QA Introduction

Quality Assurance, Testing and Test Automation

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#fund-common

Table of Contents

Software Quality Assurance

Introduction

Software Quality Assurance (QA)

- What is "**software quality assurance**" (**SQA**)?
  - Software quality assurance aims to **assure** that the **software** is **bug free** (behaves as expected)
  - Defects are reported and tracked through a **bug tracking system**
  - Performed by the Quality Assurance engineers (**QA engineers**)
- Most of the QA work is **software testing**
  - **Manual** testing (click and check the results)
  - **Automated** testing (QA automation)
- Continuous integration and delivery (**CI/CD pipeline**)

The QA Role and Its Responsibilities

Quality Assurance (QA) Engineer's Role

Software University

- **QA engineers** ensure the **software quality**
- Plan and execute **testing activities**
  - **Test** the software, its functionality, UX and usability, etc.
  - Create **test plans**, design **test cases**, **execute tests**
  - Develop and execute **test automation** scripts
- **Report** and **track bugs** and their lifecycle
  - Perform **regression testing** when bugs are resolved
- Track the **development process** and its quality
  - Review the **requirements**, **design** and **code**
  - Build and monitor **CI/CD pipeline**, track QA **metrics**

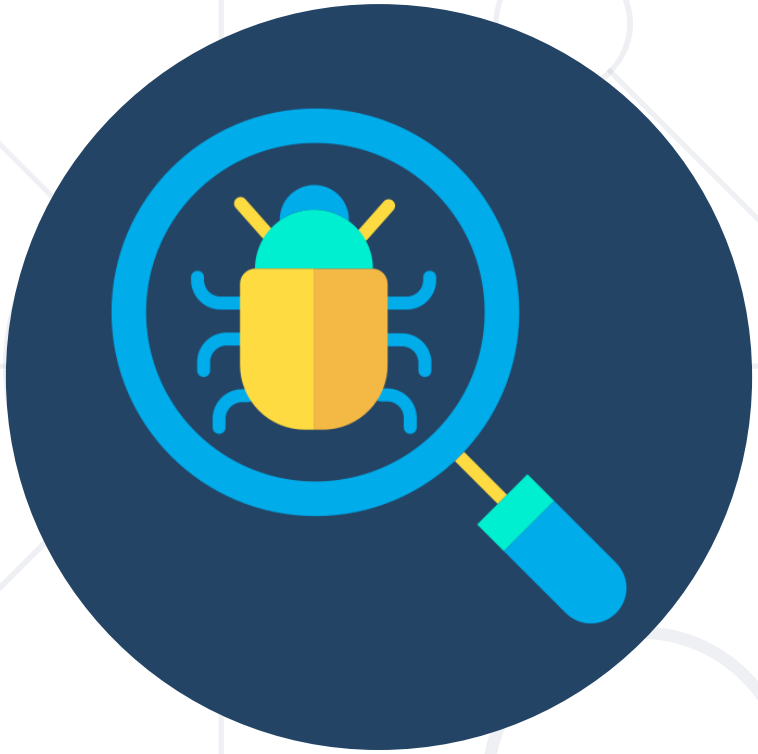
What is a Database?

- A **database** is a collection of data, organized to be easily accessed, managed and updated
- Modern databases are managed by **Database Management Systems** (DBMS)
  - Define database **structure**, e.g. tables, collections, columns, relations, indexes
  - Create / Read / Update / Delete data (CRUD operations)
  - Execute **queries** (filter / search data)

QA Job Ads

Live Demo

https://calendly.com/pages/jobs/details?gh_jid=4698556002
https://www.indeed.com/viewjob?jk=534ebdec45075857
https://www.linkedin.com/jobs/view/1949370301



Defects, Bugs, Issues

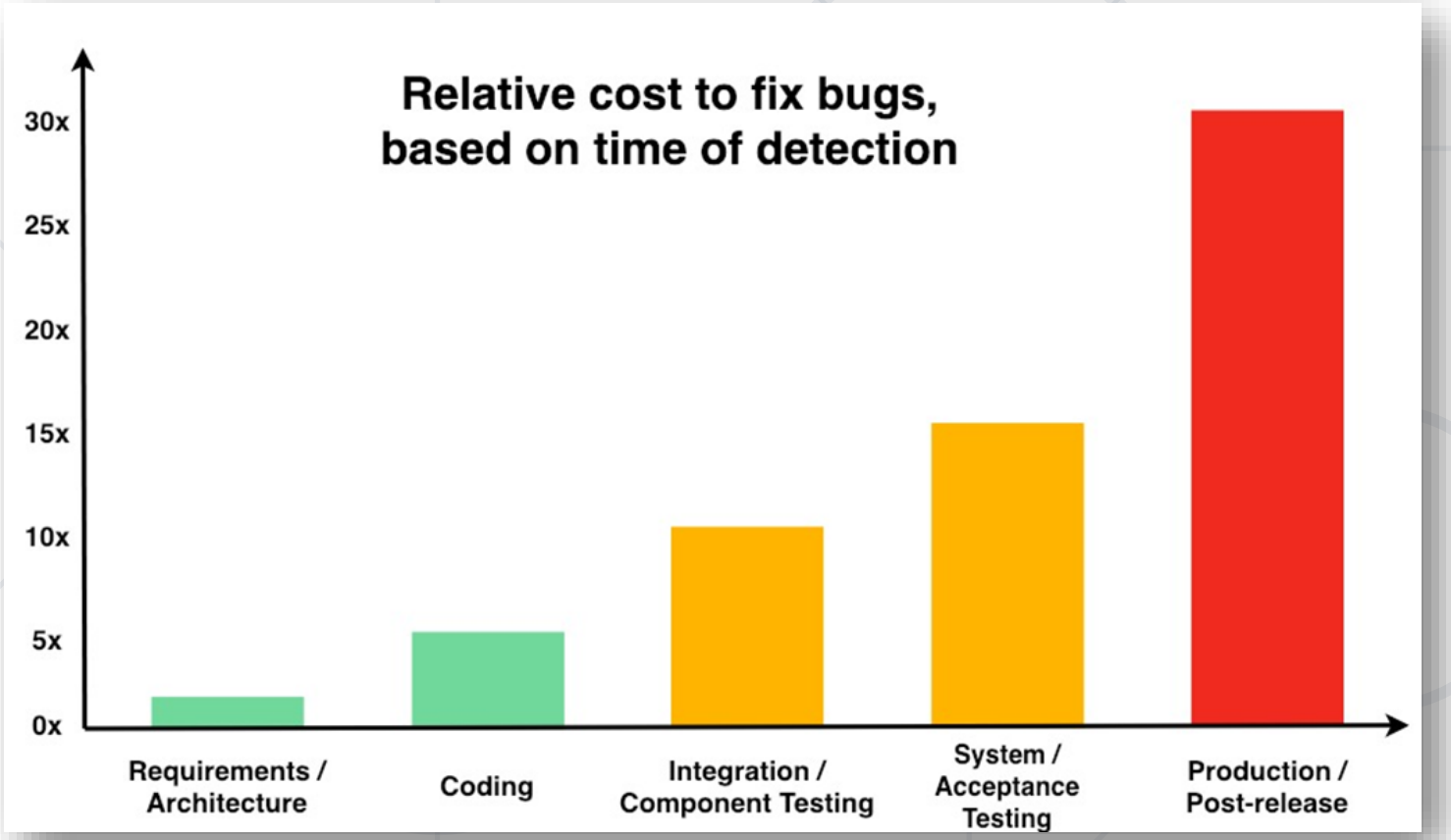
Issue Tracking Systems

Software Defects

- Humans can make **errors** (mistakes)
- Errors produce **defects**
  - **Defects** are **bugs** in the program code, or mistakes in the **requirements** / **design** / other
- If a **defect** is executed, it might cause a **failure**:
  - Fail to do what it should do / do wrong thing
- **QA** / **software testing** aims to find the **defects**
  - **Automated testing** and **CI/CD** reduce the defects

The Cost of Software Defects

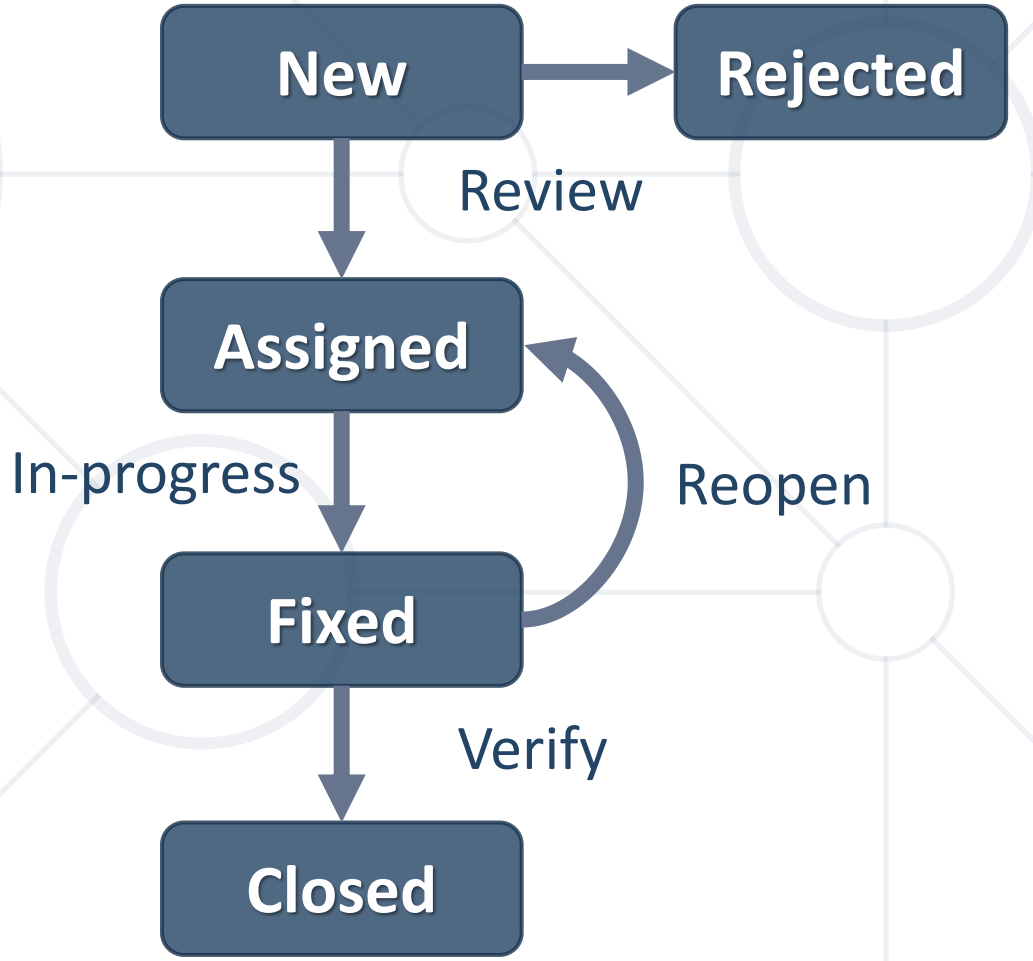
- Defects **cost less** when found **earlier**



- Agile practices (like CI/CD) find defects earlier

Bug Tracking and Issue Lifecycle

- Software defects / bugs / problems / issues
  - Are tracked in **issues trackers** (bug trackers)
- **QA engineers** manage the issue lifecycle
- Issue **lifecycle**
  - New → Assign / Reject → Fix → Verify → Close / Reopen



Issues

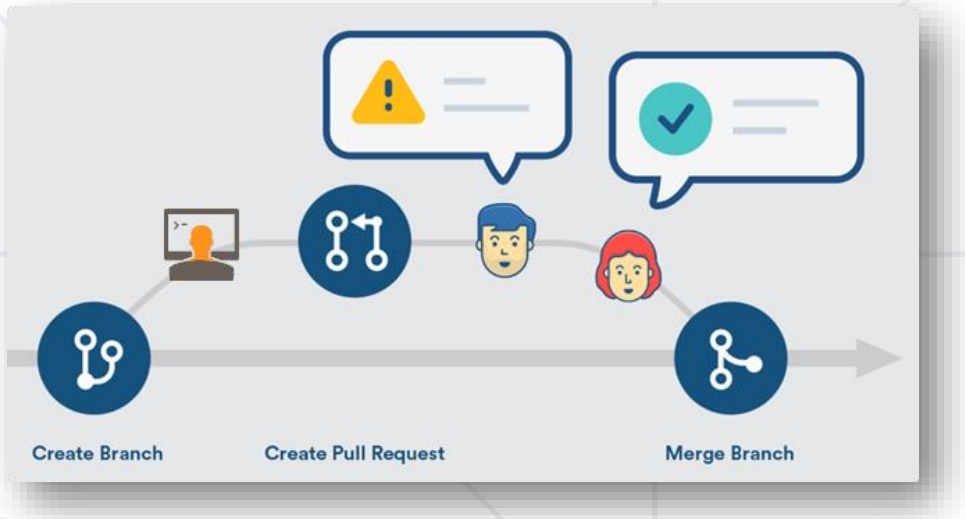
- **QAs** report, describe and **track issues** in an issue tracker
- **Issues** hold the following information
  - Title and description (with steps to reproduce)
  - State: open / closed
  - Status: new / assigned / rejected / fixed / verified
  - Priority: low, medium, high, critical
  - Assigned team members
  - Discussion / comments

Issue Tracker

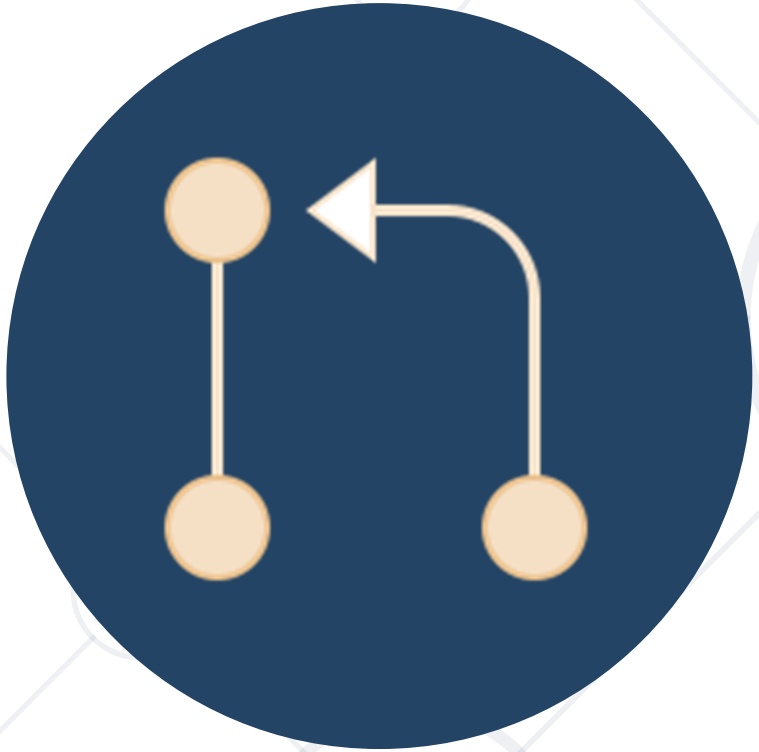
Live Demo

https://github.com/twbs/bootstrap/issues
https://github.com/twbs/bootstrap/issues/31392
https://github.com/twbs/bootstrap/issues/31459

Typical Flow for Handling an Issue



1. An **issue** is logged by someone
2. A developer is **assigned** to fix it
3. A **new branch** is created for the fix
4. The developer makes **changes and fixes** in this branch (writes code, commits changes, pushes the changes)
5. When ready, the developer sends a **pull request**
6. Other developers **review** / **comment** / **approve**
7. The changes are **merged** in the upstream branch



Pull Request Merge

Live Demo

https://github.com/twbs/bootstrap/pull/31396

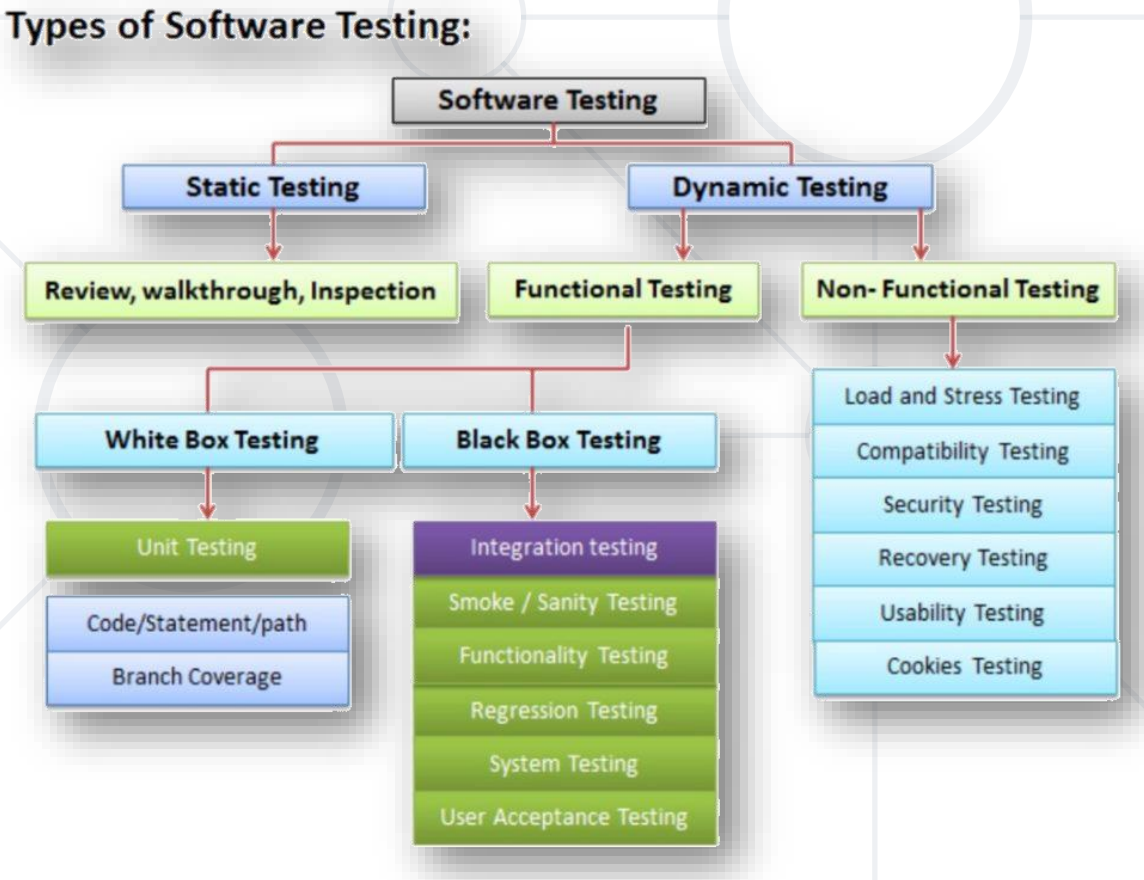
Software Testing

Test Types and Test Levels

Software Testing and Test Types

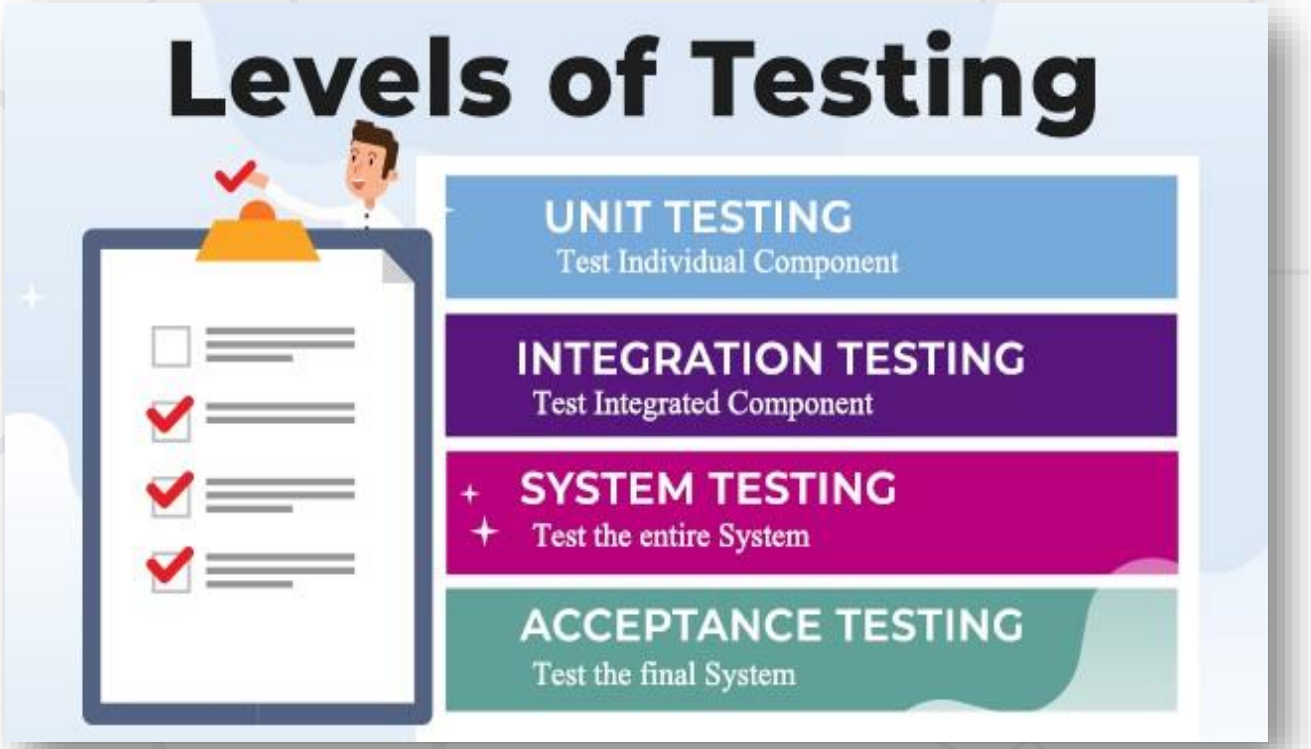
Software University

- Testing checks whether software **conforms to the requirements**, aims to **find defects**
- Types of software tests
  - Functional and non-functional
  - Black-box and white-box tests, regression tests
  - Stress tests, load tests, UX and usability tests, security tests
  - Manual vs. automated tests

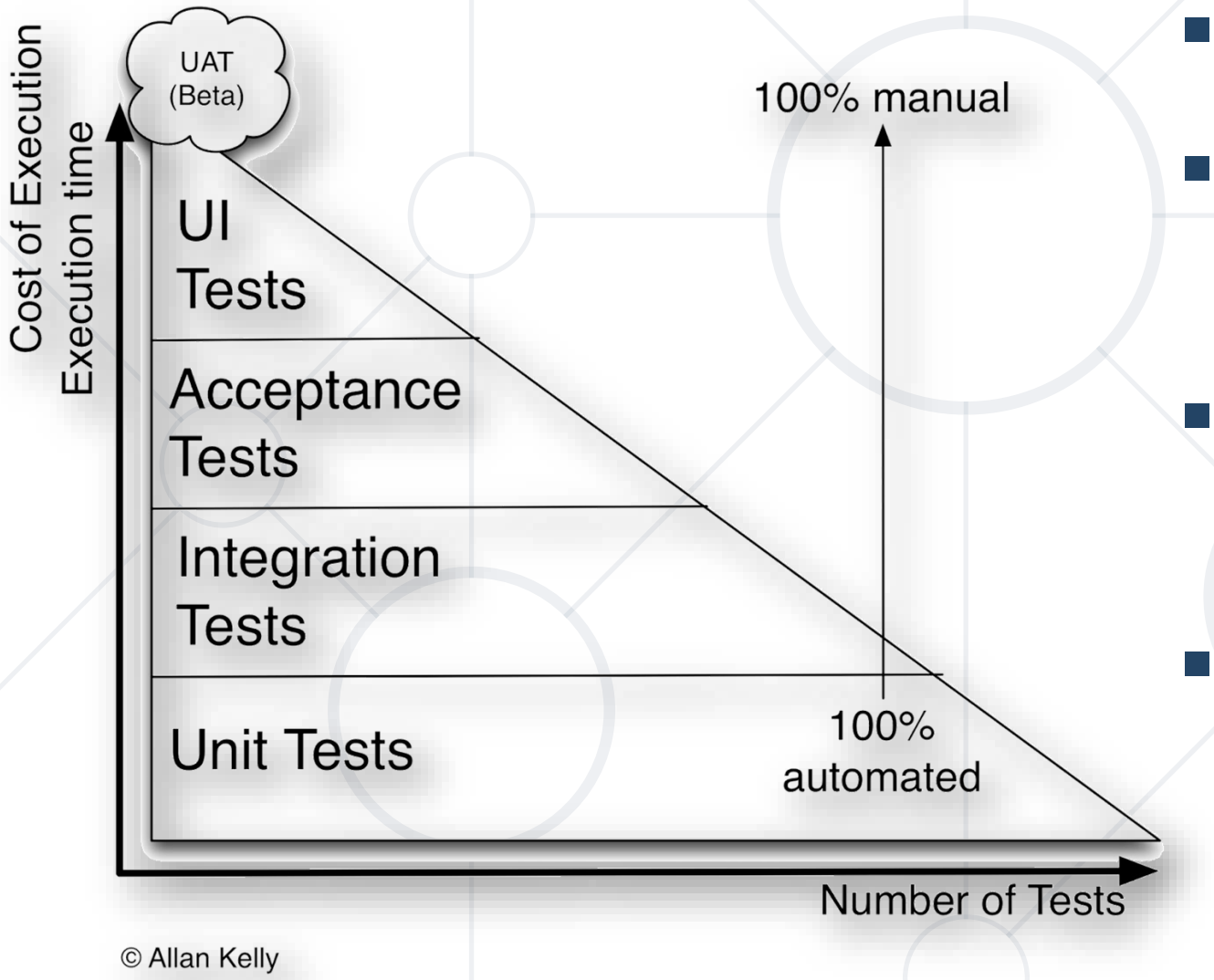


Test Levels

- **Unit tests**
  - Test single component
  - Automated by developers
- **Integration tests**
  - Test interaction between components
- **System tests / acceptance tests**
  - Test the entire system

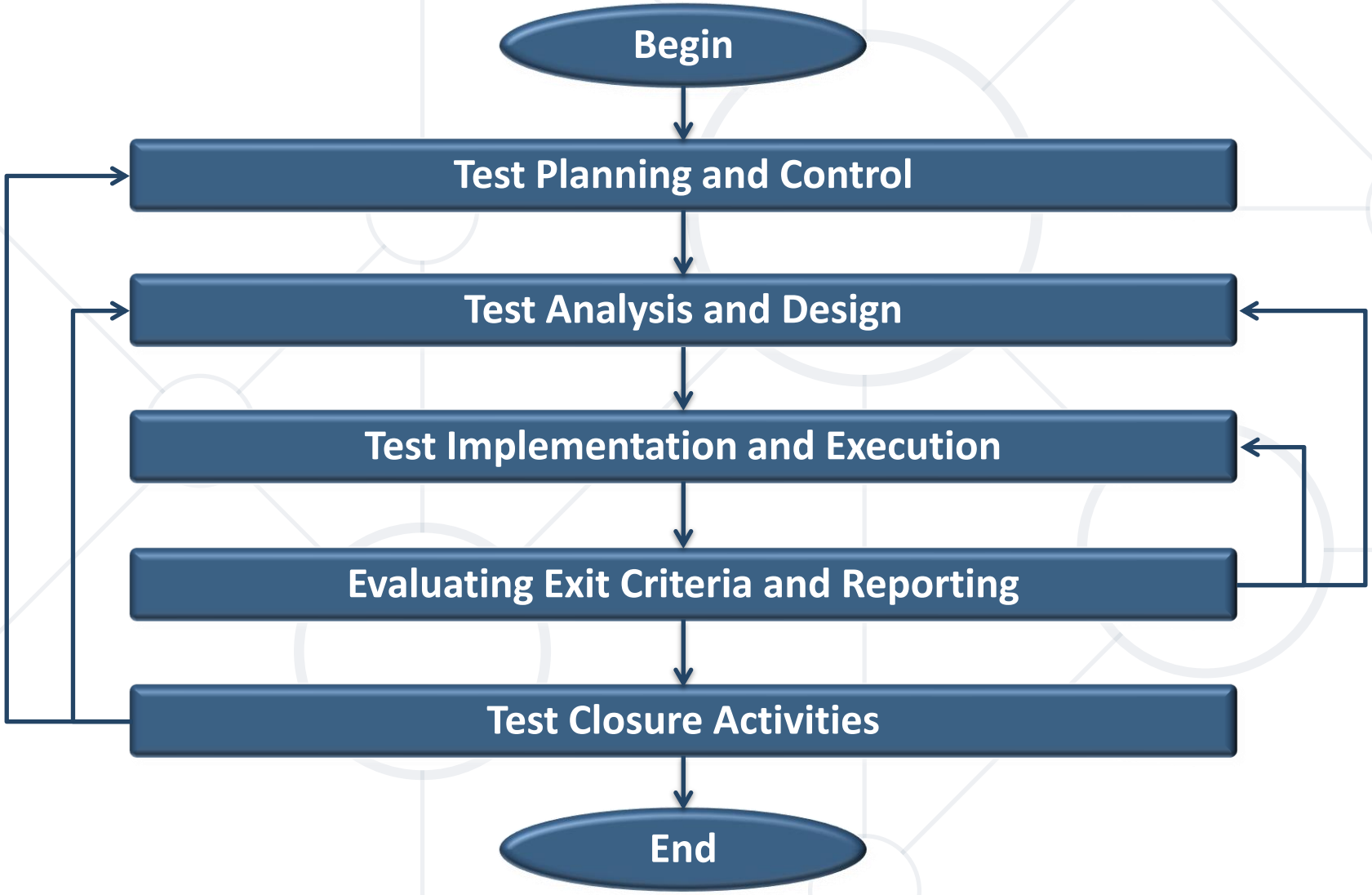


The Testing Triangle



- **Unit tests**: fully automated
- **Integration tests**: fully automated
- **System tests** / **acceptance tests**: partially automated
- **UI / UX tests**: mostly manual

Test Process and Test Activities



The Software Testing Process

- Test **planning**
  - Establish **test strategy** and **test plan**
  - What to test, how to test, when, test scenarios
- Test **development**
  - Test procedures, test scenarios,
    test cases, test scripts, test automation
- Test **execution** and reporting
- Defect tracking / **issue tracking**

Test Plan, Test Scenarios and Test Cases

Software University

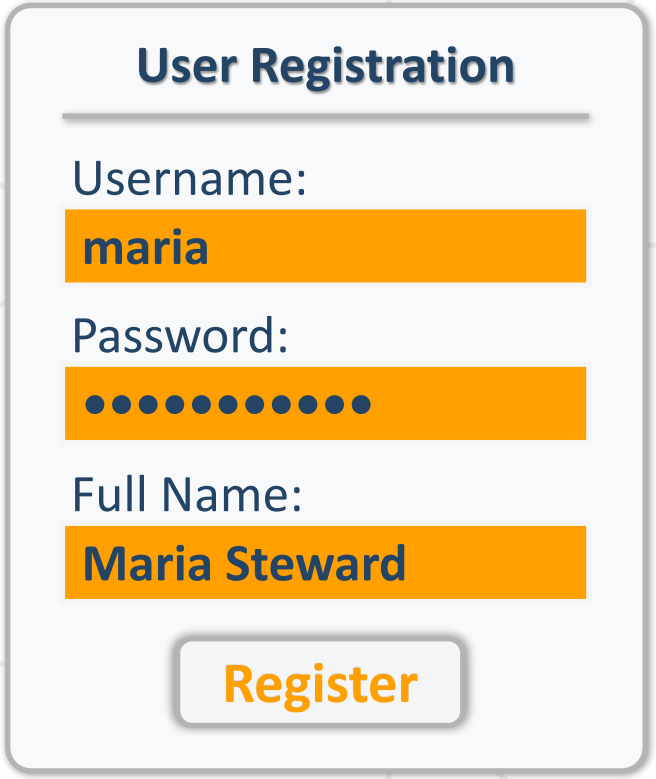
- The test plan describes how tests will be performed
  - List of QA and test activities to be performed to ensure **meeting the quality requirements** (more or less formal)
  - Features to be tested (scenarios), test cases, testing approach, test schedule, acceptance criteria
- Test scenarios and test cases
  - Test scenarios – stories to be tested
  - Test cases – tests of certain function
  - Each test scenario is covered by several test cases

Test Case

- Sequence of **steps** to check the **correct** behavior
- At **least two cases** to fully test certain scenario
  - A positive test
  - A negative test
- Test cases consist of
  - Title
  - Steps to follow
  - Expected result

Test Scenarios and Test Cases – Example

- Sample **test scenario**
  - User registration
- **Test cases** for this scenario
  - Non-existing username → success
  - Duplicated username → error
  - Empty username or password → error
  - Too long username → error
  - Invalid characters in username / password → error



Test Case – Formal Example

	A	B	C	D	E	F	G	H
1	**ID**	TC00051					**Cycle**	Major
2	**Name**	Test Login					**Category**	Regression Tests
3	**Revision**	1.0						
4								
5	**Description**	Check the basic login functionality						
6	**Precondition**	Server installed						
7	**Postcondition**	User is logged in						
8	**Expected Result**							
9								
10	**Note**	Do not skip this!						
11	**Area**	REGRESSION						
12	**Design Method**	BLACK_BOX						
13	**Variety**	NEGATIVE						
14	**Execution**	MANUAL						
15	**Priority**	MEDIUM						
16	**State**							
17	**Team**	QA						
18	**Level**	COMPONENT						
19	**Document Base**	Requirements Document 1.5 (12.7.2011)						
20	**Dependency**	-						
21	**Evaluation**	MANUAL						
22	**Traceability**	UC-112						
23								
24								
25	**Step**	**Action**	**Precondition**	**Postcondition**	**Expected Result**			
26	1	Open login page			Login page displayed			
27	2	Enter username						
28	3	Enter password			Password should not be visible			
29	4	Press ok			User is logged in			
30								

General Properties

Custom Properties

Test Steps

Test Plan

Live Demo

https://melodic.cloud/wp-content/uploads/2019/01/D5.06-Test-Strategy-and-Environment.pdf
https://www.smartdcc.co.uk/media/3609/testing-approach-document-for-june-2020-release_v03-clean.pdf

Test Automation

Unit Testing, Integration Testing, Mocha, Selenium

Test Automation

- **Test automation** is important part of software development
- Test automation is done at many levels
  - **Unit tests**: written by developers
  - **Integration tests**: written by devs and QAs
  - **UI tests**: written by QAs
- **Test automation tools** record and execute recorded tests
  - Testing **frameworks** (JUnit, NUnit, Mocha, …)
  - Automated testing **tools** (Selenium, Appium, Sikuli)
  - **Web** testing, **API** testing, **mobile** testing

Test Automation Engineers

- **Test automation engineers** (software developers in test)
  - **Developers** with **QA** automation specialization
  - **Technical** skills: coding, OOP, Web technologies, front-end, back-end, databases, services and APIs, software engineering, etc.
  - **QA** skills: testing frameworks and test automation tools
  - **DevOps** skills: containers, cloud, CI/CD pipeline
  - Logical thinking and problem-solving skills
  - Planning and organizational skills
  - Attention to details

Unit Testing

- **Unit test** == a piece of code that tests specific functionality in certain software component (unit)


```
function testSum() {
  if (sum([1, 2]) != 3)
    throw "1+2 != 3";
  if (sum([-2]) != -2)
    throw "-2 != -2";
  if (sum([]) != 0)
    throw "empty sum != 0";
}
```

```
function sum(arr) {
  let sum = 0;
  for (let item of arr)
    sum += item;
  return sum;
}
```

Unit Testing Framework

- **Unit testing frameworks** simplify unit testing and reporting
  - Example: **Mocha** JS testing framework

```
const assert = require('assert');

suite('sum(arr)', function() {
  test('sum([1+2]) == 3', function() {
    assert.equal(sum([1, 2]), 3); });
  test('sum([-2]) == -2', function() {
    assert.equal(sum([-2]), -2); });
  test('sum([]) == 0', function() {
    assert.equal(sum([]), 0); });
});
```

```
> mocha --ui tdd index.test.js

  sum(arr)
    ✓ sum([1+2]) == 3
    ✓ sum([-2]) == -2
    1) sum([]) == 0

  2 passing (10ms)
  1 failing
```

Unit Testing with Mocha

Live Demo

https://repl.it/@nakov/mocha-unit-test-example-js

Integration Testing

- **Integration testing** test several units (components) together
  - Aims to expose faults in the **interaction between integrated units**
  - Example: test user registration + data access services + database storage (check whether the new user is stored in the DB)
- **Unit testing** vs. **integration testing**
  - Integration testing tests the interaction between several units
  - Unit testing tests a single unit (component)
- Integration testing is implemented by:
  - Testing framework + test stubs / mocks

Integration Testing with Mocha

Live Demo

https://repl.it/@nakov/MVC-app-integration-tests-example-mocha
https://github.com/nakov/MVC-app-integration-tests-example-mocha/actions

Web Testing Automation and Selenium

Software University

- **System testing** tests the entire system:
  - e.g., front-end (UI logic) + back-end (business logic) + database
- Example: automated system testing for Web apps
  - Auto deploy the Web app in a **testing environment** (e.g. Docker)
  - Execute **UI test scenarios** (e.g. fill and submit forms, then check for the inserted / modified data)
- **Selenium** automates testing of Web apps
  - Automates the Web browser: test recording + asserts + execution

Web Testing with Selenium

Live Demo

https://repl.it/@nakov/selenium-webdriver-example

The CI/CD Pipeline

Continuous Integration and Continuous Delivery

Software Development Lifecycle (SLDC)

Software University

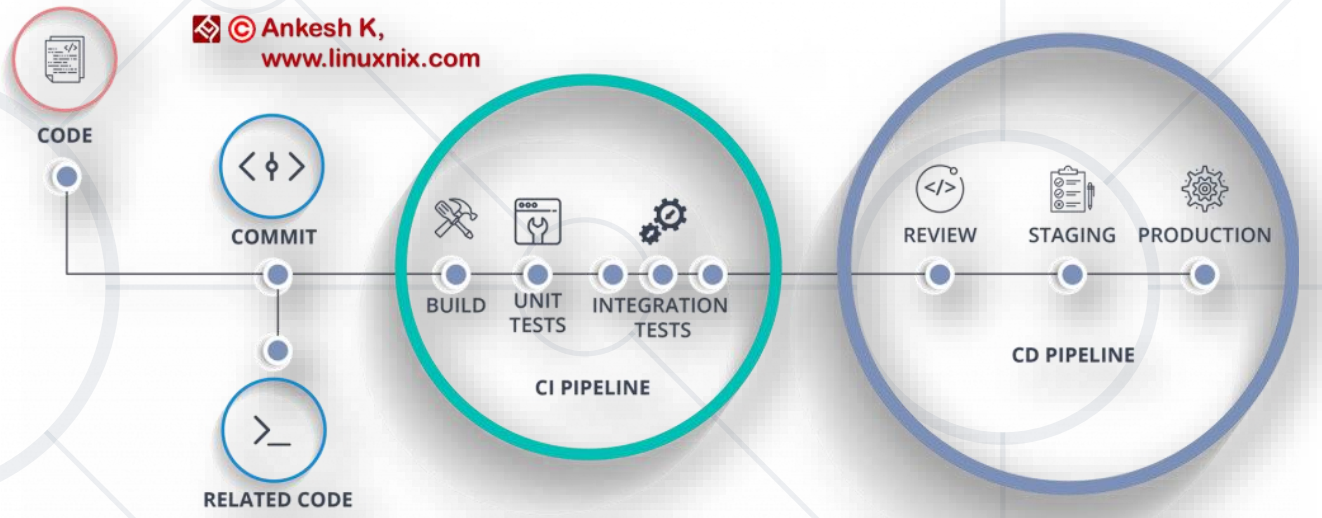
- **Software engineering** is not just coding!
- The **SDLC** includes the following activities:
  - **Requirements** analysis
  - Software **design**
  - **Construction**
  - **Release**
  - **Testing**
  - **Maintenance**

Software project **management**

- **Development processes** (Waterfall / Scrum /Kanban) define workflow and key practices

CI/CD Pipeline

- **CI/CD pipeline**
  - Continuously integrate and release new features
- **Continuous integration (CI)**
  - Write code, test and integrate it in the product
- **Continuous delivery (CD)**
  - Continuously release new features
- **QAs** maintain and monitor the CI/CD pipeline



CI/CD Pipeline with GitHub Actions

Live Demo

https://github.com/fireship-io/fireship.io/runs/924075545

https://github.com/dotnet-architecture/eShopOnWeb/runs/930547025

https://github.com/github/covid19-dashboard/runs/923863536

https://github.com/nakov/MVC-app-integration-tests-example-mocha/actions

Summary

- QA engineers ensure the **software quality**: testing, reporting and process
- Plan and execute **testing activities**
- Design **test cases** and execute **tests**
- Write **test automation** scripts
- Report **bugs** and track their lifecycle
- Build and monitor **CI/CD** pipeline

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

Flutter™ International

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

BOSCH

Postbank
Решения за твоето утре

PHAR VISION

SmartIT

DXC TECHNOLOGY

createX

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://softuni.org
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg