WIZELINE

www.wizeline.com

Automation Overview

QA Automation Certification - Week 1

Agenda

- Understanding ROI of Automation
- What Should be Automated
- Automation Pyramid
- Test Levels
    - Functional UI Automation
    - Data Testing
    - Performance Testing
    - Load Testing
    - API Testing
    - Layout Testing
    - Mobile Testing
    - Security Testing
    - Pen Testing

Understanding ROI of Automation

Test automation at all levels of testing occurs in many Agile teams. This can mean that testers spend time **creating**, **executing**, **monitoring**, and **maintaining** automated tests and results.

Because of the heavy use of **automation**, a higher percentage of the manual testing on Agile projects tends to be done using experience-based techniques such as software attacks, exploratory testing, and error guessing.

WIZELINE

Test Automation Software is the best way to increase the effectiveness, efficiency and coverage of your software testing

-ISTQB

Automated Testing of Agile Projects

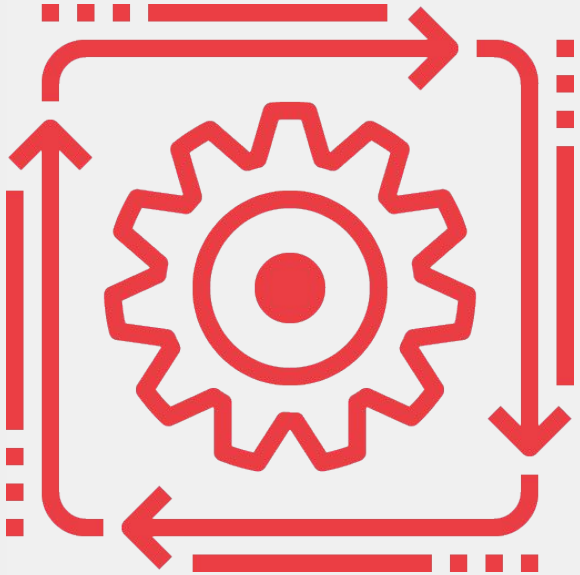
In an **Agile project**, as each **iteration** completes, the product grows. Therefore, the scope of testing also increases.

Since responding **to change** is a key Agile principle, changes can also be made to previously delivered features to meet business needs.

In order to maintain velocity without incurring a large amount of technical debt, it's critical that teams **invest in test automation** at all test levels as early as possible.

Why do We Need to Automate?

- Manual testing for all workflows, all negative test scenarios is not affordable on time and cost.
- No human intervention needed.
- Maintain a high standard of quality at all points along the pipeline.
- Wider test coverage.
- Reusable test scripts.
- Increases efficiency.
- Delivery time can be achieved.



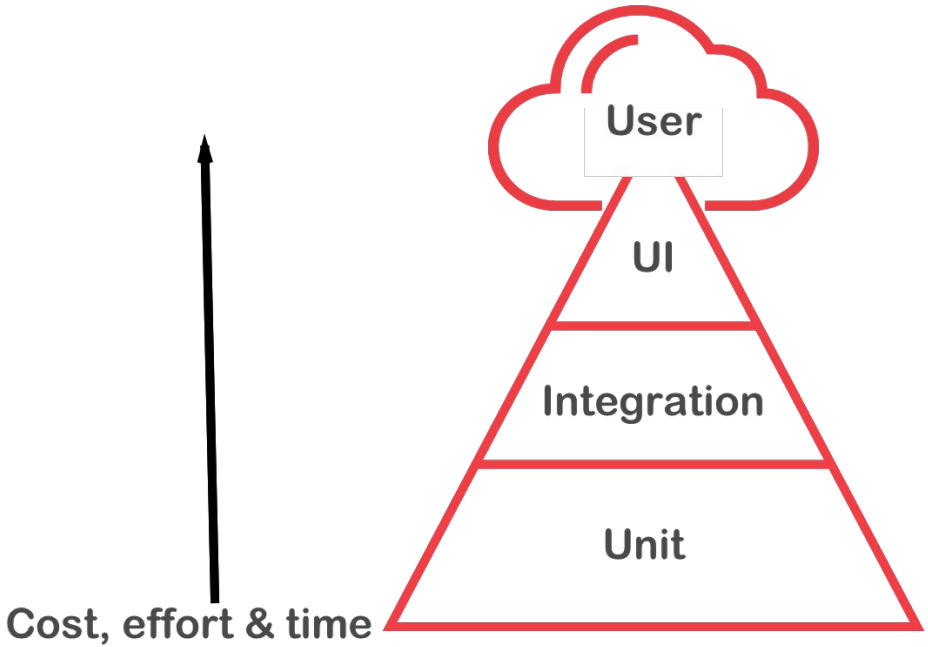
Regression Tests

Along with testing the code changes made in the current iteration, testers also need to verify no **regression** has been introduced on features that were developed and tested in previous iterations.

Regression tests are tied to human error. It's tedious to have to check the same thing again!

Test automation consists of a machine being able to execute the test cases automatically.

Automation Pyramid



Automation Ice Cream “Anti-Pattern”

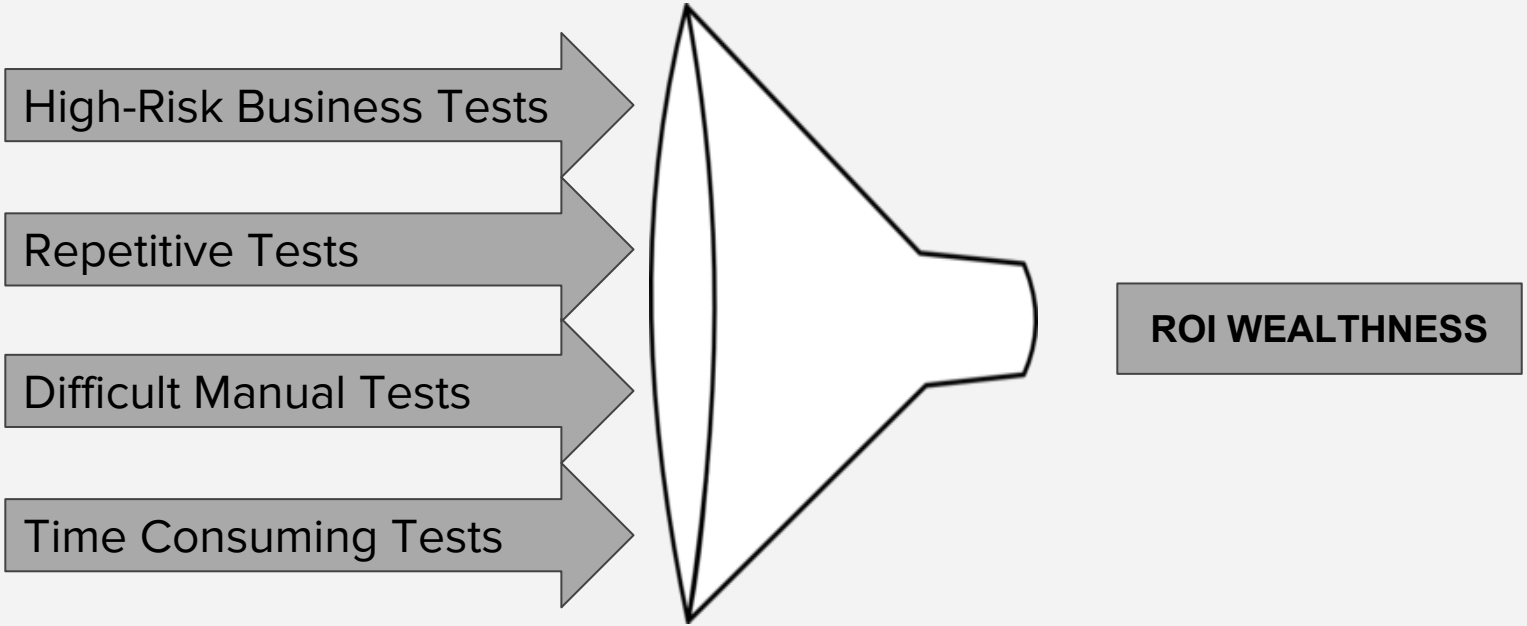
Anti-Pattern Effects

- Defect detection cycle is focused on finding issues.
- Increases regression cycle time.
- Lost of trust in automation suite.
- Delivery time increases.
- Less ROI, higher time and effort.

Ideal Test Automation Pyramid

- Focus on defect prevention of issues.
- Less setup time.
- Less test writing, setup, and maintenance cost.
- Higher stability.
- Saves times and resources (higher ROI).

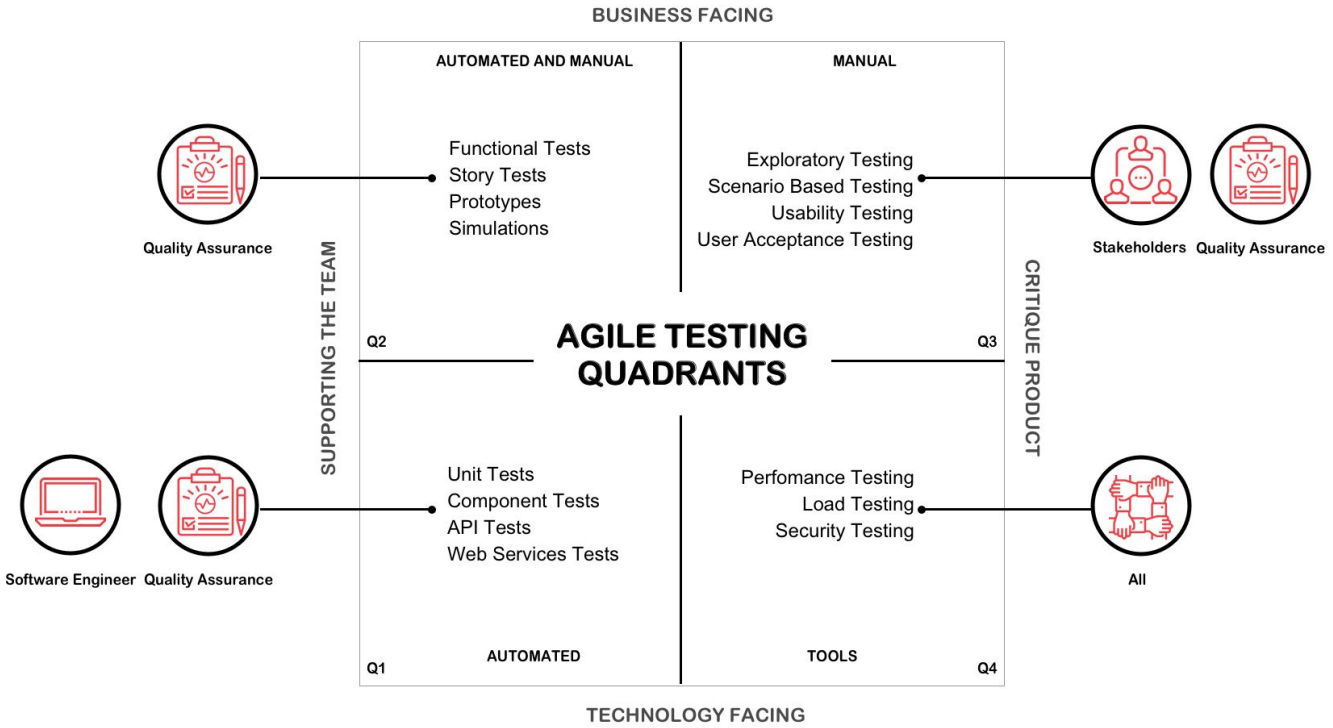
Which test cases to automate?



Automation Process

1. Identify the areas within software that need testing
2. Select the appropriate automation tool
3. Write test scripts
4. Develop test suits
5. Execute test scripts
6. Create a result report
7. Identify any potential bugs or performance issues

Agile Testing Quadrants

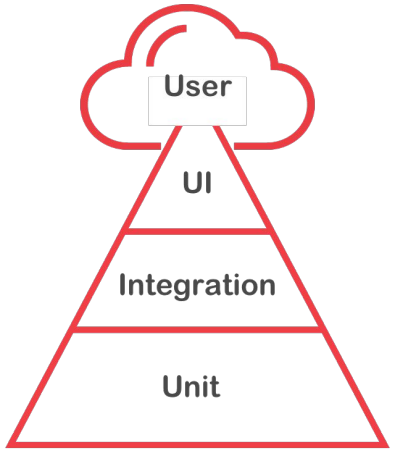


Test Levels

Test levels are groups of test activities that are organized and managed together. Each test level is an instance of the test process, consisting of the activities, performed in relation to software at a given level of development, from individual units or components to complete systems or, where applicable, systems of systems.

The test levels more commonly used are:

- **Component testing (Unit)**
- **Integration testing**
- **System testing (UI)**
- **Acceptance testing (User)**



Component Testing

Objectives	Test Basis	Test Objects	Typical Defects and Failures
• Reducing risk • Verifying whether the functional and non-functional behaviors of the component are as designed and specified • Building confidence in the component's quality • Finding defects in the component • Preventing defects from escaping to higher test levels	• Detailed design • Code • Data model • Component specifications	• Components, units, or modules • Code and data structures • Classes • Database modules	• Incorrect functionality (e.g., not as described in design specifications) • Data flow problems • Incorrect code and logic

Integration Testing

Objectives	Test Basis	Test Objects	Typical Defects and Failures
• Reducing risk • Verifying whether the functional and non-functional behaviors of the interfaces are as designed and specified • Building confidence in the quality of the interfaces • Finding defects (which may be in the interfaces themselves or within the components or systems) • Preventing defects from escaping to higher test levels	• Software and system design • Sequence diagrams Interface and communication protocol specifications • Use cases • Architecture at component or system level • Workflows • External interface definitions	• Subsystems • Databases • Infrastructure • Interfaces • APIs • Microservices	• Inconsistent message structures between systems • Incorrect data, missing data, or incorrect data encoding • Interface mismatch • Failures in communication between systems • Unhandled or improperly handled communication failures between systems • Incorrect assumptions about the meaning, units, or boundaries of the data being passed between systems • Failure to comply with mandatory security regulations

System Testing

Objectives | **Test Basis** | **Test Objects** | **Typical Defects and Failures**

- Reducing risk
- Verifying whether the functional and non-functional behaviors of the system are as designed and specified
- Validating that the system is complete and will work as expected
- Building confidence in the quality of the system as a whole
- Finding defects
- Preventing defects from escaping to higher test levels or production

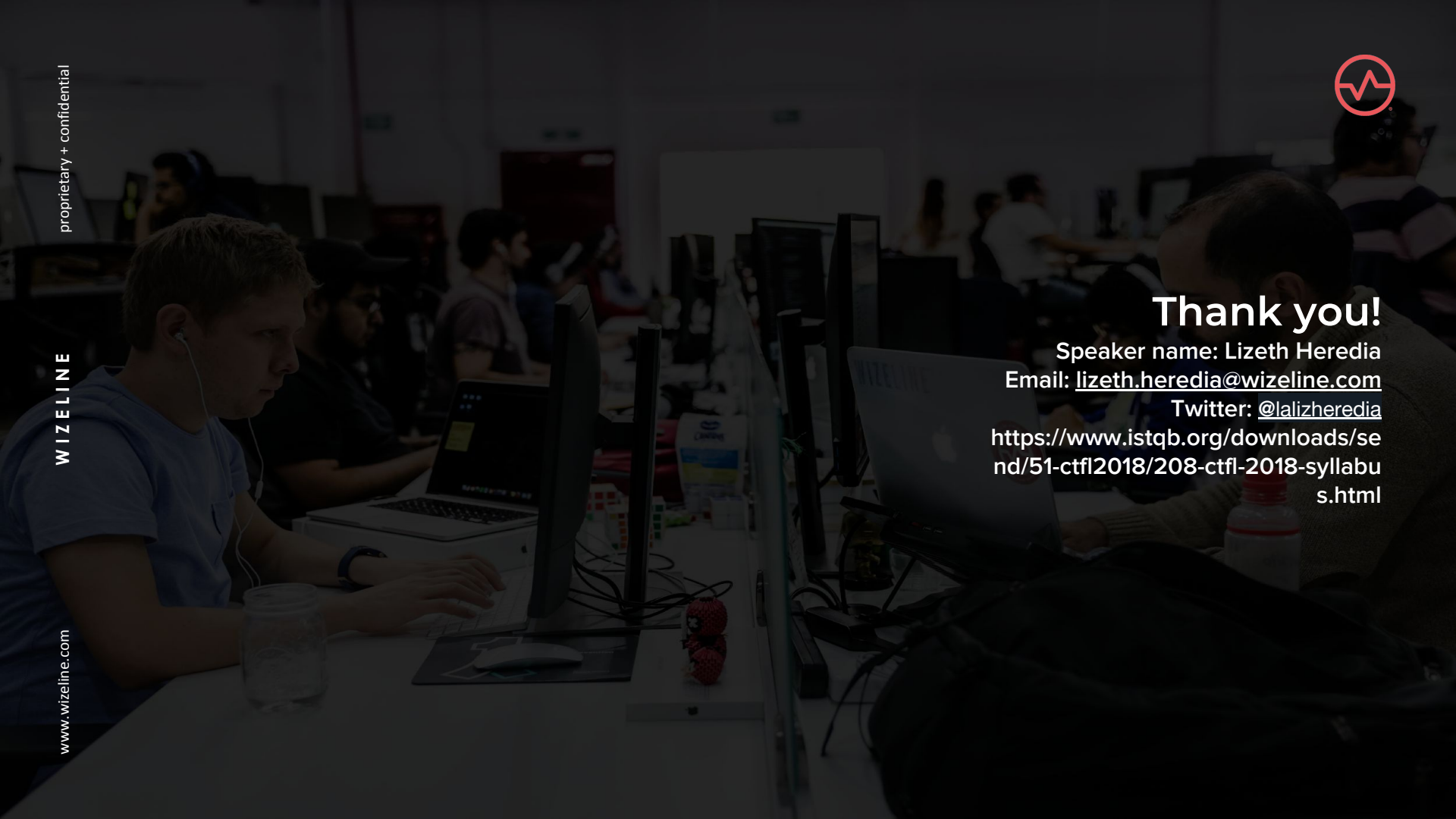
- System and software requirement specifications (functional and non-functional)
- Risk analysis reports
- Use cases
- Epics and user stories
- Models of system behavior
- State diagrams
- System and user manuals

- Applications
- Hardware or software systems
- Operating systems
- System under test (SUT)
- System configuration and configuration data

- Incorrect calculations
- Incorrect or unexpected system functional or non-functional behavior
- Incorrect control and/or data flows within the system
- Failure to properly and completely carry out end-to-end functional tasks
- Failure of the system to work properly in the production environment(s)
- Failure of the system to work as described in system and user manuals

Acceptance Testing

Objectives	Test Basis	Test Objects	Typical Defects and Failures
• Establishing confidence in the quality of the system as a whole • Validating that the system is complete and will work as expected • Verifying that functional and non-functional behaviors of the system are as specified	• Business processes • User or business requirements • Regulations, legal contracts and standards • Use cases • System requirements • System or user documentation • Installation procedures • Risk analysis reports	• System under test • System configuration and configuration data • Business processes for a fully integrated system • Recovery systems and hot sites (for business continuity and disaster recovery testing) • Operational and maintenance processes • Forms • Reports • Existing and converted production data	• System workflows do not meet business or user requirements • Business rules are not implemented correctly • System does not satisfy contractual or regulatory requirements • Non-functional failures such as security vulnerabilities, inadequate performance efficiency under high loads, or improper operation on a supported platform



proprietary + confidential

WIZELINE

www.wizeline.com

Thank you!

Speaker name: Lizeth Heredia
Email: lizeth.heredia@wizeline.com
Twitter: @lalizheredia

https://www.istqb.org/downloads/send/51-ctfl2018/208-ctfl-2018-syllabus.html