CLASSICAL WATERFALL MODEL

ECC 811 – SOFTWARE ENGINEERING

Tuesday, July 8, 2025

WHAT IS CLASSICAL WATERFALL MODEL?

1. **Waterfall Model is one of the earliest and most straightforward Software Development Life Cycle (SDLC) methodologies**.
2. **Waterfall model follows a linear, sequential approach where each phase must be completed fully before the next phase begins**, resembling a waterfall flowing steadily downward through distinct stages.

KEY FEATURES OF THE WATERFALL MODEL

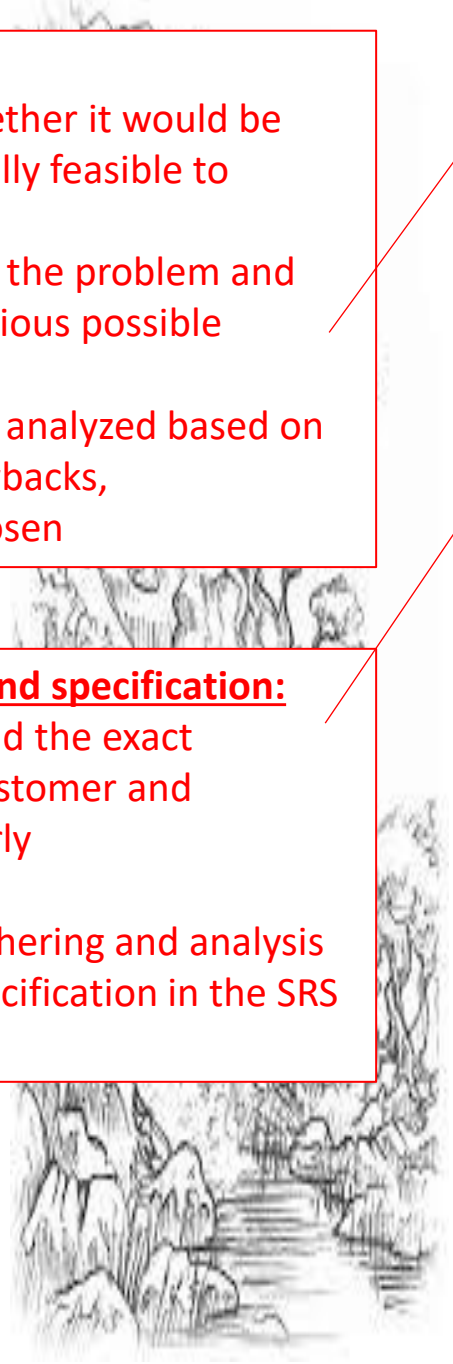
1. **Linear & Sequential:** Progress flows in one direction (like a waterfall) through defined phases.
2. **Phase-Gated:** Each phase has specific deliverables and a review process. Approval is needed to proceed to the next phase.
3. **Documentation-Driven:** Heavy emphasis on comprehensive documentation at each stage before moving forward.
4. **Minimal Customer Involvement:** Customer feedback is typically gathered only at the very beginning (requirements) and the very end (delivery).
5. **"Big Design Up Front" (BDUF):** Requires detailed requirements and design specifications to be finalized early.

1. Feasibility Study:

1. Used to determine whether it would be financially and technically feasible to develop the software.
2. Involves understanding the problem and then determine the various possible solutions.
3. Identified solutions are analyzed based on their benefits and drawbacks,
4. The best solution is chosen

2. Requirements analysis and specification:

1. The aim is to understand the exact requirements of the customer and document them properly
2. Uses 2 steps, i.e
   a) Requirement gathering and analysis
   b) Requirement specification in the SRS document.



3. Design:

Transforms the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language.

4. Coding and Unit testing:

1. Software design is translated into source code using any suitable programming language.
2. Each designed module is coded.
3. Each module is tested to ensure that it is working properly.

Integration and System testing:

1. Integration of various modules is carried out incrementally over a number of steps.
2. The full working system is tested.

Maintenance

1. Corrective, perfective and adaptive maintenance is carried out.

SOFTWARE MAINTENANCE

1. Maintenance comprises approximately 60% of the total effort spent to develop a full software.
2. There are basically three types of maintenance:
   a) **Corrective Maintenance:** corrects errors that were not discovered during the product development phase.
   b) **Perfective Maintenance:** enhances the functionalities of the system based on the customer's request.
   c) **Adaptive Maintenance:** allows work in a new environment such as work on a new computer platform or with a new operating system.

TYPES OF SYSTEM TESTING / 01

- Unit Testing:
- Integration Testing:
- System Testing:
- User Acceptance Testing:

TYPES OF SYSTEM TESTING

There are three types of system testing

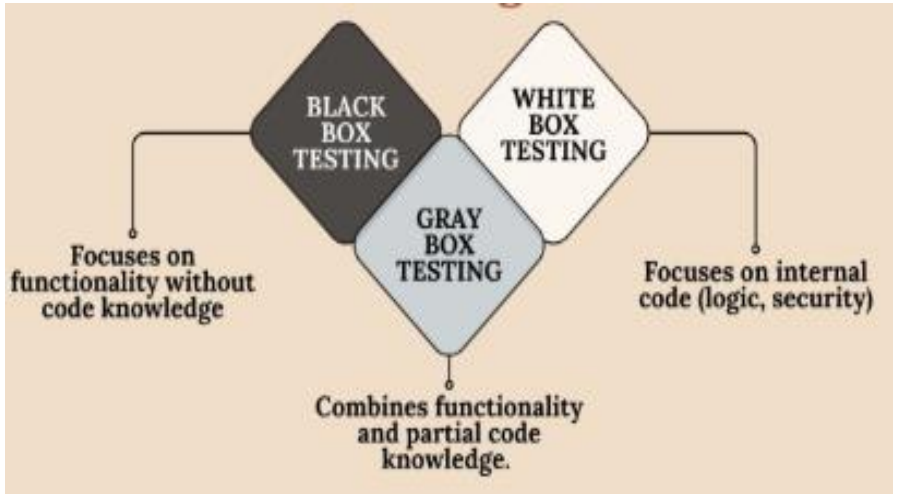
1. **Alpha testing:** Alpha testing is the system testing performed by the development team.

2. **Beta testing:** Beta testing is the system testing performed by a friendly set of customers.

3. **Acceptance testing:** After the software has been delivered, the customer performed the acceptance testing to determine whether it meets his needs.

WHITE BOX, BLACK BOX & GREY BOX TESTING

White box, black box, and grey box testing are three distinct approaches to software testing, differing primarily in the level of knowledge testers have about the system's internal workings.

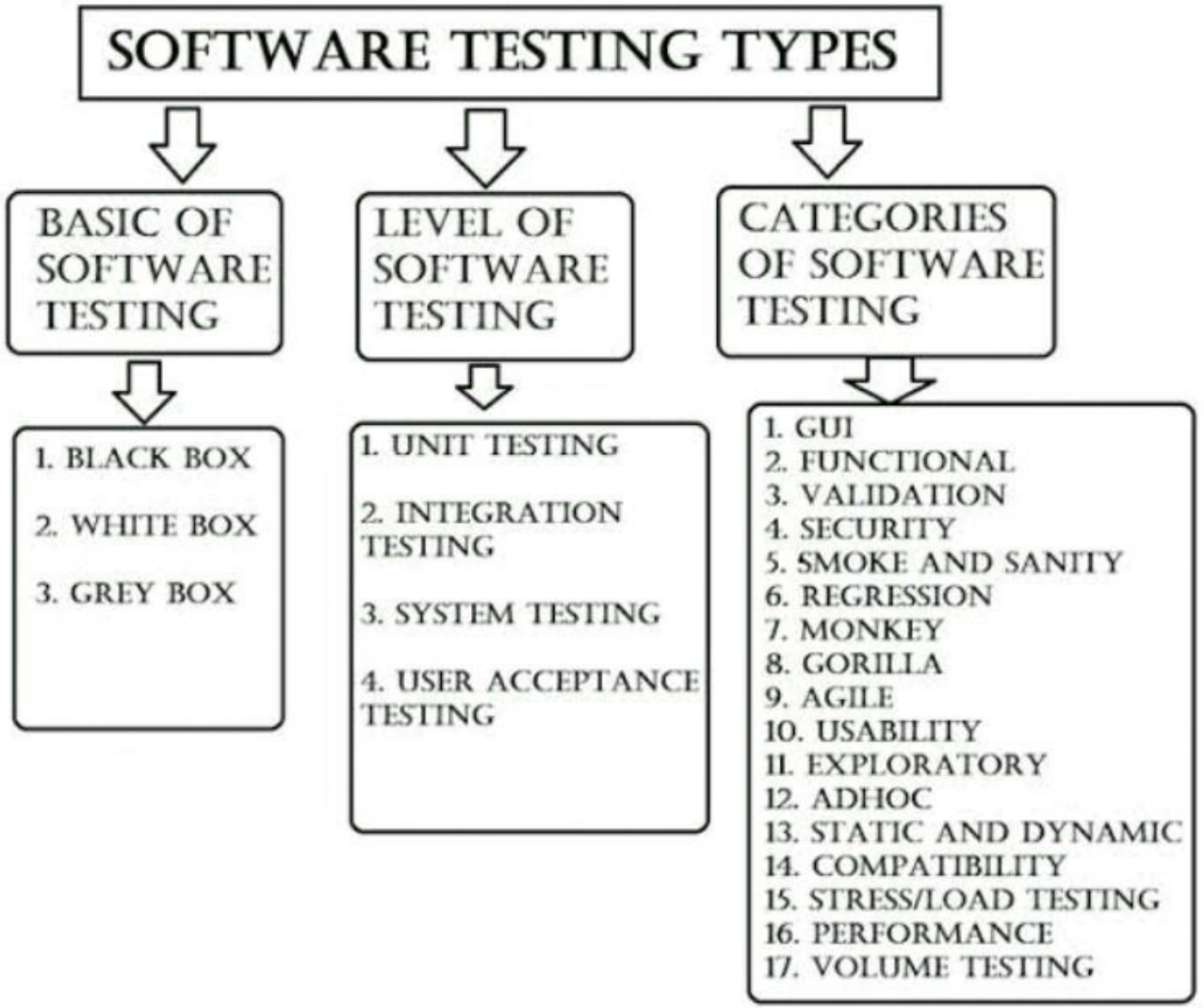


1. **White box testing** involves having full knowledge of the system's internal structure, code, and logic, allowing testers to design tests based on this knowledge.
2. **Black box testing** treats the system as a "black box," meaning testers have no knowledge of the internal workings and focus on functionality based on specifications and requirements.
3. **Grey box testing is a hybrid approach**, where testers have partial knowledge of the system's internal workings, allowing them to combine elements of both white box and black box testing.

WHITE BOX, BLACK BOX & GREY BOX TESTING COMPARISON

Aspect	Black Box testing	White Box testing	Grey Box testing
Knowledge of Code	No knowledge	Full knowledge	Partial knowledge
Focus	Functionality & behavior	Internal code & logic	Combination of both
Performed By	Testers, QA engineers	Developers	Testers with some code knowledge
UseCase	UI, APIs, system behavior	Code structure, security	Security, functional, and logic flaws
Testing Approach	Input-output validation	Code path analysis	Both functional and structural testing

TYPES OF SOFTWARE TESTING (SUMMARY)



ADVANTAGES OF THE WATERFALL MODEL

The advantages of the waterfall model are as follows.

1. This model is very simple and is easy to understand.
2. Phases in this model are processed one at a time.
3. Each stage in the model is clearly defined.
4. This model has very clear and well understood milestones.
5. Process, actions and results can be well documented.
6. Reinforces good habits:
   a) define-before- design
   b) design-before-code
7. This model works well for smaller projects and projects where requirements are well-understood.

DISADVANTAGES OF THE WATERFALL MODEL

The advantages of the waterfall model are as follows.

1. **No feedback path:** Model assumes that no error is ever committed by developers during any phases. Therefore, it does not incorporate any mechanism for error correction.
2. **Difficult to accommodate change requests:** Assumes that all the customer requirements can be completely and correctly defined at the beginning of the project, but actually customers' requirements keep on changing with time. It is difficult to accommodate any change requests after the requirements specification phase is complete.
3. **No overlapping of phases:** A new phase can start only after the completion of the previous phase. But in real projects, this can't be maintained. To increase the efficiency and reduce the cost, phases may be required overlap.

WHY DOES THE WATERFALL MODEL FAIL ON REAL PROJECTS? / 01

1. **One way street:** Model is just like the one-way street. Once phase X is completed and next phase Y has started then there is no way to going back on the previous phase. This is one of the issues to the failure of the waterfall model.
2. **Overlapping:** Model has no provisions for overlapping among phases. But in real projects, this can't be maintained. To increase the efficiency and reduce the cost, phases may overlap.
3. **Interaction:** There is no interaction among phase. Users have little interaction with project team. This feedback is not taken during development.

WHY DOES THE WATERFALL MODEL FAIL ON REAL PROJECTS? / 02

4. **Support delivery of system:** Model does not support delivery of system in pieces. After a development process starts, changes cannot accommodate easily.
5. **Feedback path:** Model has no feedback path. Hence, it does not incorporate any mechanism for error correction.
6. **Not Flexible:** Difficult to accommodate change requests. The model assumes that all the customer requirements can be completely and correctly defined at the beginning of the project, but actually customers' requirements keep on changing with time. After the requirements specification phase is completed difficult to accommodate any change requests.

WHEN IS WATERFALL SUITABLE?

1. **Projects with very clear, stable, and well-understood requirements** from the outset.
2. **Projects with fixed scope, budget, and deadlines** that are unlikely to change.
3. **Projects involving familiar technology.**
4. **Short duration projects.**
5. **Projects with stringent regulatory compliance** requiring extensive documentation (common in some industries like medical devices or aerospace, though often adapted).