SOFTWARE TESTING

CO3015 / CO5252

CH6. TEST-DRIVEN DEVELOPMENT PROCESS

Content

- Test-Driven Development process
- Functional Testing
- System Testing
- Regression testing
- Acceptance testing

Test-Driven Development process

https://www.guru99.com/test-driven-development.html

- Test Driven Development (TDD) is software development approach in which test cases are developed to specify and validate what the code will do. In simple terms, test cases for each functionality are created and tested first and if the test fails then the new code is written in order to pass the test and making code simple and bug-free.

- Test-Driven Development starts with designing and developing tests for every small functionality of an application. TDD instructs developers to write new code only if an automated test has failed. This avoids duplication of code. The full form of TDD is Test-driven development.

Test-Driven Development process

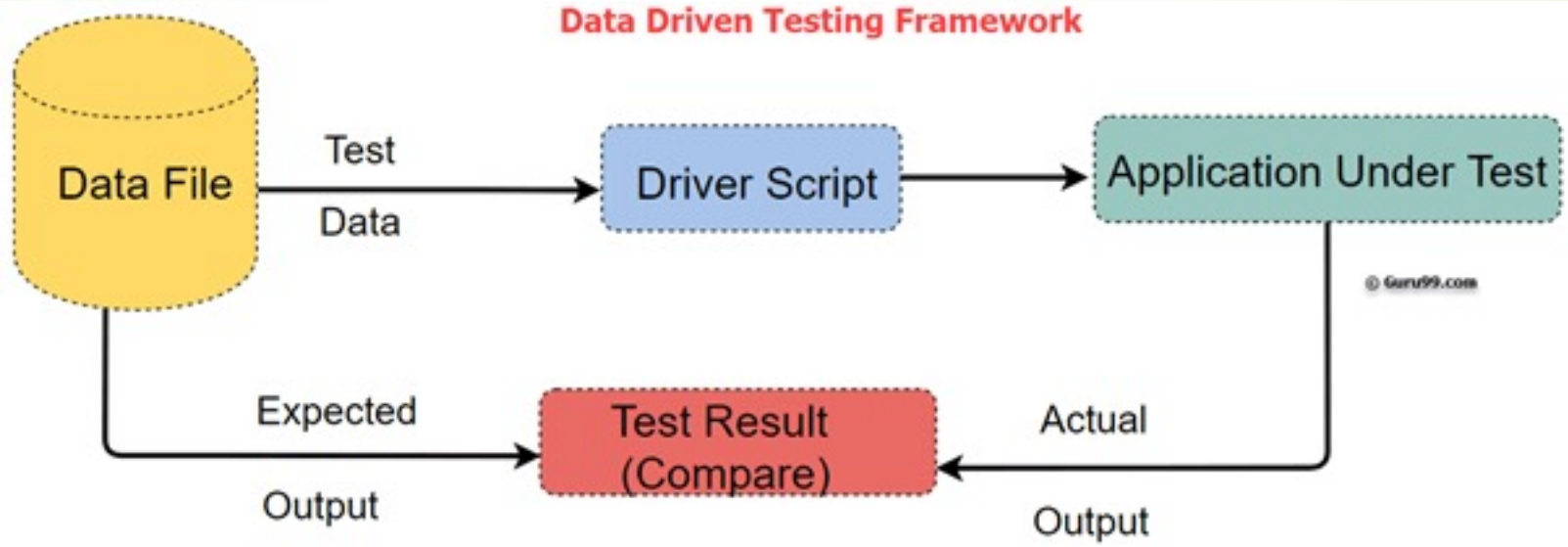
Chapter 19

Test-Driven Development

Data Driven Testing

https://www.guru99.com/data-driven-testing.html

- Data Driven Testing is a software testing method in **which test data is stored in table or spreadsheet format**. Data driven testing allows testers to input a single test script that can execute tests for all test data from a table and expect the test output in the same table. It is also called table-driven testing or parameterized testing.



Data Driven Testing

https://www.guru99.com/data-driven-testing.html

- Advantages of Data-Driven testing
  - Allows to test application with multiple sets of data values during Regression testing
  - Test data and verification data can be organized in just one file, and it is separate from the test case logic.
  - Base on the tool, it is possible to have the test scripts in a single repository. This makes the texts easy to understand, maintain and manage.
  - Actions and Functions can be reused in different tests.
  - Some tools generate test data automatically. This is useful when large volumes of random test data are necessary, which helps to save the time.
  - Data-driven testing can perform any phase of the development. A data-driven test cares are generally merged in the single process. However, it can be used in multiple test cases.
  - Allows developers and testers to have clear separation for the logic of their test cases/scripts from the test data.
  - The same test cases can be executed several times which helps to reduce test case and scripts.
  - Any changes in the test script do not effect the test data

Data Driven Testing

https://www.guru99.com/data-driven-testing.html

- Disadvantages of Data Driven testing:
  - Quality of the test is depended on the automation skills of the Implementing team
  - Data validation is a time-consuming task when testing large amount of data.
  - Maintenance is a big issue as large amount of coding needed for Data-Driven testing.
  - High-level technical skills are required. A tester may have to learn an entirely new scripting language.
  - There will be more documentation. Mostly related to scripts management tests infrastructure and testing results.
  - A text editor like Notepad is required to create and maintain data files.

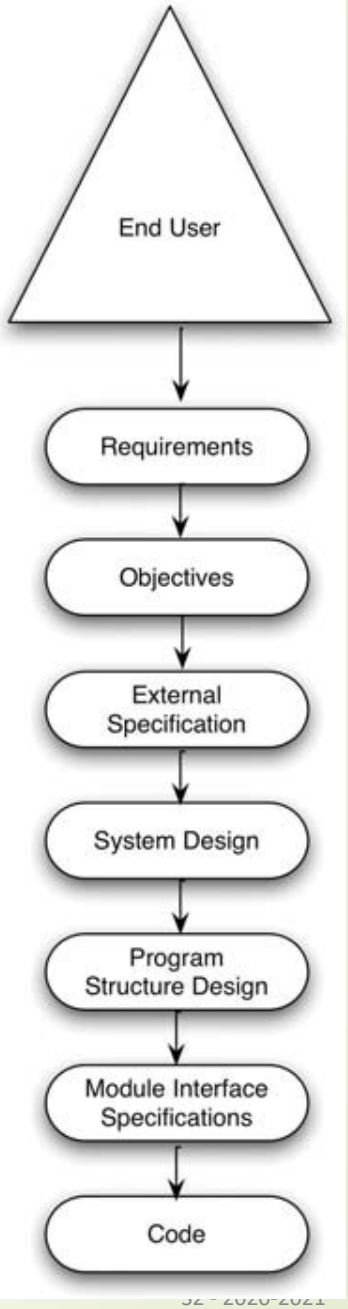
HIGHER-ORDER TESTING

Higher-Order Testing

- Function Testing
- System Testing
- Acceptance Testing
- Installation Testing
- Regression Testing

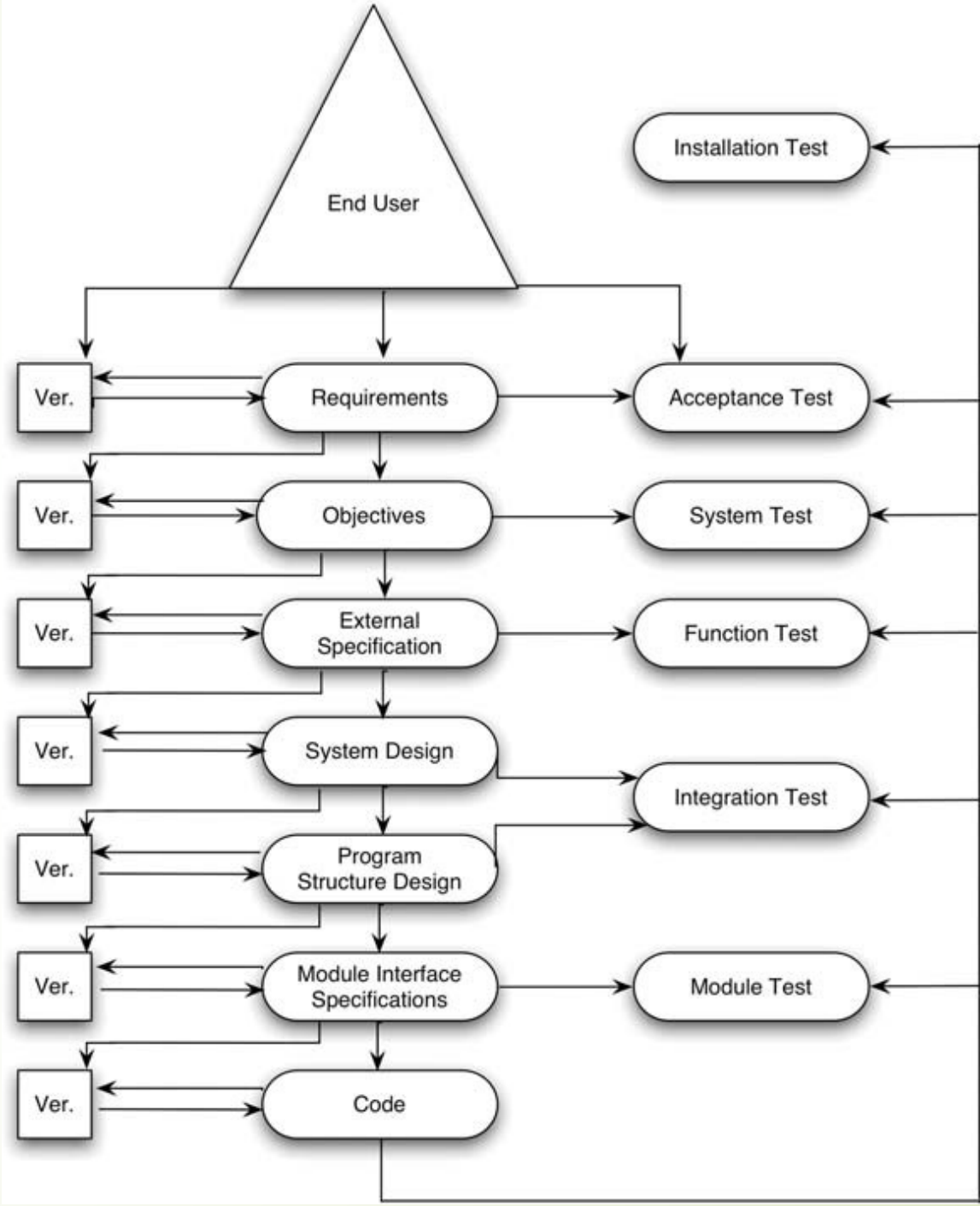
Higher-Order Testing

- Software Development Process ?
  - Requirements specify why the program is needed.
  - Objectives specify what the program should do and how well the program should do it.
  - External specifications define the exact representation of the program to users.
  - Documentation associated with the subsequent processes specifies, in increasing levels of detail, how the program is constructed.



Higher-Order Testing

- The purpose of a module test is to find discrepancies between the program's modules and their interface specifications.
- The purpose of a function test is to show that a program does not match its external specifications.
- The purpose of a system test is to show that the product is inconsistent with its original objectives.



Functional Testing

- A process of attempting to find discrepancies between the program and the external specification.
  - An external specification is a precise description of the program's behavior from the point of view of the end user.
- Function testing is normally a black-box activity
  - Equivalence-partitioning
  - Boundary-value analysis
  - Cause-effect graphing
  - Use case
  - State Diagram
  - Etc…

Functional Testing

- Some approaches for function testing:
  - User Navigation Testing
  - Transaction Screen Testing
  - Transaction Flow Testing
  - Report Screen Testing
  - Report Flow Testing
  - Database Create/Retrieve/Update/Delete Testing

Gerald D. Everett, Raymond McLeod Jr. *Software Testing: Testing Across the Entire Software Development Life Cycle*. Wiley-IEEE CS., 1st ed. (July 16, 2007), ISBN-13: 978-0471793717

https://books.google.com.vn/books?id=z8UdPmvkBHEC&pg=PA103

Functional Testing – User Navigation Testing

- Navigation screens are those
  - log on and log off screens that control access to the software
  - all menus that provide alternate activity paths through the software,
  - all screen-to-screen linkage that represents a continuum of some business activity.
- User navigation testing focuses on
  - the user's ability to log on to the software with appropriate authority,
  - traverse the application to the desired transaction screens,
  - traverse the transaction screens correctly, and log off the software

Functional Testing – Transaction Screen Testing

- The transaction screen normally has
  - Input data fields, lists of choices, options, and action buttons (Add, Change, Delete, Submit, Cancel, OK, and so forth).
  - Some kind of results may be displayed on the transaction screen after appropriate action buttons are pressed.
- The tester's job
  - Design tests that validate the operation of every field, list, option, and action button on each transaction screen against the business requirements, the user guide, and the administrator guide
  - If results are also displayed on the transaction screen, then the black box inputs versus the expected result technique is used to validate the displayed results.

Functional Testing – Transaction Flow Testing

- Takes the transaction screens that have been validated by testing and determines if their combined results of correct navigation completes the intended business activity in some specified way.
- Example: validate customer profile updates as
  - transaction screen 1 for customer name, address, and contact person
  - transaction screen 2 for customer line of credit approval
  - transaction screen 3 for customer payment terms and discounts
  - transaction screen 4 for profile summary and update action
  - transaction screen 5 for viewing updated customer profile
- The result of the sequence of file screens being completed is expected to be a master file or database file update with all the information collected on these transaction screens.

Functional Testing – Report Screen Testing

- Similar to transaction screen testing.
- You are attempting to retrieve and display data from the system using the report screens instead of entering data using the transaction screens.
- The difficulty in report screen testing usually lies in the variety of ways an end user can specify which data are retrieved (search criteria) and how these data are displayed (sorting and formatting options).
- The tester's job is to pay particular attention to the data retrieved and displayed because the wrong data may have been selected or, worse yet, not all data requested were displayed.

Functional Testing – Report Flow Testing

- Different from report screen testing when the report results are provided in other modalities besides on-screen display (for example: hardcopy output, print to file)
- Testers:
  - Does the software send exactly the same results to the printer that it displays on the report screen?
    Having a printer option implies the possibility of the report screen offering a selection of print fonts, another fertile area for testing
  - Validate the report results on all the alternative report modalities supported by the software.

Functional Testing – Database Create/ Retrieve/ Update/ Delete Testing

- Normally done in two steps:
  - Test the database design viability by successfully performing the application data manipulations outside of the application
  - Test the application software's use of the validated database design.
- Requires cooperation and collaboration between the tester and the database administrator

System Testing

https://www.guru99.com/system-testing.html

- SYSTEM TESTING is a level of testing that **validates the complete and fully integrated software product**. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

System Testing

https://www.guru99.com/system-testing.html

- SYSTEM TESTING is a level of testing that **validates the complete and fully integrated software product**. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

System Testing

https://www.guru99.com/system-testing.html

https://www.guru99.com/types-of-software-testing.html

- Typical System testing types:
  - **Usability Testing**- mainly focuses on the user's ease to use the application, flexibility in handling controls and ability of the system to meet its objectives
  - **Load Testing**- is necessary to know that a software solution will perform under real-life loads.
  - **Regression Testing**- involves testing done to make sure none of the changes made over the course of the development process have caused new bugs. It also makes sure no old bugs appear from the addition of new software modules over time.
  - **Recovery testing** - is done to demonstrate a software solution is reliable, trustworthy and can successfully recoup from possible crashes.
  - **Migration testing**- is done to ensure that the software can be moved from older system infrastructures to current system infrastructures without any issues.
  - **Functional Testing** - Also known as functional completeness testing, Functional Testing involves trying to think of any possible missing functions. Testers might make a list of additional functionalities that a product could have to improve it during functional testing.
  - **Hardware/Software Testing** - IBM refers to Hardware/Software testing as "HW/SW Testing". This is when the tester focuses his/her attention on the interactions between the hardware and software during system testing.

System Testing

- Facility Testing
- Volume Testing
- Stress Testing
- Usability Testing
- Security Testing
- Performance Testing
- Storage Testing
- Configuration Testing
- Compatibility/ Configuration/ Conversion Testing
- Installability Testing
- Reliability Testing
- Recovery Testing
- Serviceability Testing
- Documentation Testing
- Procedure Testing

[1] Glenford J. Myers, Corey Sandler, and Tom Badgett. 2011. The Art of Software Testing (3rd. ed.). Wiley Publishing.

System Testing

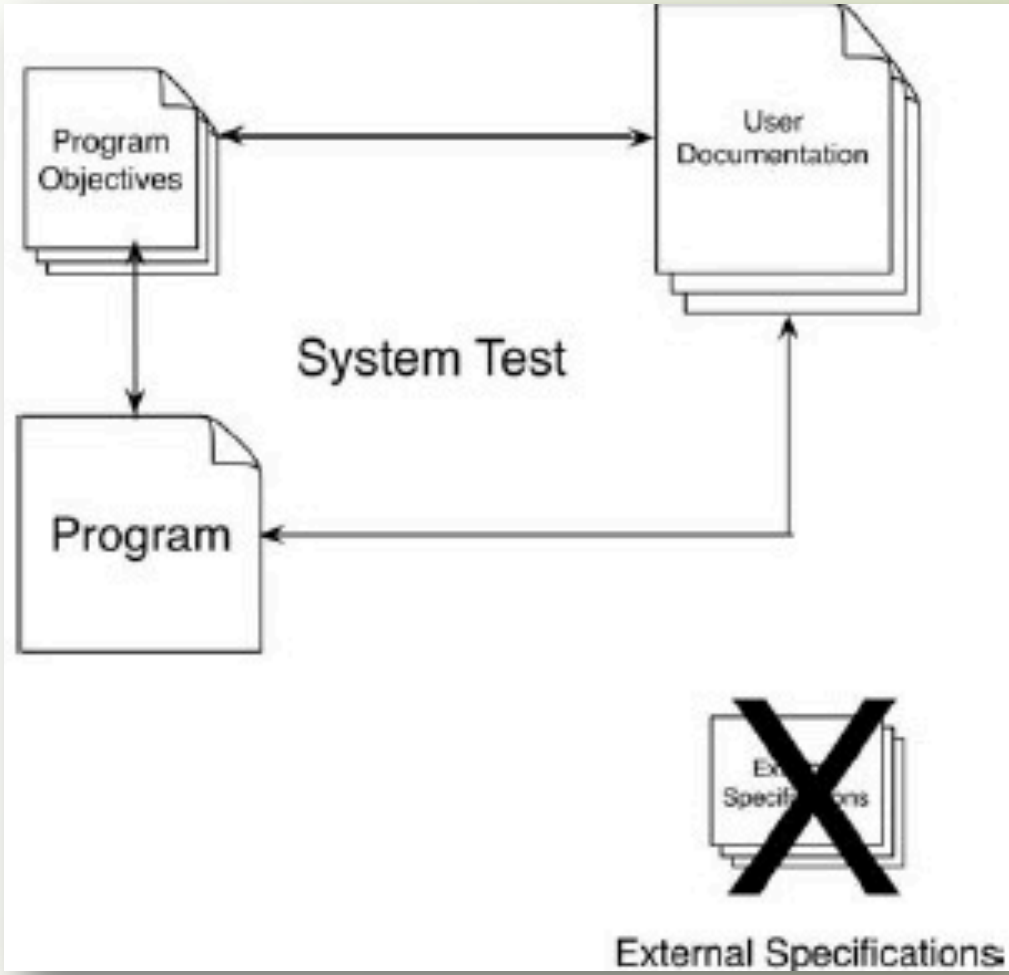
- Facility Testing
- Volume Testing
- Stress Testing
- Usability Testing
- Security Testing
- Performance Testing
- Storage Testing
- Configuration Testing

TABLE 6.1 15 Categories of Test Cases

Category	Description
Facility	Ensure that the functionality in the objectives is implemented.
Volume	Subject the program to abnormally large volumes of data to process.
Stress	Subject the program to abnormally large loads, generally concurrent processing.
Usability	Determine how well the end user can interact with the program.
Security	Try to subvert the program's security measures.
Performance	Determine whether the program meets response and throughput requirements.
Storage	Ensure the program correctly manages its storage needs, both system and physical.
Configuration	Check that the program performs adequately on the recommended configurations.
Compatibility/ Conversion	Determine whether new versions of the program are compatible with previous releases.
Installation	Ensure the installation methods work on all supported platforms.
Reliability	Determine whether the program meets reliability specifications such as uptime and MTBF.
Recovery	Test whether the system's recovery facilities work as designed.
Serviceability/ Maintenance	Determine whether the application correctly provides mechanisms to yield data on events requiring technical support.
Documentation	Validate the accuracy of all user documentation.
Procedure	Determine the accuracy of special procedures required to use or maintain the program.

System Testing

- A particular purpose: to compare the system or program to its original objectives.
  - System testing is not limited to systems. If the product is a program, system testing is the process of attempting to demonstrate how the program, as a whole, does not meet its objectives.
  - System testing, by definition, is impossible if there is no set of written, measurable objectives for the product.



System Testing – Objective examples

- A command will be provided to view, from a terminal, the contents of main-storage locations.
  - Its syntax should be consistent with the syntax of all other system commands.
  - The user should be able to specify a range of locations, via an address range or an address and a count.
  - Sensible defaults should be provided for command operands.
  - Output should be displayed as multiple lines of multiple words (in hexadecimal), with spacing between the words.
  - Each line should contain the address of the first word of that line.
  - The command is a “trivial” command, meaning that under reasonable system loads, it should begin displaying output within two seconds, and there should be no observable delay time between output lines.
  - A programming error in the command processor should, at the worst, cause the command to fail; the system and the user’s session must not be affected.
  - The command processor should have no more than one user-detected error after the system is put into production.

System testing - Facility Testing

- Determine whether each facility mentioned in the objectives was actually implemented.
- Procedures:
  - Scan the objectives sentence by sentence
  - When a sentence specifies a what (for example, “syntax should be consistent . . . ,” “user should be able to specify a range of locations . . .”), determine that the program satisfies the “what.”
- Often can be performed without a computer; a mental comparison of the objectives with the user documentation is sometimes sufficient

System testing - Volume Testing

- Subjects the program to heavy volumes of data.
- The purpose of volume testing is to show that the program cannot handle the volume of data specified in its objectives.
- Examples:
  - A compiler would be fed an absurdly large source program to compile.
  - A linkage editor might be fed a program containing thousands of modules
- Require significant resources, in terms of machine and people time, you can’t go overboard.

System testing - Stress Testing

- Subjects the program to heavy loads or stresses.
- A heavy stress is a peak volume of data, or activity, encountered over a short span of time
- Analogy:
  - A volume test would determine whether the typist could cope with a draft of a large report;
  - A stress test would determine whether the typist could type at a rate of 50 words per minute.
- Involves an element of time, it is not applicable to many programs
- Examples:
  - If an operating system is supposed to support a maximum of 15 Multi-programmed jobs, the system could be stressed by attempting to run 15 jobs simultaneously.
  - Web-based applications: ensure that your application, and hardware, can handle some volume of concurrent users

System testing - Usability Testing

- An attempt to find human-factor, or usability, problems.
- Considerations
  - Has each user interface been tailored to the intelligence, educational background, and environmental pressures of the end user?
  - Are the outputs of the program meaningful, non-abusive, and devoid of computer gibberish?
  - Are the error diagnostics, such as error messages, straightforward, or does the user need a PhD in computer science to comprehend them?
  - Does the total set of user interfaces exhibit considerable conceptual integrity, an underlying consistency, and uniformity of syntax, conventions, semantics, format, style, and abbreviations?
  - Where accuracy is vital, such as in an online banking system, is sufficient redundancy present in the input?
  - Does the system contain an excessive number of options, or options that are unlikely to be used?
  - Does the system return some type of immediate acknowledgment to all inputs?
  - Is the program easy to use?

System testing - Serviceability Testing

- The program also may have objectives for its serviceability or maintainability characteristics.
- All objectives of this sort must be tested. Such objectives might define the service aids to be provided with the system, including:
  - Storage dump programs or diagnostics
  - The mean time to debug an apparent problem
  - The maintenance procedures
  - The quality of internal logic documentation.

System testing - Security Testing

- The process of attempting to devise test cases that subvert the program's security checks.
- One way to devise such test cases is to study known security problems in similar systems and generate test cases that attempt to demonstrate similar problems in the system you are testing
- Web-based applications often need a higher level of security testing than do most applications. This is especially true of ecommerce sites.

System testing - Performance Testing

- Response times and throughput rates under certain workload and configuration conditions.
- The purpose of a system test is to demonstrate that the program does not meet its objectives, test cases must be designed to show that the program does not satisfy its performance objectives.

System testing - Storage Testing

- Programs occasionally have storage objectives that state, for example, the amount of main and secondary memory the program uses and the size of temporary or spill files.
- You should design test cases to show that these storage objectives have not been met.

System testing - Configuration Testing

- Programs such as operating systems, database management systems, and message-switching programs support a variety of hardware configurations
- Often the number of possible configurations is too large to test each one, but at the least, you should test the program with each type of hardware device and with the minimum and maximum configuration.

System testing - Compatibility/ Configuration/ Conversion Testing

- Most programs that are developed are not completely new; they often are replacements for some deficient system
- Programs often have specific objectives concerning their compatibility with, and conversion procedures from, the existing system
- Examples:
  - Upgrading a database management system.

System testing – Installability Testing

- Some types of software systems have complicated installation procedures.
- A malfunctioning installation program could prevent the user from ever having a successful experience with the main system you are charged with testing

System testing – Reliability Testing

- The goal of all types of testing is the improvement of the program reliability
- If the program's objectives contain specific statements about reliability, specific reliability tests might be devised
- Testing reliability objectives can be difficult.
  - For example, a modern online system such as a corporate wide area network (WAN) or an Internet service provider (ISP) generally has a targeted uptime of 99.97 percent over the life of the system.
  - There is no known way that you could test this objective with a test period of months or even years.

System testing – Recovery Testing

- Programs such as operating systems, database management systems, and teleprocessing programs often have recovery objectives that state how the system is to recover from programming errors, hardware failures, and data errors
- Objective:
  - Show that these recovery functions do not work correctly
  - Show that the system fails to meet the service-level agreement for the mean time to recovery (MTTR).

System testing – Documentation Testing

- The system test also is concerned with the accuracy of the user documentation
- The principle way of accomplishing this is to use the documentation to determine the representation of the prior system test cases.
- The user documentation should be the subject of an inspection (similar to the concept of the code inspection), checking it for accuracy and clarity

System testing – Procedure Testing

- Many programs are parts of larger, not completely automated systems involving procedures people perform
- Any prescribed human procedures, such as procedures for the system operator, database administrator, or end user, should be tested during the system test

Installation Testing

- Its purpose is not to find software errors but to find errors that occur during the installation process.
- Many events occur when installing software systems
  - User must select a variety of options.
  - Files and libraries must be allocated and loaded.
  - Valid hardware configurations must be present.
  - Programs may need network connectivity to connect to other programs.
- Test cases might check to ensure that
  - A compatible set of options has been selected
  - All parts of the system exist
  - All files have been created and have the necessary contents
  - The hardware configuration is appropriate
- Should be developed by the organization that produced the system, delivered as part of the system, and run after the system is installed

Regression testing

- Regression testing means rerunning test cases from existing test suites to build confidence that software changes have no unintended side-effects.
  - Normally started while new code is being written and existing code is being corrected.
  - Also done from version to version.
- The “ideal” process would be to create an extensive test suite and run it after each and every change.
- Regression test selection (RTS) techniques
  - Unpredictable performance
  - Incompatible process assumptions
  - Inappropriate evaluation models
- Can be automated

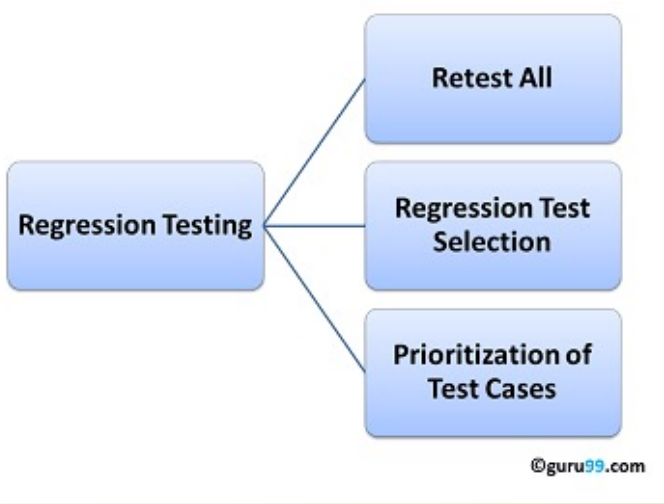
Regression testing

https://www.guru99.com/regression-testing.html

- a type of software testing to confirm that a recent program or code change has not adversely affected existing features.
- Regression Testing is nothing but a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.
- This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that the old code still works once the latest code changes are done.

Regression testing

https://www.guru99



- Retest All
  - This is one of the methods for Regression Testing in which all the tests in the existing test bucket or suite should be re-executed. This is very expensive as it requires huge time and resources.
- Regression Test Selection
  - some selected test cases from test suite are executed to test whether the modified code affects the software application or not.
  - Test cases are categorized into two parts, reusable test cases which can be used in further regression cycles and obsolete test cases which can not be used in succeeding cycles.
- Prioritization of Test Cases
  - depending on business impact, critical & frequently used functionalities.
  - Selection of test cases based on priority will greatly reduce the regression test suite.

Regression testing

https://www.guru99.com/regression-testing.html

- Selecting test cases for regression testing:
  - Test cases which have frequent defects
  - Functionalities which are more visible to the users
  - Test cases which verify core features of the product
  - Test cases of Functionalities which has undergone more and recent changes
  - All Integration Test Cases
  - All Complex Test Cases
  - Boundary value test cases
  - A sample of Successful test cases
  - A sample of Failure test cases

Acceptance testing

- The process of comparing the program to its initial requirements and the current needs of its end users
- Usually performed by the program's customer or end user and normally not considered the responsibility of the development organization.
- In the case of a contracted program, the contracting (user) organization performs the acceptance test by comparing the program's operation to the original contract
- In the case of a program product, such as a computer manufacturer's operating system or compiler, or a software company's database management system, the sensible customer first performs an acceptance test to determine whether the product satisfies its needs.

Acceptance testing

https://www.guru99.com/user-acceptance-testing.html

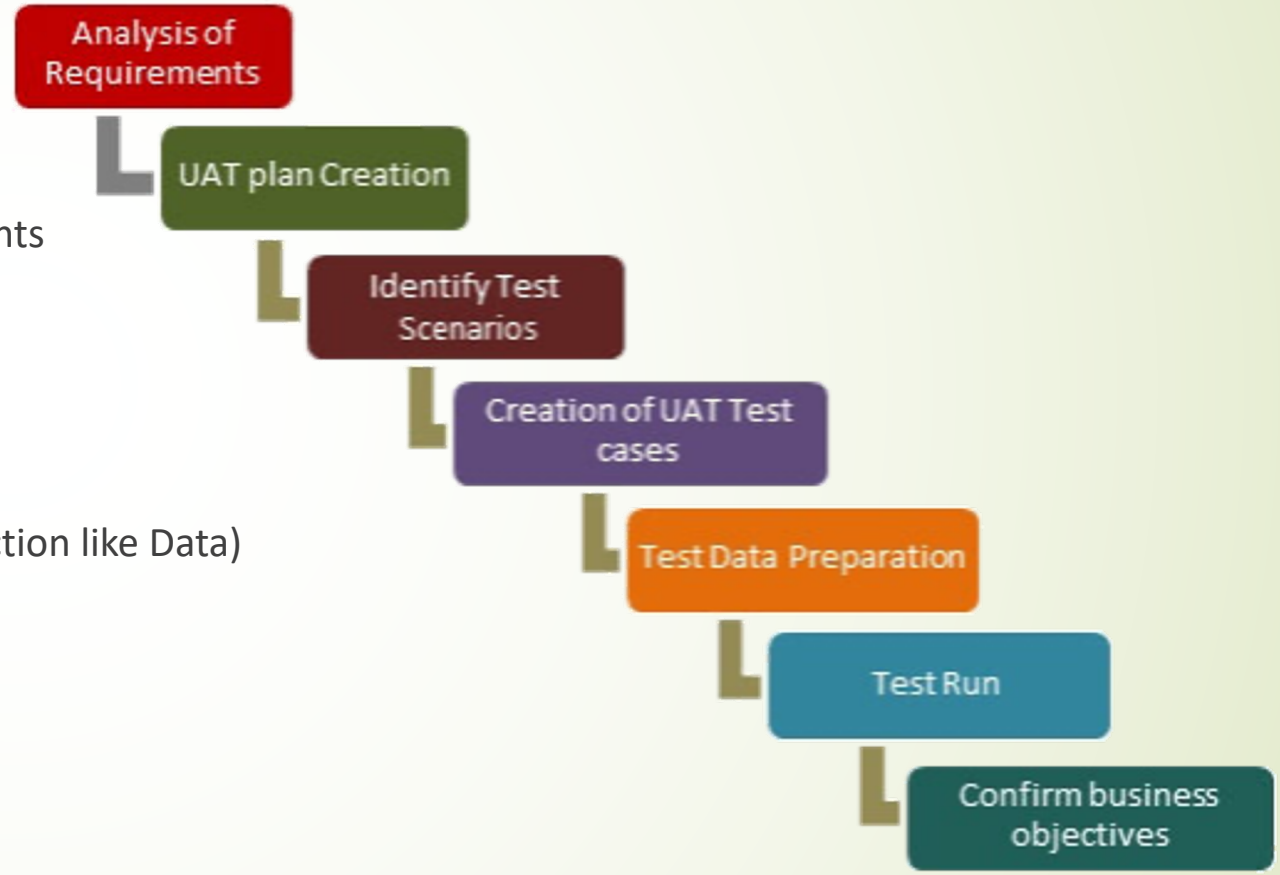
- User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done.

- The main Purpose of UAT is to validate end to end business flow. It does not focus on cosmetic errors, spelling mistakes or system testing. User Acceptance Testing is carried out in a separate testing environment with production-like data setup. It is kind of black box testing where two or more end-users will be involved.

Acceptance testing

https://www.guru99.com/user-acceptance-testing.html

- Process:
  - Analysis of Business Requirements
  - Creation of UAT test plan
  - Identify Test Scenarios
  - Create UAT Test Cases
  - Preparation of Test Data(Production like Data)
  - Run the Test cases
  - Record the Results
  - Confirm business objectives



Summary

- Test-Driven Development process
  - Test-case first, test, then add code, then test, then refactor, ...
- Data-Driven testing ~ automation testing
- Functional Testing ~ Black-box techniques
  - Some suggestion approaches:
    - User Navigation Testing
    - Transaction Screen Testing
    - Transaction Flow Testing
    - Report Screen Testing
    - Report Flow Testing
    - Database Create/Retrieve/Update/Delete Testing

Summary

- System Testing
  - Facility Testing
  - Volume Testing
  - Stress Testing
  - Usability Testing
  - Security Testing
  - Performance Testing
  - Storage Testing
  - Configuration Testing
- Compatibility/ Configuration/ Conversion Testing
- Installability Testing
- Reliability Testing
- Recovery Testing
- Serviceability Testing
- Documentation Testing
- Procedure Testing

Summary

- Regression testing
  - Test for new or updating code
  - Ways:
    - Retest All
    - Regression Test Selection
    - Prioritization of Test Cases
  - Selection of the test cases
    - Test cases which have frequent defects
    - Functionalities which are more visible to the users
    - Test cases which verify core features of the product
    - Test cases of Functionalities which has undergone more and recent changes
    - All Integration Test Cases
    - All Complex Test Cases
    - Boundary value test cases
    - A sample of Successful test cases
    - A sample of Failure test cases

Summary

- Acceptance testing
  - User Acceptance Testing (UAT)
  - Process:
    - Analysis of Business Requirements
    - Creation of UAT test plan
    - Identify Test Scenarios
    - Create UAT Test Cases
    - Preparation of Test Data(Production like Data)
    - Run the Test cases
    - Record the Results
    - Confirm business objectives