Testing Techniques

Exploring Different Approaches to Testing

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

http://softuni.bg

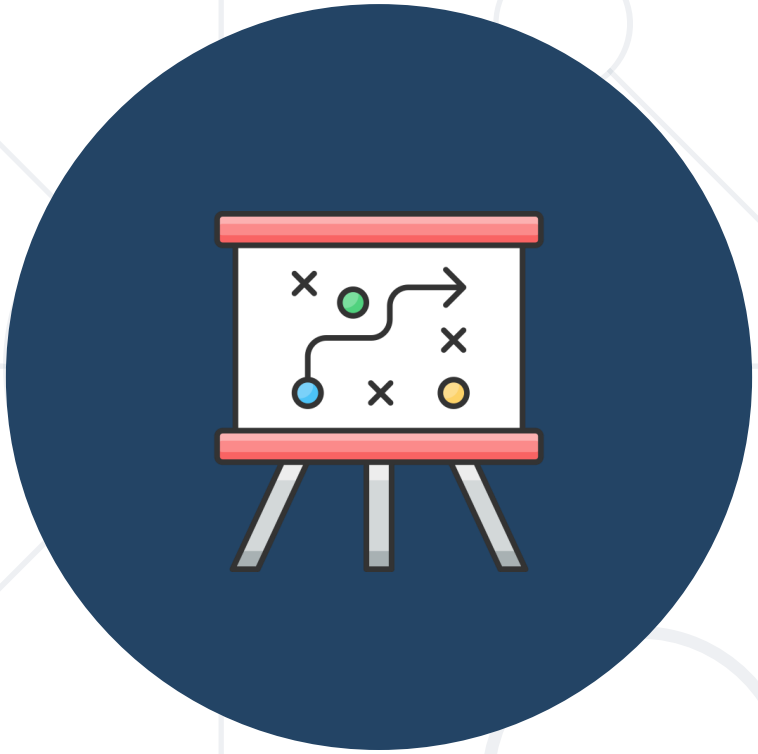
Have a Question?

Software University

sli.do

#QA-Fund

Table of Contents



Testing Techniques

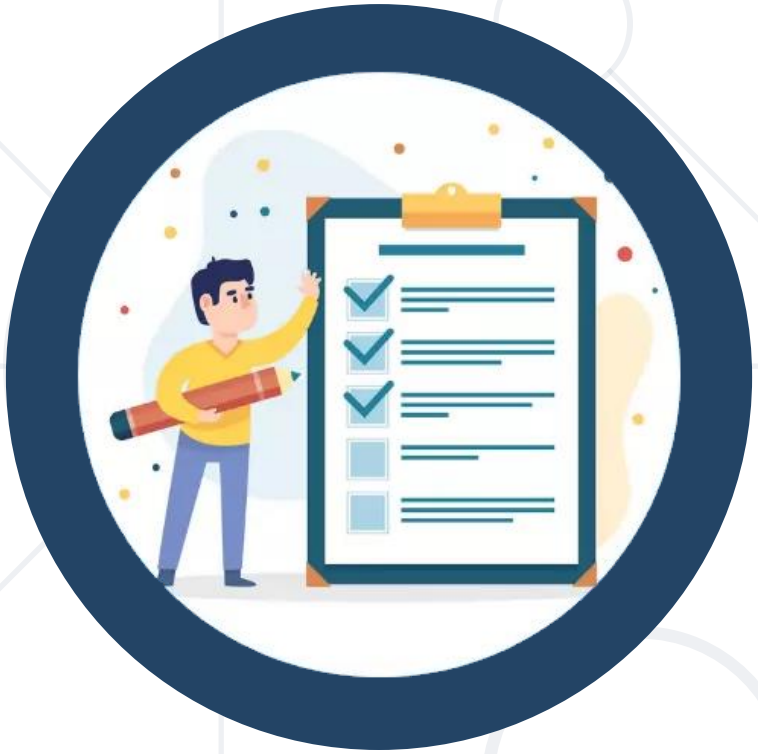
Definition, Purpose, Categories

Testing Techniques Overview

- **Systematic approaches** for software testing
- **Why We Need Different Test Techniques:**
  - Address **diverse** and **complex** software **systems**
  - Detect **varying types** of defects
  - **Optimize resources** and testing efforts
- **Categories:**
  - **Static** - Analyzing code, requirements, and design without execution
  - **Dynamic** - Executing software and observing outcomes

Static vs. Dynamic Techniques

- **Static**
  - Emphasizes **early defect detection** and prevention
  - **Analyzing software** (code, design documents, requirements) **without execution**
  - Identifies **syntax** errors, **logical** errors, coding **standards violations**, and **document errors**
- **Dynamic**
  - Validates software **functionality / performance**
  - Testing the software by **executing** it
  - Identifies **functional errors**, **performance issues**, **runtime errors**, and **missing functionality**

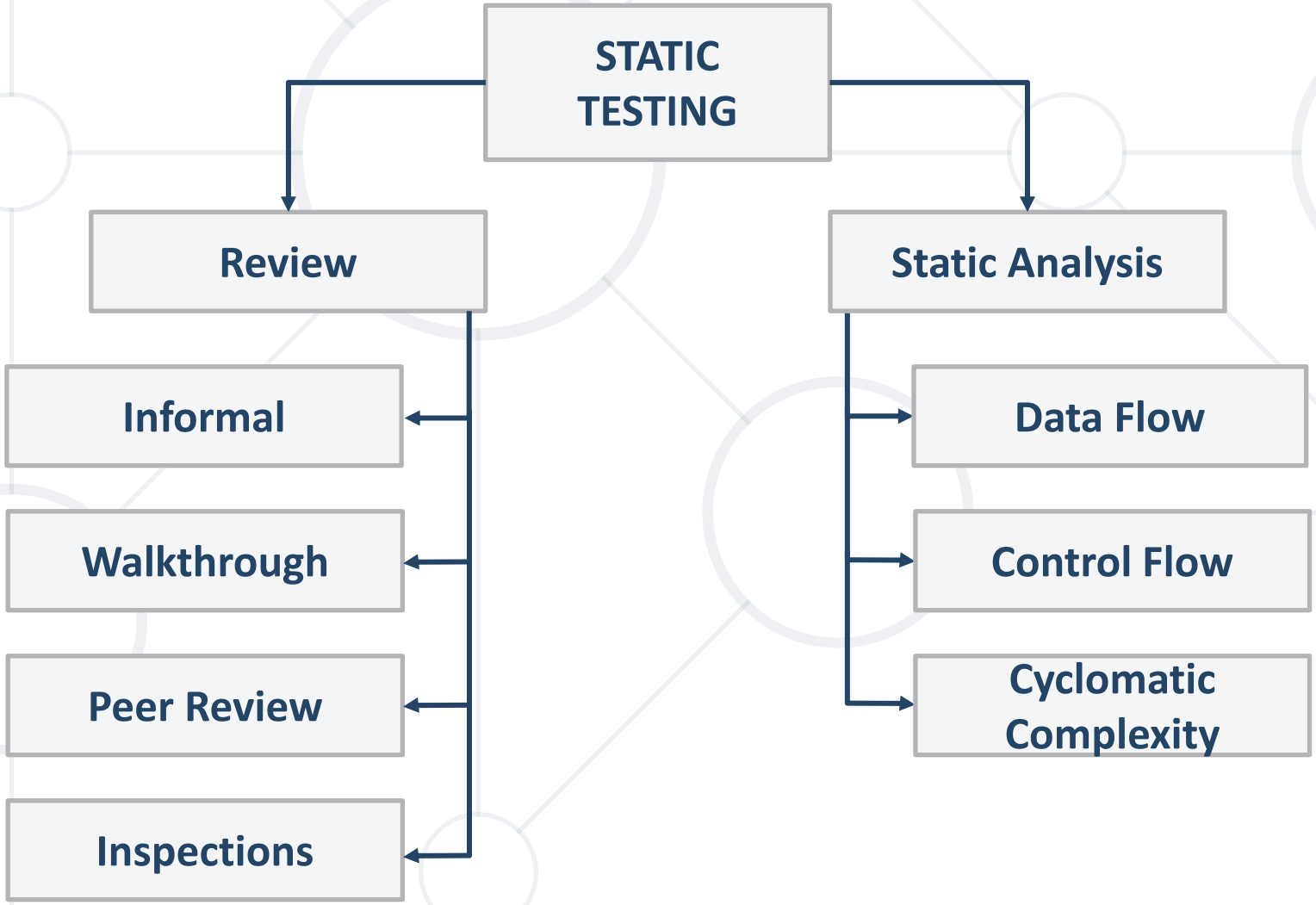


Static Testing Techniques

Analyzing Code Without Execution

Static Testing Techniques

- **Static techniques** improve quality and productivity



Reviews

- **Systematic examination** of the software or document to identify defects and improve quality
- Typically follow a **well-defined process**, with specific roles and responsibilities for participants
- Catch defects and issues in the **early stages** of development
- Promote **knowledge sharing** among team members
- Help in **reducing rework** and late-stage bug fixes
- **Save time** and also **minimize project delays**

Types of Reviews

- **Informal Reviews**
  - **Flexible** and **lightweight** approach to reviewing documents, code, or other artifacts
  - **Casual** and often unstructured
  - Typically driven by a sense of **collaboration** and **knowledge sharing** among team members
- **Example:**
  - A QA engineer testing a mobile app notices a button with a different label compared to similar buttons. They mention it during a team discussion, addressing a UI inconsistency early on

Types of Reviews

- **Walkthroughs**
  - **Systematic examination** of the product's logic, structure, and functionality
  - **Step-by-step exploration** of the artifact being reviewed
  - **Detailed evaluations** of test plans, test cases, or process documentation
- **Example:**
  - A QA lead conducts a walkthrough of a test plan, reviewing test strategy, objectives, and test case selection with the testing team to ensure project alignment

Types of Reviews

- **Peer Reviews**
  - Colleagues working at **similar level** providing focused feedback and improvements
  - QA team members **review each other's** test cases, scripts, or results, share best practices, and ensure testing consistency
- **Example:**
  - Two QA engineers, responsible for different application modules conduct peer reviews of test cases, verifying test coverage and requirement compliance

Types of Reviews

- **Inspections**
  - Most **formal** and **structured** quality assessments
  - Follow **predefined** checklists and criteria
  - Ensure **compliance** with QA **standards**, industry **regulations**, and project-specific **requirements**
- **Example:**
  - In a healthcare software project, a QA inspection team conducts a formal review of validation documentation to ensure compliance with industry regulations, including traceability and completeness

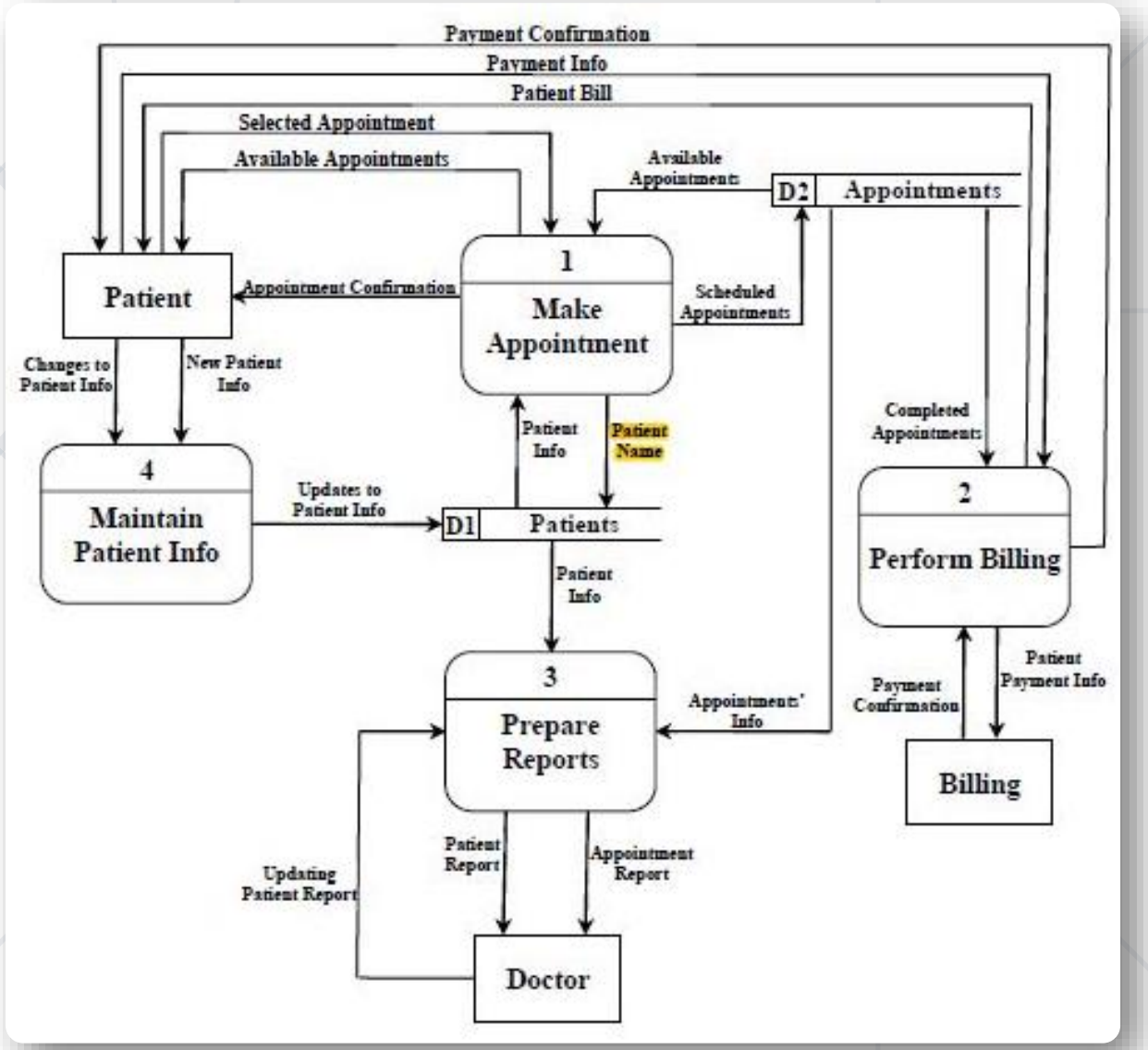
Static Testing Techniques

Software University

- **Static Analysis**
  - **Looking** at software code, design, or documents **without running the program**
  - The main aim is to **find** problems like **coding mistakes**, **security risks**, or **design issues**
  - It can involve checking the **code** for **rule violations** and ensuring the **design matches** the project's **requirements**
  - Also includes **confirming** that the project **documents** are clear and match what the project is supposed to do

Types of Static Analysis

- **Data Flow**
  - Focuses on the paths and states data can take through a program, aiming to identify potential data-related errors
    - Uninitialized variables
    - Data leaks
    - Incorrect variable usage
    - Helps ensure data integrity

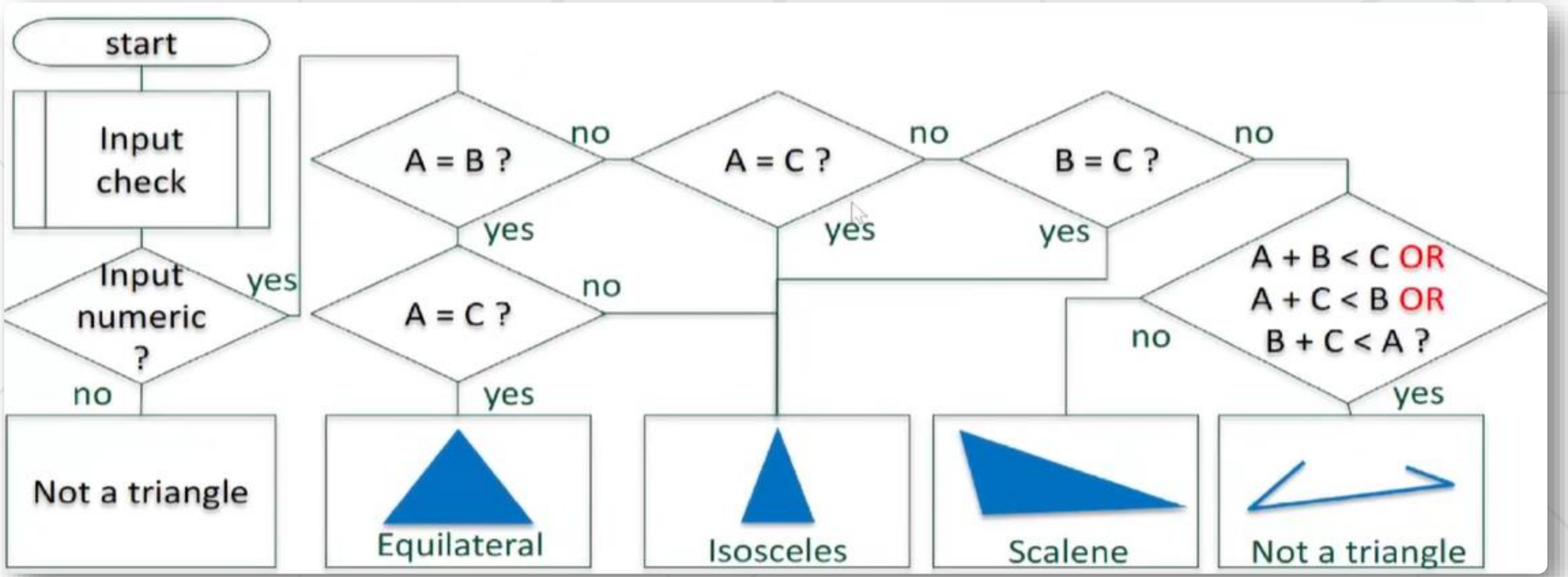


Types of Static Analysis

- **Control Flow**
  - Examines the order in which individual statements, instructions, or function calls within a program are executed or evaluated
  - Helps in identifying various execution paths through the code, including branches and loops
  - Valuable for detecting logic errors, unreachable code, or potential infinite loops

Control Flow

- Process for Triangle Application



Types of Static Analysis

- **Cyclomatic Complexity**
  - Measures program's complexity, counting its execution paths
  - Higher values mean more complexity, leading to increased testing and potential defects
  - It helps determine the minimum test cases needed for good code coverage
  - Simplifying code to reduce complexity enhances maintainability and reduces defects

Static Testing Tools

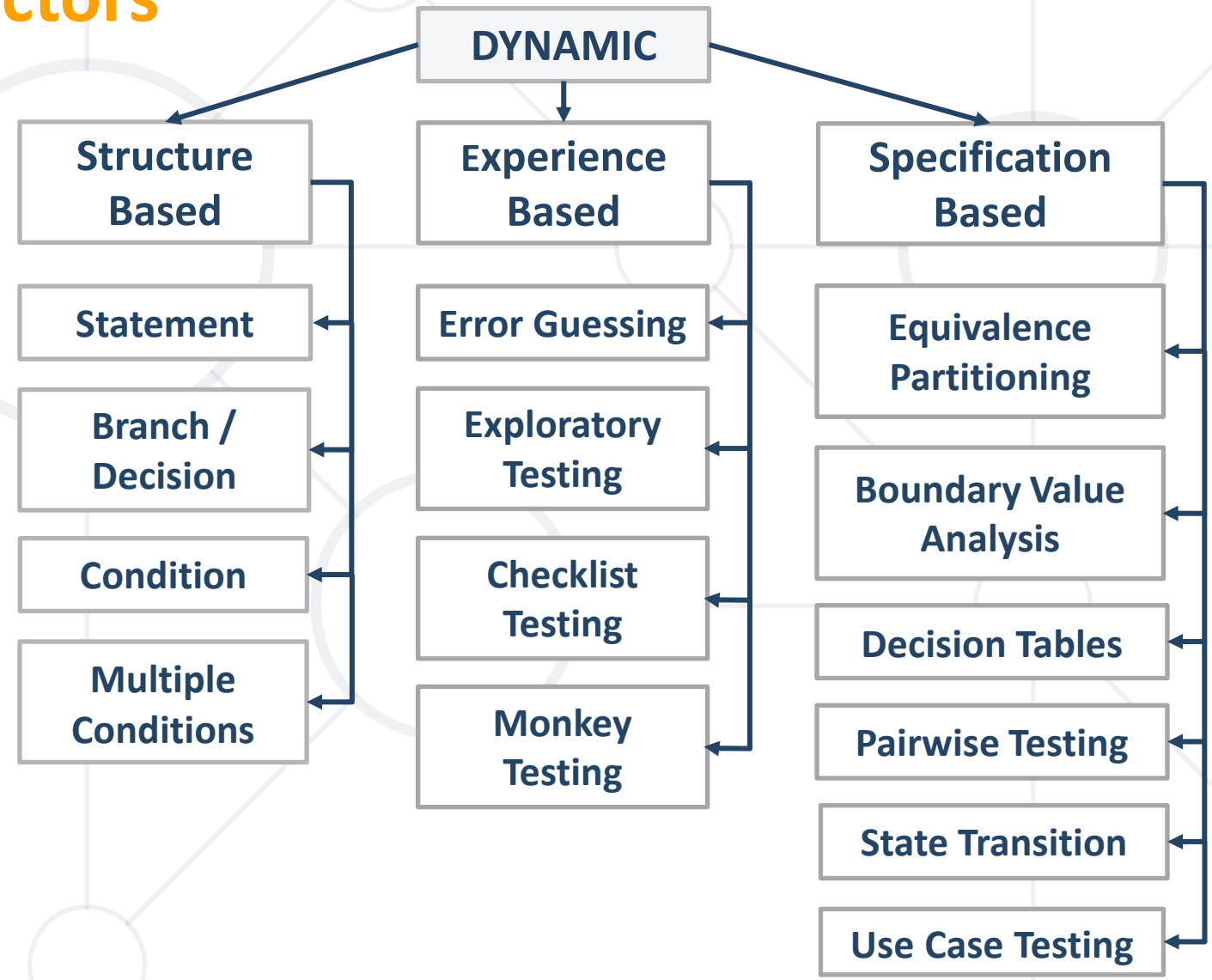
- SonarQube: comprehensive code quality and static code analysis tool
  - Static Code Analysis
  - Code Quality Metrics
  - Security Scanning
  - Customizable Rules
  - Historical Data
  - IDE Integrations
  - Multiple Language Support

Dynamic Testing Techniques

Testing in Action: Testing Through Execution

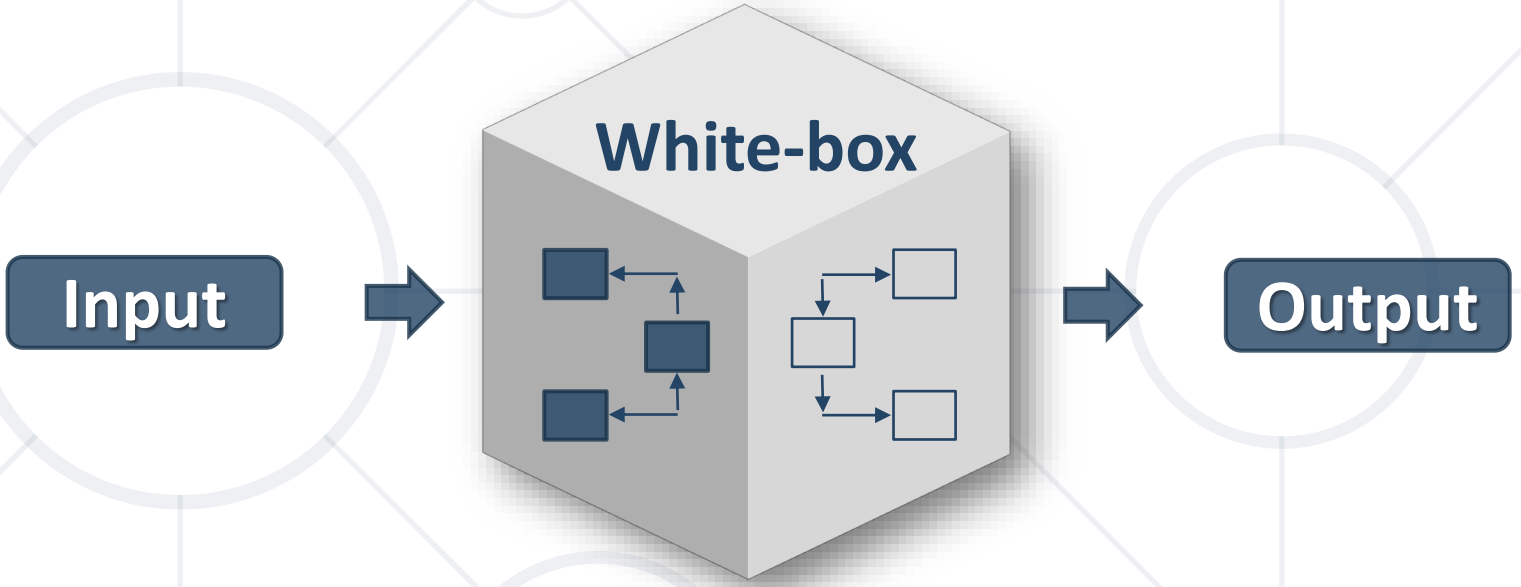
Dynamic Testing Techniques

- Based on **three factors**
  - Structure
  - Experience
  - Specification

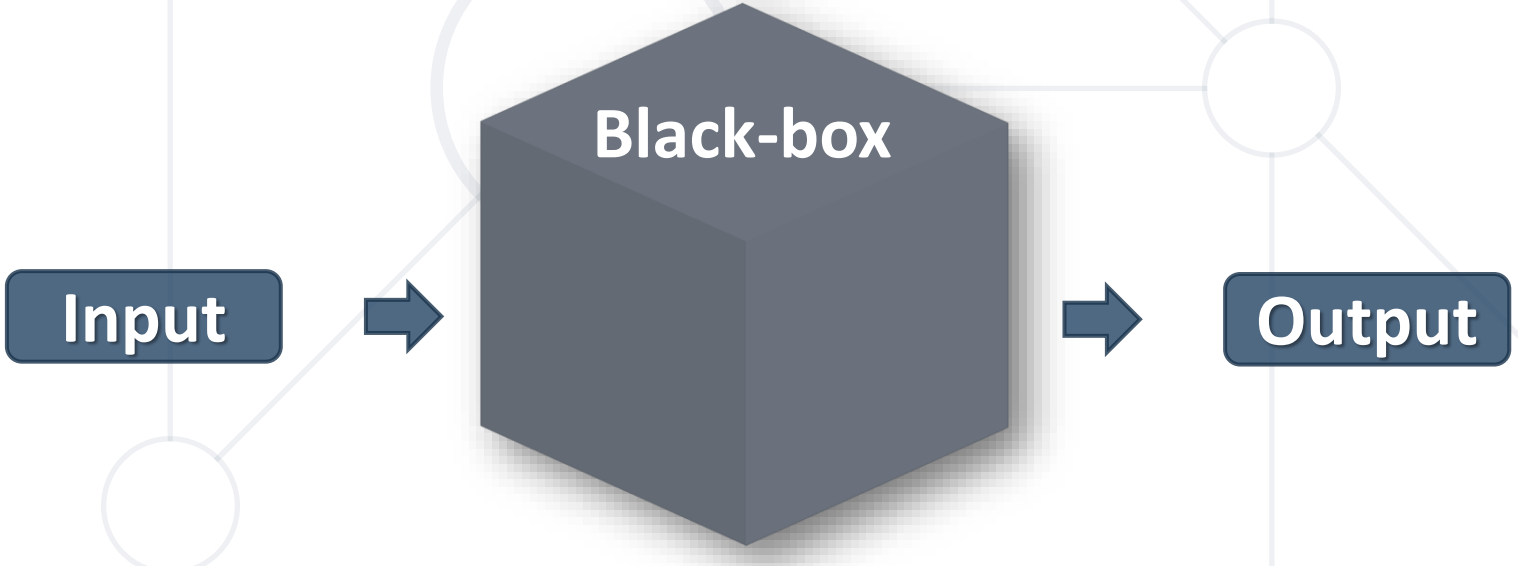


Structure-Based vs Specification-Based

- Structure-Based techniques are **white-box**



- Specification-Based techniques are **black-box**



Black-Box, Grey-Box, White-Box

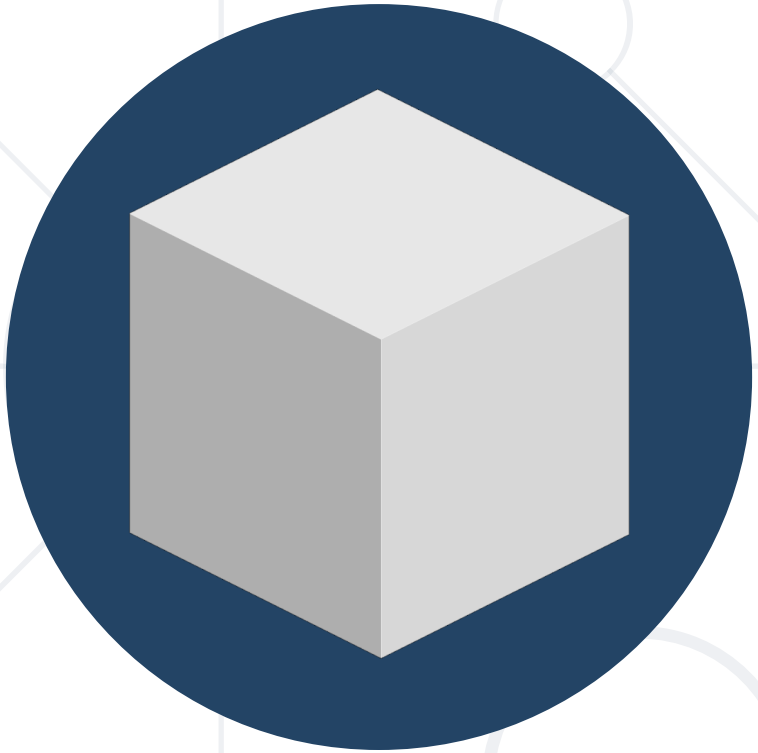
Software University

- Structure Based
- Statement
- Branch / Decision
- Condition
- Multiple Conditions

- Experience Based
- Error Guessing
- Exploratory Testing
- Monkey Testing

* Experience-based techniques can be considered as grey box testing techniques, but not all grey box testing techniques are experience-based

- Specification Based
- Equivalence Partitioning
- Boundary Value Analysis
- Decision Tables
- Pairwise Testing
- State Transition
- Use Case Testing



White-Box Testing Techniques

Structure-Based Techniques

Structure-Based Techniques

Software University

- **White-box techniques**
- Test cases are chosen based on an analysis of the internal structure of a component or a system
- Aims to assess the amount of testing performed by specific tests, often in terms of code coverage
- After the initial set of tests are run and their coverage is analyzed, additional tests are designed to cover parts of the code that have not been tested yet
- The aim is to **increase** the test **coverage**

Coverage

Software University

- Code coverage is defined by the number of **items covered** in testing divided by the **total** number of **items**

$$\text{Coverage} = \frac{\text{Number of coverage items exercised}}{\text{Total number of coverage items}} \times 100\%$$

- The objective of testing is to achieve **maximum** code coverage
- **100% coverage** does **not** mean **100% tested**
- Measuring coverage requires tool support

Coverage Types

- Statement Coverage

$$\text{Statement Coverage} = \frac{\text{Number of statements exercised}}{\text{Total number of statements}} \times 100\%$$

- Branch/Decision Coverage

$$\text{Desicion Coverage} = \frac{\text{Number of desicion outcomes exercised}}{\text{Total number of desicion outcomes}} \times 100\%$$

Example: Statement Coverage

Software University

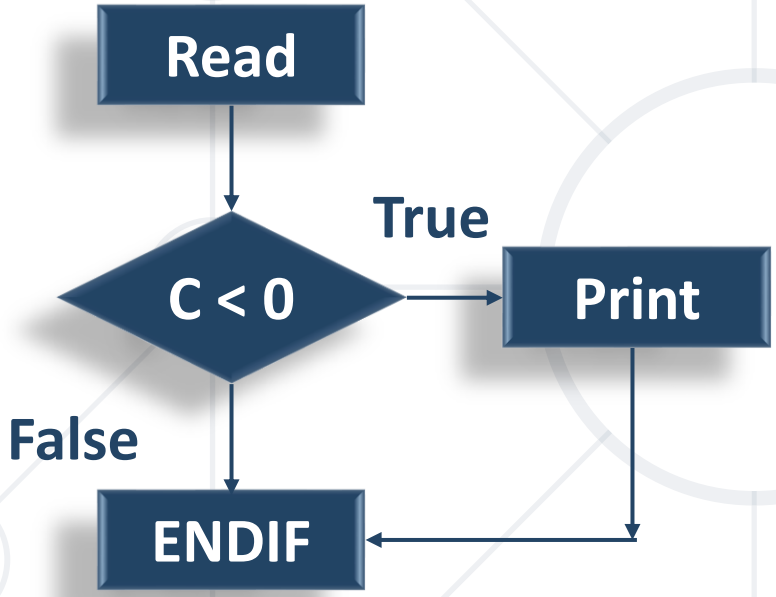
```
Code sample:

READ A
READ B
IF A > B THEN C = 0
ENDIF
```

- 100% statement coverage can be achieved with one test case
- It must ensure that A is greater than B, for example:

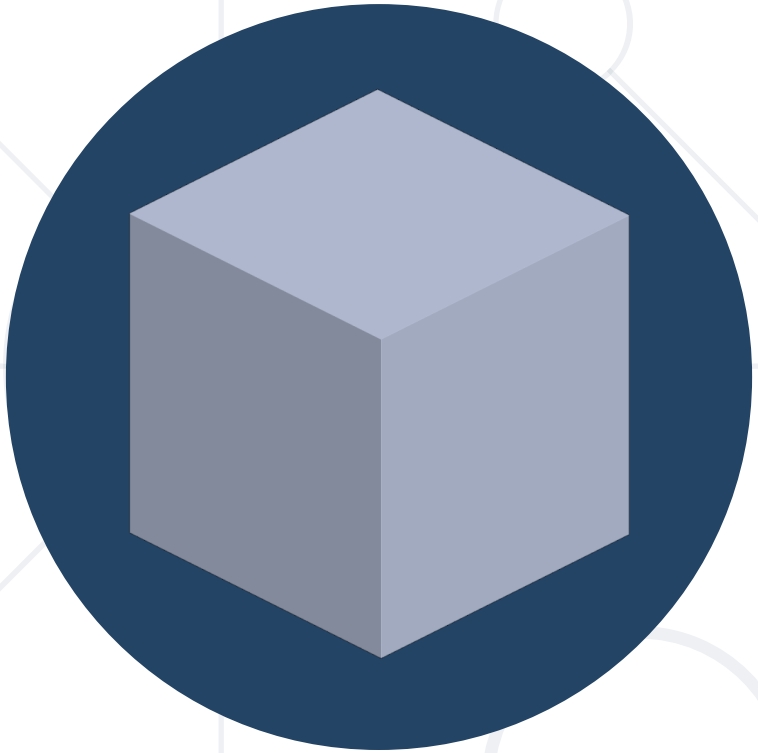
  A = 20, B = 10

Example: Branch/Decision Coverage




```
1 READ A
2 READ B
3 C = A-2*B
4 IF C<0 THEN
5 PRINT "C negative"
6 ENDIF
```

- We have a test that gives us **100% statement coverage** and covers the "**True**" outcome: **A = 20, B = 15**
- In order to cover the "**False**" outcome and achieve **100% branch/decision coverage**, we can use this test: **A = 10, B = 2**



Experience-Based Testing Techniques

Learning from Experience

Experience-Based Techniques

Software University

- Rely on the knowledge and expertise of the testers
- The more experienced the tester, the more effective these techniques tend to be
- Offer **flexibility** as they are not constrained by a rigid testing plan
- Effectively target **known problem** areas and potential weaknesses in the system
- Can be used on their own, but **often complement** specification-based and structure-based testing techniques
- Useful in situations where the **system's documentation may be incomplete** or where the system is too complex

Experience-Based Techniques

Software University

- **Exploratory testing**
  - **Adaptive Approach:** Emphasizes minimal planning and maximum test execution flexibility
  - **Specs or Time Constraints:** Particularly valuable when there are no formal specifications available or when time is severely limited
  - **Hands-on Exploration:** Software actively explored with a combination of domain knowledge and creativity
  - **Rapid Feedback:** Provides rapid feedback, allowing for quick identification and rectification of issues

Experience-Based Techniques

Software University

- **Error guessing**
  - **Complementary Technique**: Used alongside formal methods to enhance testing coverage
  - **Human Creativity**: Leverages tester intuition and creativity to identify potential software issues
  - **Realistic Scenarios**: Testers simulate real-world scenarios and user interactions to uncover hidden or unexpected software defects
  - **Enhanced Quality**: Thinking from a user's perspective

Experience-Based Techniques

Software University

- **Checklist-based Testing**
  - **Guided Testing:** Involves using a predefined checklist of common issues and areas to test as a guide
  - **Experience-Driven:** Checklists are built based on the tester's previous experience, common issues in similar systems, known problem areas, or general testing best practices
  - **Comprehensive Coverage:** Systematically verifying critical aspects of the software
  - **Efficiency and Consistency:** a valuable approach for both experienced and novice testers

Experience-Based Techniques

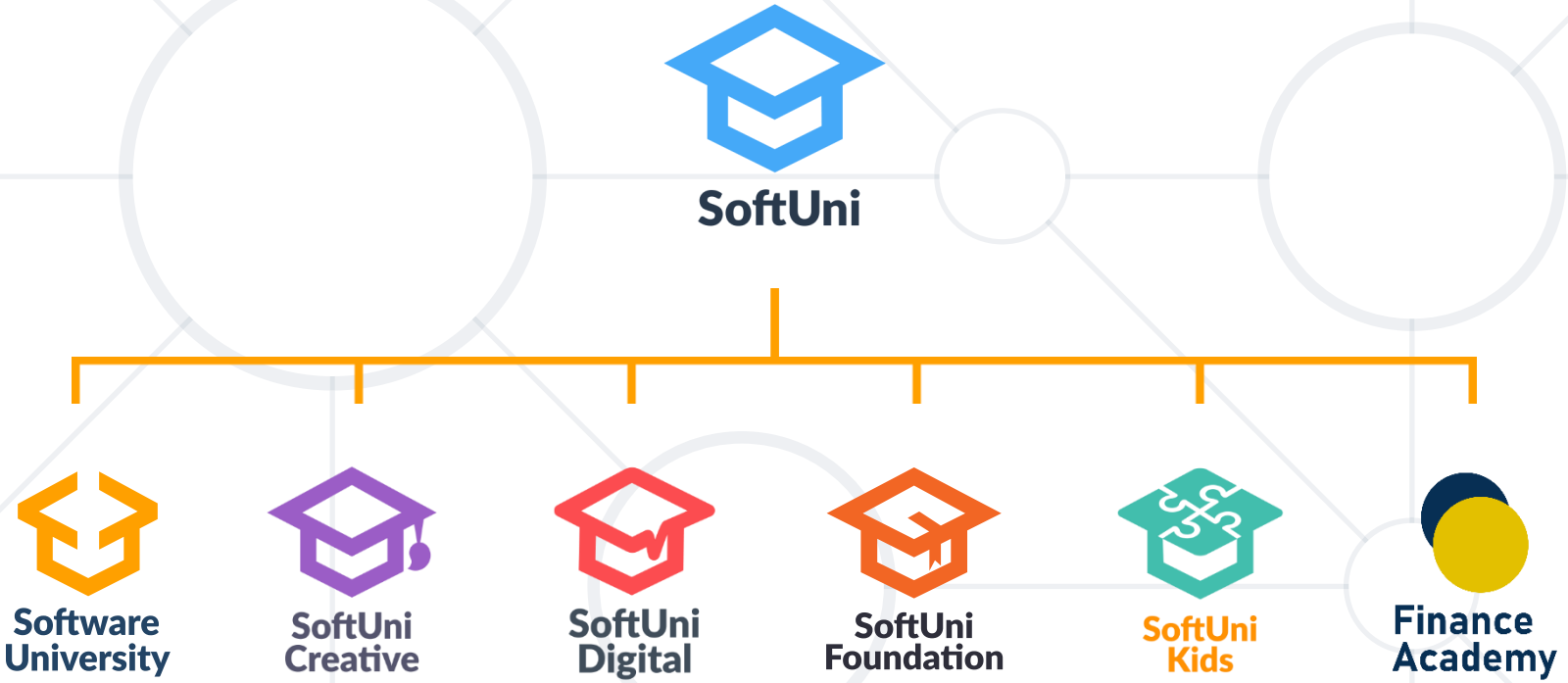
Software University

- **Random Testing or Monkey Testing**
  - **Random Input Generation:** Involves generating inputs to the system randomly, aiming to uncover hard-to-discover bugs
  - **Varied "Smartness" Levels:** Monkeys can have varying levels of "smartness." Some may use completely random data ("dumb" monkeys), while others may incorporate knowledge of the system to guide their inputs ("smart" monkeys)
  - **Unbiased Exploration:** Dumb monkeys provide unbiased exploration
  - **Balance of Creativity and Randomness:** Smart monkeys strike a balance between creativity and randomness

Summary

- There are **static** and **dynamic** testing techniques
  - Static techniques include reviews which increase quality and productivity
  - Dynamic techniques are based on three factors – **structure**, **specification**, **experience**
- Structure-based techniques are called **white box techniques**
- **Experience-based** techniques
- Specification-based techniques are called **black box techniques**

Questions?



SoftUni Diamond Partners

Software University

Trainings @ Software University (SoftUni)

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

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://about.softuni.bg/
- © Software University – https://softuni.bg