Departamento de Lenguajes y Sistemas Informáticos

Graphical User Interface (GUI) Testing

Sergio Segura
sergiosegura@us.es

1

Path

- Introduction
- Place in the software development lifecycle
- Functional vs non-functional testing
- Test case execution techniques
- Test case design techniques
- Tools
- Best practices
- Challenges

2

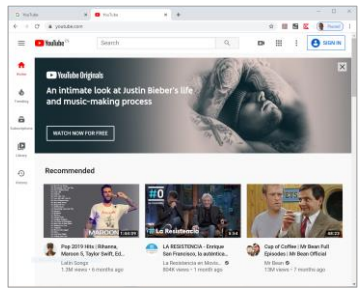
Path

- **Introduction**
- Place in the software development lifecycle
- Functional vs non-functional testing
- Test case execution techniques
- Test case design techniques
- Tools
- Best practices
- Challenges

3

Introduction

Graphical user interface (GUI) testing is the process of testing a product's graphical user interface (buttons, icons, forms, etc.) to ensure it meets its specifications.



- Does log in work as expected?
- Do GUI elements have the correct size and position?
- Are error messages displayed correctly?
- Do users find the GUI attractive?
- Do users find the GUI intuitive?
- …

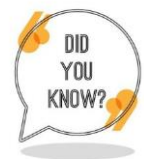
4

Introduction

Do GUI elements work and look as expected in different platforms, devices, and screen resolutions?

Image source: https://www.perfecto.io/

5



Netflix's streaming service is available on more than 800 different device types!

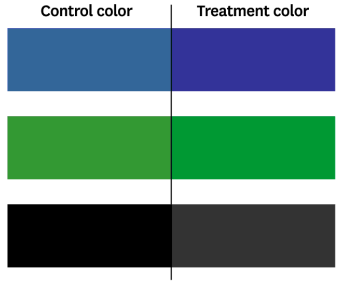
NETFLIX

6

Introduction
Why is it important?

Small Changes with a Huge Impact

Bing's experiments showed that slightly darker blues and greens in titles and a slightly lighter black in captions improved the users' experience. When rolled out to all users, the color changes boosted revenue by more than $10 million annually.



FROM "THE SURPRISING POWER OF ONLINE EXPERIMENTS," SEPTEMBER–OCTOBER 2017, BY RON KOHAVI AND STEFAN THOMKE © HBR.ORG

Figure 1: Ads with site link experiment. Treatment (bottom) has site links. The difference might not be obvious at first but it is worth tens of millions of dollars

FROM: Ron Kohavi, Alex Deng, Brian Frasca, Toby Walker, Ya Xu, and Nils Pohlmann. 2013. Online controlled experiments at large scale. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (*KDD '13*). Association for Computing Machinery, New York, NY, USA, 1168–1176. DOI:https://doi.org/10.1145/2487575.2488217

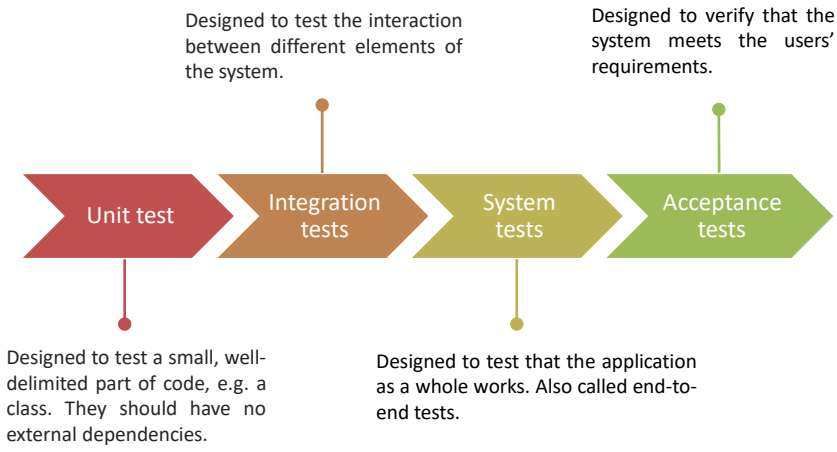
7

Path

- Introduction
- **Place in the software development lifecycle**
- Functional vs non-functional testing
- Test case execution techniques
- Test case design techniques
- Tools
- Best practices
- Challenges

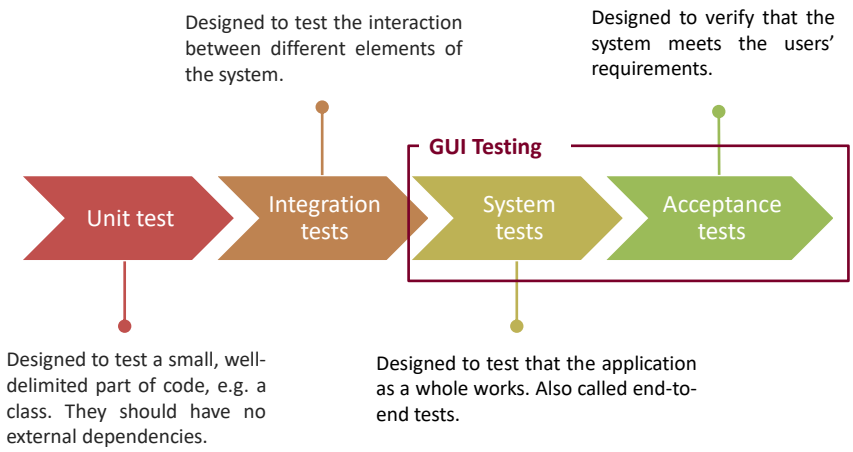
8

Place in the software development lifecycle



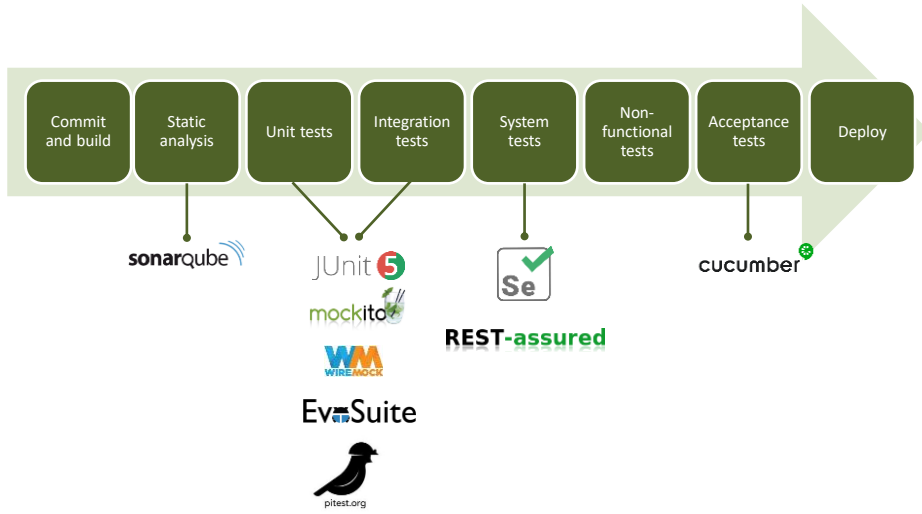
9

Place in the software development lifecycle



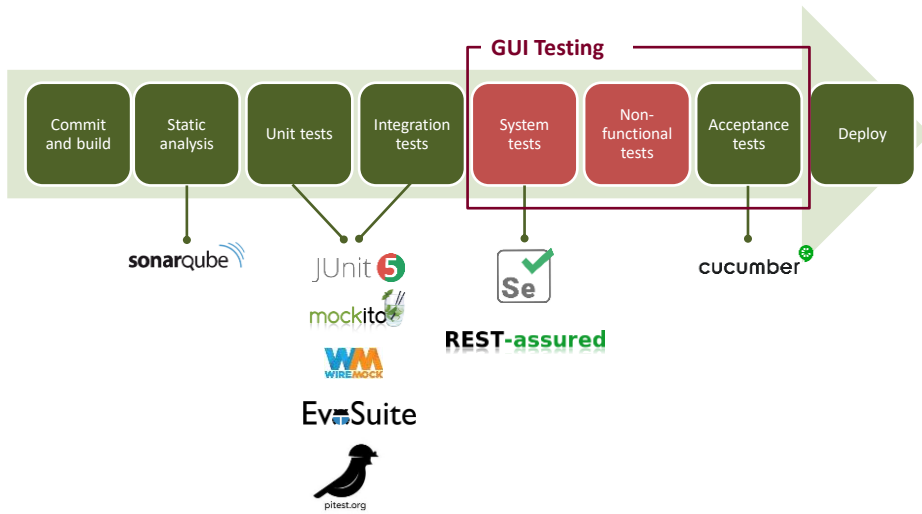
10

Place in the software development lifecycle



11

Place in the software development lifecycle



12

Path

- Introduction
- Place in the software development lifecycle
- **Functional vs non-functional testing**
- Test case execution techniques
- Test case design techniques
- Tools
- Best practices
- Challenges

13

Functional vs Non-functional testing

Functional test

They aim to detect faults related to system functionality.

- Does log in work as expected?
- Is the workflow correct?
- Is the menu showing all the necessary items?

Non-functional tests

They aim to detect bugs related to non-functional aspects such as performance, usability, security, etc.

- Is the GUI intuitive?
- Is the GUI accessible?
- Are asynchronous calls taking too long?

14

Path

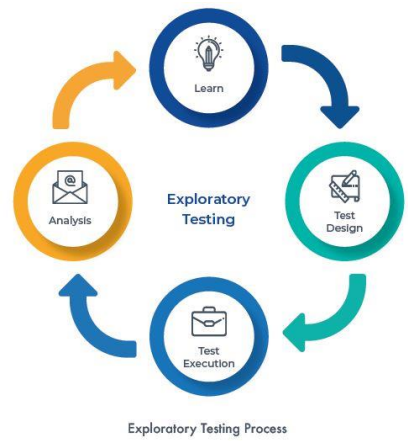
- Introduction
- Place in the software development lifecycle
- Functional vs non-functional testing
- **Test case execution techniques**
  - **Exploratory testing**
  - **Scripted testing**
  - **User-driven testing**
- Test case design techniques
- Tools
- Best practices
- Challenges

15

Test case execution techniques
Exploratory testing

Exploratory testing is about *exploring* the software without a previous plan. As the tester learns how it works, (s)he design and execute new test cases based on his/her previous experience and creativity.

"What happens if I do this?"



Exploratory Testing Process

Image's source: https://www.rapidvaluesolutions.com/whitepapers/exploratory-testing/

16

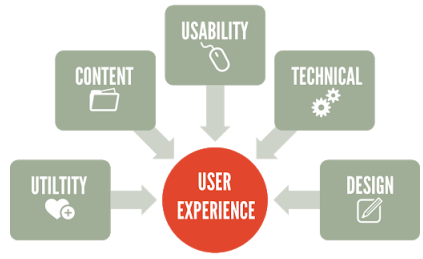
Test case execution techniques
Scripted testing

- **Scripted testing** is about executing pre-planned scripts to uncover defects and verify that an application meets its requirements.
- The script defines the inputs that the tester introduces on each screen (click events, submitting forms, etc.) and the expected outcome of each entry.
- Scripted testing may be performed manually or supported by test automation.

17

Test case execution techniques
User-driven testing

In **user-driven testing**, actual end-users or user representatives evaluate an application for its usability, visual appeal, and ability to meet their needs. For example, users can be asked to use the application and express their opinion through questionnaires.



Image's source: http://www.resounddigital.com/blog/website-visitor-surveys-the-questions-you-need-to-ask.html

18

Test case execution techniques

User-driven testing

19

Path

- Introduction
- Place in the software development lifecycle
- Functional vs non-functional testing
- Test case execution techniques
- **Test case design techniques**
  - **Risk-based testing**
  - **Model-based testing**
  - **Random testing**
  - **Metamorphic testing**
- Tools
- Best practices
- Challenges

20

Test case design techniques
Risk-based testing

Risk-based testing. Testing focuses on the functionality which has the highest impact and probability of failure.

Frequency	Impact			
	Catastrophic	Critical	Moderate	Minor
Constant	Critical	Critical	High	High
Frequent	Critical	High	High	Medium
Occasional	High	High	Medium	Medium
Infrequent	High	Medium	Medium	Low
Rare	Medium	Medium	Low	Low

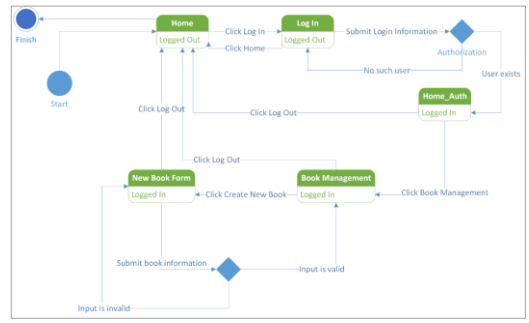
Critical High Medium Low

Image's source: https://www.ranorex.com/resources/testing-wiki/gui-testing/

21

Test case design techniques
Model-based testing

In **model-based testing** test cases−inputs and expected outputs− are derived from a model of the system under test, manually or automatically. A **model** is a kind of specification, which models some aspect of the system's behavior in a simplified, abstract way, e.g. state machine. Coverage metrics can be used to decide when to stop testing.



Image's source: https://www.inflectra.com/support/knowledgebase/kb284.aspx

22

Test case design techniques
Random testing

Random testing is about testing the software with (pseudo) random inputs. Since an automated oracle is not usually available, tests are mostly used to detect crashes, e.g. unhandled exceptions.

Android Developers › Android Studio › User guide

☆☆☆☆☆

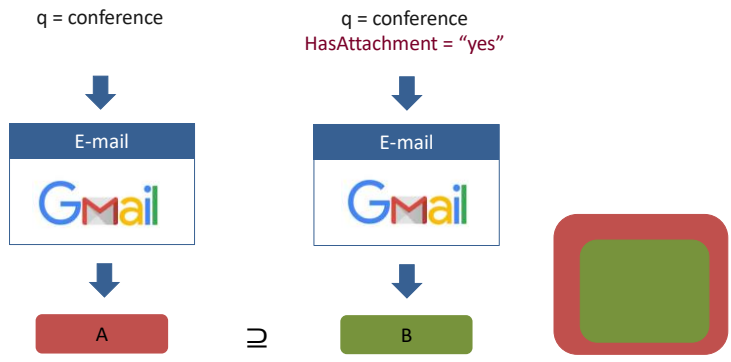
UI/Application Exerciser Monkey

The Monkey is a program that runs on your emulator or device and generates pseudo-random streams of user events such as clicks, touches, or gestures, as well as a number of system-level events. You can use the Monkey to stress-test applications that you are developing, in a random yet repeatable manner.

23

Test case design techniques
Metamorphic testing

Metamorphic testing aims to detect bugs by checking expected relations (called metamorphic relations) between the inputs and outputs of two or more test cases.



24

Path

- Introduction
- Place in the software development lifecycle
- Functional vs non-functional testing
- Test case execution techniques
- Test case design techniques
- **Tools**
- Best practices
- Challenges

25

Tools

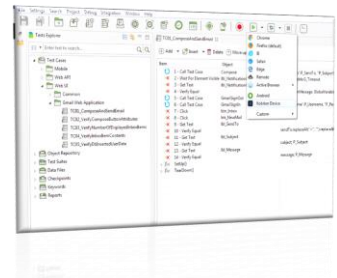
Most GUI testing tools follow a **Record-and-Replay** strategy. The user's actions on the GUI (e.g. clicking, typing, etc.) are recorded as test steps during Record, and recorded steps are then executed on the application under test during Replay. This can be done visually (codeless) or programmatically.

26

Tools

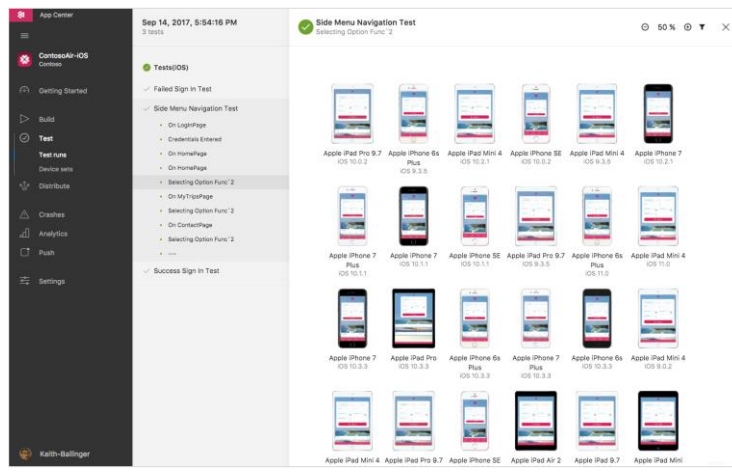
Three types of tools:

1. **Browser extensions.** Selenium IDE, Katalon Recorder, etc.
2. **Visual IDEs**. Katalon Studio, Ranorex, TestArchitect, etc.
3. **Code libraries** for JUnit, NUnit, TestNG, etc.



27

Tools
Cross-platform GUI testing in the cloud



29

Path

- Introduction
- Place in the software development lifecycle
- Functional vs non-functional testing
- Test case execution techniques
- Test case design techniques
- Tools
- **Best practices**
- Challenges

30

Best practices

- **Separate test data from test cases.**
  Example: Using a CSV files to store pairs of username and password.
- **Separate the location of GUI elements from test cases.**
  Example: Saving the location of the login button in a reusable test object.
- **Write positive and negative test cases.**
  Example: Entering a valid (positive) and invalid (negative) credit card number.
- **Keep test cases modular.**
  Example: Log in, log out, add item to shopping cart, cancel order…
- **Use standard test data design techniques.**
  Example: Equivalence partitioning + boundary values.

31

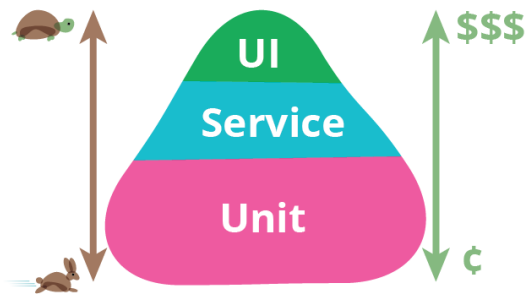
Path

- Introduction
- Place in the software development lifecycle
- Functional vs non-functional testing
- Test case execution techniques
- Test case design techniques
- Tools
- Best practices
- **Challenges**

32

Challenges

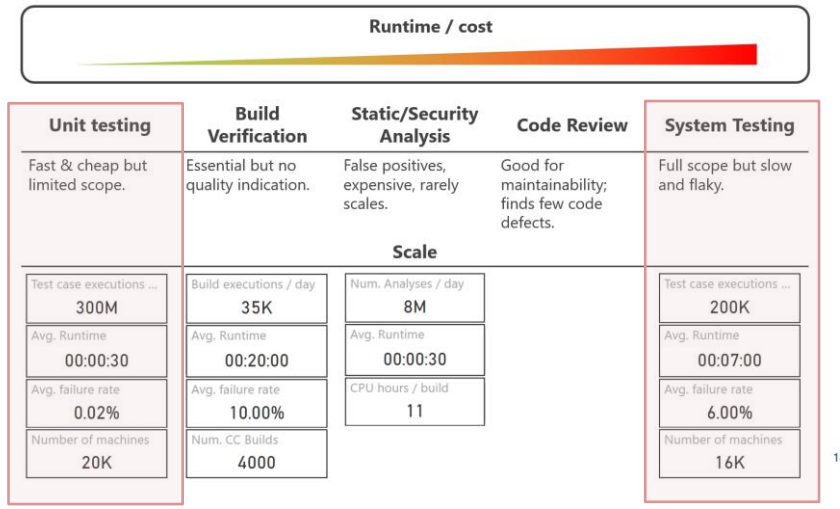
- **GUI tests are costly and slow.**



Source: https://martinfowler.com/bliki/TestPyramid.html

33

Relations and orders not universally accurate/correct. Only for illustration purposes.

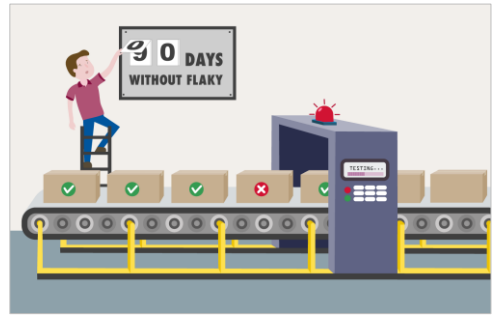


K. Herzig, "Testing and continuous integration at scale: limits, costs, and expectations," in *Proceedings of the 11th international workshop on search-based software testing*, New York, NY, USA, 2018, p. 38–38

34

Challenges

- **GUI tests are fragile.**
  Any small change in the GUI is likely to make them fail.
- **GUI tests are often flaky.**
  An expected pop-up or a slow asynchronous response could make tests fail erratically.



35

References

- GUI Testing. The Beginner's Guide for User Interface (UI) Testing
https://www.ranorex.com/resources/testing-wiki/gui-testing/

- GUI Testing Tutorial: User Interface (UI) TestCases with Examples
https://www.guru99.com/gui-testing.html

- GUI Testing
https://www.professionalqa.com/gui-testing

- Katalon Docs
https://docs.katalon.com/katalon-studio/docs/index.html

36

References

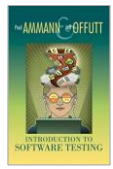
- The Practical Test Pyramid
https://martinfowler.com/articles/practical-test-pyramid.html

- ISO/IEC/IEEE 29119
http://www.softwaretestingstandard.org/

- ISTQB
https://www.istqb.org/

37

References



P. Ammann and J. Offutt. *Introduction to Software Testing. Cambridge University Press*. 2008. (Second edition available, 2016)

G. Fraser and J. M. Rojas. Software Testing. Handbook of Software Engineering. Springer. 2019.

38

Disclaimer and Terms of Use

All material displayed on this presentation is for teaching and personal use only.

Many of the images that have been used in the presentation are Royalty Free images taken from http://www.everystockphoto.com/. Other images have been sourced directly from the Public domain, from where in most cases it is unclear whether copyright has been explicitly claimed. Our intention is not to infringe any artist's copyright, whether written or visual. We do not claim ownership of any image that has been freely obtained from the public domain. In the event that we have freely obtained an image or quotation that has been placed in the public domain and in doing so have inadvertently used a copyrighted image without the copyright holder's express permission we ask that the copyright holder writes to us directly, upon which we will contact the copyright holder to request full written permission to use the quote or images.

39