GIU
GERMAN
INTERNATIONAL
UNIVERSITY
الجامعة الألمانية الدولية

Faculty of Informatics and Computer Science

Software Construction & Testing

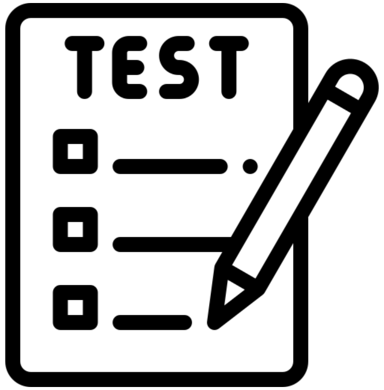
Topic 1

Introduction to Software Construction and Testing

Dr. Ahmed Maghawry

Contents

1. Welcome & course overview
2. Software construction
3. Software Testing
4. Construction and testing methodologies
5. Tools and technologies
6. Best practices and ethics
7. Assessment
8. Conclusion



1. Welcome and course overview

2. Software construction

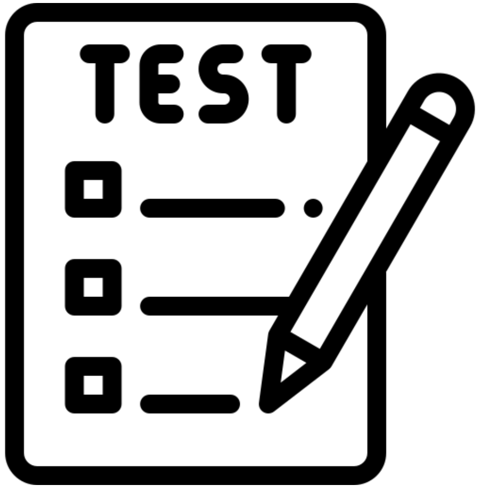
- Software construction is the process of creating software, including the design, development, testing, and deployment of software systems.

Software construction involves a series of activities

- Requirements analysis:
  - Identifying the needs and requirements of the software system, including the functional and non-functional requirements.
- Design:
  - Creating a detailed design of the software system, including its architecture, data structures, algorithms, and user interfaces.
- Implementation:
  - Writing the code for the software system, using programming languages, development tools, and software engineering principles.
- Testing:
  - Verifying and validating the software system to ensure it meets the requirements and works as expected, using various testing techniques and tools.
- Deployment:
  - Installing, configuring, and deploying the software system in the production environment.
- Maintenance:
  - Updating, modifying, and fixing the software system over time, to ensure it continues to meet the changing needs of its users.

3. Software testing

- Software testing is the process of evaluating the quality, functionality, and performance of software to ensure it meets the requirements and expectations of its users.



Software testing can be performed at various stages

- Unit testing:
  - This involves testing individual components or modules of the software to ensure they function correctly.
- Integration testing:
  - This involves testing how different components or modules of the software work together.
- System testing:
  - This involves testing the entire software system to ensure it meets the specified requirements.
- Acceptance testing:
  - This involves testing the software to ensure it meets the acceptance criteria of the customer or end-user.
- Regression testing:
  - This involves testing the software after changes or updates have been made to ensure that no new defects have been introduced.

5 mins

Break

4. Construction and testing methodologies

Software construction methodologies include

- Waterfall:
  - A linear approach where requirements are gathered, designed, implemented, tested, and deployed in a sequential manner.
- Agile:
  - An iterative and incremental approach that focuses on delivering working software in short iterations, with frequent feedback and adaptation to changing requirements.
- V-model:
  - A development process that follows the shape of the V diagram, where requirements are gathered at the top, followed by design, implementation, testing, and deployment.
- Spiral model:
  - A risk-driven approach that iteratively builds and evaluates software, with a focus on identifying and mitigating risks.
- Extreme programming (XP):
  - An iterative and incremental methodology that emphasizes customer satisfaction, teamwork, and frequent deliveries.

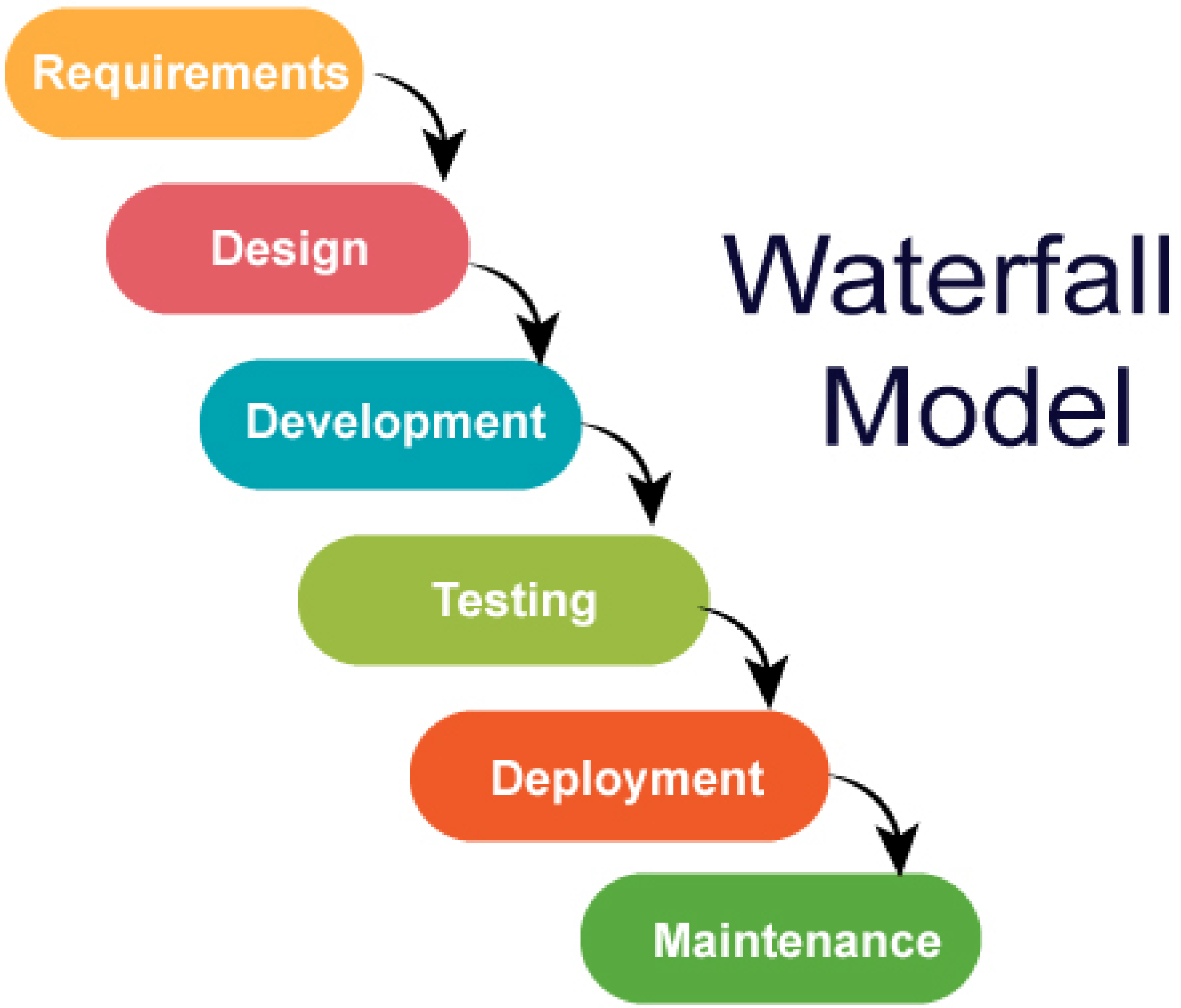
Testing methodologies include

- Black box testing:
  - Testing software without knowledge of the internal workings or code.
- White box testing:
  - Testing software with knowledge of the internal workings or code.
- Gray box testing:
  - Testing software with some knowledge of the internal workings or code.
- Functional testing:
  - Testing software for its intended functionality.
- Non-functional testing:
  - Testing software for its performance, scalability, security, and other non-functional requirements.
- Regression testing:
  - Testing software after changes or updates have been made to ensure that no new defects have been introduced.
- Acceptance testing:
  - Testing software to ensure it meets the acceptance criteria of the customer or end-user.
- Exploratory testing:
  - Testing software in an unscripted and unstructured way to discover new issues or defects.

4. Construction and testing methodologies

Software construction methodologies include

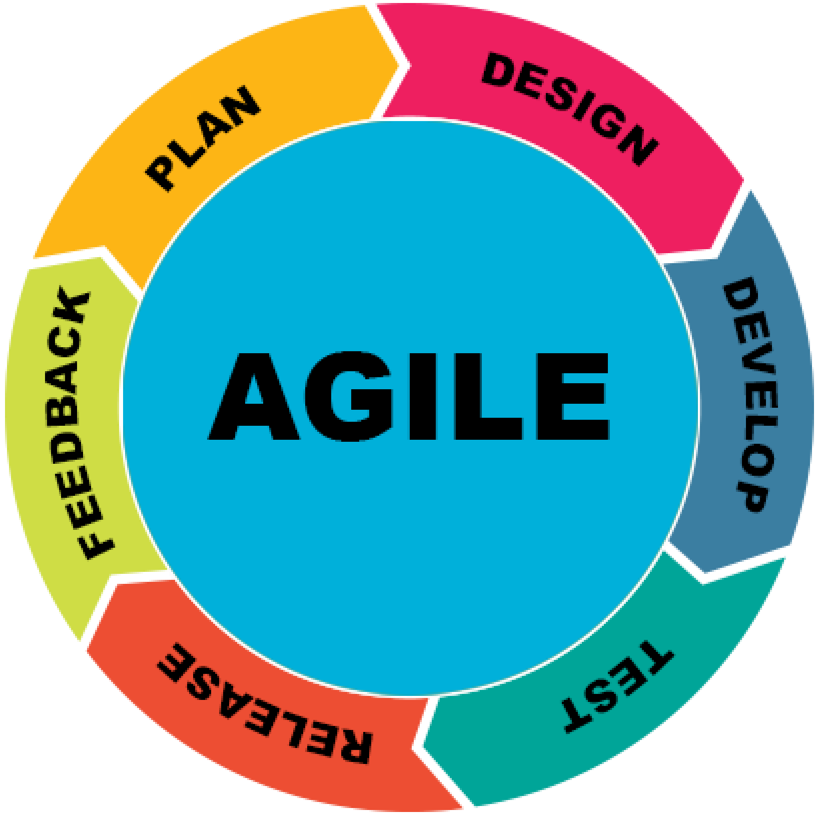
- Waterfall:
  - A linear approach where requirements are gathered, designed, implemented, tested, and deployed in a sequential manner.



4. Construction and testing methodologies

Software construction methodologies include

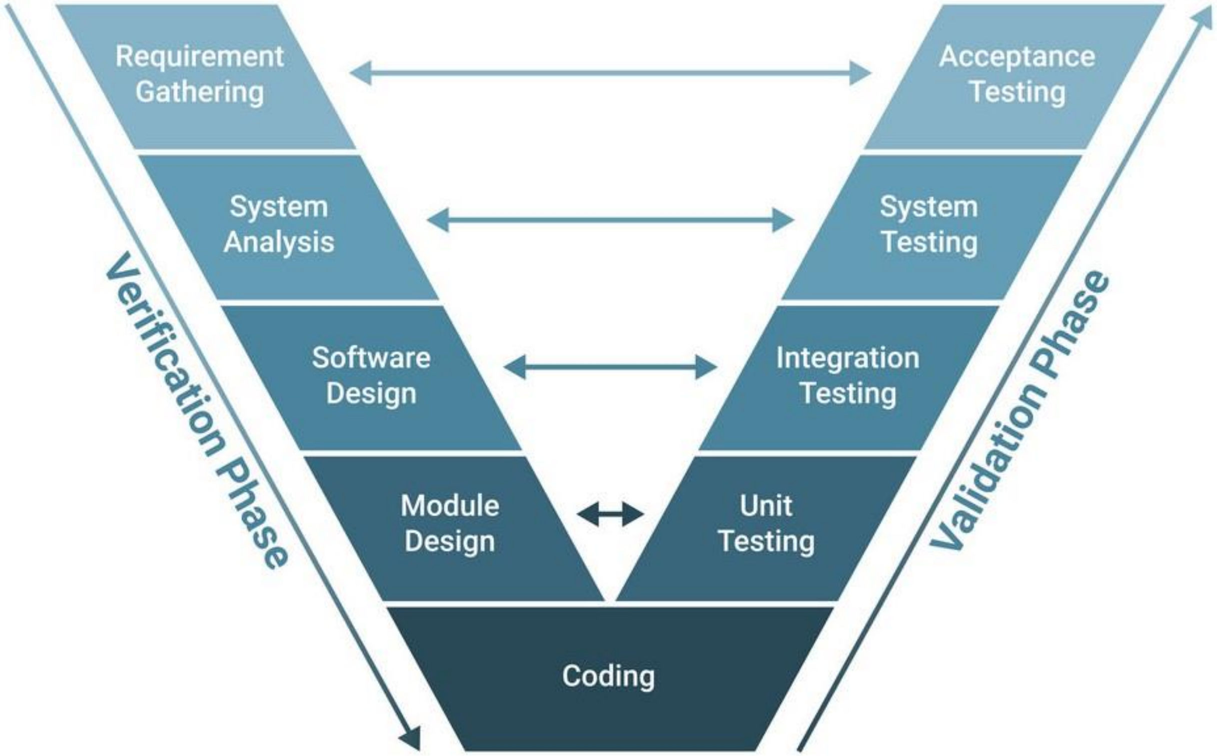
- Agile:
  - An iterative and incremental approach that focuses on delivering working software in short iterations, with frequent feedback and adaptation to changing requirements.



4. Construction and testing methodologies

Software construction methodologies include

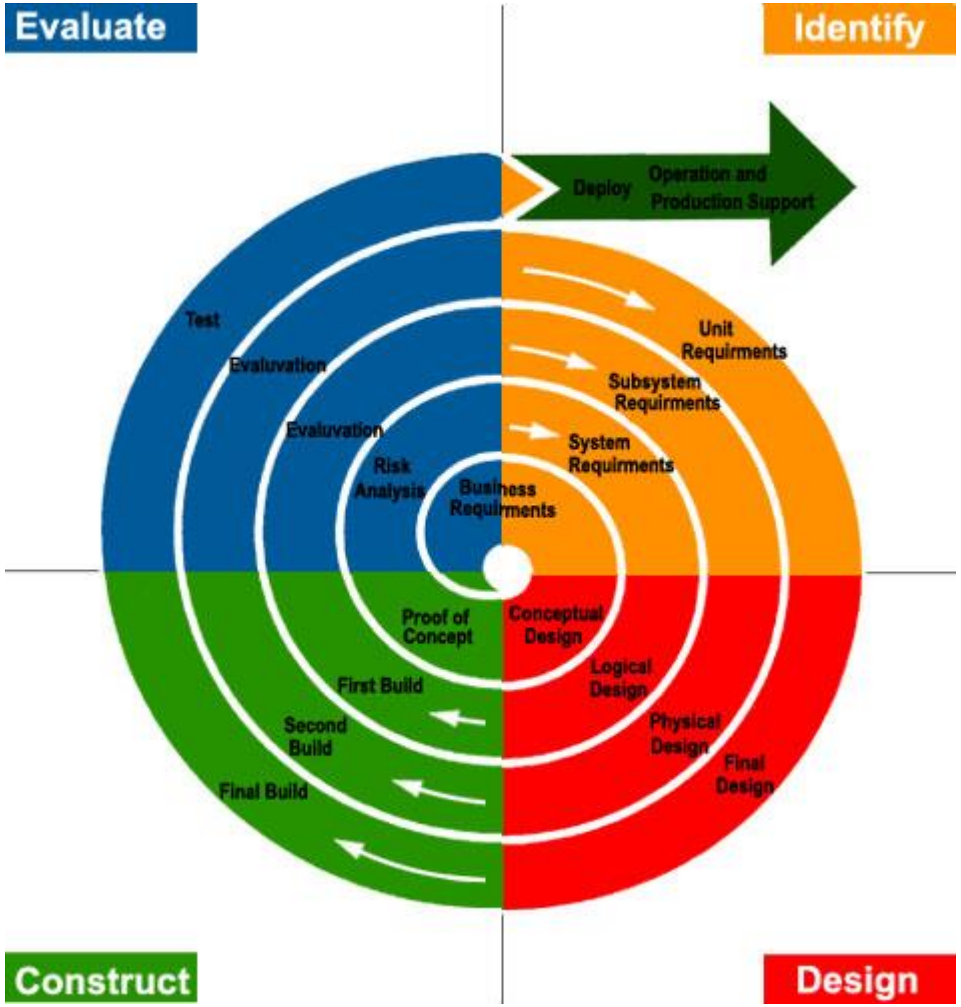
- V-model:
  - A development process that follows the shape of the V diagram, where requirements are gathered at the top, followed by design, implementation, testing, and deployment.



4. Construction and testing methodologies

Software construction methodologies include

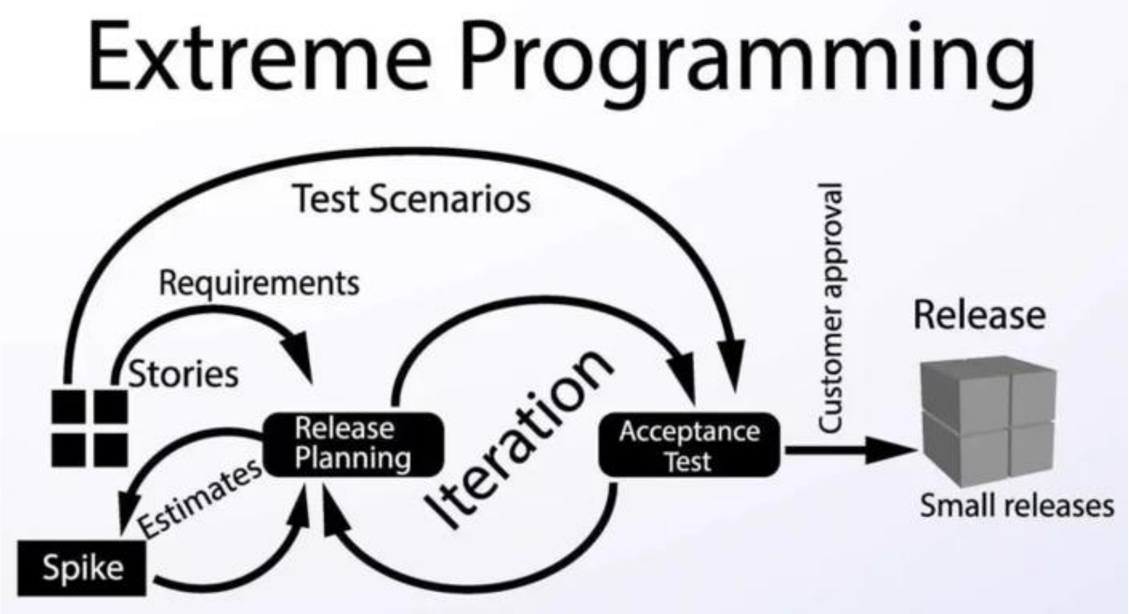
- Spiral model:
  - A risk-driven approach that iteratively builds and evaluates software, with a focus on identifying and mitigating risks.



4. Construction and testing methodologies

Software construction methodologies include

- Extreme programming (XP):
    - An iterative and incremental methodology that emphasizes customer satisfaction, teamwork, and frequent deliveries.



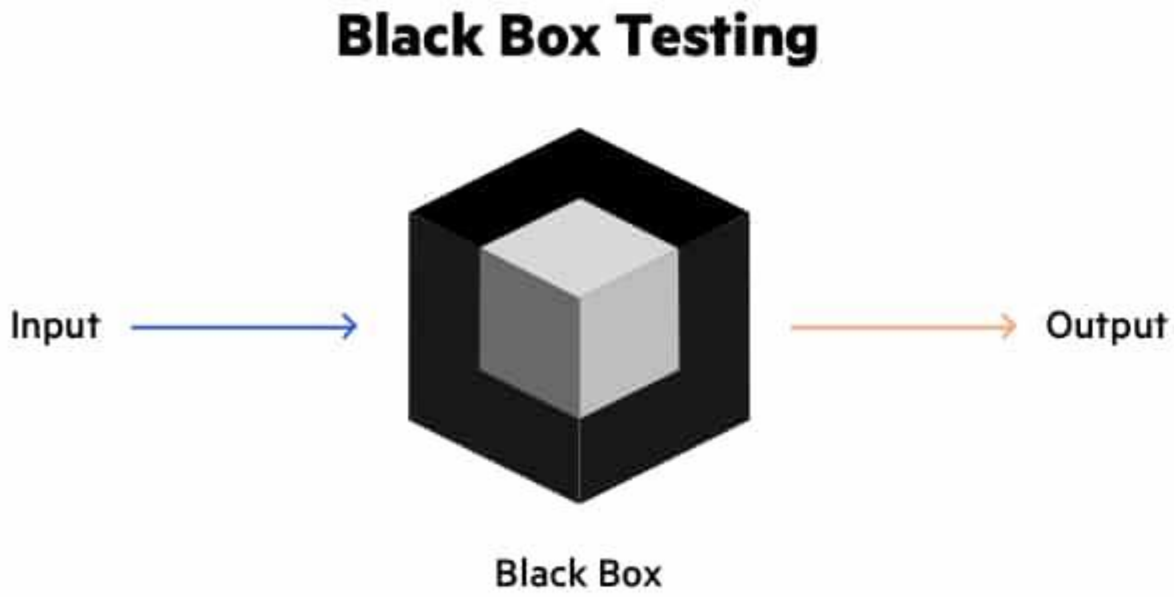
5 mins

Break

4. Construction and testing methodologies

Testing methodologies include

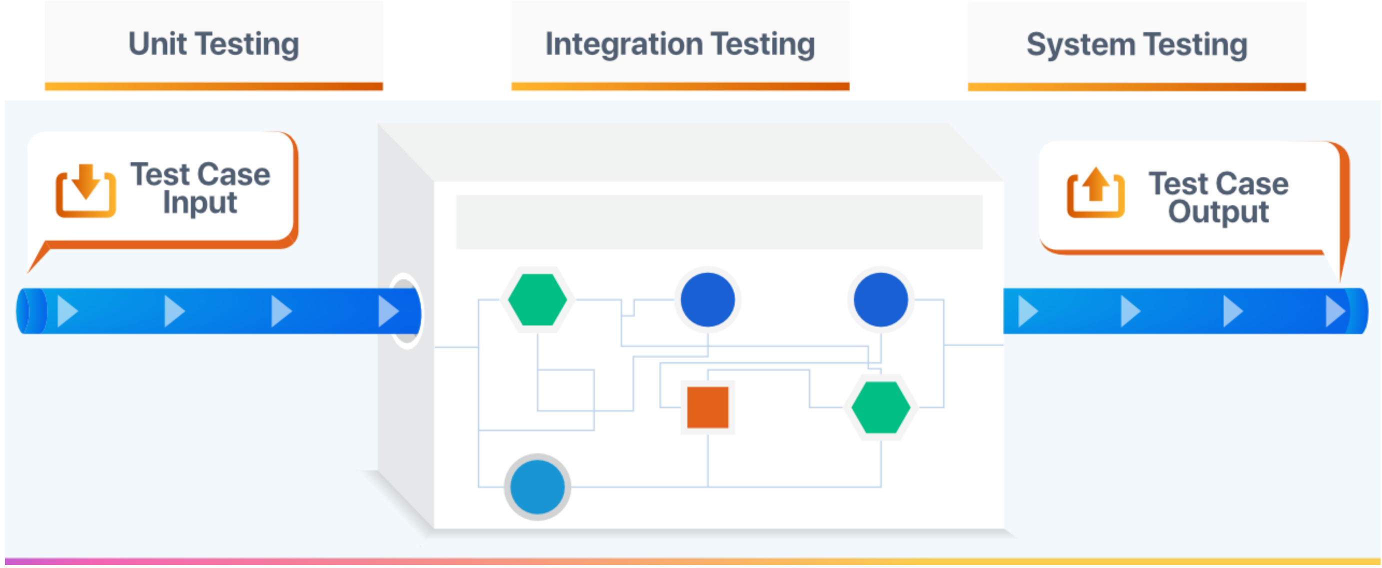
- Black box testing:
  - Testing software without knowledge of the internal workings or code.



4. Construction and testing methodologies

Testing methodologies include

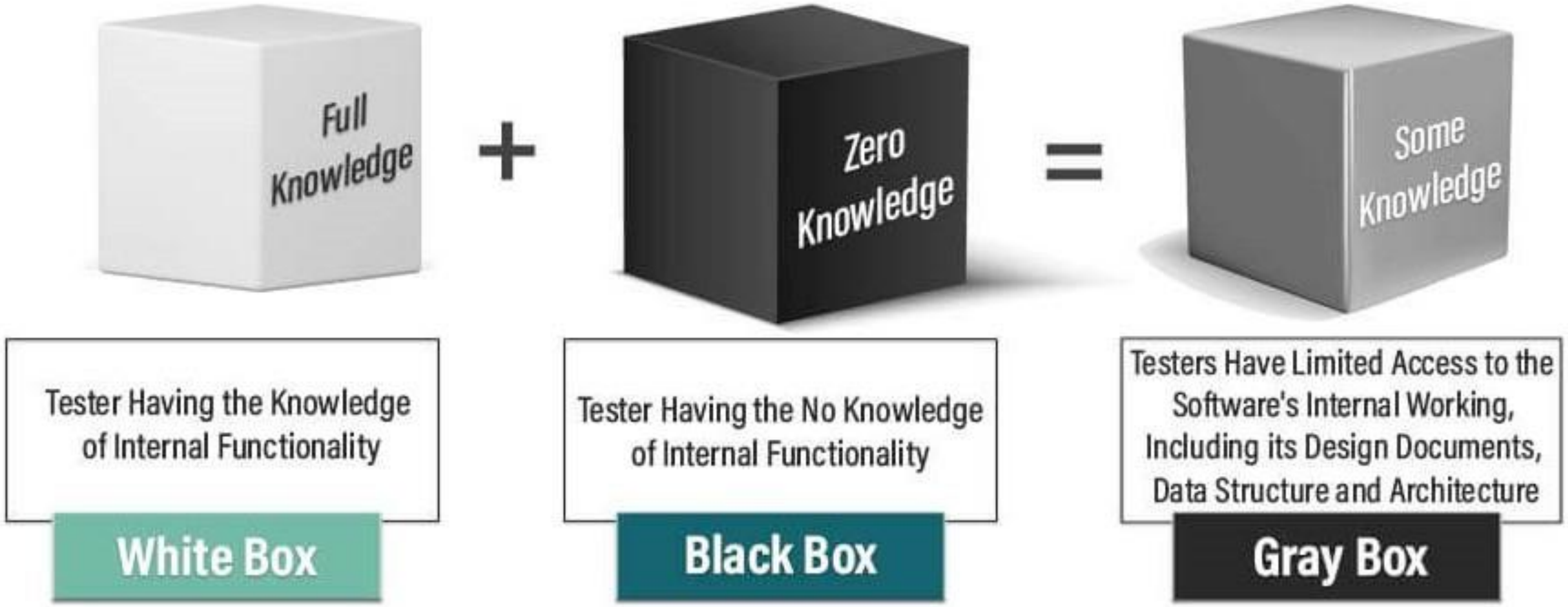
- White box testing:
  - Testing software with knowledge of the internal workings or code.



4. Construction and testing methodologies

Testing methodologies include

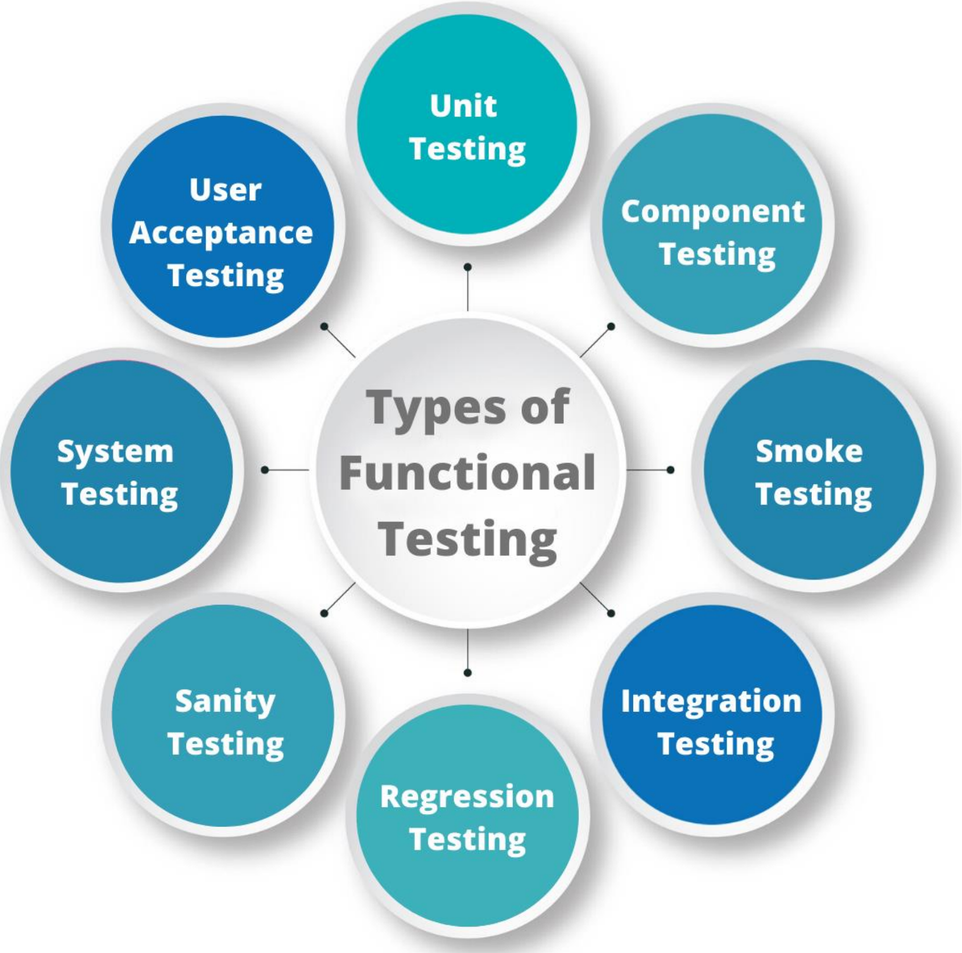
- Gray box testing:
  - Testing software with some knowledge of the internal workings or code.


Full Knowledge
+
Zero Knowledge
=
Some Knowledge
Tester Having the Knowledge of Internal Functionality
White Box
Tester Having the No Knowledge of Internal Functionality
Black Box
Testers Have Limited Access to the Software's Internal Working, Including its Design Documents, Data Structure and Architecture
Gray Box

4. Construction and testing methodologies

Testing methodologies include

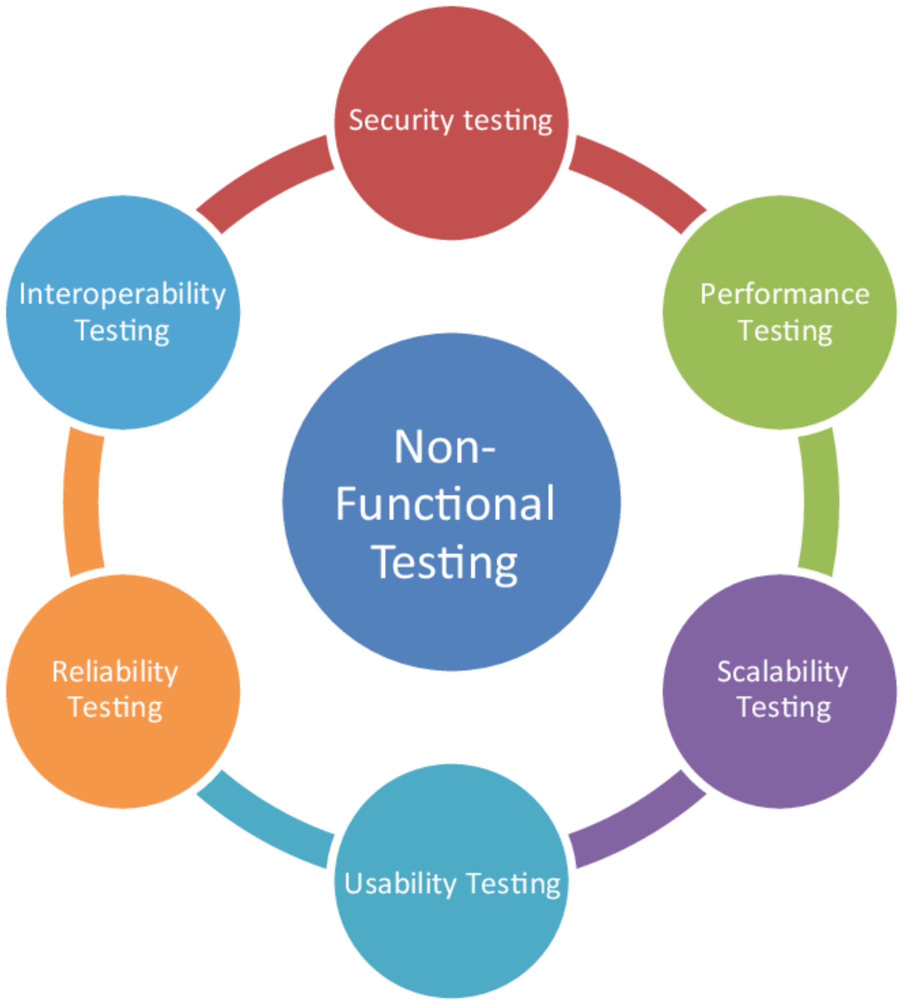
- Functional testing:
  - Testing software for its intended functionality.



4. Construction and testing methodologies

Testing methodologies include

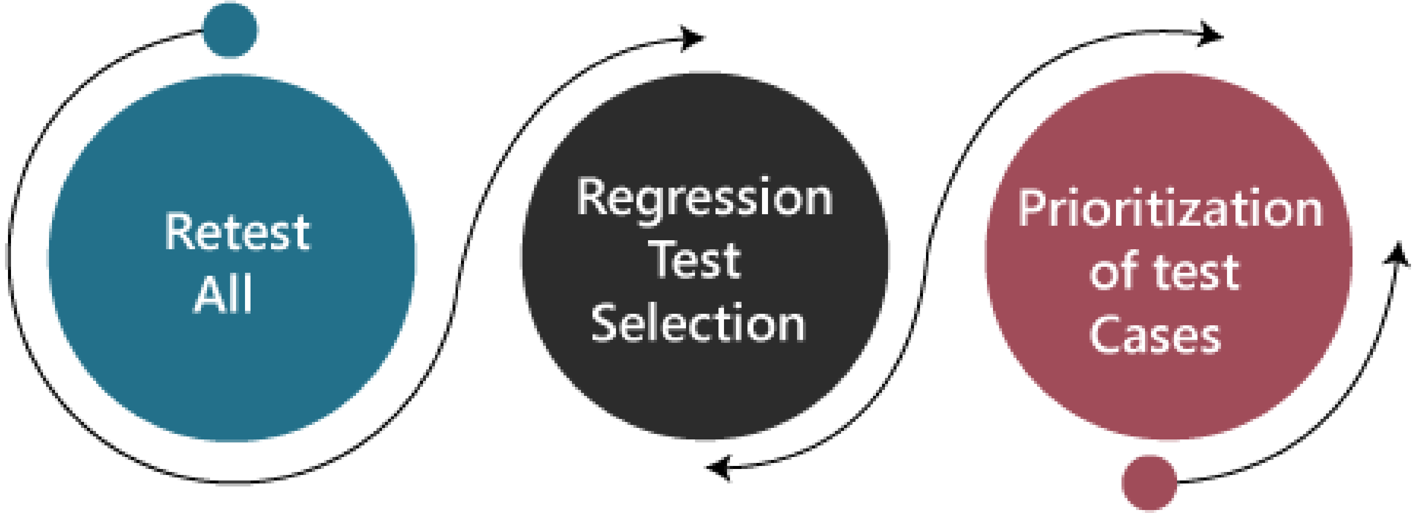
- Non-functional testing:
  - Testing software for its performance, scalability, security, and other non-functional requirements.



4. Construction and testing methodologies

Testing methodologies include

- Regression testing:
  - Testing software after changes or updates have been made to ensure that no new defects have been introduced.



4. Construction and testing methodologies

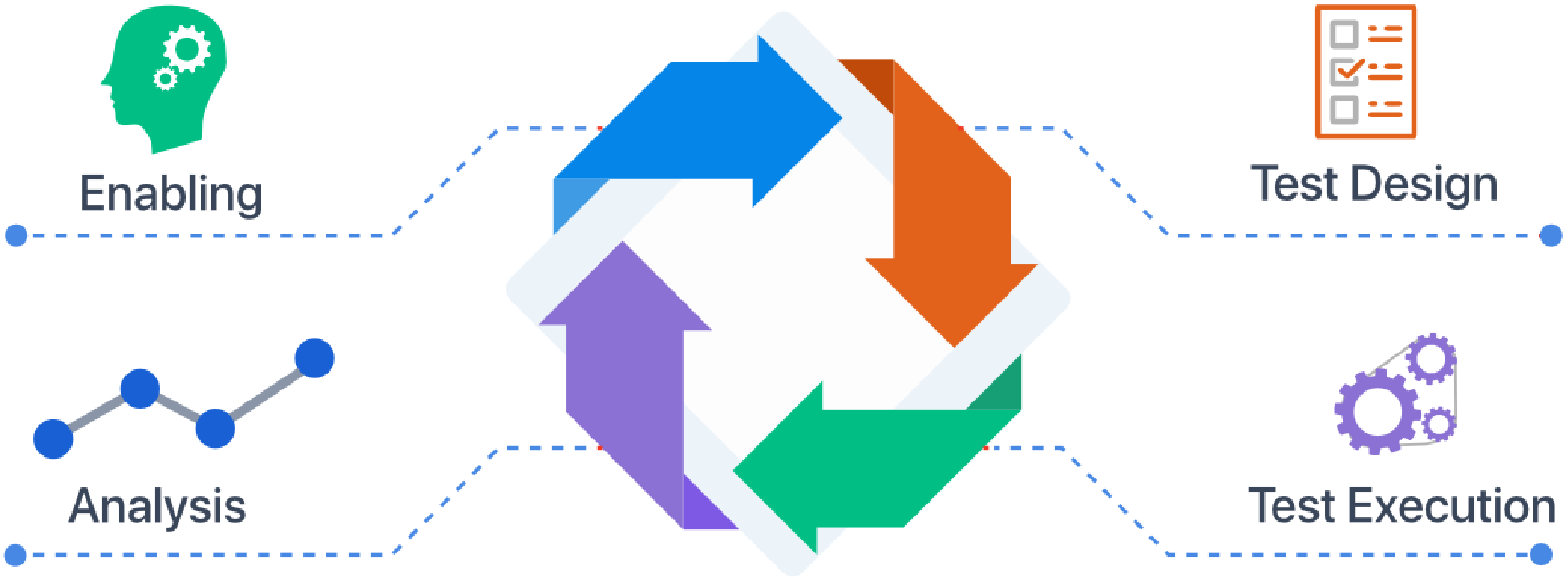
Testing methodologies include

- Acceptance testing:
  - Testing software to ensure it meets the acceptance criteria of the customer or end-user.

4. Construction and testing methodologies

Testing methodologies include

- Exploratory testing:
  - Testing software in an unscripted and unstructured way to discover new issues or defects.



Break

10 mins

5. Tools and technologies

Software construction

- Integrated Development Environments (IDEs)
  - Eclipse, Visual Studio, IntelliJ IDEA, etc.
- Version Control Systems
  - Git, SVN, Mercurial, etc.
- Agile Project Management Tools
  - Jira, Trello, Asana, etc.
- Code Editors
  - Sublime Text, Atom, Emacs, etc.
- Build Automation Tools
  - Maven, Gradle, Ant, etc.
- Deployment Automation Tools
  - Ansible, Puppet, Chef, etc.
- Continuous Integration/Continuous Deployment (CI/CD) Tools
  - Jenkins, Travis CI, CircleCI, etc.
- Code Analysis Tools
  - SonarQube, CodeCoverage, etc.
- Code Generation Tools
  - Eclipse Code Generation, Visual Studio Code Generator, etc.
- Test-Driven Development (TDD) Tools
  - JUnit, TestNG, NUnit, etc.

Software Testing

- Test Management Tools
  - TestRail, TestLink, PractiTest, etc.
- Automated Testing Tools
  - Selenium, Appium, TestComplete, etc.
- Performance Testing Tools
  - JMeter, LoadRunner, Gatling, etc.
- Security Testing Tools
  - Burp Suite, OWASP ZAP, Vega, etc.
- Testing Frameworks
  - JUnit, TestNG, NUnit, etc.
- Acceptance Testing Tools
  - Cucumber, JBehave, SpecFlow, etc.
- Regression Testing Tools
  - TestComplete, TestLink, etc.
- Defect Tracking Tools
  - JIRA, Bugzilla, Redmine, etc.
- Test Data Management Tools
  - TestData Manager, TestData Pro, etc.
- Continuous Testing Tools
  - Jenkins, Travis CI, CircleCI, etc.

6. Best practices and ethics

Software construction

- **Follow a software development process:**
  - Use a software development process such as Agile, Scrum, or Waterfall to ensure that your software is well-structured, maintainable, and meets the requirements.
- **Use version control:**
  - Use version control systems such as Git or SVN to track changes, collaborate with others, and maintain a history of changes.
- **Write clean and readable code:**
  - Write code that is easy to read, understand, and maintain. Follow coding standards and use meaningful variable and function names.
- **Use modular design:**
  - Design software that is modular, with separate components that can be easily modified, replaced, or extended.
- **Use abstraction and encapsulation:**
  - Use abstraction and encapsulation to hide implementation details and make software components more flexible and reusable.
- **Test-Driven Development (TDD):**
  - Write automated tests before writing code to ensure that the code meets the requirements and is of high quality.
- **Continuous Integration/Continuous Deployment (CI/CD):**
  - Use CI/CD tools such as Jenkins or Travis CI to automate the build, test, and deployment process.

Software Testing

- **Test early and often:**
  - Test software early and often to catch defects and bugs before they become more serious problems.
- **Test thoroughly:**
  - Test software thoroughly, including functional testing, performance testing, security testing, and usability testing.
- **Use automated testing:**
  - Use automated testing tools such as Selenium or Appium to automate regression testing and improve testing efficiency.
- **Test for accessibility:**
  - Test software for accessibility to ensure that it can be used by people with disabilities.
- **Test for security:**
  - Test software for security vulnerabilities and ensure that it meets security standards.
- **Use testing frameworks:**
  - Use testing frameworks such as JUnit or TestNG to write and run tests efficiently.
- **Use defect tracking tools:**
  - Use defect tracking tools such as JIRA or Bugzilla to track and manage defects and bugs.

Ethics

- **Honesty and integrity:**
  - Be honest and have integrity in your work, and avoid any actions that could compromise the quality or reliability of the software.
- **Respect for users:**
  - Respect users' privacy and data, and ensure that software is designed with their needs and expectations in mind.
- **Responsibility:**
  - Take responsibility for the quality and reliability of the software, and be accountable for any defects or bugs that may occur.
- **Transparency:**
  - Be transparent in your work, and provide clear and accurate information about the software and its capabilities.
- **Collaboration:**
  - Collaborate with others, including developers, testers, and users, to ensure that the software meets their needs and expectations.
- **Continuous learning:**
  - Continuously learn and improve your skills and knowledge, and stay up-to-date with the latest trends and technologies.
- **Compliance with laws and regulations:**
  - Ensure that software complies with relevant laws and regulations, such as data privacy laws and security standards.

7. Assessment

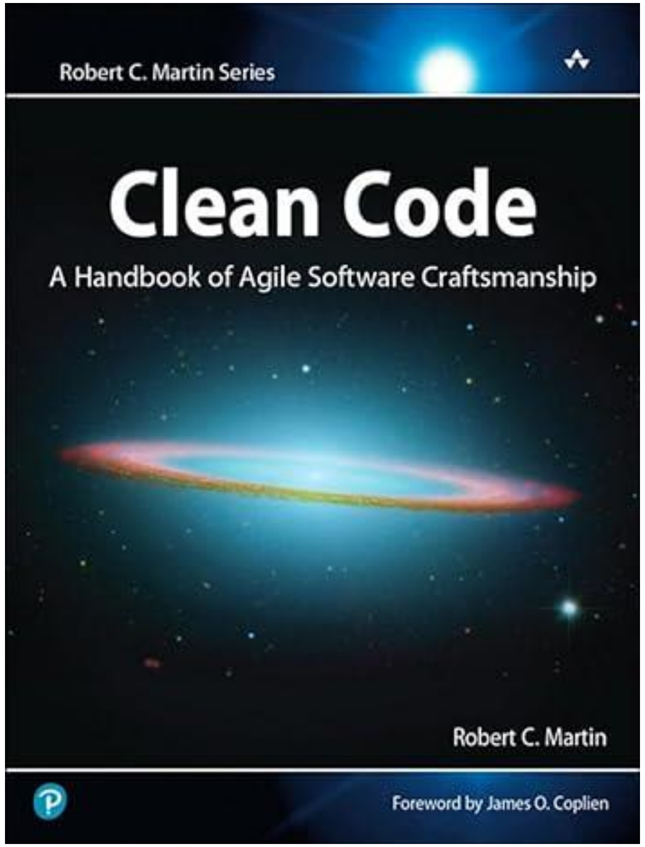
Student Assessment Methods	Assessment Weights
Project	40 %
Midterm Exam	25 %
Final Exam	35 %

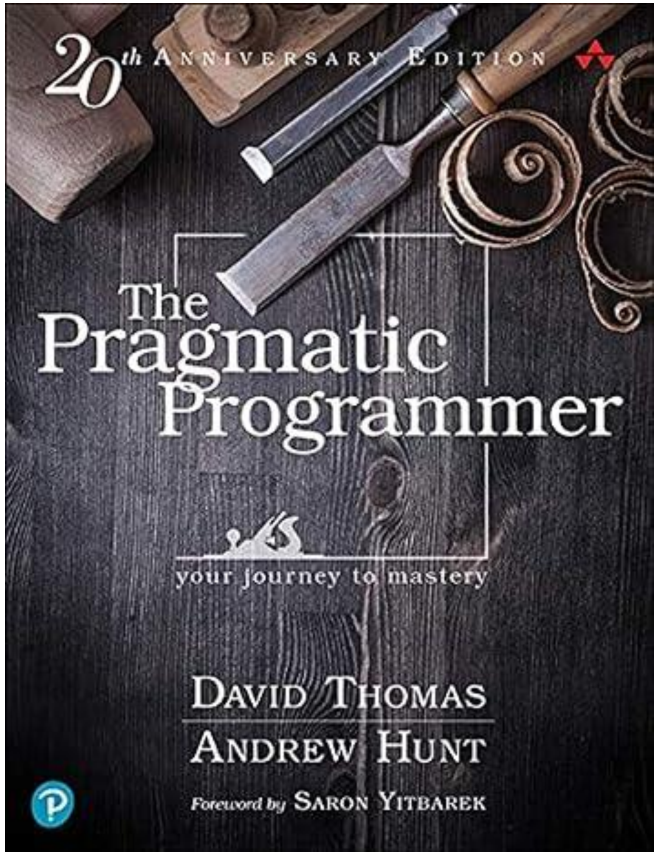
8. Conclusion

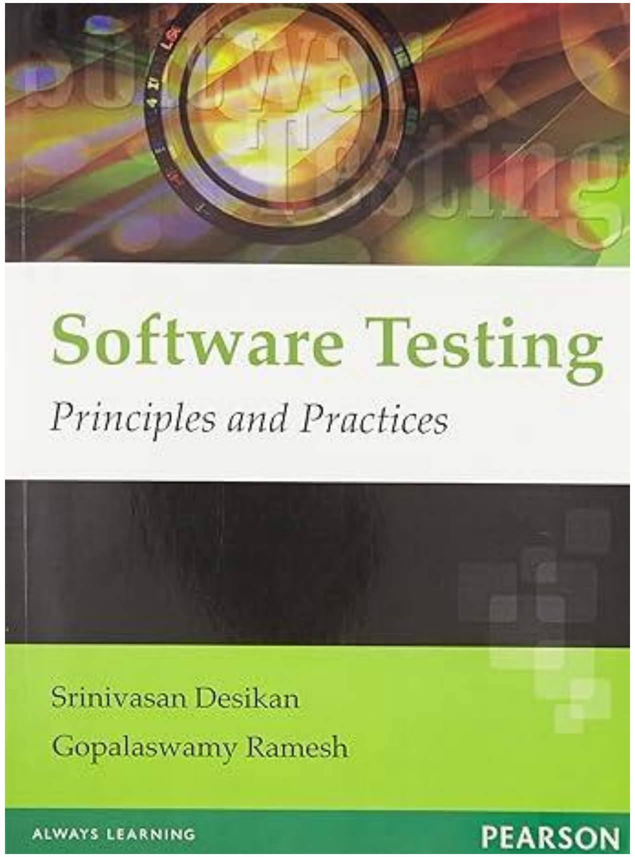
- **Software construction and testing are**
  - essential phases of the software development process
- **High-quality software requires**
  - careful construction and thorough testing
- **Software construction involves**
  - designing, implementing, and integrating software components
- **Software testing ensures that**
  - software meets requirements, is reliable, and performs as expected
- **Best practices and ethical considerations are essential**
  - for effective software construction and testing
- **Continuous learning and improvement are necessary**
  - to stay up-to-date with evolving technologies and trends

References

- Martin, R. C. Clean code: A handbook of agile software craftsmanship. Prentice Hall.
- Hunt, A., & Thomas, D. The pragmatic programmer: Your journey to mastery. Addison-Wesley.
- Desikan, S., & Ramesh, G. Software testing: Concepts and practices. Pearson Education.


Robert C. Martin Series
Clean Code
A Handbook of Agile Software Craftsmanship
Robert C. Martin
Foreword by James O. Coplien





GIU GERMAN INTERNATIONAL UNIVERSITY

الجامعة الألمانية الدولية

Faculty of Informatics and Computer Science

Questions?