RAD Model

RAD model is Rapid Application Development model. It is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.If the project is large, it is divided into a series of smaller projects. Each of these smaller projects is planned and delivered individually. Thus, with a series of smaller projects, the final project is delivered quickly and in a less structured manner. The major characteristic of the RAD model is that it focuses on the reuse of code, processes, templates, and tools.

Phases in RAD Model:

- Business Modeling
- Data Modeling
- Process Modeling
- Application Modeling
- Testing and Turnover

1)Business Modeling: The business model for the product under development is designed in terms of flow of information and the distribution of information between various business channels. A complete business analysis is performed to find the vital information for business, how it can be obtained, how and when is the information processed and what are the factors driving successful flow of information.

2)Data Modeling: Once the business modeling phase over and all the business analysis completed, all the required and necessary data based on business analysis are identified in data modeling phase.

3)Process modeling: Data objects defined in data modeling are

converted to achieve the business information flow to achieve some specific business objective. Description are identified and created for CRUD of data objects.

4)Application Generation: The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes.

5)Testing and turnover: All the testing activates are performed to test the developed application.

Advantages of RAD Model:

a)Fast application development and delivery.
b)Lest testing activity required.
c)Visualization of progress.
d)Less resources required.
e)Review by the client from the very beginning of development so very less chance to miss the requirements.
f)Very flexible if any changes required.
g)Cost effective.
h)Good for small projects.

Disadvantages of RAD model:

a)Depends on strong team and individual performances for identifying business requirements.
b)Only system that can be modularized can be built using RAD
c)Requires highly skilled developers/designers.
d)High dependency on modeling skills
e)Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

When to use RAD model:

a)RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
b)It should be used if there’s high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
c)RAD SDLC model should be chosen only if resources with high

business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).
d)If technical risks are low.
e)If development needed to complete in specified time.
f)RAD Model is suitable if the functionality have less dependencies on other functionality.

Software Testing

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of A software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

Verification

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

Validation

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

Testing can either be done manually or using an automated testing tool:

- **Manual** – This testing is performed without taking help

of automated testing tools. The software tester prepares test cases for different sections and levels of the code, executes the tests and reports the result to the manager.Manual testing is time and resource consuming. The tester needs to confirm whether or not right test cases are used. Major portion of testing involves manual testing.

- **Automated** This testing is a testing procedure done with aid of automated testing tools. The limitations with manual testing can be overcome using automated test tools.

Tests can be conducted based on two approaches –

- Functionality testing
- Implementation testing

When functionality is being tested without taking the actual implementation in concern it is known as black-box testing. The other side is known as white-box testing where not only functionality is tested but the way it is implemented is also analyzed.

1)Black Box Testing: Black Box Testing, also known as Behavioral Testing, is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

The above Black-Box can be any software system you want to test. For example: an operating system like Windows, a website like

Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

This method attempts to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access
- Behavior or performance errors
- Initialization and termination errors

There are many types of Black Box Testing but following are the prominent ones –

- **Functional testing** – This black box testing type is related to functional requirements of a system; it is done by software testers.
- **Non-functional testing** – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements such as performance, scalability, usability.
- **Regression testing** – Regression testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

Black box testing has its own life cycle called Software Test Life Cycle (STLC) and it is relative to every stage of Software Development Life Cycle.Some famous Black Box testing techniques are Boundary value analysis, state transition testing, equivalence partitioning.

2)White Box Testing: It is also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing.

It is a software testing method in which the internal

structure/ design/ implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

This method is named so because the software program, in the eyes of the tester, is like a white/ transparent box; inside which one clearly sees.

White box testing, on its own, cannot identify problems caused by mismatches between the actual requirements or specification and the code as implemented but it can help identify some types of design weaknesses in the code. Examples include control flow problems (e.g., closed or infinite loops or unreachable code), and data flow problems (e.g., trying to use a variable which has no defined value). Static code analysis (by a tool) may also find these sorts of problems, but doesn’t help the tester/developer understand the code to the same degree that personally designing white-box test cases does.

3)Gray Box Testing: Gray Box Testing is a software testing method which is a combination of Black Box Testing method and White Box Testing method. In Black Box Testing, the internal structure of the item being tested is unknown to the tester and in White Box Testing the internal structure in known. In Gray Box Testing, the internal structure is partially known. This involves having access to internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level.

Gray Box Testing is named so because the software program, in the eyes of the tester is like a gray/ semi-transparent box; inside which one can partially see.

Gray Box Testing gives the ability to test both sides of an application, presentation layer as well as the code part. It is primarily useful in Integration Testing and Penetration Testing.Grey-box testing is a perfect fit for Web-based applications.Grey-box testing is also a best approach for functional or domain testing.

Techniques used for Grey box Testing are-

- **Matrix Testing:** This testing technique involves defining all the variables that exist in their programs.
- **Regression Testing**: To check whether the change in the previous version has regressed other aspects of the program in the new version. It will be done by testing strategies like retest all, retest risky use cases, retest within firewall.
- **Orthogonal Array Testing or OAT**: It provides maximum code coverage with minimum test cases.
- **Pattern Testing:** This testing is performed on the historical data of the previous system defects. Unlike black box testing, gray box testing digs within the code and determines why the failure happened

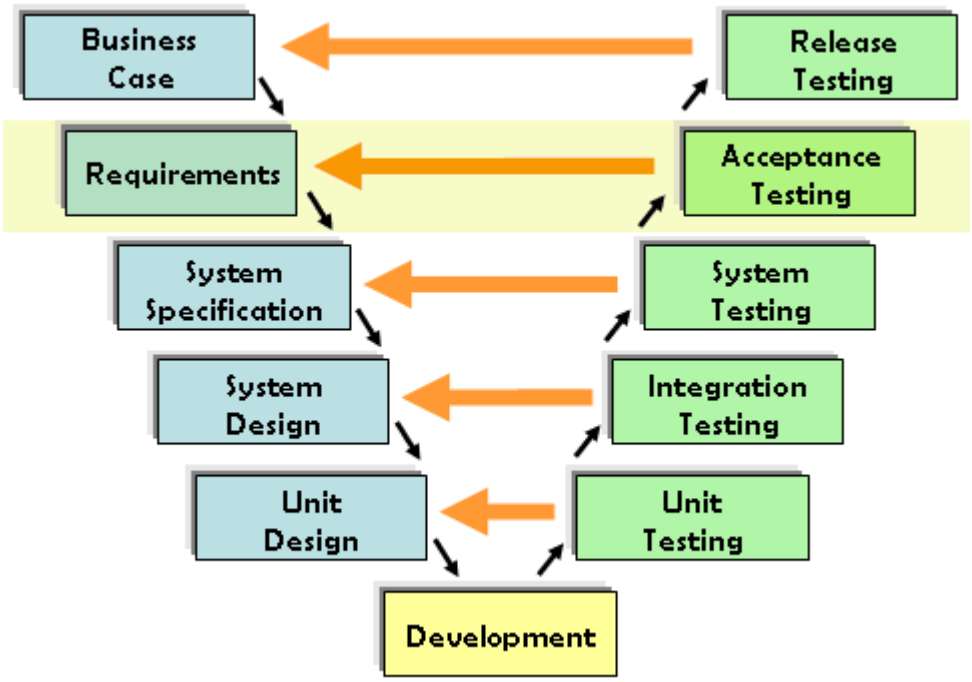
Usually, Grey box methodology uses automated software testing tools to conduct the testing. Stubs and module drivers are created to relieve tester to manually generate the code.

S.N.	Black Box Testing	Grey Box Testing	White Box Testing
1	The Internal Workings of an application are not required to be known	Somewhat knowledge of the internal workings are known	Tester has full knowledge of the Internal workings of the application
2	Also known as closed box testing, data driven testing and functional testing	Another term for grey box testing is translucent testing as the tester has limited knowledge of the insides of the application	Also known as clear box testing, structural testing or code based testing
3	Performed by end users and also by testers and developers	Performed by end users and also by testers and developers	Normally done by testers and developers
4	Testing is based on external expectations - Internal behavior of the application is unknown	Testing is done on the basis of high level database diagrams and data flow diagrams	Internal workings are fully known and the tester can design test data accordingly
5	This is the least time consuming and exhaustive	Partly time consuming and exhaustive	The most exhaustive and time consuming type of testing
6	Not suited to algorithm testing	Not suited to algorithm testing	Suited for algorithm testing
7	This can only be done by trial and error method	Data domains and Internal boundaries can be tested, if known	Data domains and Internal boundaries can be better tested

There are many other types of testing like:

Acceptance Testing

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.



Regression Testing

Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing.

Beta Testing

Beta testing is the testing which is done by end users, a team outside development, or publicly releasing full pre-version of the product which is known as beta version. The aim of beta testing is to cover unexpected errors. It falls under the class of black box testing.

Unit Testing

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.Statements, functions, methods, interfaces i.e

units of the code are individually tested for proper execution. It can be automated or can be done manually. Usually **small data** is used for unit testing.

Integration Testing

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing. Different approaches used in integration testing are: **top down & bottom up** integration testing, **sandwich testing** (combination of both).

Stress Testing

Stress testing is the testing to evaluate how system behaves under unfavorable conditions. Testing is conducted at beyond limits of the specifications. It falls under the class of black box testing.

Performance Testing

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

Functional Testing

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

System Testing

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of

black box testing. It is performed **after integration testing.** Various approaches used are: **load testing, smoke testing, security testing, migration testing** etc.

Usability Testing

Usability testing is performed to the perspective of the client, to evaluate how the GUI is user-friendly? How easily can the client learn? After learning how to use, how proficiently can the client perform? How pleasing is it to use its design? This falls under the class of black box testing.

Introduction of Software Engineering and Waterfall Model

Software Engineering is an engineering approach for software development.The basic principle of software engineering is to use structured, formal and disciplined methods for building and using systems.The outcome of software engineering is an efficient and reliable software product.

Without using software engineering principles it would be difficult to develop large programs. In industry it is usually needed to develop large programs to accommodate multiple functions. A problem with developing such large commercial programs is that the complexity and difficulty levels of the programs increase exponentially with their sizes. Software engineering helps to reduce this programming complexity. Software engineering principles use two important techniques to reduce problem complexity: abstraction and decomposition. The principle

of abstraction implies that a problem can be simplified by omitting irrelevant details. In other words, the main purpose of abstraction is to consider only those aspects of the problem that are relevant for certain purpose and suppress other aspects that are not relevant for the given purpose. Once the simpler problem is solved, then the omitted details can be taken into consideration to solve the next lower level abstraction, and so on. Abstraction is a powerful way of reducing the complexity of the problem.

The other approach to tackle problem complexity is decomposition. In this technique, a complex problem is divided into several smaller problems and then the smaller problems are solved one by one. However, in this technique any random decomposition of a problem into smaller parts will not help. The problem has to be decomposed such that each component of the decomposed problem can be solved independently and then the solution of the different components can be combined to get the full solution. A good decomposition of a problem should minimize interactions among various components.

System Requirement Specification(SRS):

It is obtained after excessive discussions with the users.Software requirement specification (SRS) is a document that completely describes what the proposed software should do without describing how software will do it.SRS is important and difficult task of a System Analyst.

Characteristics of SRS:

- Correct
- Complete and Unambiguous
- Verifiable
- Consistent
- Traceable
- Modifiable

Software Life Cycle Models:

A software life cycle model (also called process model) is a descriptive and diagrammatic representation of the software life cycle. A life cycle model represents all the activities required to make a software product transit through its life cycle phases. It also captures the order in which these activities are to be undertaken. In other words, a life cycle model maps the different activities performed on a software product from its inception to retirement. Different life cycle models may map the basic development activities to phases in different ways. Thus, no matter which life cycle model is followed, the basic activities are included in all life cycle models though the activities may be carried out in different orders in different life cycle models. During any life cycle phase, more than one activity may also be carried out. A software life cycle model is a particular abstraction representing a software life cycle.Such a model may be:

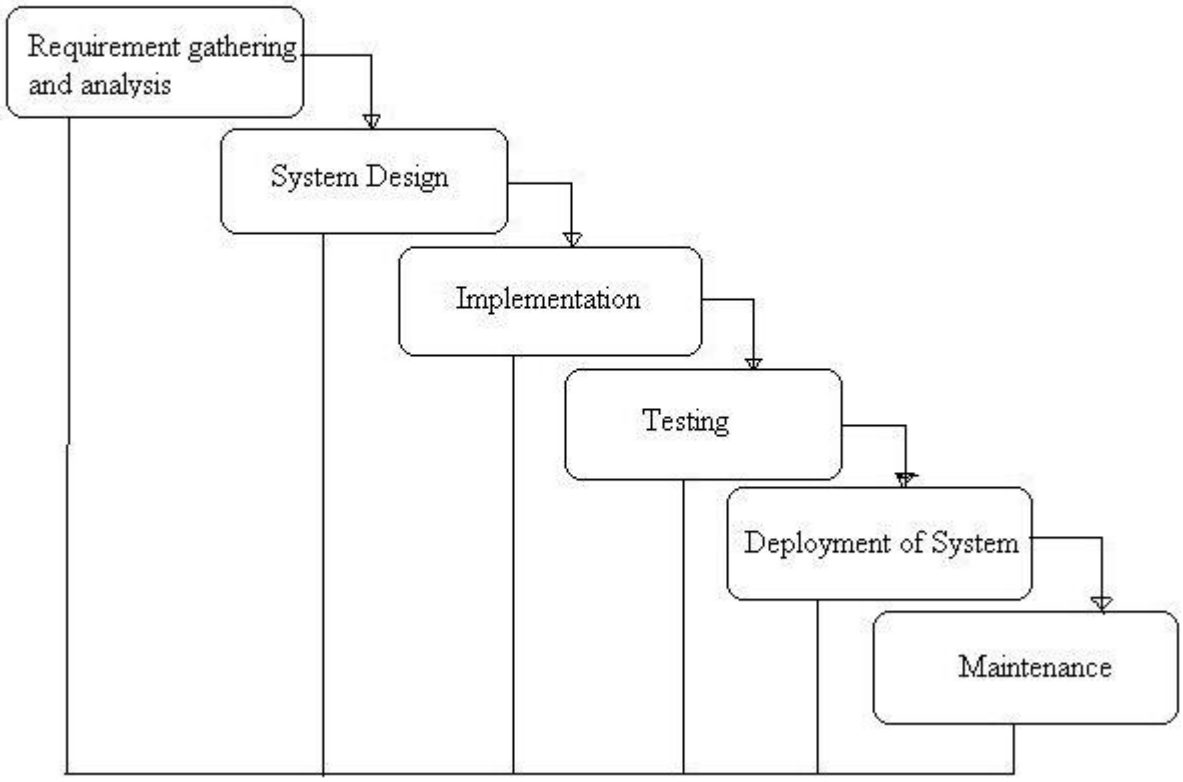
- Activity-centered—-Focusing on the activities of software development
- Entity-centered—-Focusing on the work products created by these activities

A software life cycle model is often referred to as a Software Development Life Cycle(SDLC).ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

Waterfall Model:

The Waterfall Model was first Process Model to be introduced. The Waterfall Model is a linear sequential flow. In which progress is seen as flowing steadily downwards (like a waterfall) through the phases of software implementation. This means that any phase in the development process begins only if the previous phase is complete. The waterfall approach does not define the process to go back to the previous phase to handle changes in requirement. The waterfall approach is the earliest approach that was used for software development.

General Overview of "Waterfall Model"



- **Requirement Gathering and Analysis:**Capture all the possible requirement of the system to be developed and documented in a software requirement.
- **System Design:**Helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation:**With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- **Integration and Testing:**All the units developed in the implementation phase are integrated into a system after testing of each unit. During this phase, each module is unit tested to determine the correct working of all the individual modules. It involves testing each module in isolation as this is the most efficient way to debug the errors identified at this stage.

- **Integration and System Testing:** During the integration and system testing phase, the modules are integrated in a planned manner. The different modules making up a software product are almost never integrated in one shot. Integration is normally carried out incrementally over a number of steps. During each integration step, the partially integrated system is tested and a set of previously planned modules are added to it. Finally, when all the modules have been successfully integrated and tested, system testing is carried out. The goal of system testing is to ensure that the developed system conforms to its requirements laid out in the SRS document.System testing usually consists of three different kinds of testing activities:α – testing: It is the system testing performed by the development team.
  β –testing: It is the system testing performed by a friendly set of customers.

  Acceptance testing: It is the system testing performed by the customer himself after the product delivery to determine whether to accept or reject the delivered product.

- **Deployment of System:**Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market.
- **Maintenance:** Maintenance of a typical software product requires much more than the effort necessary to develop the product itself. Many studies carried out in the past confirm this and indicate that the relative effort of development of a typical software product to its maintenance effort is roughly in the 40:60 ratios. Maintenance involves performing any one or more of the following three kinds of activities: Correcting errors that were not discovered during the product development phase. This is called

corrective maintenance.Improving the implementation of the system, and enhancing the functionalities of the system according to the customer's requirements. This is called perfective maintenance.

Porting the software to work in a new environment. For example, porting may be required to get the software to work on a new computer platform or with a new operating system. This is called adaptive maintenance.

Advantages of waterfall model:

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages of waterfall model:

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

When to use the waterfall model:

- This model is used only when the requirements are very well known, clear and fixed.

- Product definition is stable.
- Technology is understood.
- There are no ambiguous requirements
- Ample resources with required expertise are available freely
- The project is short.

Very less customer enter action is involved during the development of the product. Once the product is ready then only it can be demoed to the end users. Once the product is developed and if any failure occurs then the cost of fixing such issues are very high, because we need to update everywhere from document till the logic.