MODULE 1

PHASES OF SOFTWARE PROJECT

REQUIREMENT GATHERING AND ANALYSIS

- Specific requirements of the software to be built are gathered and documented in the form of system requirement specification(SRS)

PLANNING

- This phase comes up with schedule, the scope and resource requirement of project
- At the end ,project plan and test plan documents are delivered

DESIGN

- This phase figure out how to meet the requirements
- It is divided into 2 levels. High level design and low level design
- System design description (SDD) document is created at end of this phase

CODING

- It includes coding the program in choosen programming language

TESTING

- It includes identifying and removing defects in the software
- Different kinds of testing are
- Unit testing: Modules are tested individually
- Integration testing: Interconnection among modules are tested
- System testing: System as a whole is tested
- Acceptance testing: Tested with real life data

DEPLOYMENT AND MAINTAINCE

- Defects occur after the deployment of software in customer's environment should be corrected
- There are 3 kinds of maintaince
- Corrective: To correct errors that were not discovered during development
- Adaptive: For porting software to work in new environment
- Preventive: for eg changing the application program code to avoid security hole in operating system code

Comparison of quality control and quality assurance

Quality assurance

- concentrated on process of producing the product
- defect prevention oriented
- usually done throughout life cycle
- this is usually staff function
- eg reviews and audits

Quality control

- concentrating on specific product
- defect detection and correction orientex
- usually done after the product is built
- this is a line function
- eg software testing at various levels

VERIFICATION

- It check whether we are building product right
- It is done to prevent defects before they take shape
- It includes requirement review, design review, code review etc
- We can assume verification and quality assurance to be one

VALIDATION

- It check whether we are building right product
- It finds defect and fix them
- It includes different kind of testing like unit testing, integration testing etc
- Validation and quality control are assumed to be one

PROCESS MODEL TO REPRESENT DIFFERENT PHASES

- ETVX model(entry ,task ,verification ,exit)

_entry criteria:

specify when that phase can be started and conditions that input to phase should satisfy

Tasks:

activities to be carried out in that phase

Verification:

Specify the methods of checking that the tasks have been carried out correctly

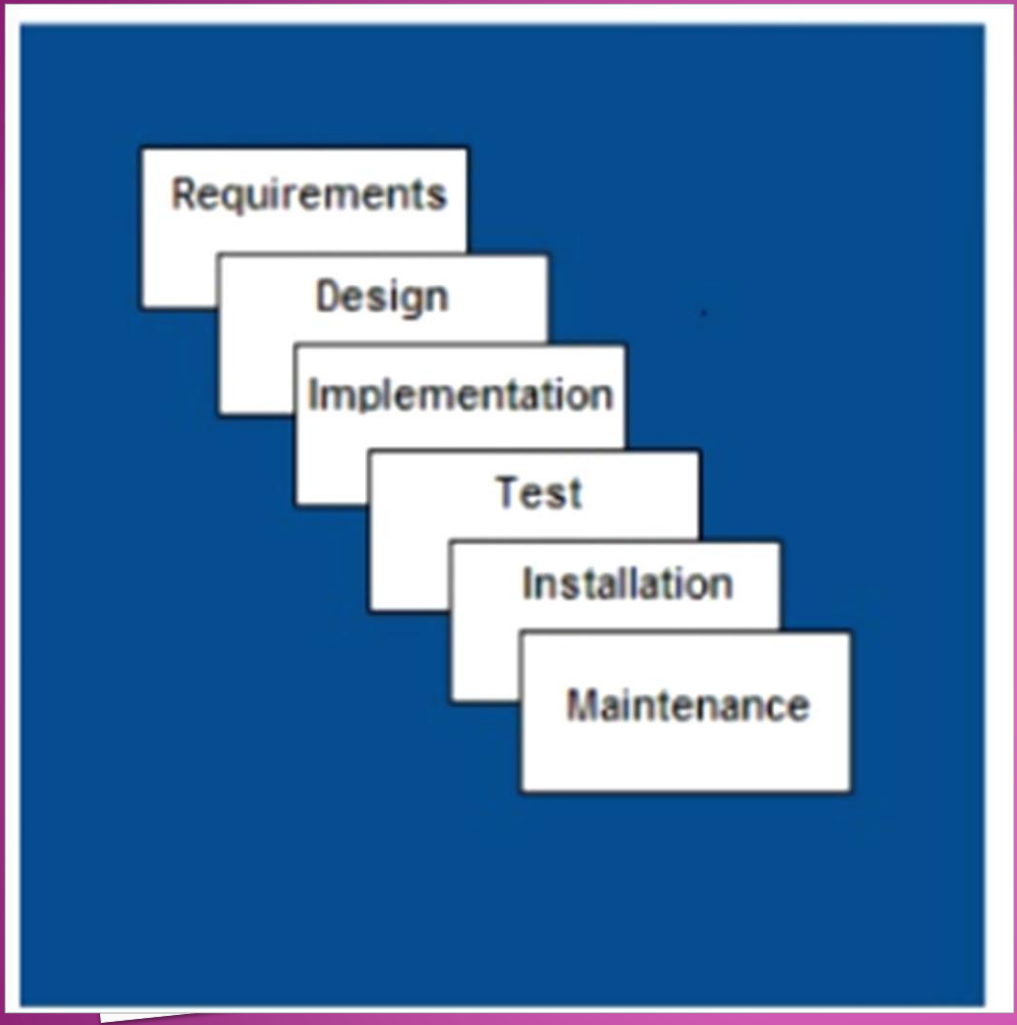
Exit criteria:

conditions that output of each phase should satisfy

Life Cycle Model

- It provides a fixed generic framework that can be tailored to a specific project.
- Project specific parameters will include:
- Size, (person-years)
- Budget,
- Duration

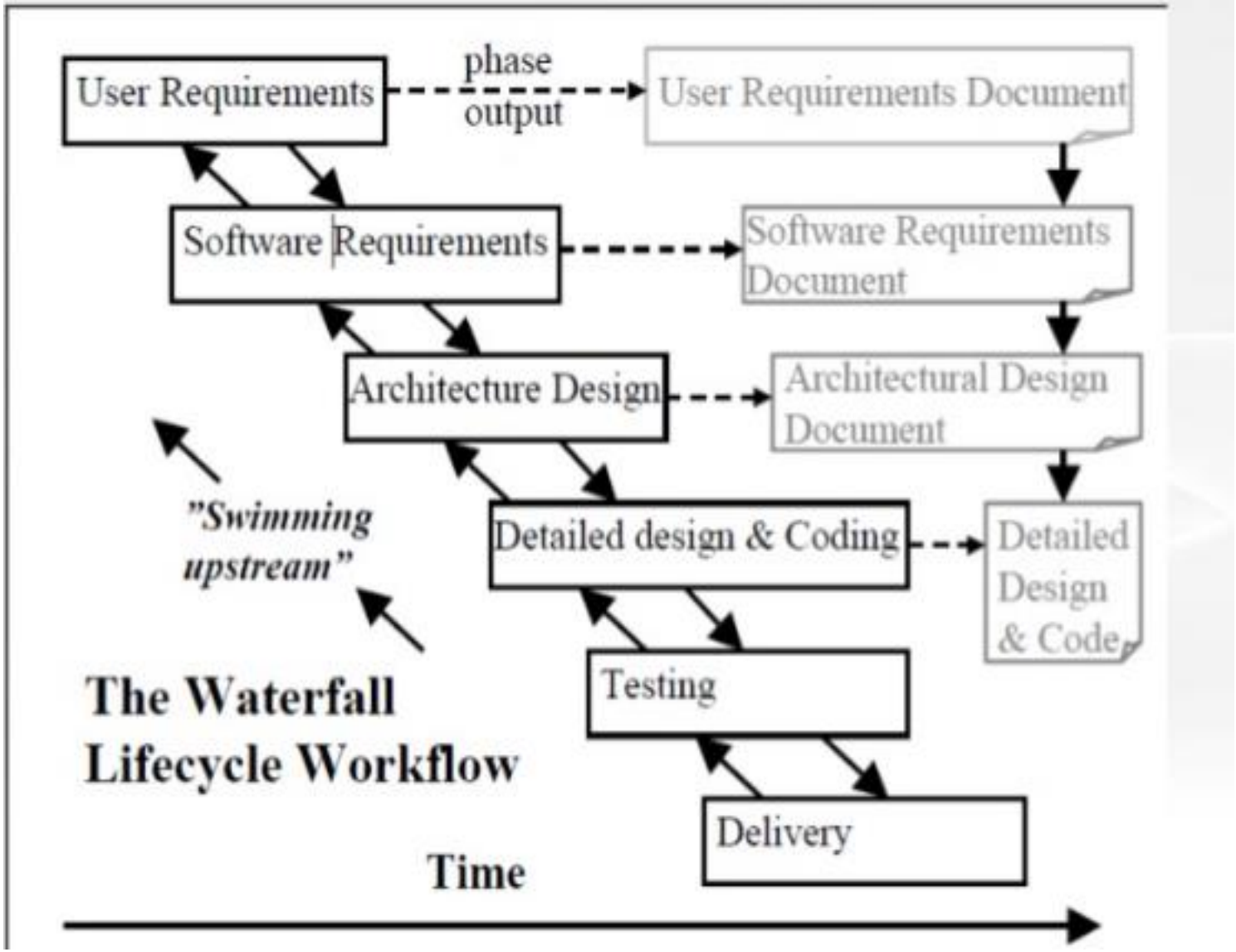
project plan = lifecycle model + project parameters



WATERFALL MODEL

Phases are organized in linear order

At end of each phase certification is done to the o/p formed


User Requirements
phase output
User Requirements Document
Software Requirements
Software Requirements Document
Architecture Design
Architectural Design Document
"Swimming upstream"
Detailed design & Coding
Detailed Design & Code
Testing
Delivery
The Waterfall Lifecycle Workflow
Time

ADVANTAGES

- Simple
- Straight forward
- Best for projects where requirements are well understood

DISADVANTAGES

- Requirements should be given first . can’t add requirements in between development
- It follow big bang theory. Software is delivered in one shot . It have great risk
- Formal documents are needed at end of each phase

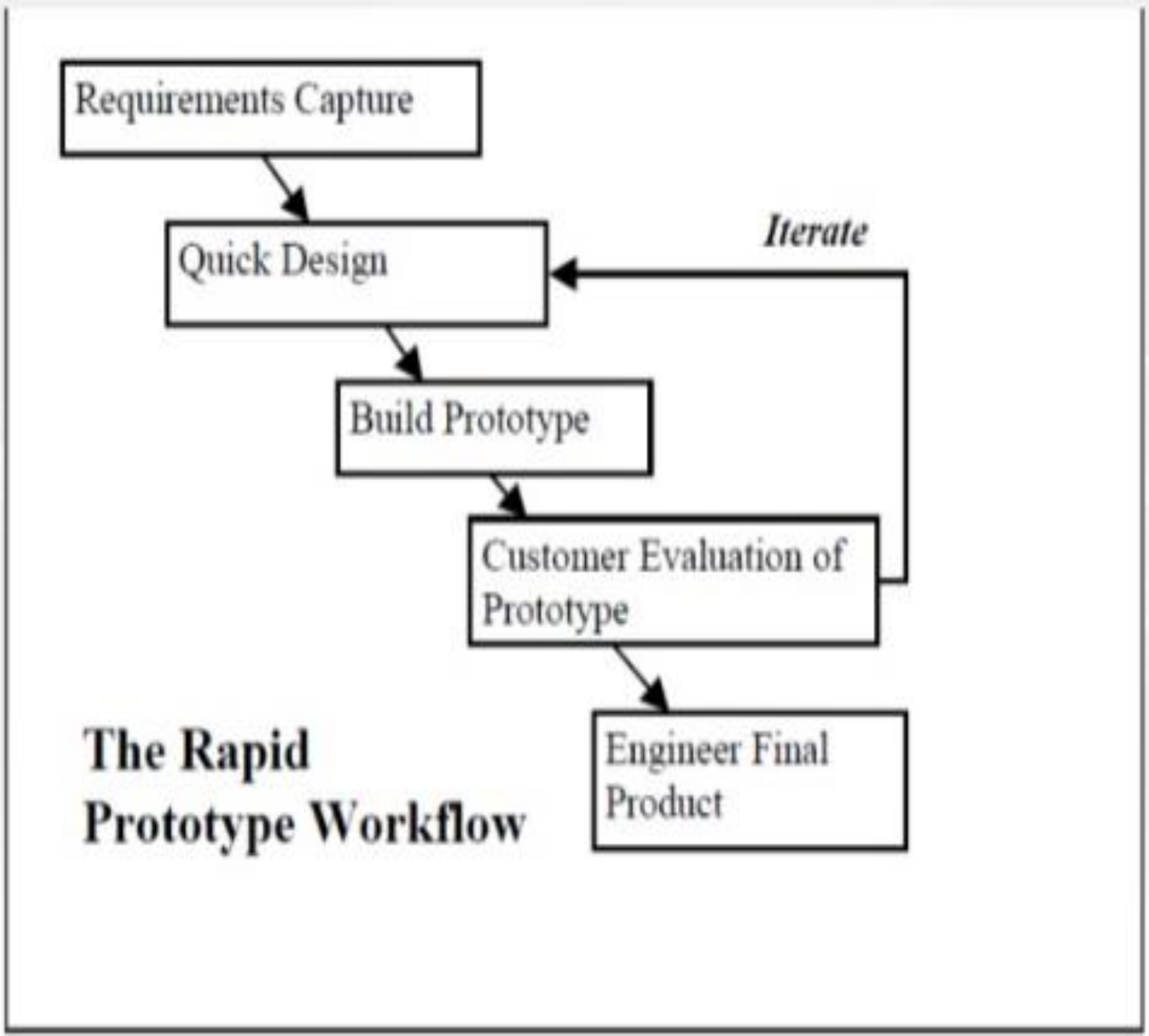
PROTOTYPE MODEL

- A prototype is made with known requirements and remaining phases are done informally.

- After this the prototype is provided to client and according to their suggestions changes are made

- Once the requirements are obtained, prototype is discarded

- It is suitable for projects which has no clear idea about requirements at first



The Rapid Prototype Workflow

ADVANTAGES

- Requirements need not given first
- Minimal documentation is needed

DISADVANTAGES

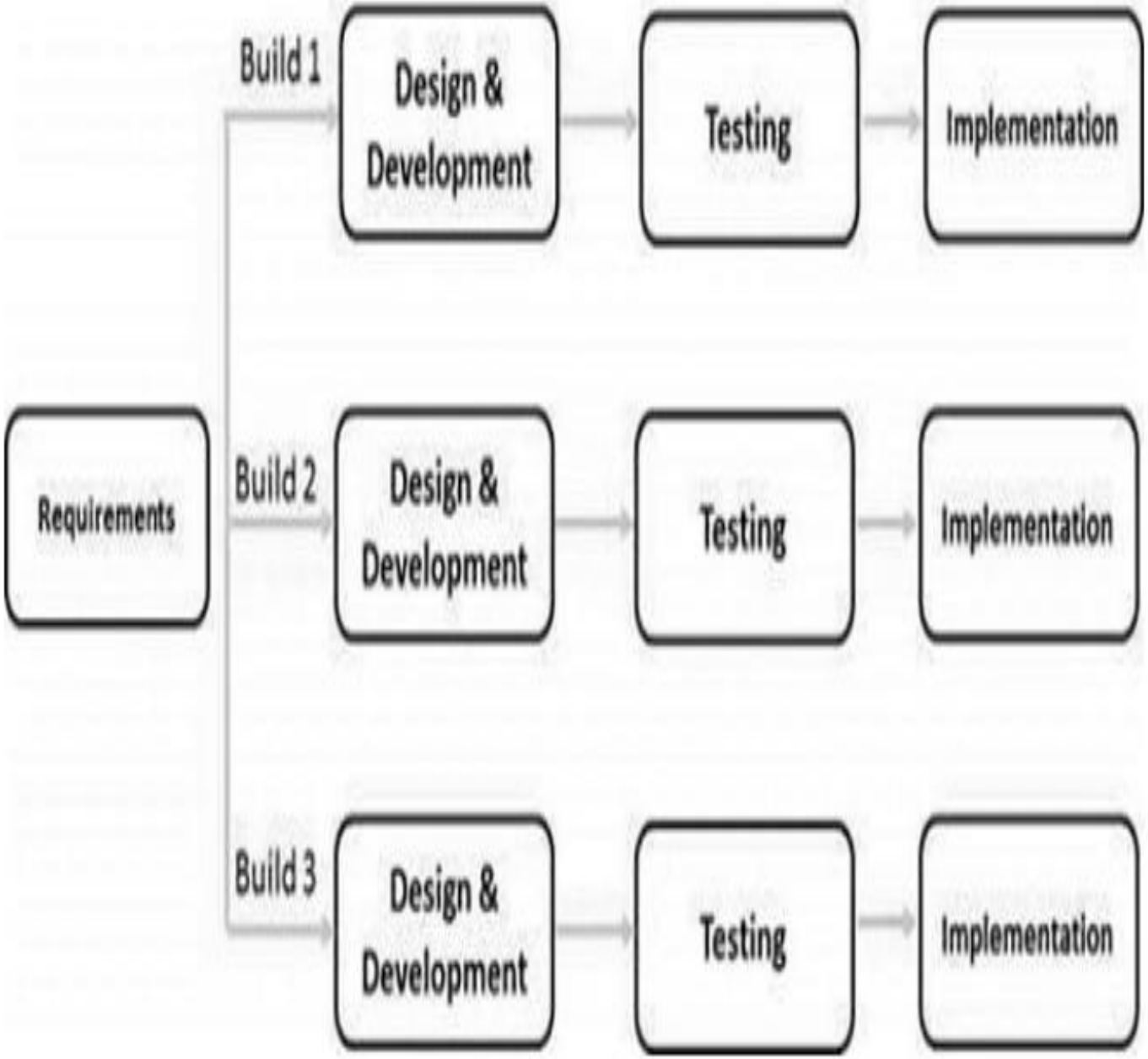
- Cost is high
- Quick and dirty method. Because it focus on quick development rather than quality

RAPID APPLICATION DEVELOPMENT(RAD)

- In this it is not a prototype that is built but actual product itself.
- The built application is not discarded
- CASE(computer aided software engineering) tools are used through out the life cycle

ITERATIVE MODEL

- Software is developed in increments, each increment adding some functional capabilities to system


Requirements
Build 1
Design & Development
Testing
Implementation
Build 2
Design & Development
Testing
Implementation
Build 3
Design & Development
Testing
Implementation

ADVANTAGES

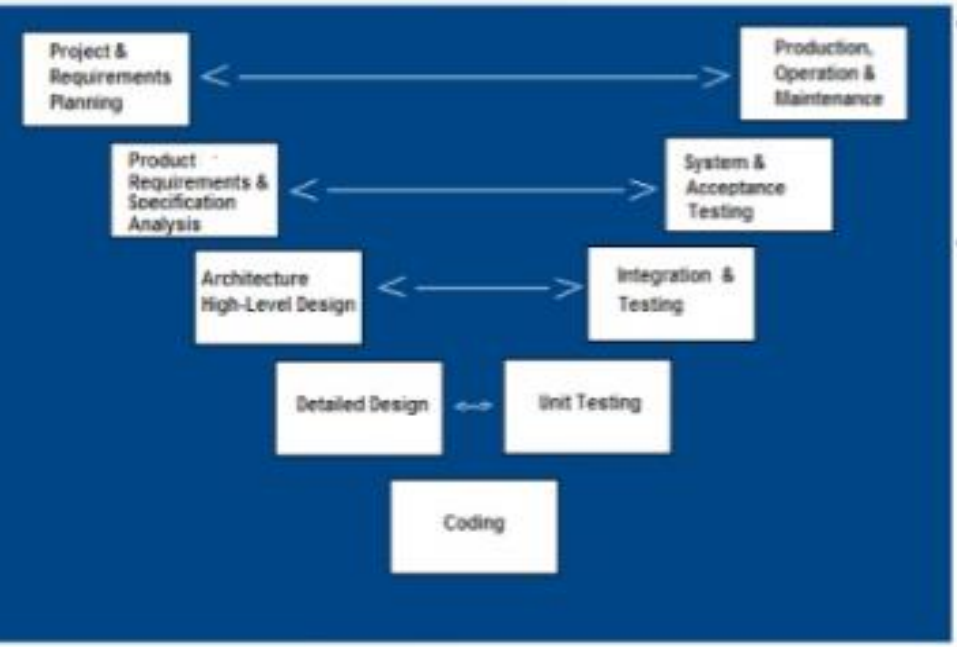
- Feedback path– in each phase corrections are made and reflected in later phase
- Each release delivers an operational product

DISADVANTAGES

- Not suitable for small projects
- Requirements should be given first
- Limited customer interaction

V MODEL

V-Shaped SDLC Model



- A variant of the Waterfall that emphasizes the verification and validation of the product.
- Testing of the product is planned in parallel with a corresponding phase of development

V-Shaped Steps

- Project and Requirements Planning - allocate resources
- Product Requirements and Specification Analysis - complete specification of the software system
- Architecture or High-Level Design - defines how software functions fulfill the design
- Detailed Design - develop algorithms for each architectural component
- Production, operation and maintenance - provide for enhancement and corrections
- System and acceptance testing - check the entire software system in its environment
- Integration and Testing - check that modules interconnect correctly
- Unit testing - check that each module acts as expected
- Coding - transform algorithms into software

- It executes phases in sequential manner in V shape
- For each phase there is a testing activity corresponding to it

- It involves static analysis technique (review)done without executing code and dynamic analysis technique done by executing code

- early design of tests enables better validation of individual phases

ADVANTAGES

- Emphasize planning for verification and validation of the product in early stages of product development
- Each deliverable must be testable
- Project management can track progress by milestones
- Easy to use

DISADVANTAGES

- Does not easily handle concurrent events
- Does not handle iterations or phases
- Does not easily handle dynamic changes in requirements
- Does not contain risk analysis activities

When to use the V-Shaped Model

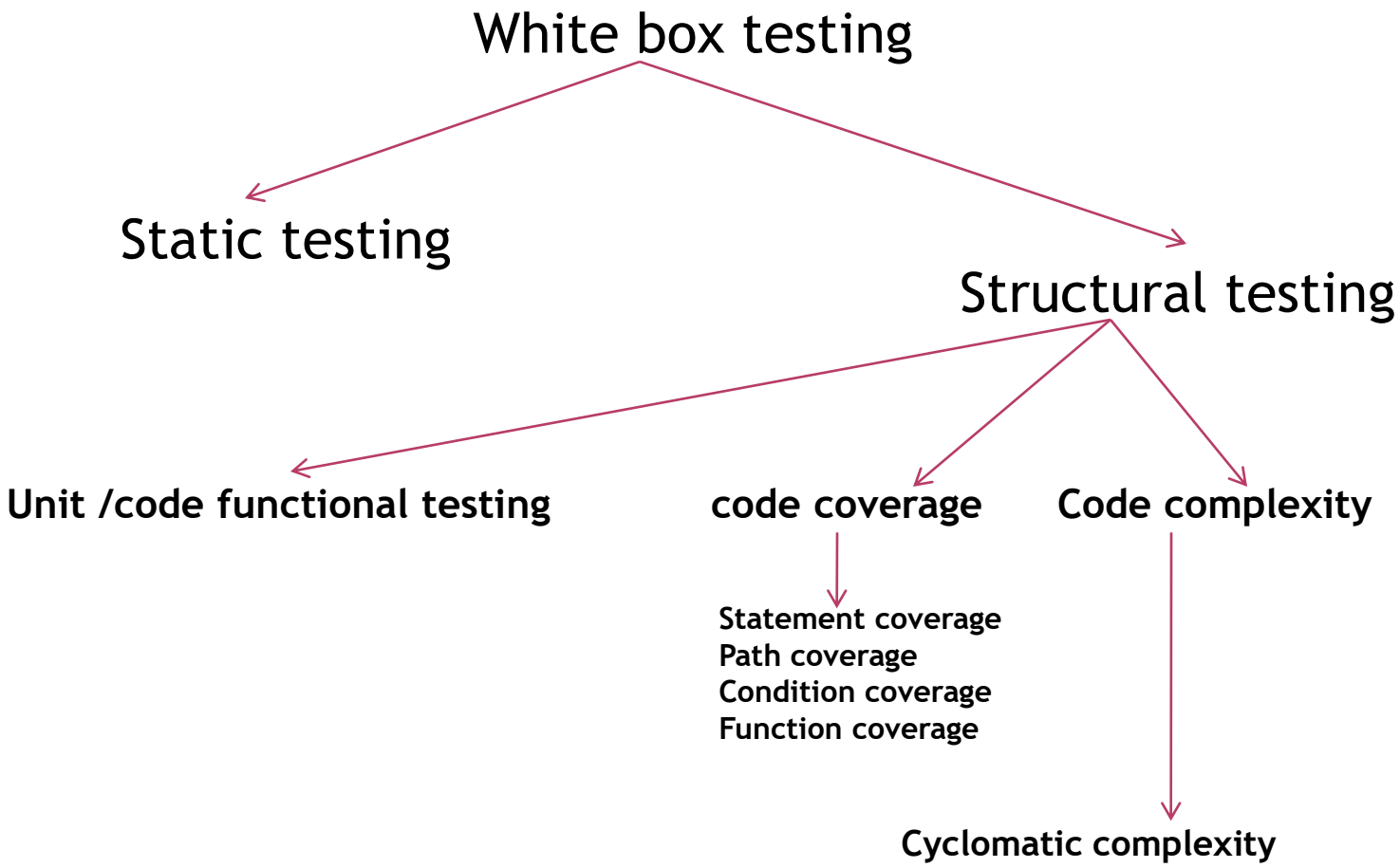
- Excellent choice for systems requiring high reliability - hospital patient control applications
- All requirements are known up-front
- Solution and technology are known

MODIFIED V MODEL

- It bring parallelism in different parts of product
- When one module satisfy a given phase of testing , can move to next phase of testing , without waiting for all modules to move in from one phase of testing to another

WHITE BOX TESTING

- It is a way of testing external functionality of code by examing and testing program code
- It takes into account the program code, code structure and internal design flow
- It is also known as clear box, glass box or open box


White box testing
Static testing
Structural testing
Unit /code functional testing
code coverage
Code complexity
Statement coverage
Path coverage
Condition coverage
Function coverage
Cyclomatic complexity

STATIC TESTING

- This type of testing requires only source code of product not the binaries or executables
- It does not involve executing the programs on computer but involve select people going through code
- Static testing by humans: Humans read the program code to detect error rather than computers executing the code to find errors

DESK CHECKING OF CODE

- It is done by author of code
- It is done by comparing code with design to make sure that code does what it is supposed to do
- There is no process or structure that verify the effectiveness of desk checking

ADVANTAGES

- The programmer who knows the code well is well equipped to understand his own code
- There are fewer scheduling and logistics overhead
- Defects are detected and corrected with minimum time delay

DISADVANTAGES

- Developer is not the best person to detect problems in his own code
- Developers generally prefer to write new code rather than do any form of testing
- It is person dependent and informal

CODE WALKTHROUGH

- These are group oriented and less formal than inspection method
- It bring multiple perspective
- A set of people look at program code and raise questions .Author explains logic and answer the questions. if the author is unable to answer he/she take those questions and find answers

FORMAL INSPECTION

- This method is to detect all faults,violations and other side effect
- It has high degree of formalism
- There are 4 roles in inspection
- ***Author***:
- ***A moderator*** who is expected to formally run the inspection according to process
- ***Inspectors*** are the people who actually provides review comments for code
- ***Scribe*** takes detailed notes during the inspection meeting and circulates them to inspection team after meeting

DISADVANTAGES

- It is time consuming
- The logistics and shedulings can become an issue
- It is not always possible to go through every lines of code

STRUCTURAL TESTING

- It is actually run by the computer on the built product
- It takes into account code structure, internal design and how they are coded

UNIT/CODE FUNCTIONAL TESTING

- It involves quick test that checks out any obvious mistakes .This is done prior to formal reviews of static testing so that review mechanism does not waste time
- Initial test can be done either by running the product under a debugger or IDE by building a debug version of product

CODE COVERAGE TESTING

- It involves designing and executing test cases and finding out percentage of code that is covered by testing
- Percentage of code covered by testing can be found by technique called ‘instrumentation of code’
- Instrumented code can monitor and keep audit of what portions of code are covered

Uses of code coverage technique

- Performance analysis and optimization
- Resource usage analysis
- Checking of critical sections or concurrency related parts of code
- Identifying memory leaks
- Dynamically generated code

Different types of coverage are

- Statement coverage
- Path coverage
- Condition coverage
- Function coverage

STATEMENT COVERAGE

- It is a technique in which all the executable statements in the source code are executed atleast once
- It is used for calculation of number of statements in the source code which have been executed
- Statement coverage = $\frac{\text{Number of executed statements}}{\text{Total number of statements}} * 100$

PATH COVERAGE

- It test all paths of the program
- This is a technique which ensures that all the paths of the program are traversed atleast once
- Path coverage = $\frac{\text{Total paths exercised}}{\text{Total paths in program}} * 100$

CONDITION COVERAGE

- It cover all the possible outcomes(true and false) of each condition of decision point at least once

- Condition coverage = $\frac{\text{Total decisions exercised}}{\text{Total number of decisions in the program}} * 100$

FUNCTION COVERAGE

- This technique cover all functions in a program
- It is easier to achieve 100 percent function coverage than 100 percent coverage in any of earlier methods
- Function coverage gives more focus on functions which are frequently called and hence it help in improving the performance and quality of the product
- Function coverage = $\frac{\text{Total functions exercised}}{\text{Total number of functions in the program}} * 100$

CODE COMPLEXITY TESTING

- This testing finds the complexity of code
- '***Cyclomatric complexity***' is a metric that quantifies the program's complexity
- A program is represented in the form of flow graph

To convert a flow chart to flow graph following steps are done

- Identify predicates or decision points in program
- Ensure that predicates are simple
- Combine all sequential statements into a single node
- When a set of sequential statements are followed by a single predicate, combine all sequential statements and predicate into one node and have 2 edges emanating from this one node . such nodes are called ‘predicate nodes’
- Make sure all nodes terminate at some node

- Cyclomatic complexity=E-N+2(where E=edges,N=node)
- Cyclomatic complexity=P+1(where P=predicate)

CHALLENGES IN WHITE BOX TESTING

- It requires knowledge about program code and programming language
- Human tendency of a developer being unable to find the defects in his or her code
- Fully tested code may not correspond to realistic scenarios