STANNS COLLEGE OF ENGINEERING &TECHNOLOGY
SOFTWARE TESTING METHODOLOGIES

Unit-6

Automation and Testing Tools: need for automation, categorization of testing tools, selection of testing tools, Cost incurred, Guidelines for automated testing, overview of some commercial testing tools.
Testing Object Oriented Software: basics, Object oriented testing
Testing Web based Systems: Challenges in testing for web based software, quality aspects, web engineering, testing of web based systems, Testing mobile systems.

Automation and Testing Tools

Need for Automation:

If an organization needs to choose a testing tool, the following benefits of an automation must be considered:

--Reduction of testing Effort
--Reduces the testers' involvement in executing tests
--Facilitates Regression Testing
--Avoids Human mistakes
--Reduces overall cost of the software
--Simulated testing
--Internal Testing
--Test case Design

Categorization of Testing Tools:

Static Testing Tools

Dynamic Testing Tools

Testing Activity Tools

Static Testing Tools:

For Static Testing, there are static program analysers which scan the source program and detect all possible faults and anomalies. These static tools parse the program text, recognize the various sentences, and detects the following:
--Statements are well formed.
--Inferences about the control flow of the program.
-- Compute the set of all possible values for program data.

Static tools perform the following types of static analysis:

--Control Flow Analysis
--Data use Analysis
--Interface Analysis
--Path Analysis

Dynamic Testing Tools:

Automated test tools enable the test team to capture the state of events durinf the execution of a program by preserving a snapshot of the conditions. These tools are sometimes called ***Program Monitors***. These monitors perform the following functions:

--List the number of times a component is called of line of code is executed. This information is used by testers about the statement or path coverage of their test cases.

--Report on whether a decision point has branched in all directions, thereby providing information about branch coverage.

--Report summary statistics providing a high level view of the percentage of statements, paths, and branches that have been covered by the collective set of test cases run. This information is important when test objectives are stated in terms of coverage.

Testing Activity Tools: These tools are based on the testing activities or tasks in a particular phase of the SDLC. Testing activities can be categorized as:
--Tools for Review and Inspections
--Tools for Test Planning
--Tools for Test Design and Development
--Test Execution and Evaluation Tools

Tools for Review and Inspections: Since these tools are for static analysis on many items, some tools are designed to work with specifications but there are far too many tools available that work exclusively with code. In this category, the following types of tools are required:

Complexity Analysis Tools: It is important for testers that complexity is analysed so that testing time and resources can be estimated.
Code Comprehension: These tools help in understanding dependencies, tracing, program logic, viewing graphical representations of the program.

Tools for Test Planning: The types of tools required for test planning are:
-- Templates for test plan documentation.
-- Test schedule and staffing estimates.
-- Complexity analyser.

Tools for Test Design and Development:

Test data generator: It automates the generation of test data based on a user defined format.
Test case generator: It automates the procedure of generating the test cases. But it works with a requirement management tool which is meant to capture requirements information.

Test Execution and Evaluation Tools:
Capture/Playback Tools: These tools record events (keystrokes, mouse activity, display output) at the time of running the system and place the information into a script.

Coverage Analysis Tools: These tools automate the process of throughly testing the software and provide a quanttive measure of the coverage of the system being tested.

Memory Testing Tools: These tools verify that an application is properly using its memory resources.

Test management Tools: Some test management tools such as Rationals TestStudio are integrated with requirement and configuration management and defect tracking tools, in order to simplify the entire testing life cycle.

Network-testing Tools: These tools monitor, meausre, test and dignose performance across an entire network.

Performance testing Tools: These tools helps in measuring the response time and load capabilities of a sytem.

Selection of Testing Tools:

--Match the tool to its appropriate use
--Select the tool to its appropriate SDLC phase
--Select the tool to the skill of the tester
--Select a tool which is affordable
--Determine how many tools are required for testing the system
--Select the tool after having the schedule of testing

Cost incurred in Testing Tools:

Automated Script Development: Auotmated test tools do not create test scripts. Therefore,a significant time is needed to program the tests.

Training is required: Testers require training regarding the tool, otherwise the tools may end up on the shelf or implemented inefficiently.

Configuration Management: It is necessary to track large number of files and test related artifacts.

Learning Curve for the Tools: Tests scripts generated by the tool during recording must be modified manually, requiring tool-scripting knowledge in order to make the script robust, reusable and maintainable.

Testing Tools can be Intrusive: It may be necessary that for automation some tools require that a special code is inserted in the system to work correctly and to be integrated with the testing tools. These types of tools are called as intrusive tools.

Multiple Tools are required: It may be possible that your requirement is not satisfied with just one tool for automation.

Guidelines for Automated Testing:

Consider building a tool instead of buying one if possible: If the requirement is small and sufficient resources allow then go for building the tool instead of buying.

Test the tool on an application prototype: While purchasing the tool it is important to verify that it works properly with the system being developed.

Not all the tests should be automated: Automated testing is an enhancement of manual testing, but it cannot be expected that all test on a project can be automated.

Select the tools according to Organization needs: Focus on the needs of the Organization and know the resources (budget, schedule) before choosing the automation tool.

Use proven test-script development techniques: Automation can be effective if proven techniques are used to produce efficient, maintainable and reusable test-script.

Automate the regression tests whenever feasible: Regression testing consumes a lot of time. If tools are used for this testing, the testing time can be reduced to a greater extent.

Overview of Some Commercial Testing Tools:

Mercury Interactive's WinRunner:

WinRunner, Mercury Interactive's enterprise functional testing tool. It is used to quickly create and run sophisticated automated tests on your application. Winrunner helps you automate the testing process, from test development to execution. You create adaptable and reusable test scripts that challenge the functionality of your application. Prior to a software release, you can run these tests in a single overnight run- enabling you to detect and ensure superior software quality.

Main Features of **Win Runner** are:
--Developed by Mercury Interactive
--Functionality testing tool
--Supports C/s and web technologies such as (VB, VC++, D2K, Java, HTML, Power Builder, Delphe, Cibell (ERP))
--To Support .net, xml, SAP, Peoplesoft, Oracle applications, Multimedia we can use QTP.
--Winrunner run on Windows only.
--Tool developed in C on VC++ environment.

Segue Software's SilkTest:
Robust and portable test automation for web, native, and enterprise software applications.
Silk Test's portability enables users to test applications more effectively with lower complexity and cost in comparison to other functional testing tools on the market. Silk Test's role based testing enables business stakeholders, QA engineers, and developers to contribute to the whole automation testing process, which drives collaboration and increases the effectiveness of software testing.

IBM rational SQA Robot:
Rational Robot is an automated functional, regression testing tool for automating Windows, Java, IE and ERP applications under windows platform. Rational Robot provides test cases for common objects such as menus, lists, bitmaps and specialized test cases for objects specific to the development environment.

Mercury Interactive's LoadRunner:
Mercury LoadRunner is an automated performance and load testing tool from Hewlett-Packard (HP). An industry standard, Mercury LoadRunner is used to predict an application's behavior and performance prior to live release. It is an enterprise-class solution for analyzing system behavior and performance.

Apache's Jmeter
The **Apache Jmeter** application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions.

Mercury Interactive's TestDirector
TestDirector is a test management tool which includes:
--*Requirements Management* – Helps in the recording and linking of software requirements in TestDirector to ensure traceability of requirements through to test cases and to defects.
--*Test Plan* - Helps in the documentation of test plans (IEEE 829 test procedures and test cases) using TestDirector. In the case of manual tests, Gillogley Services is experienced in the realisation of test strategy, test planning as well as test design through the creation of TestDirector test cases.
--*Test Lab* – Helps in managing the execution of tests including the reporting of progress of testing using Mercury Test Lab.

Testing Web based Systems

Challenges in testing for web based software, quality aspects, web engineering, testing of web based systems, Testing mobile systems.

Challenges in testing for web based software:

Diversity and Complexity: Web applications interact with many components that run on diverse hardware and Software platforms.

Dynamic Environment: The dynamic nature of web software creates challenges for the analysis, testing, and maintenance for these systems.

Very short development time: Clients of web based systems impose very short development time, compared to other software systems.

Continuous evolution: Demand for more funcitonality and capacity after the system has been designed and deployed to meet the extended scope and demands i.e Scalability issues.

Compatibility & Interoperability: Web applications often are affected by factors that may cause incompatability and interoperability issues.

Quality Aspects:

Reliability: Extensive research literature and a collection of commercial tools have been devoted to testing, ensuring, assuring, and measuring software reliability. Safety-critical software applications such as telecommunications, aerospace, and medical devices demand highly reliable software, but although many researchers are reluctant to admit it, most software currently produced does not need to be highly reliable.

Performance: Performance testing is largely on the basis of load testing, response time, download time of a web page, or transactions performed per unit time.

Security: We have all heard about Web sites being cracked and private customer information distributed or held for ransom. This is only one example of the many potential security flaws in Web software applications. When the Web functioned primarily to distribute online brochures, security breaches had relatively small consequences. Today, however, the breach of a company's Web site can cause significant revenue losses, large repair costs, legal consequences, and loss of credibility with customers:

Usability: Web application users have grown to expect easy Web transactions—as simple as buying a product at a store. Although much wisdom exists on how to develop usable software and Web sites, many Web sites still do not meet most customers' usability expectations.

Scalability: We must engineer Web software applications to be able to grow quickly in terms of both how many users they can service and how many services they can offer. The need for scalability has driven many technology innovations of the past few years. The industry has developed new software languages, design strategies, and communication and data transfer protocols in large part to allow Web sites to grow as needed.

Availability: Availability not only means that the web software is available at any time, but in todays environemnt it also must be available when accessed by diverse browsers.

Maintainability: One novel aspect of Web-based software systems is the frequency of new releases. Traditional software involves marketing, sales, and shipping or even personal installation at customers' sites. Because this process is expensive, software manufacturers usually collect maintenance modifications over time and distribute them to customers simultaneously. For a software product released today, developers will start collecting a list of necessary changes. For a simple change, (say, changing a button's label), the modification might be made immediately. But the delay in releases means that customers won't get more complex (and likely important) modifications for months, perhaps years. Web-based software, however, gives customers immediate access to maintenance updates - both small changes (such as changing the label on a button) and critical upgrades can be installed immediately.

Web Engineering:

Analysis and Design of Web Based Systems:

Keeping in consideration the nature of web applicaions, the design of these systems must consider the following:
--Design for Usability-Interface, design, navigation
--Design for comprehension
-- Design for Performance
-- Design for Security and Integrity
-- Design for evolution, maintainability
-- Design for testability.

Conceptual Modeling:
A conceptual model can be defined as a model which is made of concepts and their relationships.

- A conceptual model is the first step before drawing a UML diagram. It helps to understand the entities in the real world and how they interact with each other.

As UML describes the real time systems it is very important to make a conceptual model and then proceed gradually. Conceptual model of UML can be mastered by learning the following three major elements:

- UML building blocks
- Rules to connect the building blocks
- Common mechanisms of UML

Navigation Modeling : The navigation space model specifies, which objects can be visited through navigation in the hypermedia application. In the process of building the navigation space model the developer takes design decisions that are crucial, such as which view of the conceptual model is needed for the application and what navigation paths are required to ensure the application's functionality. The decisions of the designer are based on the conceptual model, use case model and the navigation requirements that the application must satisfy. The navigational structure model shows how to navigate through the navigation space by using the access elements

Presentation Modeling: In the next step of our methodology we describe how the information within the navigation space and the access structures are presented to the user. This is done by constructing a presentation model which shows an abstract interface design similarly to a user interface sketch. The presentation model focuses on the structural organization of the presentation and not on the physical appearance in terms of special formats, colors.

Web Scenarios Modeling: The dynamic behaviour of any web scenarios including the navigation scenarios are modeled here. The navigation structure model is detailed here showing all the sequences. This modeling is done with the help of sequence, interaction diagram / collaboration digrams and state chart diagrams.

Task Modeling: The task performed by the user or system are modeled here. Basically, the use-cases as tasks are refined here. Since use-cases are refined by activity diagrams in UML, these diagrams are used for task modeling. The modeling elements for task medeling are those for activity diagrams i.e activities, transitions and branches.

Configuration Modeling: Since web systems are drivers and complex in nature, we need to consider all the configuration and attributes that may be present on the client as well as on servers. Deployement diagrams are used for configuration modeling.

Design Activities:

Interface Design: It considers the design of all interfaces recognized in the system including screen layouts. Interfaces should be designed such that they are flexible, consistent, and readable.

Content Design: Content design is often a type of Web design and development that includes selecting the right elements and organizing them in a coherent fashion. Content typically includes text, static and animated graphics, sound and page layout.

Architecture Design: An overall system architecture describes how the network and variuous servers (Web servers, application servers and database servers) interact.

Presentation Design: This design is related to the look and feel of the application. The design issues are related to layout of the web pages, graphics to be inserted, color schemes, fonts etc..

Navigation Design: Navigational paths are designed such that usrs are able to navigate from one place to another in the web application.

Testing of Web Based Systems:

Interface Testing: The main interfaces are:

--Web server and application server interface
--Application server and database server interface

Interface Testing includes different activitieslike:

--Check if all the interactions between these servers are executed properly.
--Errors are handled properly.
--If database or web server returns any error message for any query by application server, then application server should catch and display these error messages appropriately to users.
--Check what happens if user interrupts any transaction in-between?
--Check what happens if connection to web server is reset in between?

Usability Testing: Usability Testing concentrates on testing the web application in user point of view. It includes different scenarios like:

Content checking: Content should be logical and easy to understand. Check for spelling errors.

Use of dark colors annoys users and should not be used in site theme. You can follow some standards that are used for web page and content building. Content should be meaningful. All the anchor text links should be working properly. Images should be placed properly with proper sizes. These are some basic standards that should be followed in web development. Your task is to validate all for UI testing

The general guidelines for usability testing are:

--Present information in a natural and logical order.
--Indicate similar concepts through identical terminology and graphics. Adhere to uniform conventions for layout, formatting, typefaces, labeling, etc.
--Do not force users to remember key information across documents.
--Keep in consideration that users may be from diverse categories with various goals. Provide understandable instructions where useful.
--Lay out screens in such a manner that frequently accessed information is easily found.
--The user should not feel irritating while navigating through the web application. Create visually pleasing displays.
-- Eliminate information which is irrelevant or distracting.

Content Testing:
Static contents can be checked as part of verification. For instance, Forms are the integral part of any web site. Forms are used to get information from users and to keep interaction with them. First check all the validations on each field. Check for the default values of fields and also wrong inputs to the fields in the forms. Options to create forms if any, form delete, view or modify the forms must also be checked.
There may be dynamic contents on a web page also. Largely dynamic testing will be suitable in testing these dynamic contents. These dynamic contents can be in many forms. One possibility is that constantly changing contents are there, e.g. weather information web pages or online news paper. Another case may be that web applications are generated dynamically from information contained in a data base or in a cookie.
Web page content should be correct without any spelling or grammatical errors

- All fonts should be same as per the requirements.
- All the text should be properly aligned.
- All the error messages should be correct without any spelling or grammatical errors and the error message should match with the field label.
- Tool tip text should be there for every field.
- All the fields should be properly aligned.
- Enough space should be provided between field labels, columns, rows, and error messages.
- All the buttons should be in a standard format and size.
- Home link should be there on every single page.
- Disabled fields should be grayed out.
- Check for broken links and images.
- Confirmation message should be displayed for any kind of update and delete operation.
- Check the site on different resolutions (640 x 480, 600x800 etc.?)
- Check the end user can run the system without frustration.

Navigation Testing: Navigation means how the user surfs the web pages, buttons, boxes or how user using the links on the pages to surf different pages. To ensure the functioning of correct sequence of navigations, navigation testing is performed on various possible paths. Design the test cases such that the following navigations are correctly executing: Internal Links, External Links, Redirected links ets..

The following errors can be checked during the navigation testing for the following:
--The links should not be broken due to any reasons.
--The redirected links should be with proper messages displayed to the user.
--Check that all possible navigation paths are active.
--Check that all possible navigation paths are relevant.
--Check the navigations for the Back and Forward buttons, whether these are properly working if allowed.

Compatibility Testing: Compatibility of your web site is a very important testing aspect. Different types of features for compatability are:

•Browser compatibility
•Operating system compatibility
•Mobile browsing

Browser Compatibility
Some applications are very dependent on browsers. Different browsers have different configurations and settings that your web page should be compatible with. Your web site coding should be cross browser platform compatible. We can perform Testing of web application on different browsers like Internet Explorer, Firefox, Netscape Navigator, AOL, Safari, Opera browsers with different versions.

OS Compatibility:
Some functionality in your web application may not be compatible with all operating systems. All new technologies used in web development like graphics designs, interface calls like different APIs may not be available in all Operating Systems. Test your web application on different operating systems like Windows, Unix, MAC, Linux, Solaris with different OS flavors.

Mobile Browsing:
This is the new technology age. So in future, Mobile browsing will rock. Test your web pages on mobile browsers.

Security Testing: Today, the web applications store more vital data and the number of transactions on the web has increased tremendously, with the increasing number of users. Therefore, in the internet environemnt, the most challenging issue is to protect the web applications from hackers, virus launchers.

Security Test Plan: Securing testing can be planned into two categories:

--Testing the security of the infrastructure hosting the Web application and
--Testing for vulnerabilities of the web application.

Firewalls and port scans can be the solution for security of infrastructure. For vulnerabilities, user authentication, restricted and encrypted use of cookies, data communicated must be planned. Moreover, users should not be able to browse through the directories in the server. Security Testing involves the test to identify any flaws and gaps from a security point of view.

Test Scenarios for Security Testing:
1. Verify the web page which contains important data like password, credit card numbers, secret

answers for security question etc should be submitted via HTTPS (SSL).
2.Verify the important information like password, credit card numbers etc should display in encrypted format.
3.Verify password rules are implemented on all authentication pages like Registration, forgot password, change password.
4.Verify if the password is changed the user should not be able to login with the old password.
5.Verify the error messages should not display any important information.
6.Verify if the user is logged out from the system or user session was expired, the user should not be able to navigate the site.
7.Verify to access the secured and non secured web pages directly without login.
8.Verify the "View Source code" option is disabled and should not be visible to the user.
9.Verify the user account gets locked out if the user is entering the wrong password several times.
10.Verify the cookies should not store passwords.

Performance Testing:

Performance testing is conducted to evaluate the compliance of a system or component with specified performance requirements.

General Test scenarios:

- To determine the performance, stability and scalability of an application under different load conditions.
- To determine if the current architecture can support the application at peak user levels.
- To determine which configuration sizing provides the best performance level.
- To identify application and infrastructure bottlenecks.
- To determine if the new version of the software adversely had an impact on response time.
- To evaluate product and/or hardware to determine if it can handle projected load volumes.

Performance Parameters:
--Resource utilization
--Throughput
--Response time
--Database load
--Scalability
--Round trip time

Types of Performance Testing:
-- Load Testing
-- Stress Testing

Testing mobile systems:

Mobile application testing is a process by which application software developed for hand held mobile devices is tested for its functionality, usability and consistency. Mobile application testing can be automated or manual type of testing. Mobile applications either come pre-installed or can be installed from mobile software distribution platforms. Mobile devices have witnessed a phenomenal growth in the past few years.

Key Challenges in Mobile Application Testing:

1. **Variety of Mobile Devices**: Mobile devices differ in screen sizes, input methods (QWERTY, touch, normal) with different hardware capabilities.

2. **Diversity in Mobile Platforms/OSes**: There are different mobile operating systems in the market. The major ones are Android, iOS, Symbian, Windows Phone, and BlackBerry(RIM). Each

operating system has its own limitations. Testing a single application across multiple devices running on the same platform and every platform poses a unique challenge for testers.
3.**Device Availability**: Access to the right set of devices when there is an ever-growing list of devices and operating system versions is a constant mobile application testing challenge. Access to devices can become even more challenging if testers are spread across different locations.

4. **Mobile network operators**: There are over 400 mobile network operators in the world;out of which some are CDMA, some GSM.

5. **Scripting**: The variety of devices makes executing a test script (scripting) a key challenge. As devices differ in keystrokes, input methods, menu structure and display properties single script does not function on every device.

6. **Choosing how to test**: There are two main ways of testing mobile applications: testing on real devices or testing on emulators. Unfortunately, neither method is flawless.[5] Emulators often miss issues that can only be caught by testing on real devices, but because of the multitude of different devices in the market, real devices can be expensive to purchase and time-consuming to use for testing.

Types of Mobile Application Testing:

1. **Functional Testing**: Functional testing ensures that the application is working as per the requirements. Most of the test conducted for this is driven by the user interface and call flows.

2. **Laboratory Testing**: Laboratory testing, usually carried out by network carriers, is done by simulating the complete wireless network. This test is performed to find out any glitches when a mobile application uses voice and/or data connection to perform some functions.

3. **Performance Testing**: This testing process is undertaken to check the performance and behavior of the application under certain conditions such as low battery, bad network coverage, low available memory, simultaneous access to application’s server by several users and other conditions.

4. **Memory Leakage Testing**: Memory leakage happens when a computer program or application is unable to manage the memory it is allocated resulting in poor performance of the application and the overall slowdown of the system. As mobile devices have significant constraints of available memory, memory leakage testing is crucial for the proper functioning of an application

5. **Interrupt Testing**: An application while functioning may face several interruptions like incoming calls or network coverage outage and recovery. The different types of interruptions are:
•Incoming and Outgoing SMS and MMS
•Incoming and Outgoing calls
•Incoming Notifications
•Battery Removal
•Cable Insertion and Removal for data transfer

6. **Usability testing**: Usability testing is carried out to verify if the application is achieving its goals and getting a favorable response from users. This is important as the usability of an application is its key to commercial success (it is nothing but user friendliness).

7. **Installation testing**: Certain mobile applications come pre-installed on the device whereas others have to be installed from the store. Installation testing verifies that the installation process goes smoothly without the user having to face any difficulty. This testing process covers installation,

updating and uninstalling of an application

Testing Object Oriented Software

Basics:

Terminology:

Object: An entity that has state and behaviour is known as an object e.g. chair, bike, marker, pen, table, car etc. It can be physical or logical (tengible and intengible).
An object has three characteristics:

- •state: represents data (value) of an object.
- •Behavior: represents the behavior (functionality) of an object such as deposit, withdraw etc.
- •Identity: Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But,it is used internally by the JVM to identify each object uniquely.

Class:
A class is a group of objects that has common properties. It is a template or blueprint from which objects are created.
A class in java can contain:

- •data member
- •method
- •constructor
- •block
- •class and interface

Inheritance: When one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

Polymorphism: When one task is performed by different ways i.e. known as polymorphism. For example: to convense the customer differently, to draw something e.g. shape or rectangle etc.

Abstraction: Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing.

Encapsulation: Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.

Inheritance: Object-oriented programming allows classes to *inherit* commonly used state and behavior from other classes. Different kinds of objects often have a certain amount in common with each other. Mountain bikes, road bikes, and tandem bikes, for example, all share the characteristics of bicycles (current speed, current pedal cadence, current gear). Yet each also defines additional features that make them different.

Object-Oriented Modeling and UML:

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997. OMG is continuously putting effort to make a truly industry standard.
--UML stands for **U**nified **M**odeling **L**anguage.
--UML is different from the other common programming languages like C++, Java, COBOL etc.

--UML is a pictorial language used to make software blue prints.
So UML can be described as a general purpose visual modeling language to visualize, specify, construct and document software system. Although UML is generally used to model software systems but it is not limited within this boundary. It is also used to model non software systems as well like process flow in a manufacturing unit etc.

UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization UML is become an OMG (Object Management Group) standard.

User View:
The UML user model view encompasses the models which define a solution to a problem as understood by the client or stakeholders. This view is often also referred to as the Use Case or scenario view.

Structural View:
The UML structural view encompasses the models which provide the static, structural dimensions and properties of the modelled system. This view is often also referred to as the static or logical view. particular point in time.

Behavioural View:
These UML models describe the behavioural and dynamic features and methods of the modelled system. This view is often also referred to as the dynamic, process, concurrent or collaborative view. after completion of the elements as flows of processing within a system.

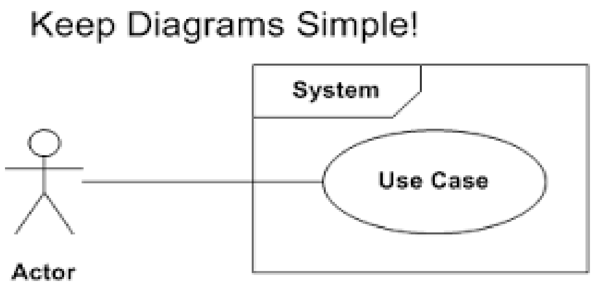
ImplementationView:
The UML Implementation View combines the structural and behavioural dimensions of the solution realisation or implementation. The view is often also referred to as the component or development view.

Environment View:
These UML models describe both structural and behavioural dimensions of the domain or environment in which the solution is implemented. This view is often also referred to as the deployment or physical view.

Use Case Model: These models depict the functionality required by the system and the interaction of users and other elements (known as actors) with respect to the specific solution.

Use Case: A Use Case is a scenario that identifies and describe how the system will be used in given situation.

Actors: Use Cases are typically related to 'actors', which are human or machine entities that use or interact with the system to perform a piece of meaningful work that helps them to achieve a goal.



Class Diagrams: These models describe the static structure and contents of a system using elements such as classes, packages and objects to display relationships such as containment, inheritance and associations.

Object Diagrams: An object diagram represents the static structure of the system at a particular instance of time.

Sequence Diagram: Describe timing sequence of the objects over a vertical time dimension with interactions between objects depicted on a horizontal dimension.

Collaboration Diagrams: Describe the interactions and relationships between objects and sequences of a system organised in time and space. Numbers are used to show the sequence of messages.

State Diagrams: Describe the sequence, status conditions and appropriate responses or actions to conditions during the life of the objects within the system.

Activity Diagrams: Describe the methods, activities and resulting transitions

Component Diagrams: These depict the high level organisation and dependencies of source code components, binary components and executable components and whether these components exist at compile, link or run time.

Deployment Diagrams: These UML Models depict and describe the environmental elements and configuration of runtime processing components, libraries and objects that will reside on them.

Object Oriented Testing:

Testing is a continuous activity during software development. In object-oriented systems, testing encompasses three levels, namely, unit testing, subsystem testing, and system testing.

Unit Testing: In unit testing, the individual classes are tested. It is seen whether the class attributes are implemented as per design and whether the methods and the interfaces are error-free. Unit testing is the responsibility of the application engineer who implements the structure.

Subsystem Testing: This involves testing a particular module or a subsystem and is the responsibility of the subsystem lead. It involves testing the associations within the subsystem as well as the Interaction of the subsystem with the outside. Subsystem tests can be used as regression tests for each newly released version of the subsystem.

System Testing: System testing involves testing the system as a whole and is the responsibility of the quality-assurance team. The team often uses system tests as regression tests when assembling new releases.

Object-Oriented Testing Techniques:

Grey Box Testing: The different types of test cases that can be designed for testing object-oriented programs are called grey box test cases. Some of the important types of grey box testing are:

State model based testing: This encompasses state coverage, state transition coverage, and state transition path coverage.

Use case based testing: Each scenario in each use case is tested.
Class diagram based testing: Each class, derived class, associations, and aggregations are tested.

Sequence diagram based testing : The methods in the messages in the sequence diagrams are tested.

Techniques for Subsystem Testing: The two main approaches of subsystem testing are:

Thread based testing : All classes that are needed to realize a single use case in a subsystem are integrated and tested.

Use based testing : The interfaces and services of the modules at each level of hierarchy are tested. Testing starts from the individual classes to the small modules comprising of classes, gradually to larger modules, and finally all the major subsystems.

Categories of System Testing:

Alpha testing : This is carried out by the testing team within the organization that develops software.

Beta testing : This is carried out by select group of co-operating customers.

Software Quality Assurance:

Software Quality: Schulmeyer and McManus have defined software quality as “the fitness for use of the total software product”. A good quality software does exactly what it is supposed to do and is interpreted in terms of satisfaction of the requirement specification laid down by the user.

Quality Assurance: Software quality assurance is a methodology that determines the extent to which a software product is fit for use. The activities that are included for determining software quality are:

- Auditing
- Development of standards and guidelines
- Production of reports
- Review of quality system

Quality Factors:

Correctness: Correctness determines whether the software requirements are appropriately met.
Usability: Usability determines whether the software can be used by different categories of users (beginners, non-technical, and experts).
Portability: Portability determines whether the software can operate in different platforms with different hardware devices.
Maintainability: Maintainability determines the ease at which errors can be corrected and modules can be updated.
Reusability: Reusability determines whether the modules and classes can be reused for developing other software products.

Object-Oriented Metrics: Metrics can be broadly classified into three categories: project metrics, product metrics, and process metrics.

Project Metrics: Project Metrics enable a software project manager to assess the status and performance of an ongoing project. The following metrics are appropriate for object-oriented software projects:

- Number of scenario scripts
- Number of key classes
- Number of support classes
- Number of subsystems

Product Metrics: Product metrics measure the characteristics of the software product that has been developed. The product metrics suitable for object-oriented systems are:

Methods per Class: It determines the complexity of a class. If all the methods of a class are assumed to be equally complex, then a class with more methods is more complex and thus more susceptible to errors.

Inheritance Structure: Systems with several small inheritance lattices are more well–structured than systems with a single large inheritance lattice. As a thumb rule, an inheritance tree should not have more than 7 (± 2) number of levels and the tree should be balanced.

Coupling and Cohesion: Modules having low coupling and high cohesion are considered to be better designed, as they permit greater reusability and maintainability.

Response for a Class: It measures the efficiency of the methods that are called by the instances of the class.

Process Metrics: Process metrics help in measuring how a process is performing. They are collected over all projects over long periods of time. They are used as indicators for long-term software process improvements. Some process metrics are:

- Number of KLOC (Kilo Lines of Code)
- Defect removal efficiency
- Average number of failures detected during testing
- Number of latent defects per KLOC