Software Testing, Verification and Validation

November 24, 2023
Week #11 — Lecture #9

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Last week, we introduced data flow testing as part of our set of white-box techniques and closed that chapter. This week we will (re-)visit Integration Testing and System Testing, Acceptance Testing, and Regression Testing.

White-box Testing

(White-box) Testing: Rule of thumb

- Do not start designing white-box test cases!

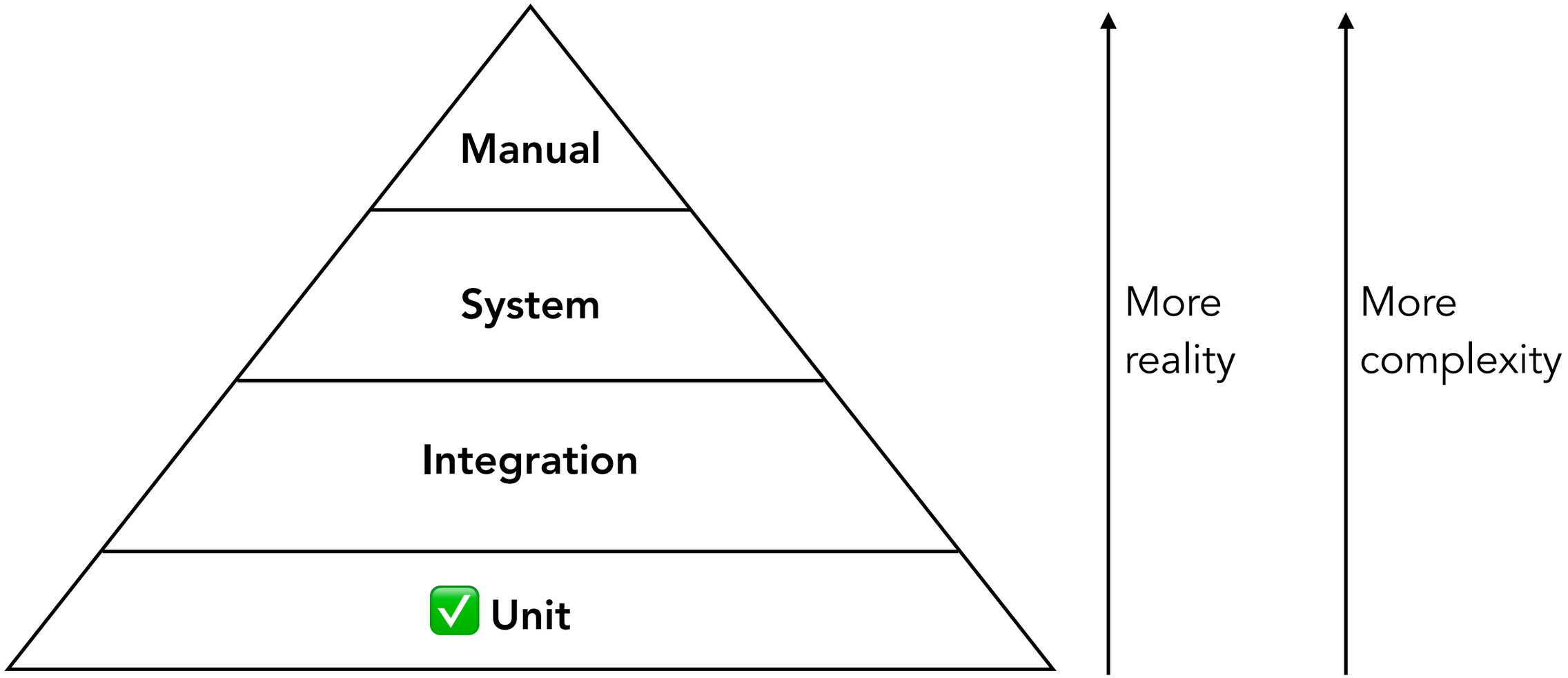
- Start with black-box test cases: equivalence partitioning, boundary value analysis, …

- Check white-box coverage (line, decision, condition, path, …)

- Use a testing coverage tool.

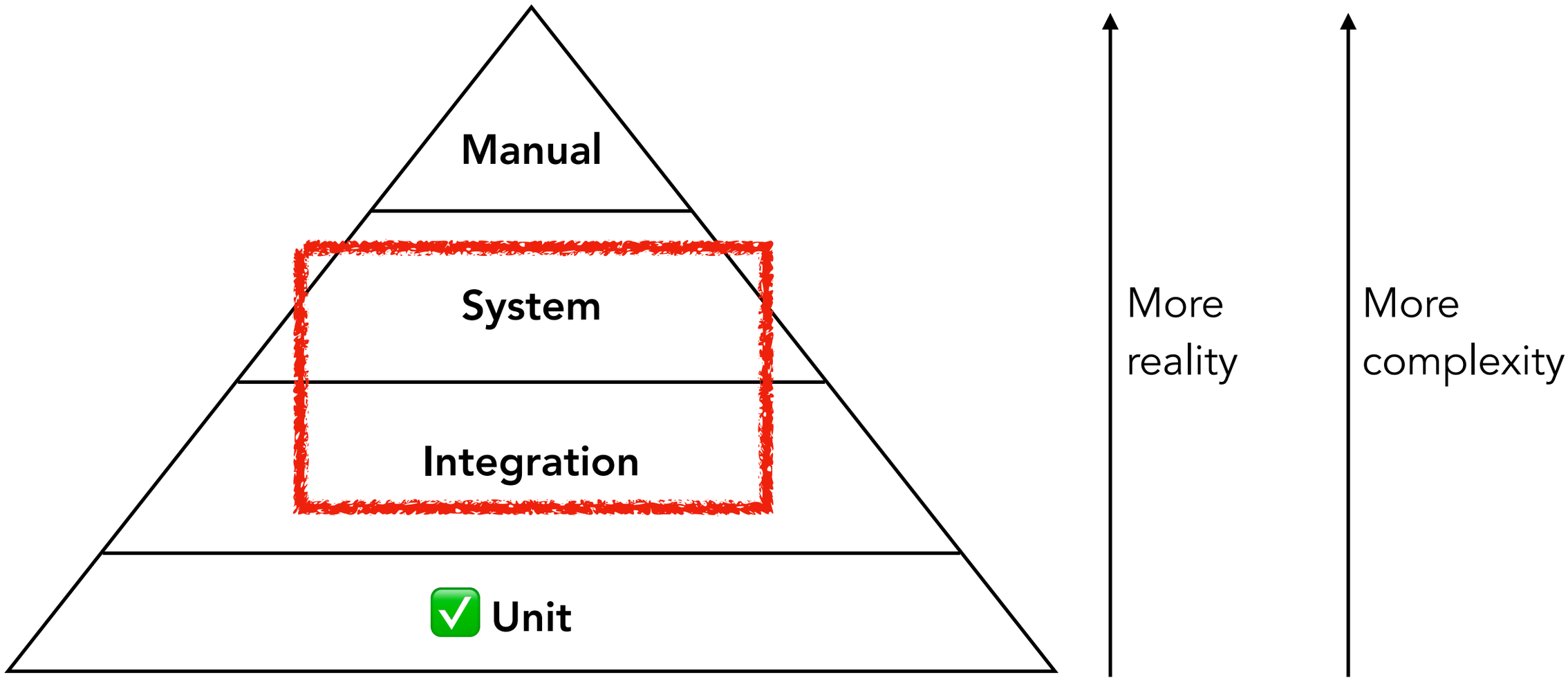
- Design additional white-box test cases for not covered code.

Testing Pyramid



by Martin Fowler, https://martinfowler.com/bliki/TestPyramid.html

Testing Pyramid



by Martin Fowler, https://martinfowler.com/bliki/TestPyramid.html

Integration Testing

Integration Testing

- **Testing groups of components integrated to create a sub-system.** Components should be tested previously.

- Usually the responsibility of an independent testing team (except sometimes in small projects).

- Integration testing should be **black-box** testing with tests derived from the technical specification.

- A principal goal is to **detect defects that occur on the interfaces of units**.

- Main difficulty is localising errors.

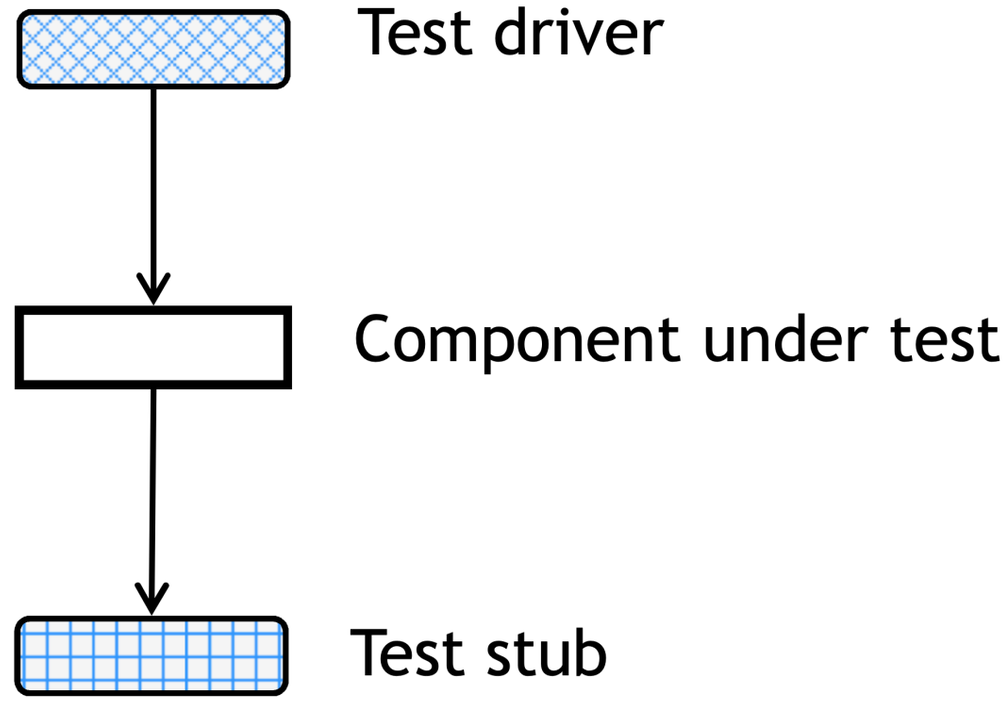
- Incremental integration testing (as opposed to big-bang integration testing) reduces this difficulty.

Integration Testing: Terminology

- **Test harness**: auxiliary code developed to support testing.

- **Test drivers**: call the target code, simulating calling units or a user.

- **Test stubs**: Simulate modules/ units/systems called by the target code. Mock objects can be used for this purpose. (We strongly recommend you to read *Practical Unit Testing with JUnit and Mockito* by Tomek Kaczanowsk.)



Integration Testing

The objective is to take unit tested components and build a program structure that has been dictated by design. Integration testing is testing in which a group of components is combined to produce output.

Integration testing is of four types:

- **Top-down:** Start with high-level system and integrate from the top-down replacing individual components by stubs (or mocks) where appropriate.

- **Bottom-up:** Integrate individual components in levels until the complete system is created.

- **Sandwich:** is the combination of bottom-up approach and top-down approach, so it uses the advantage of both bottom up approach and top down approach. Initially it uses the stubs and drivers where stubs simulate the behavior of missing components.

- **Big-Bang**

Integration Testing: Big-Bang

It is the simplest integration testing approach, where multiple modules are combining and verifying the functionality after the completion of individual module testing. In simple words, **multiple modules of the system are simply put together and tested**. This approach is practicable only for very small systems. If once an error is found during the integration testing, it is very difficult to localize the error as the error may potentially belong to any of the modules being integrated. So, debugging errors reported during big bang integration testing are very expensive to fix.

Advantages:

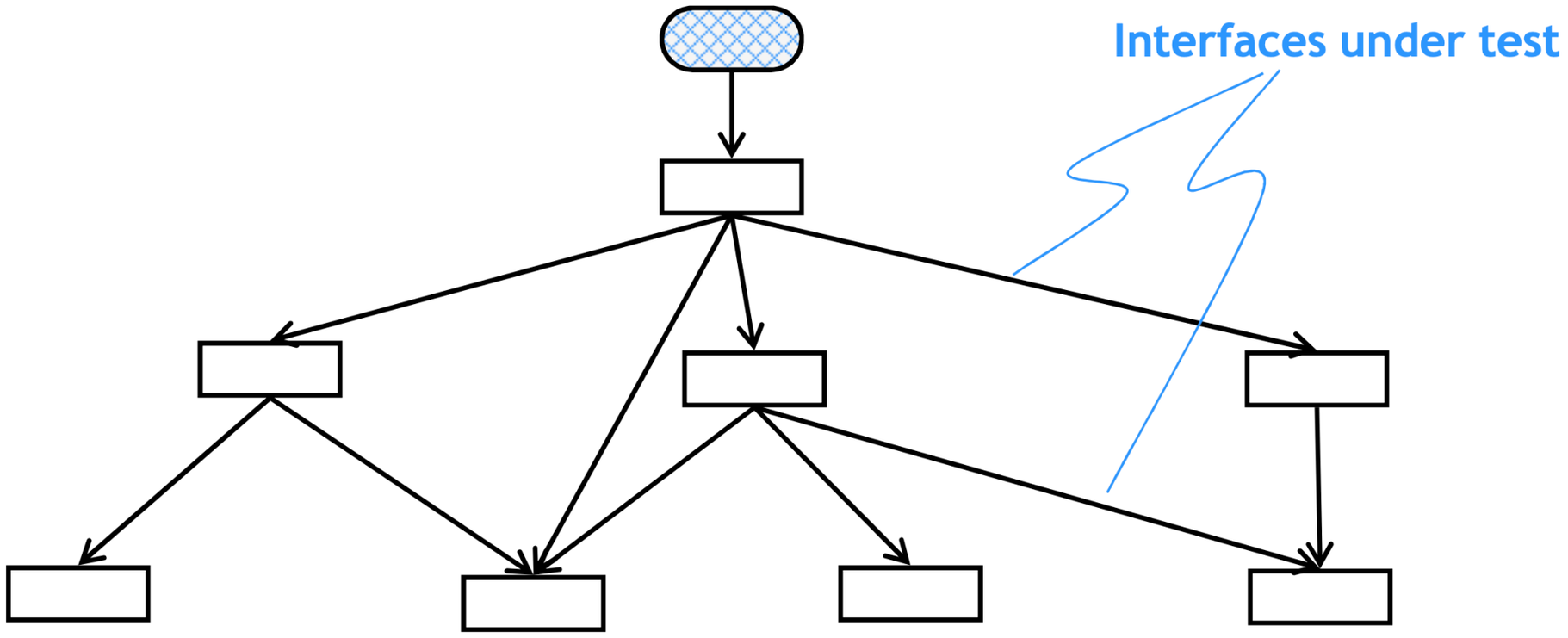
- It is convenient for small systems.

Disadvantages:

- There will be quite a lot of delay because you would have to wait for all the modules to be integrated.

- High risk critical modules are not isolated and tested on priority since all modules are tested at once.

Integration Testing: Big-Bang



Integration Testing: Top-down

Top-down integration testing technique used in order to simulate the behavior of the lower-level modules that are not yet integrated. In this integration testing, testing takes place from top to bottom. First high-level modules are tested and then low-level modules, and finally integrating the low-level modules to a high level to ensure the system is working as intended.

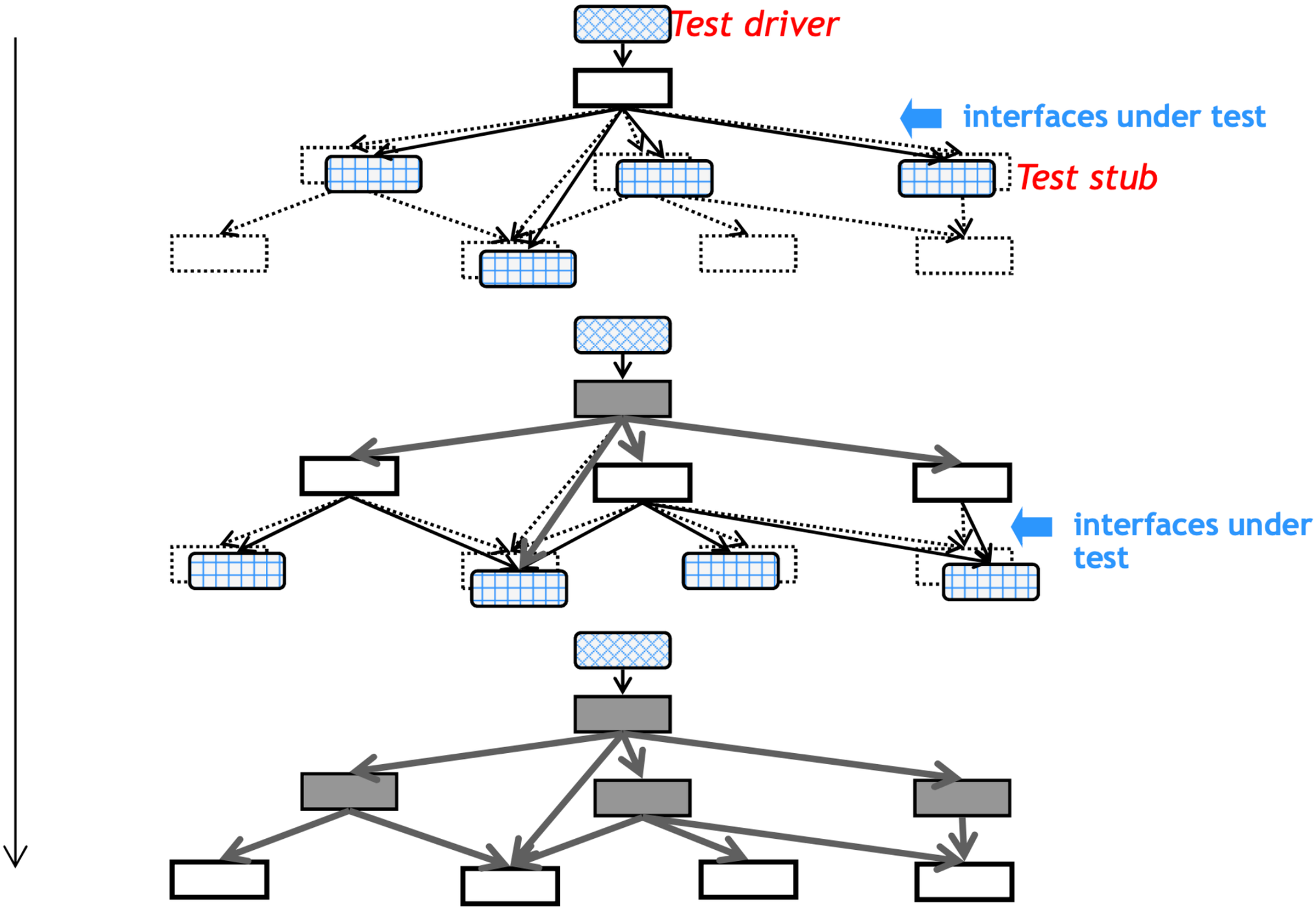
Advantages:

- Separately debugged module.
- Few or no drivers needed.
- It is more stable and accurate at the aggregate level.

Disadvantages:

- Needs many Stubs.
- Modules at lower level are tested inadequately.

Integration Testing: Top-down



Integration Testing: Bottom-up

In bottom-up testing, each module at lower levels is tested with higher modules until all modules are tested. The **primary purpose** of this integration testing is, each subsystem **is to test the interfaces among various modules making up the subsystem**. This integration testing uses test drivers to drive and pass appropriate data to the lower level modules.

Advantages:

- In bottom-up testing, no stubs are required.

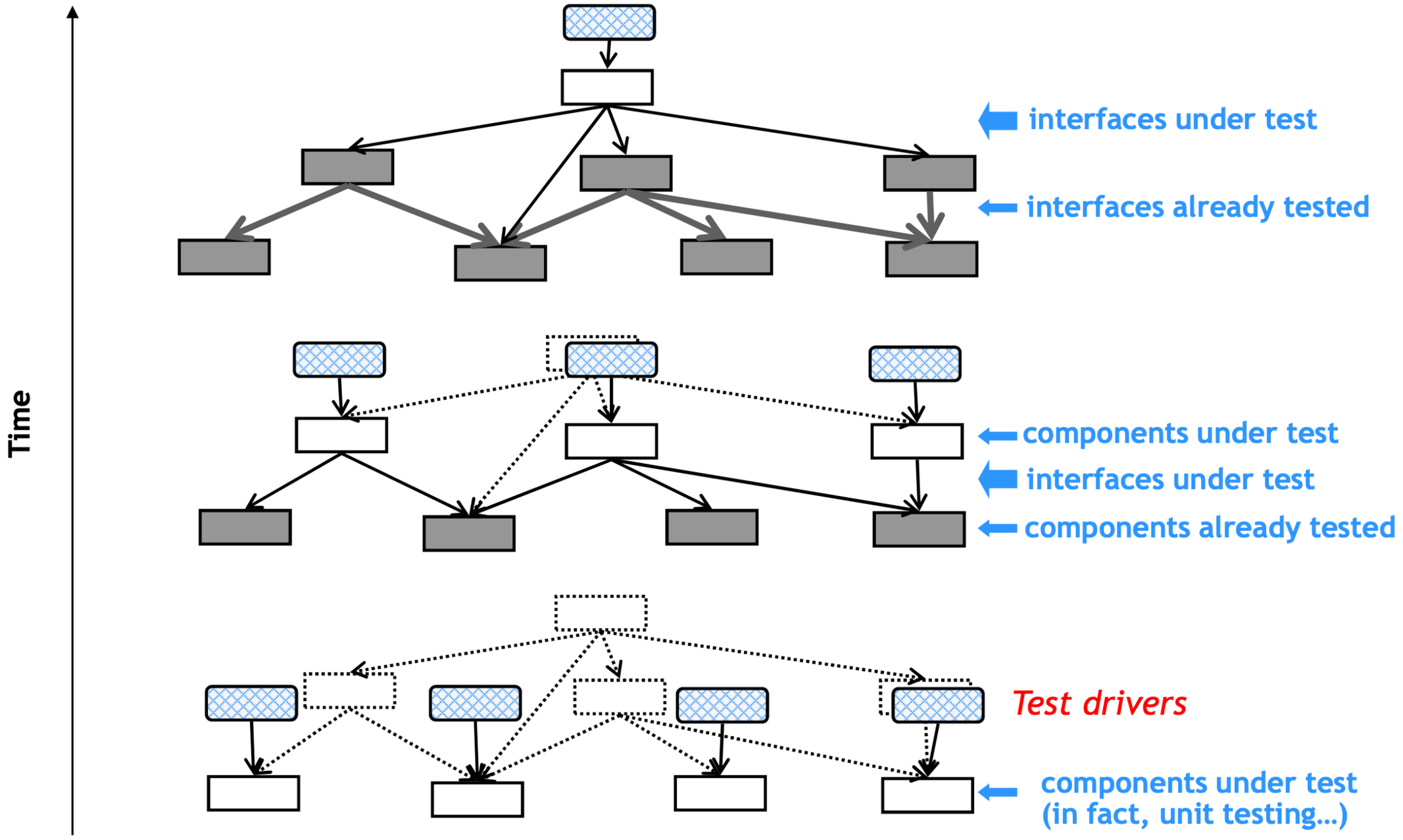
- A principle advantage of this integration testing is that several disjoint subsystems can be tested simultaneously.

Disadvantages:

- Driver modules must be produced.

- In this testing, the complexity that occurs when the system is made up of a large number of small subsystem.

Integration Testing: Bottom-up



Bottom-up vs Top-down

- Architectural validation: Top-down integration testing is better at discovering errors in the system architecture.

- System demonstration: Top-down integration testing allows a limited demonstration at an early stage in the development.

- Test implementation: Top-down integration requires the development of complex stubs to drive significant data upward while bottom-up integration requires drivers.

System Testing

System Testing

- Testing the system as a whole, usually, by an independent testing team.

- Often requires many resources: laboratory equipment, long test times, etc.

- Usually based on a requirements document, specifying both functional and non-functional (quality) requirements.

- Preparation should begin at the requirements phase with the development of a master test plan and requirements-based tests (black-box tests).

- The goal is to ensure that the system performs according to its requirements, by evaluating both functional behavior and quality requirements such as reliability, usability, performance and security.

- Especially useful for detecting external hardware and software interface defects, for example, those causing race conditions, deadlocks, problems with interrupts and exception handling, and ineffective memory usage.

- Tests implemented on the parts and subsystems may be reused/repeated, and additional tests for the system as a whole may be designed.

System Testing: Performance Testing

Performance Testing is a type of software testing that ensures software applications to perform properly under their expected workload. It is a testing technique carried out to determine system performance in terms of sensitivity, reactivity and stability under a particular workload.

Goals:

- Check whether the software meets the performance requirements.
- See whether there are any hardware or software factors that impact on the system's performance.
- Provide valuable information to tune the system.
- Predict the system's future performance levels.

Results of performance tests should be quantified, and the corresponding environmental conditions should be recorded. Resources usually needed:

- A source of transactions to drive the experiments, typically a load generator.
- An experimental test bed that includes hardware and software the system under test interacts with
- Instrumentation of probes that help to collect the performance data (event logging, counting, sampling, memory allocation counters, etc.).
- A set of tools to collect, store, process and interpret data from probes.

System Testing: Stress Testing

Stress Testing is a software testing technique that **determines the robustness of software by testing beyond the limits of normal operation**. Stress testing is particularly important for critical software but is used for all types of software. Stress testing emphasizes on robustness, availability and error handling under a heavy load rather than on what is correct behavior under normal situations. Stress testing is defined as a type of software testing that verifies the stability and reliability of the system. This test particularly determines the system on its robustness and error handling under extremely heavy load conditions. It even tests beyond the normal operating point and analyses how the system works under the extreme conditions. Stress testing is performed to ensure that the system would not crash under crunch situations.

Characteristics of Stress Testing:

- Stress testing analyzes the behavior of the system after a failure.
- Stress testing makes sure that the system recovers after failure.
- It checks whether the system works under the abnormal conditions.
- It ensures to display appropriate error message when the system is under stress.
- It verifies that unexpected failures do not cause security issues.
- It verifies whether the system has saved the data before crashing or not.

System Testing: Load Testing

Load Testing determines the performance of a system, software product or software application under real life based load conditions. **Basically load testing determines the behavior of the application when multiple users use it at the same time.** It is the response of the system measured under varying load conditions. The load testing is carried out for normal and extreme load conditions.

Objectives of Load Testing is to identify performance congestion before the software product is launched in market.

- To maximize the operating capacity of a software application.
- To determine whether the latest infrastructure is capable to run the software application or not.
- To determine the sustainability of application with respect to extreme user load.
- **To find out the total count of users that can access the application at the same time.**
- **To determine scalability of the application.**
- To allow more users to access the application.

System Testing: Load Testing vs Stress Testing

Load Testing	Stress Testing
Load Testing is performed to test the performance of the system or software application under extreme load.	Stress Testing is performed to test the robustness of the system or software application under extreme load.
In load testing load limit is the threshold of a break.	**In stress testing load limit is above the threshold of a break.**
In load testing, the performance of the software is tested under multiple number of users.	In stress testing, the performance is tested under varying data amounts.
Huge number of users.	Too much users and too much data.
Load testing is performed to find out the upper limit of the system or application.	Stress testing is performed to find the behavior of the system under pressure.
The factor tested during load testing is *performance*.	**The factor tested during stress testing is *robustness* and *stability*.**
Load testing determines the operating capacity of a system or application.	Stress testing ensures the system security.

System Testing: Configuration Testing

Configuration Testing is the type of software testing which verifies the performance of the system under development against various combinations of software and hardware to find out the best configuration under which the system can work without any flaws or issues while matching its functional requirements.

Goals:

- To determine whether the software application fulfils the configurability requirements.
- To identify the defects that were not efficiently found during different testing processes.
- To determine an optimal configuration of the application under test.
- To analyze the performance of software application by changing the hardware and software resources.
- To analyze the system efficiency based on the prioritization.
- To verify the degree of ease to how the *bugs* are reproducible irrespective of the configuration changes.

System Testing: Usability / Accessibility Testing

Accessibility Testing is the process of testing the degree of ease of use of a software application for individuals with certain disabilities. It is performed to ensure to that any new component can easily be accessible by physically disabled individuals despite any respective handicaps.

Accessibility testing is part of the system testing process and is somehow similar to usability testing. In the accessibility testing process, the tester uses the system or component as it would be used by individuals with disabilities. The individuals can have the disabilities like visual disability, hearing disability, learning disability or non-functional organs.

Accessibility testing is a subset of usability testing where in the users under consideration are specifically the people with disabilities. This testing focuses to verify both usability and accessibility.

Acceptance Testing

Acceptance Testing

Acceptance Testing is a formal testing according to user needs, requirements and business processes **conducted to determine whether a system satisfies the acceptance criteria** or not and to enable the users, customers or other authorized entities to determine whether to accept the system or not. Acceptance Testing is the last phase of software testing performed after System Testing and before making the system available for actual use.

Acceptance Testing

Alpha Testing is used to test the product in the development testing environment by a specialized testers team usually called alpha testers.

Beta Testing is used to assess the product by exposing it to the real end-users, usually called beta testers in their environment. Feedback is collected from the users and the defects are fixed. Also, this helps in enhancing the product to give a rich user experience.

Acceptance Testing: Alpha vs Beta

Alpha Testing	Beta Testing
Alpha testing involves both the white box and black box testing.	**Beta testing commonly uses black box testing.**
Alpha testing is performed by testers who are usually internal employees of the organization.	**Beta testing is performed by clients who are not part of the organization.**
Alpha testing is performed at developer's site.	**Beta testing is performed at end-user of the product.**
Alpha testing ensures the quality of the product before forwarding to beta testing.	Beta testing also concentrates on the quality of the product but collects users input on the product and ensures that the product is ready for real time users.
Alpha testing requires a testing environment or a lab.	Beta testing does not require a testing environment or lab.
Developers can immediately address the critical issues or fixes in alpha testing.	Most of the issues or feedback collected from beta testing will be implemented in future versions of the product.

Regression Testing

Regression Testing

- Regression testing is not a level of testing, but it is the *retesting* of software that occurs when changes are made to ensure that the new version of the software has retained the capabilities of the old version and that no new defects have been introduced due to the changes.

- Regression tests are especially important when multiple software releases are developed.

- Sometimes the execution of all tests is not feasible so there is the need to select a subset of those tests in order to reduce the time for regression testing. Some techniques include: selection, minimization, and prioritization of such test cases.

Regression Testing

When should we do regression testing?

- When a new functionality is added to the system and the code has been modified to absorb and integrate that functionality with the existing code.

- When some defect has been identified in the software and the code is debugged to fix it.

- When the code is modified to optimize its working.

Advantages:

- It ensures that no new *bugs* have been introduced after adding new functionalities to the system.
- As most of the test cases used in Regression Testing are selected from the existing test suite and we already know their expected outputs. Hence, it can be easily automated by the automated tools.
- It helps to maintain the quality of the source code.

Disadvantages:

- It can be time and resource consuming if automated tools are not used.
- It is required even after very small changes in the code.

Regression Testing vs Retesting

Regression Testing is a type of software testing, which is used to verify that modifications in the software or the environment have not caused any unintended adverse side effect.

Retesting is done to make sure that a *bug* is fixed and/or a failed functionality is working fine or not.

References

- I. Burnstein. Practical software testing a process-oriented approach. (Chapter 6)
- P. Ammann, J. Offutt. Introduction to Software Testing. (Chapter 13)
- P. Jorgensen. Software Testing A Craftsman's Approach. (Chapters 13 and 14)
- M. Pezze, M. Young. Software Testing and Analysis Process, Principles and Techniques. (Chapters 21 and 22)
- S. Yoo, Mark Harman; Regression testing minimization, selection and prioritization: a survey, 2012. https://doi.org/10.1002/stv.430
- Gordon Fraser and José Miguel Rojas; Software Testing, 2019. ISBN 978-3-030-00262-6.