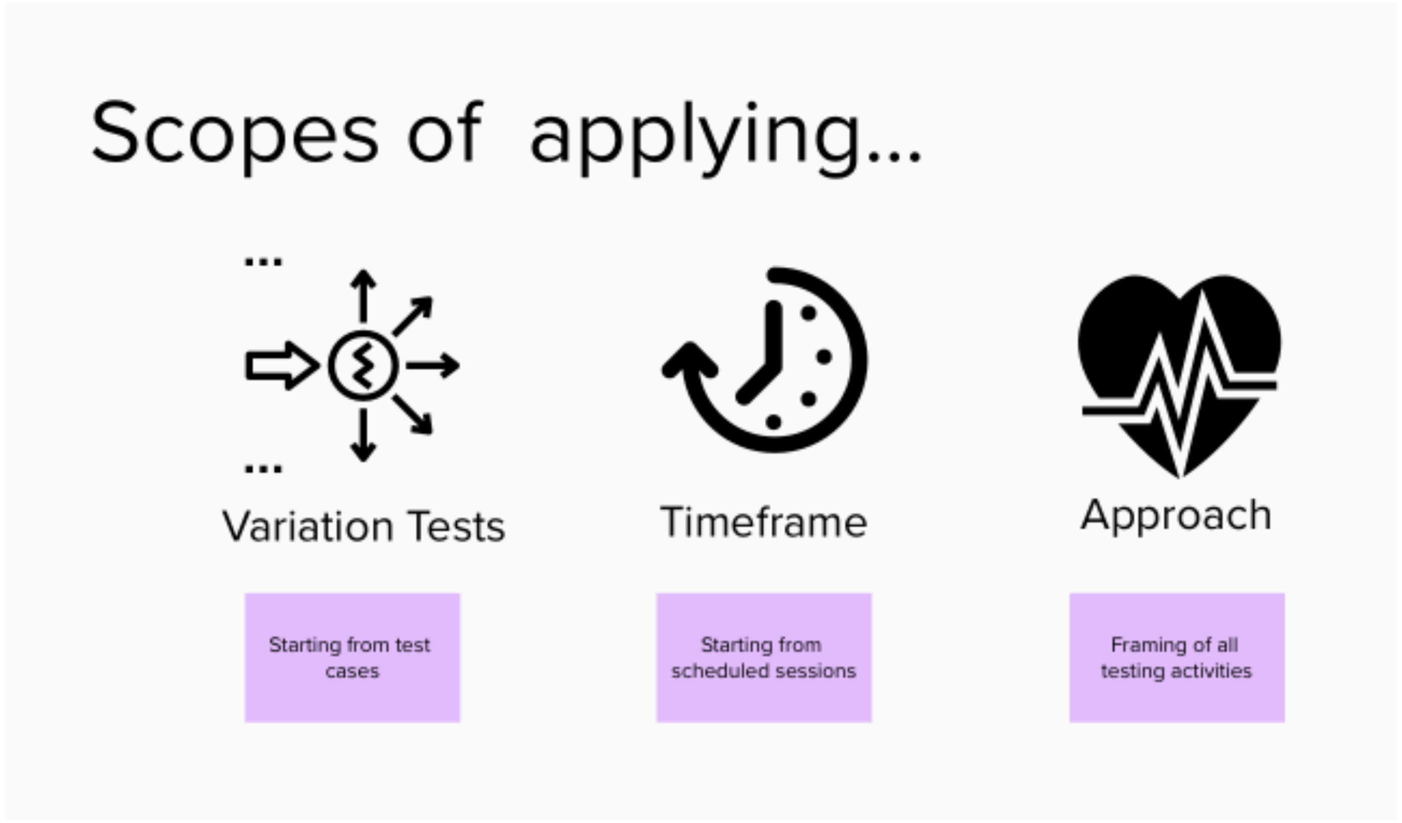
3.0

Protect from "checking"

Contemporary

Values kindness and automation

What is exploratory testing



Scopes of applying...

Variation Tests

Starting from test cases

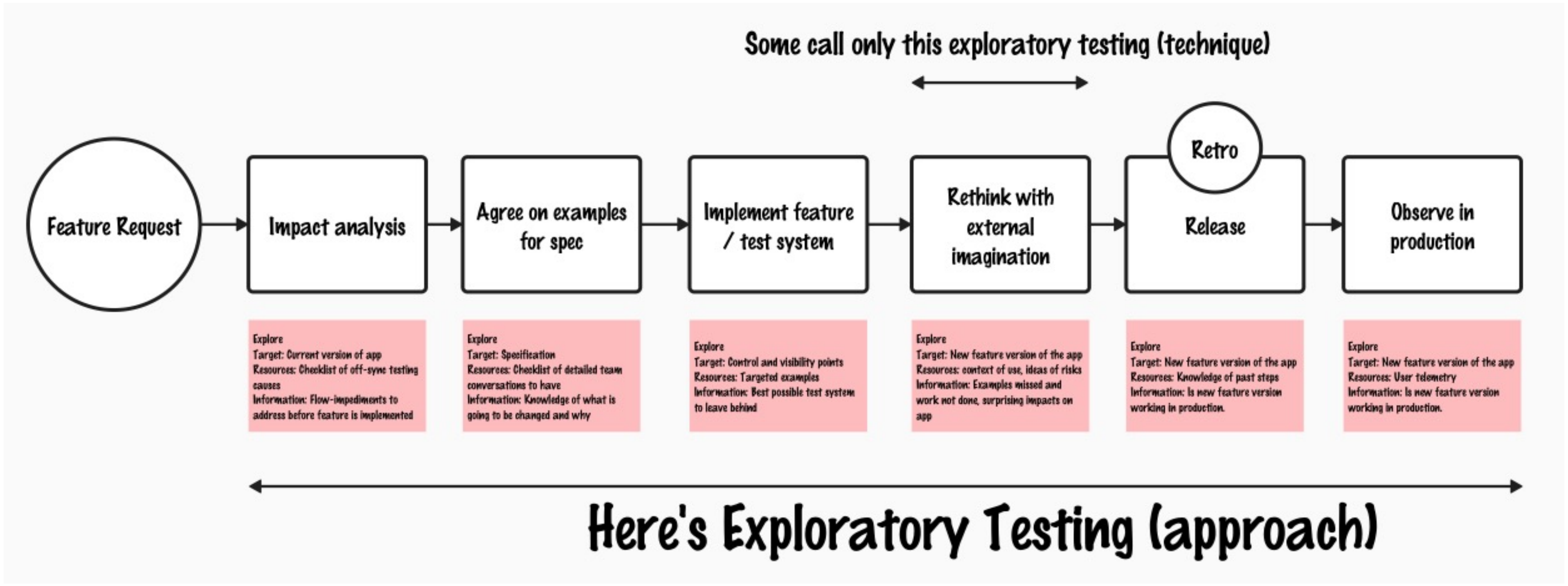
Timeframe

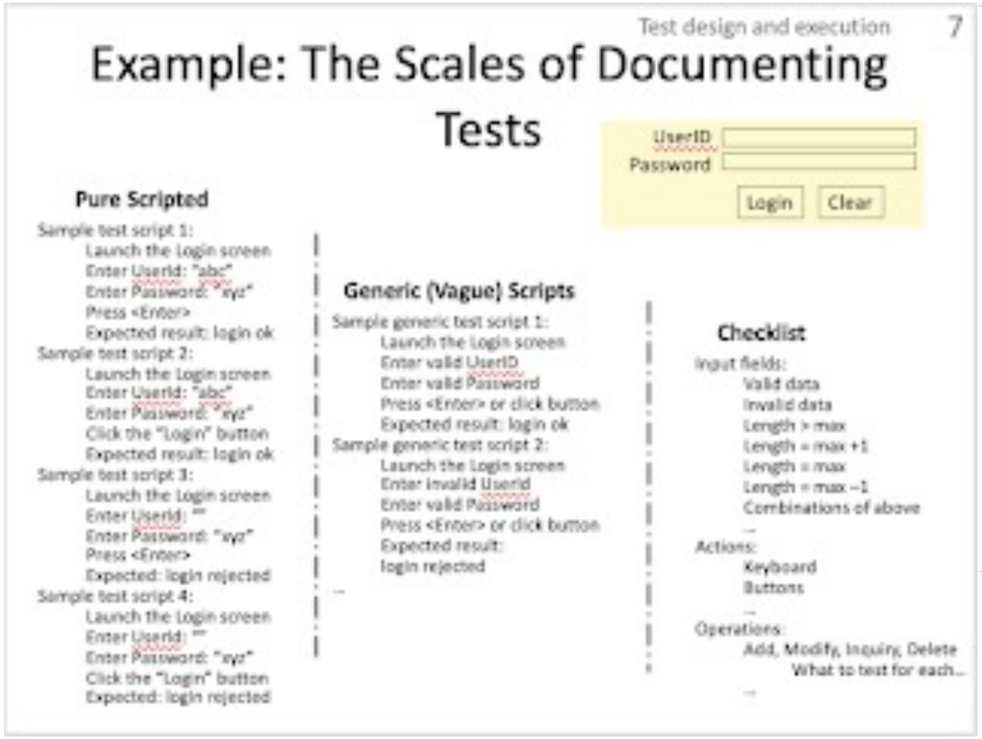
Starting from scheduled sessions

Approach

Framing of all testing activities

Where is exploratory testing


Some call only this exploratory testing (technique)
Feature Request
Impact analysis
Agree on examples for spec
Implement feature / test system
Rethink with external imagination
Retro
Release
Observe in production
Explore
Target: Current version of app
Resources: Checklist of off-sync testing causes
Information: Flow-impediments to address before feature is implemented
Explore
Target: Specification
Resources: Checklist of detailed team conversations to have
Information: Knowledge of what is going to be changed and why
Explore
Target: Control and visibility points
Resources: Targeted examples
Information: Best possible test system to leave behind
Explore
Target: New feature version of the app
Resources: context of use, ideas of risks
Information: Examples missed and work not done, surprising impacts on app
Explore
Target: New feature version of the app
Resources: Knowledge of past steps
Information: Is new feature version working in production.
Explore
Target: New feature version of the app
Resources: User telemetry
Information: Is new feature version working in production.
Here's Exploratory Testing (approach)


Test design and execution 7

Example: The Scales of Documenting Tests

UserID
Password
Login
Clear

Pure Scripted

Sample test script 1:
  Launch the Login screen
  Enter UserId: "abc"
  Enter Password: "xyz"
  Press <Enter>
  Expected result: login ok
Sample test script 2:
  Launch the Login screen
  Enter UserId: "abc"
  Enter Password: "xyz"
  Click the "Login" button
  Expected result: login ok
Sample test script 3:
  Launch the Login screen
  Enter UserId: ""
  Enter Password: "xyz"
  Press <Enter>
  Expected: login rejected
Sample test script 4:
  Launch the Login screen
  Enter UserId: ""
  Enter Password: "xyz"
  Click the "Login" button
  Expected: login rejected

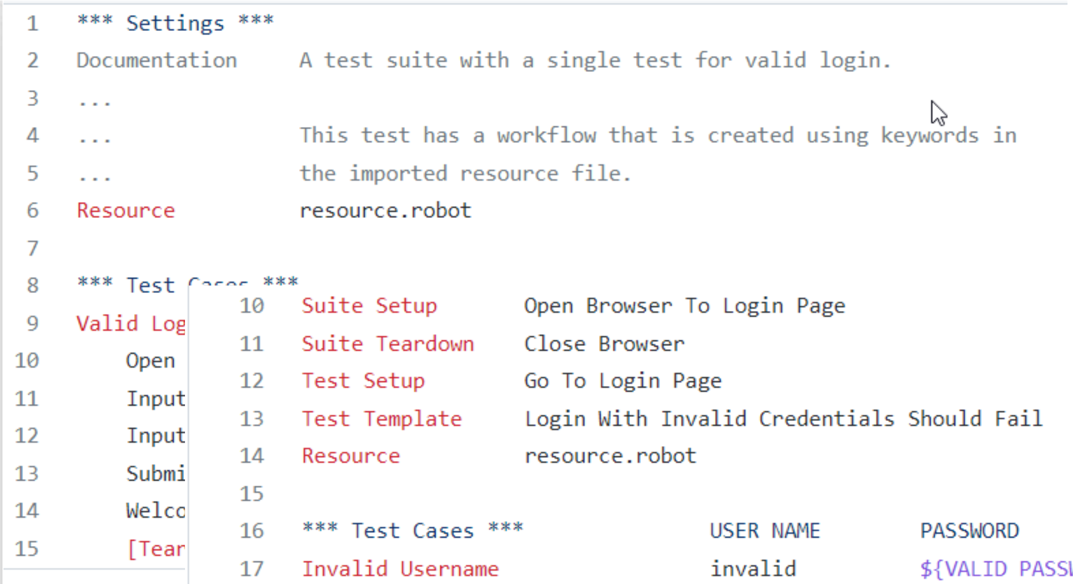
Generic (Vague) Scripts

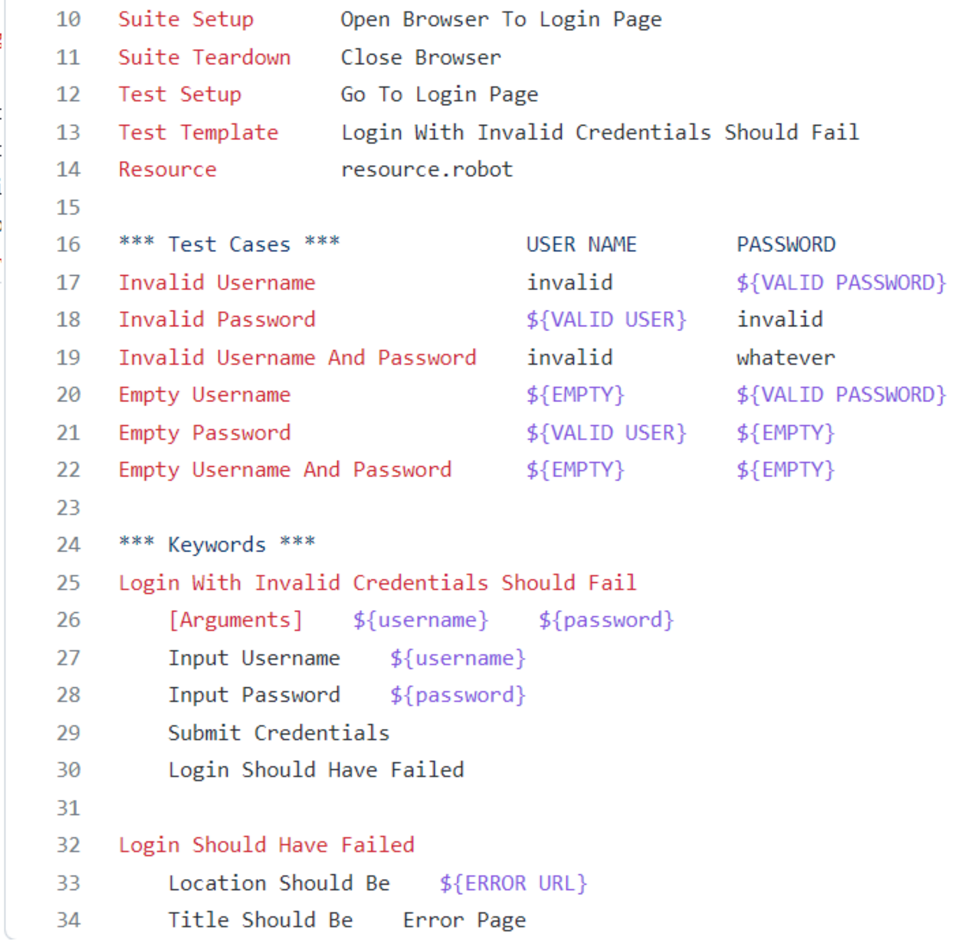
Sample generic test script 1:
  Launch the Login screen
  Enter valid UserID
  Enter valid Password
  Press <Enter> or click button
  Expected result: login ok
Sample generic test script 2:
  Launch the Login screen
  Enter invalid UserId
  Enter valid Password
  Press <Enter> or click button
  Expected result:
  login rejected
...

Checklist

Input fields:
  Valid data
  Invalid data
  Length > max
  Length = max +1
  Length = max
  Length = max –1
  Combinations of above
  ...
Actions:
  Keyboard
  Buttons
  ...
Operations:
  Add, Modify, Inquiry, Delete
    What to test for each...
  ...

My course on Test Case Design in 200x


```
*** Settings ***
Documentation     A test suite with a single test for valid login.
...
...               This test has a workflow that is created using keywords in
...               the imported resource file.
Resource          resource.robot

*** Test Cases ***
Valid Log
    Open
    Input
    Input
    Submi
    Welco
    [Tear
```


```
Suite Setup       Open Browser To Login Page
Suite Teardown    Close Browser
Test Setup        Go To Login Page
Test Template     Login With Invalid Credentials Should Fail
Resource          resource.robot

*** Test Cases ***               USER NAME        PASSWORD
Invalid Username                 invalid          ${VALID PASSWORD}
Invalid Password                 ${VALID USER}    invalid
Invalid Username And Password    invalid          whatever
Empty Username                   ${EMPTY}         ${VALID PASSWORD}
Empty Password                   ${VALID USER}    ${EMPTY}
Empty Username And Password      ${EMPTY}         ${EMPTY}

*** Keywords ***
Login With Invalid Credentials Should Fail
    [Arguments]    ${username}    ${password}
    Input Username    ${username}
    Input Password    ${password}
    Submit Credentials
    Login Should Have Failed

Login Should Have Failed
    Location Should Be    ${ERROR URL}
    Title Should Be    Error Page
```

The Login Example

http://robotframework.org/WebDemo/

Exploring Discovered Problems

Complementing functions. While it did log me in, it did not log me out but pretended it did.

Performance. While it did log me in, it took its time.

Session length. While it did log me in, the two different parts of it disagreed on how long I was supposed to be in, resulting in fascinating symptoms while being logged in long enough combined with selected use of features.

Concurrency. While it did log me in, it also logged me in a second time. And when it did so, it got really confused on which one of me did what.

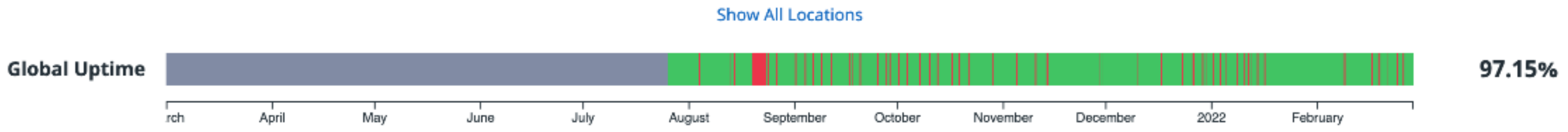
Security controls. While I could log in, the scenarios around forgetting passwords weren't quite what I would have expected.

Multi-user. While it logged me in, it did not log me out fully and sharing a computer for two different usernames was interesting experience.

Browser functions. While it logged me in, it did not play nicely with browser functions remembering usernames and passwords and password managers.

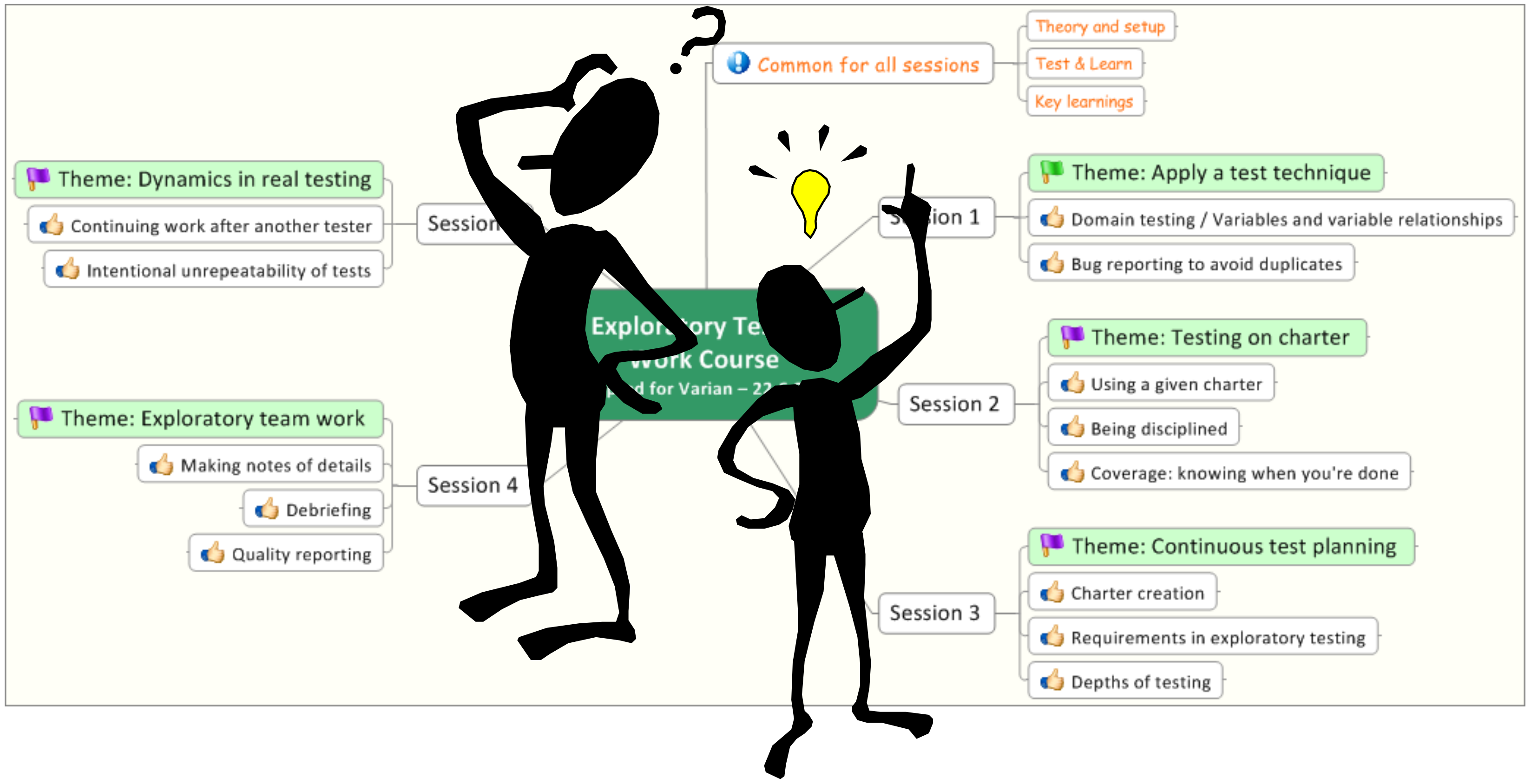
Environment. While it worked on test environment, it stopped working on test environment when a component got upgraded. And it did not work in production environment without ensuring it was setup (and tested) before depending on it.

Cookies. We save stuff available in previous versions.



360d x 24h/d x 2.85% = 246,24h

Exploratory Testing Work Course



Common for all sessions
Theory and setup
Test & Learn
Key learnings
Session 1
Theme: Apply a test technique
Domain testing / Variables and variable relationships
Bug reporting to avoid duplicates
Session 2
Theme: Testing on charter
Using a given charter
Being disciplined
Coverage: knowing when you're done
Session 3
Theme: Continuous test planning
Charter creation
Requirements in exploratory testing
Depths of testing
Session 4
Theme: Exploratory team work
Making notes of details
Debriefing
Quality reporting
Session
Theme: Dynamics in real testing
Continuing work after another tester
Intentional unrepeatability of tests
Explor ory Te
Work Course
d for Varian – 22

Insights you can act on

Founded in 1976, CGI is among the largest IT and business consulting services firms in the world.

We are insights-driven and outcomes-focused to help accelerate returns on your investments. Across hundreds of locations worldwide, we provide comprehensive, scalable and sustainable IT and business consulting services that are informed globally and delivered locally.

cgi.com