PBA TEST - EXAM QUESTIONS (Spring 2018)

Page 4 and forward in this document, are the questions you can encounter for the exam. We guarantee that this is the exact phrasing you will find in the questions, but do not provide any guarantee about how they are assembled into individual questions.

The three examples below illustrate how real questions could look:

Example Question 1

For this exercise, you will need your code for **Exercise-5**, **UNIT Testing, Testable Code, Mocking and Code Coverage**
Info: You should spend approximately 80% of the time on Part-1 and 20% on Part-2

Part-1 Unit Testing

- Explain what makes a test a unit test.
- Explain properties that make a unit test a good unit test
- Provide examples (in words, not code) of JUnit tests which are not unit tests
- What is a test fixture?

Demonstrate your solution to Exercise-5.
You should (as a minimum):

- Explain the necessary steps you did to make the code testable, and some of the patterns involved in this step
- Execute your test cases
- Explain basically about JUnit, Hamcrest, Mockito and Jacoco, and what problems they solve for testers
- Demonstrate how you used Mockito to mock away external Dependencies
- Demonstrate how/where you did state-based testing and how/where you did behavior-based testing
- Explain about Coverage Criterias, using the results presented by running Jacoco (or a similar tool) against you final test code.
- Explain/demonstrate what was required to make this project use, JUnit (Hamcrest), Mockito and Jacoco

Part-2 Final Assignment

- Explain about your Final Assignment solution.

Example Question 2

Info: You should spend approximately 80% of the time on Part-1 and 20% on Part-2.

Part-1 Final Assignment

- Explain about your Final Assignment solution.

Part-2 General Test Questions

- Explain about Mike Cohn's Test Pyramid and why it's relevant for test planning
- Discuss some of the problems with automated GUI tests and what makes such test "vulnerable"
- Explain about the Selenium, its purpose, and how it "fits into" the testing world
- Explain how, or if, tools like Jacoco or similar can be used to measure the "quality" of our test

---

Example Question 3

For this exercise, you will need your code for **Exercise-5**, **UNIT Testing, Testable Code, Mocking and Code Coverage**
Info: You should spend approximately 80% of the time on Part-1 and 20% on Part-2.

Part-1 Unit Testing

- Explain what makes a test a unit test.
- Explain properties that make a unit test a good unit test
- Provide examples (in words, not code) of JUnit tests which are not unit tests
- What is test fixture?

Demonstrate your solution to Exercise-5.
You should (as a minimum):

- Explain the necessary steps you did to make the code testable, and some of the patterns involved in this step
- Execute your test cases
- Explain basically about JUnit, Hamcrest, Mockito and Jacoco, and what problems they solve for testers
- Demonstrate how you used Mockito to mock away external Dependencies
- Demonstrate how/where you did state-based testing and how/where you did behavior-based testing
- Explain about Coverage Criterias, using the results presented by running Jacoco (or a similar tool) against you final test code.
- Explain/demonstrate what was required to make this project use, JUnit (Hamcrest), Mockito and Jacoco

Part-2 General Test Questions

- Explain about Mike Cohn's Test Pyramid and why it's relevant for test planning
- Discuss some of the problems with automated GUI tests and what makes such test "vulnerable"
- Explain about the Selenium, its purpose, and how it "fits into" the testing world
- Explain how, or if, tools like Jacoco or similar can be used to measure the "quality" of our test

---

These are the exercises, done throughout the semester, which exam questions will refer to.

You must be able to locate these exercises fast, given the exercise-number + title, once you see a question.

Exercise -1	***Static Testing Techniques***	https://github.com/datsoftlyngby/soft2018spring-test-teaching-material/blob/master/exercises/Static%20Test%20Techniques%20Exercises.pdf
Exercise -2	***Test Case Design***	https://github.com/datsoftlyngby/soft2018spring-test-teaching-material/blob/master/exercises/Test%20Case%20Exercises.pdf
Exercise -3	***Unit Testing 2***	https://github.com/datsoftlyngby/soft2018spring-test-teaching-material/blob/master/exercises/DAT%20first%20semester%20exam%202.pdf
Exercise -4	***Midterm assignment Project 1 (Testing Real Life Code)***	https://github.com/datsoftlyngby/soft2018spring-test-teaching-material/blob/master/exercises/Midterm%20AssignmentNEW.pdf
Exercise -5	***Midterm assignment Project 2 (Unit Testing, Testable Code, Mocking and Code Coverage)***	https://github.com/datsoftlyngby/soft2018spring-test-teaching-material/blob/master/exercises/Midterm%20AssignmentNEW.pdf
Exercise -6	***Automated System Testing with Selenium2***	https://github.com/datsoftlyngby/soft2018spring-test-teaching-material/blob/master/exercises/SeleniumExerciseV2.pdf
Exercise -7	***Technical proof of concept***	Exercise description found in the end of these slides: https://github.com/datsoftlyngby/soft2018spring-test-teaching-material/blob/master/slides/Cucumber.pdf

Fundamentals of testing; Testing throughout the software life cycle

- Explain the common objectives of testing
- Explain why testing is necessary by giving examples
- Explain the difference between testing and debugging
- Explain some of the seven principles of testing
- Explain the objectives of testing in different phases of the software life cycle (in general)
- Explain the characteristics of good testing that are applicable to any life cycle model
- Explain the major objectives of the different test levels: unit test, integration test, system test and acceptance test
- Explain the 4 test types: functional testing, non-functional testing, structural testing and change related testing
- Identify and describe relevant non-functional tests from one of your exercises/projects during the course
- Describe the purpose of regression testing
- Explain the V-model:

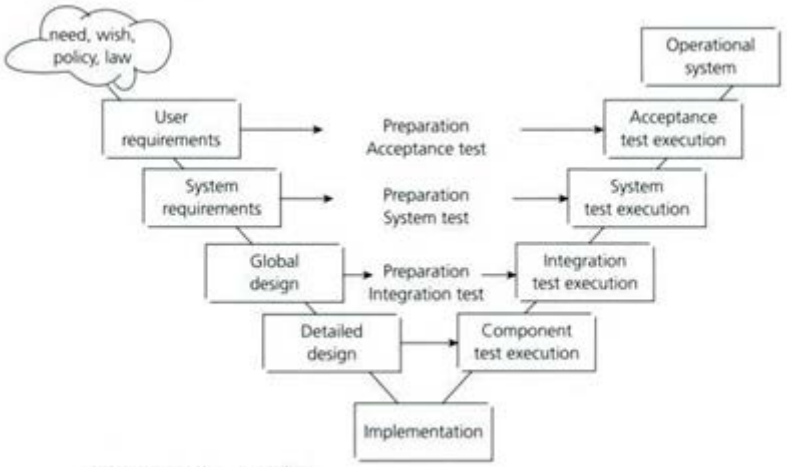
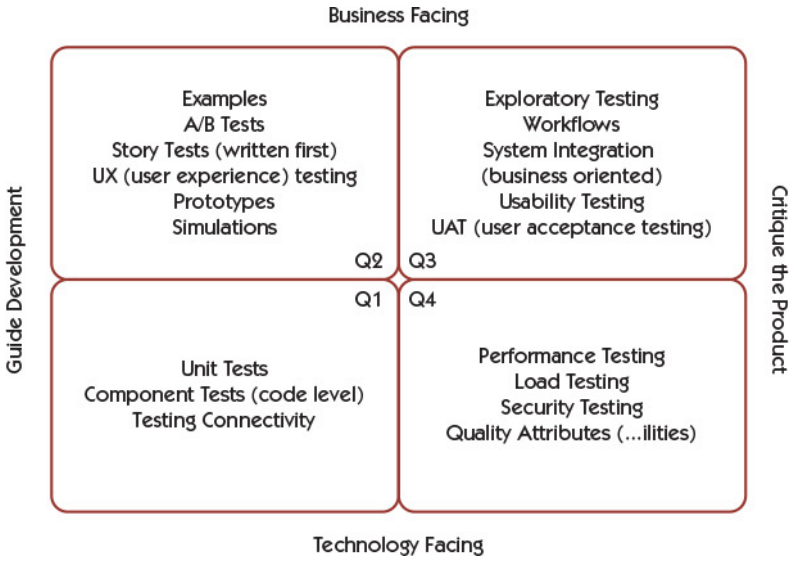


FIGURE 2.2 V-model

- Describe tools that can support test activities at the different test levels in the V-model. You may demonstrate one or more tools.
- Describe strengths and weaknesses of the V-model. What are the alternatives?
- Explain about Mike Cohn's Test Pyramid and why it's relevant for test planning
- Explain the objectives of testing in the Agile Testing Quadrants:



- Explain how agile development and testing differ from traditional approaches
- Explain the test activities in agile projects, including when which test activities are initiated
- Explain how risk is coped with in agile project as compared to traditional approaches
- Explain the roles and skills of a tester in agile projects
- Explain agile development and testing in the context of the Agile Manifesto's 4 statements of values
    - Individuals and interactions *over* processes and tools
    - Working software *over* comprehensive documentation
    - Customer collaboration *over* contract negotiation
    - Responding to change *over* following a plan

Static testing techniques

- Explain what static testing means
- Recognize software artifacts that can be examined by different static techniques
- Describe the importance and value of using static techniques for the assessment of software artifacts
- Explain how static techniques can improve software quality
- Explain the difference between static and dynamic techniques, considering objectives, types of defects to be identified, and the role of these techniques within the software life cycle
- Explain when in the development life cycle static techniques can be used
- List some alternatives to static techniques that can help reduce costs for defects in software projects
- Explain the following cost curve and discuss how static techniques can help reduce cost of defects

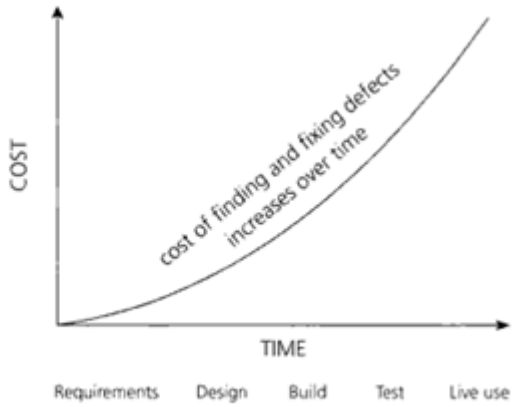


FIGURE 1.2 Cost of defects

- List the activities, roles and responsibilities of a formal review
- Explain the differences between the different types of reviews: informal review, technical review, walkthrough and inspection
- Explain the factors for successful performance of reviews
- Describe the benefits of static analysis in a tool. Use examples by demonstrating static analysis in a tool and explain the results
- Explain what typical defects can be identified by static analysis in a tool and compare them to reviews and dynamic testing

---

Demonstrate your solution to the study point exercise "Static Test Techniques" OR another similar project that you have made.

- You should:
    - Demonstrate how your Triangle code follows the coding standards defined in your development tool
    - Demonstrate the central code metrics for your Triangle program in your development tool
    - Explain what to do if your tool reports poor metrics results for your code (e.g. cyclomatic complexity)
    - List the best practices and code conventions that you find most important for a team to follow and that should be put into a coding standard document

Test design techniques

- Point out the main differences between white box and black box testing
- List some of the elements that a high quality test case in your opinion is comprised of
- Elaborate on the role of oracles and their usage with regards to test cases
- Explain the overall focus and important considerations in relation to test case design
- Explain which tasks to carry out and decisions to take in the test implementation process
- Describe how the techniques equivalence partitioning and boundary value analysis are applied in test case design and outline the reasons for using each technique
- Describe the benefits of decision tables and state transition models in the context of test case design
- How, or if, tools like Jacoco or similar can be used to measure the "quality" of our tests
- Explain the approach of Exploratory Testing, including its purpose and role in agile testing

---

Demonstrate your solution to the study point exercise "Test Case Design"

- You should (as a minimum):
    - Explain how you have designed your different test cases
    - Explain which equivalence partitions and boundary values your test cases stem from, plus why you have chosen these specifically
    - Account for any open boundaries in your test cases
    - Demonstrate whether you have chosen a two or three value approach in your boundary value analysis
    - Demonstrate how a decision table is created and used in your test cases

Unit testing and Mocking

- Explain what makes a test a unit test
- Explain a few properties that make a unit test a good unit test and rules for what makes code testable (or untestable).
- Provide examples (in words, not code) of JUnit tests which are not Unit tests
- What is a test fixture?
- Explain, using a short example, the rationale behind the *Arrange - Act - Assert* strategy for testing
- Explain about the JUnit Framework, how to include it in a project, and how to write tests
- Explain using examples you have provided, how to unit test:
    - A method with no return value
    - Exceptional behavior
    - A Single method, with multiple input values (and corresponding expected results)
    - A test with dependencies of other classes
- Why would you consider alternative matchers like for example Hamcrest matchers in a JUnit project? Provide examples to demonstrate, how to set up a project to use Hamcrest, the effect of using Hamcrest matchers compared to JUnit's basic Asserts.
- Explain about the SOLID principles plus a number of the additional rules given in Effective Unit Testing focusing on WHY following these rules makes code more testable
- Explain why testers just love the Dependency Injection Pattern
- Explain the topic “data driven testing” and ways to do it, using both “plain” JUnit and possibly alternative libraries which easily let you read test data from external sources files (cvs, Excel, etc.)

- Demonstrate, preferably using a real life example, ways to handle many input values, and expected values
- Explain strategies/frameworks used to unit test single classes with dependencies
- Explain about Test Doubles and specific types like Dummy Objects, Test Stubs, Mock Objects and their purpose for unit testing objects with dependencies.
- Explain the difference between state-based testing versus behavior-based testing, and provide practical examples using JUnit and a relevant mocking framework.
- Explain about the development process TDD in general, and the steps involved when using this process.
- Explain about Test-Driven Development and its role as a design tool
- Explain briefly about the architecture in JUnit 5.X, how to set up a project for JUnit 5.X testing and provide a few examples to demonstrate some of the new features.

Demonstrate your solution to the study point exercise "Unit testing 2".

You should (as a minimum):

- Explain how you have set up your test cases with JUnit and how your test cases are executed
- Execute your test cases
- List some of the different JUnit asserts that exist and which you have used
- Account for the need for JUnit test parameters and use of data-driven testing
- Demonstrate how you have tested for exceptions

Demonstrate your solution to Midterm assignment Project 1 (Testing Real Life Code)

You should (as a minimum):

- Explain the purpose of the Test (what the previous test exposed, and what your test expose)
- Explain about Parameterized Tests in JUnit and how you have used it in this exercise
- *Explain the topic Data Driven Testing, and why it often makes a lot of sense to read test-data from a file*
- Your answers to the question; whether what you implemented was a Unit Test or a JUnit Test, the problems you might have discovered with the test and, your suggestions for ways this could have been fixed.
- The steps you took to include Hamcrest matchers in the project, and the difference they made for the test

---

Demonstrate your solution to Midterm assignment Project 2 (Unit Testing, Testable Code, Mocking and Code Coverage)

You should (as a minimum):

- Explain the necessary steps you did to make the code testable, and some of the patterns involved in this step
- Execute your test cases
- Explain basically about JUnit, Hamcrest, Mockito and Jacoco, and what problems they solve for testers
- Demonstrate how you used Mockito to mock away external Dependencies
- Demonstrate how/where you did state-based testing and how/where you did behavior-based testing
- Explain about Coverage Criterias, using the results presented by running Jacoco (or a similar tool) against you final test code.
- Explain/demonstrate what was required to make this project use, JUnit (Hamcrest), Mockito and Jacoco

---

System Testing; Automated System Testing

- Define the term System Testing and it purpose, both in terms of the (old-fashioned) V-model and in an agile context
- Explain the "variants of tests" that are often associated with System Testing and the difference between functional and non-functional testing
- Who is (often) carrying out the System Test (would your answer be the same for an organization following the (old-fashioned) V-model versus a modern agile organization?)
- What is the difference between System and Acceptance testing
- Discuss Pros and Cons with manual versus automated tests
- Discuss some of the problems with automated GUI tests and what makes such test "vulnerable"
- Explain about the Selenium, its purpose, and how it "fits into" the testing world
- Demonstrate, using practical examples, how you have tested Web-UI's using Selenium
- Explain what kinds of tests can be set up with Apache JMeter or similar tool.
- Explain the purpose of Behavior-Driven-Development (BDD) as a style of writing test cases and how it relates to Acceptance-Test-Driven-Development (ATTD).
- Explain what it meant by “executable specification” and how Gherkin and Cucumber can be used in this context.

Demonstrate your solution to the study point exercise "Automated System Testing with Selenium2".

You should (as a minimum):

- Discuss Pros and Cons with manual versus automated tests
- Explain about the Test Pyramid and whether this exercise supported the ideas in the "pyramid"
- Discuss some of the problems with automated GUI tests and what makes such tests "vulnerable"
- Demonstrate details in how to create a Selenium Test using the code for the exercise
- Explain shortly about the DOM, and how you have read/manipulated DOM-elements in your test
- Explain how (and why it was necessary) you have solved "waiting" problems in your test

---

Demonstrate your solution to the study point exercise "Technical proof of concept ".

You should (as a minimum) on a project of your own choice:

- Demonstrate profiling by Apache JMeter or similar tool
- Explain pairwise testing method and show example of usages
- Demonstrate how you have used pairwise testing technique to create a reduced number of test cases based on a combination of criteria
- Demonstrate how you have set up a Cucumber project with Feature, Scenarios, and Step Definitions

Integration testing

- Explain the focus and purpose of integration testing
- Explain when integration (testing) is done and under which conditions
- List different types of integration tests and elaborate on the differences between them
- Explain the most important implications involved in performing integration tests that involve databases