IT Bootcamp

Introduction to Software Testing

Miroslav Lončarević

Beograd, septembar 2018

Sadržaj

- Osnove testiranja
- Manuelno testiranje
- JIRA
- Database i SQL

- Dva testa
  - Osnove testiranja i SQL
  - Praktičan rad iz manuelnog testiranja

Software Development Life Cycle (SDLC)

- Šta je software?
- Koji su tipovi software-a?
  Desktop (Windows, MacOS, Linux), Mobile (Android, iOS, Windows), Web
- Kako izgleda životni ciklus software-a?
  - Scope identification
  - Plan
  - Analysis
  - Design
  - Development
  - Testing
  - Deployment
  - UAT – User Acceptance Testing

Software Development Life Cycle (SDLC)

Phases:

	What?	Who?	DOC?
❖ Scope identification	Business requirements	BA	BRD: Business Requirement Doc
❖ Plan	schedule, Budget	PM	Project Plan
❖ Analysis	Specifications	BA	FSD: Functional Specification doc
❖ Design	Programming logic	Architect	Design doc
❖ Development	Programming	Developers	Programs/source code
❖ Testing	defects, requirements	QA	Test docs
❖ Deployment	release	Developers, System admin	Installation Manual.
❖ UAT	user testing	Users, BA, QA	User Manual

Software QA Testing and related terminologies

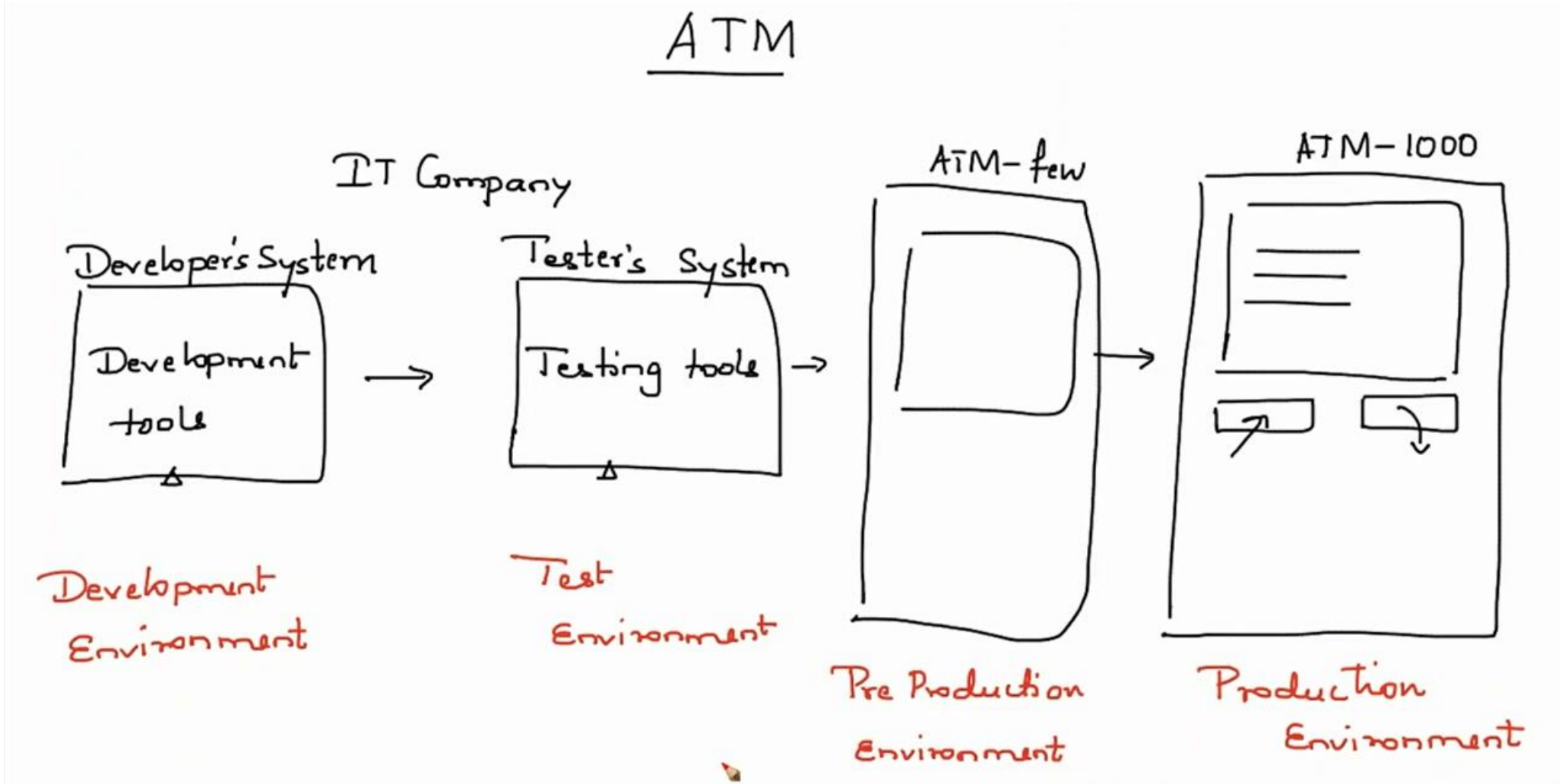
- Šta je testiranje?
  - Proces kojim se utvrdjuje da li je predmet testiranja u skladu sa zahtevom
- Šta je testiranje software-a?
  - Proces kojim se utvrdjuje da li je software u skladu sa zahtevom klijenta
- Zašto je testiranje potrebno?
- Kada početi testiranje a kada završiti?
- Primeri grešaka prilikom programiranja
  - Mariner 1 raketa (1962)
  - Mars Climate Orbiter (1999)
  - AT&T (1998)
  - USSR (1983)

Software QA Testing and related terminologies

- Zašto SQA lakše dolazi do posla u odnosu na developera
- QA tim se sastoji od:

  QA menadžera, QA Team lead, QA Testeri
- Okruženja koja se koriste u SDLC:
  - Develpoment,
  - Test,
  - Pre-Production (Beta, UAT),
  - Production

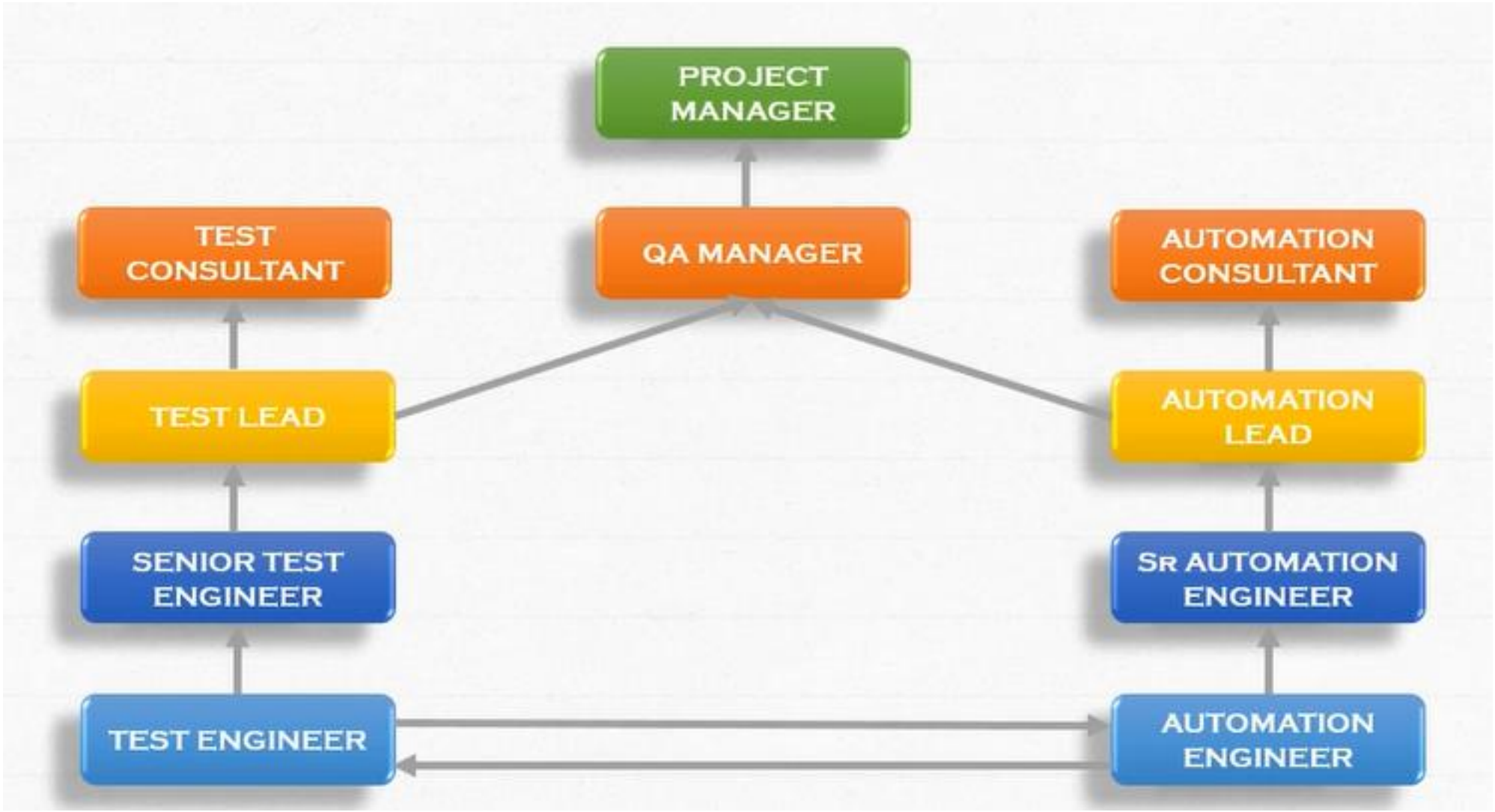
Software QA Testing and related terminologies



Tester roles

- Functional Tester
- Automation Tester
- Performance Tester
- Mobile Tester
- Pen tester – Security tester
- Test Architect

Career Paths in SQA



Software Testing Principles

1. Testing shows presence of defects, but cannot prove the absence of any defect.
2. Exhaustive testing is impossible
3. Early Testing
4. Defect Clustering
5. Pesticide paradox
6. Testing is context dependant
7. Absence – of – errors fallacy

Myths about Software Testing

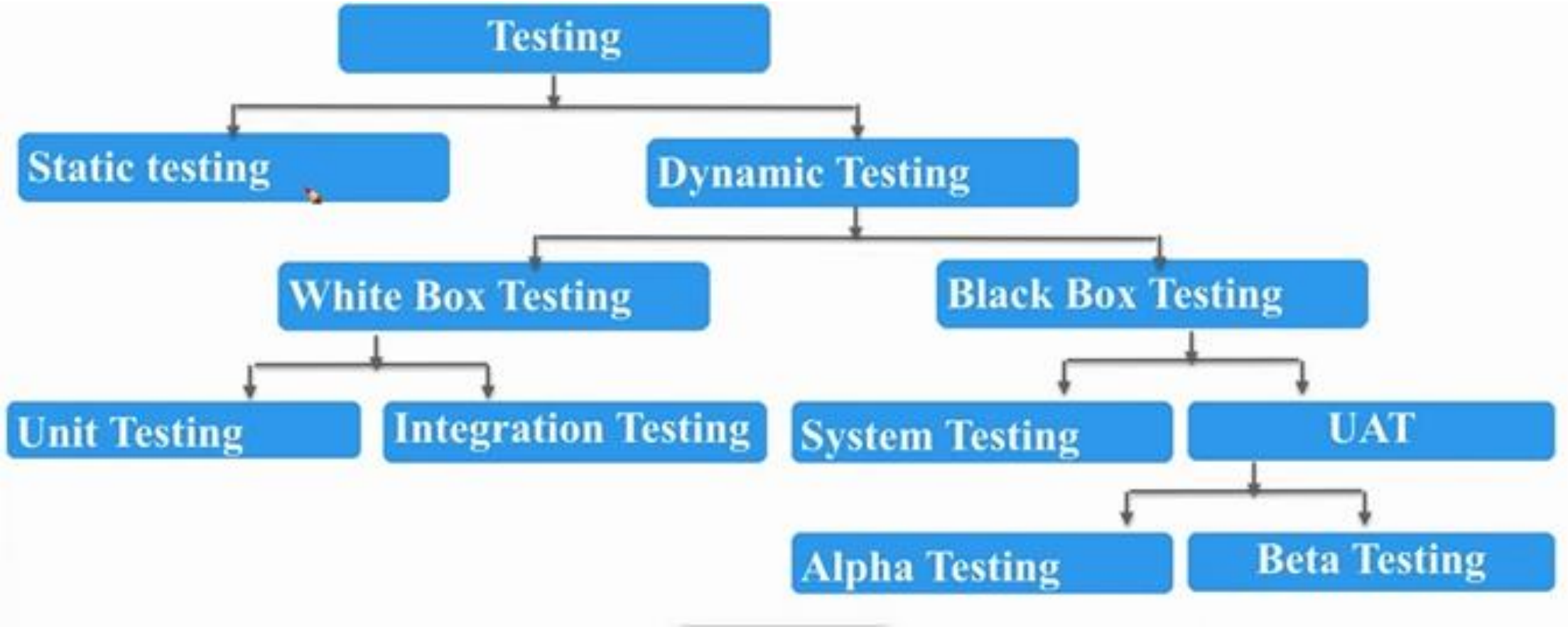
- Tačno ili ne
  - Testiranje je preskupo?
  - Testiranje nije potrebno dok se proizvod potpuno ne napravi?
  - Tesitiranje oduzima mnogo vremena?
  - Kompletno testiranje software-a je moguće?
  - Posao testera je samo da traži bugove?
  - Za neotkrivene bugove tokom testiranja krivica je na testeru?
  - Samo testeri su odgovorni za kvalitet proizvoda?
  - Ako je software testiran, znači da je 100% bug free?
  - Automatsko testiranje će zameniti manuelno testiranje?
  - Software može da testira bilo ko?

Myths about Software Testing

Dev said: It only visual update, nothing wrong will happen

Software QA Testing and related terminologies

- Forms of testing



Software QA Testing and related terminologies

Static Testing:

QA reviews the project documents to identify the errors

Static Testing Techniques:

- Informal Review
- Formal/ Inspection
- Walkthrough

Software QA Testing and related terminologies

- Informal
  - No formal process
  - Inexpensive
  - Results may be documented
  - Reviewer dependent
- Formal
  - Test lead -> Moderator
  - Formal process
  - Planned & controlled by moderator
  - Roles & documents to be reviewed assigned to QA
  - Based on rules & checklist
  - Includes metric gathering
  - Entry & exit criteria specified
  - Pre meeting preparation
  - Review Meeting- Scribe makes meeting notes
  - Review Report submitted

Walkthrough

- Led by author
- Conducted to gain knowledge and provide project information
- And also find defects

Forms of testing

WHAT IS STATIC TESTING?

- Testing without running the AUT
- Goal is to perform Sanity Check
- Not a Detailed Test
- Can be done by Developers
- Easy to fix Bugs

TYPES OF STATIC TESTING

- Walkthroughs
- Code Reviews (Peer Reviews)
- Inspections

ANALYSIS BY TOOLS

- Syntax violations
- Undefined and unused Variables.
- Unreachable or Dead Code
- Security Violations
- Standards Violations

Forms of testing – Static testing

WALKTHROUGHS

- Detect Defects
- Improve Software Product
- Alternate implementations
- Conformance to standards
- Usability and Accessibility of Software Product

WHAT ARE REVIEWED?

- Software requirement specification
- Design Description
- Test plans
- User documentation, Maintenance manuals
- Installation procedures, Release notes, Licenses

INPUTS

- Software Product
- Statement of Objectives
- Standards, Regulations, Procedures

OUTPUTS

- Walkthrough Report

Forms of testing – Static testing

FORMAL CODE REVIEW

- Detailed process
- Multiple participants and phases
- Series of meetings
- Exhaustive review
- Effective in finding defects

LIGHT WEIGHT CODE REVIEW

- Over the shoulder
- Email thread
- Pair-programming
- Tool assisted review

Forms of testing – Static testing

INSPECTION

- Detect anomalies
- Specifications
- Quality attributes
- Conformance to standards
- Collect data
- Input to management decisions

WHAT ARE REVIEWED?

- Software requirement specification
- Source code
- Software user documentation, Maintenance manuals
- Installation procedures, Release notes
- Policies, Strategies
- Marketing documents

INPUTS

- Statement of objectives
- Software product
- Inspection procedure, Checklists
- Quality criteria
- Predecessor software

OUTPUTS

- Inspection report
- Anomaly summary
- Estimate of rework effort
- Estimate of the savings

Software QA Testing and related terminologies

Dynamic Testing:

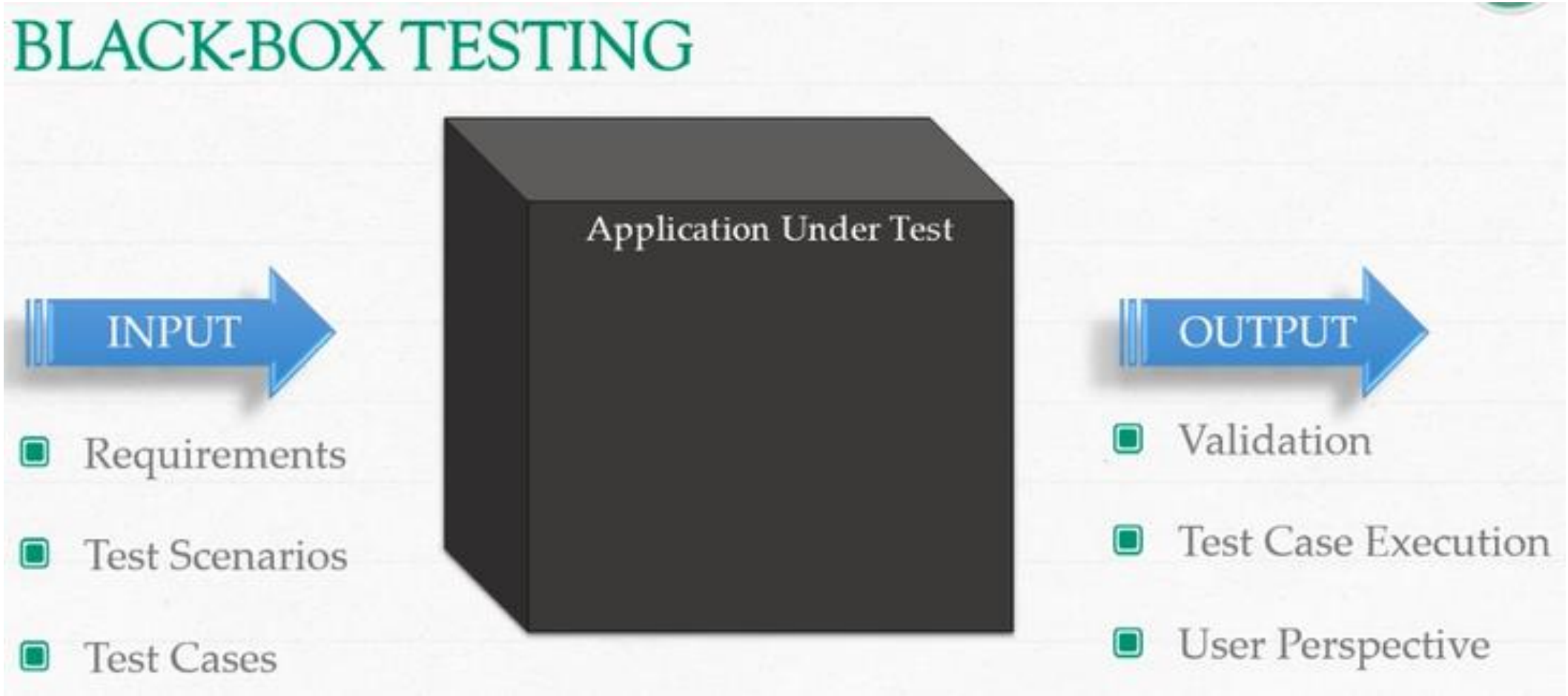
Software is tested by executing it.

Dynamic testing levels:

-White box testing- Unit / Component testing, Integration testing

-Black box testing - System testing & UAT

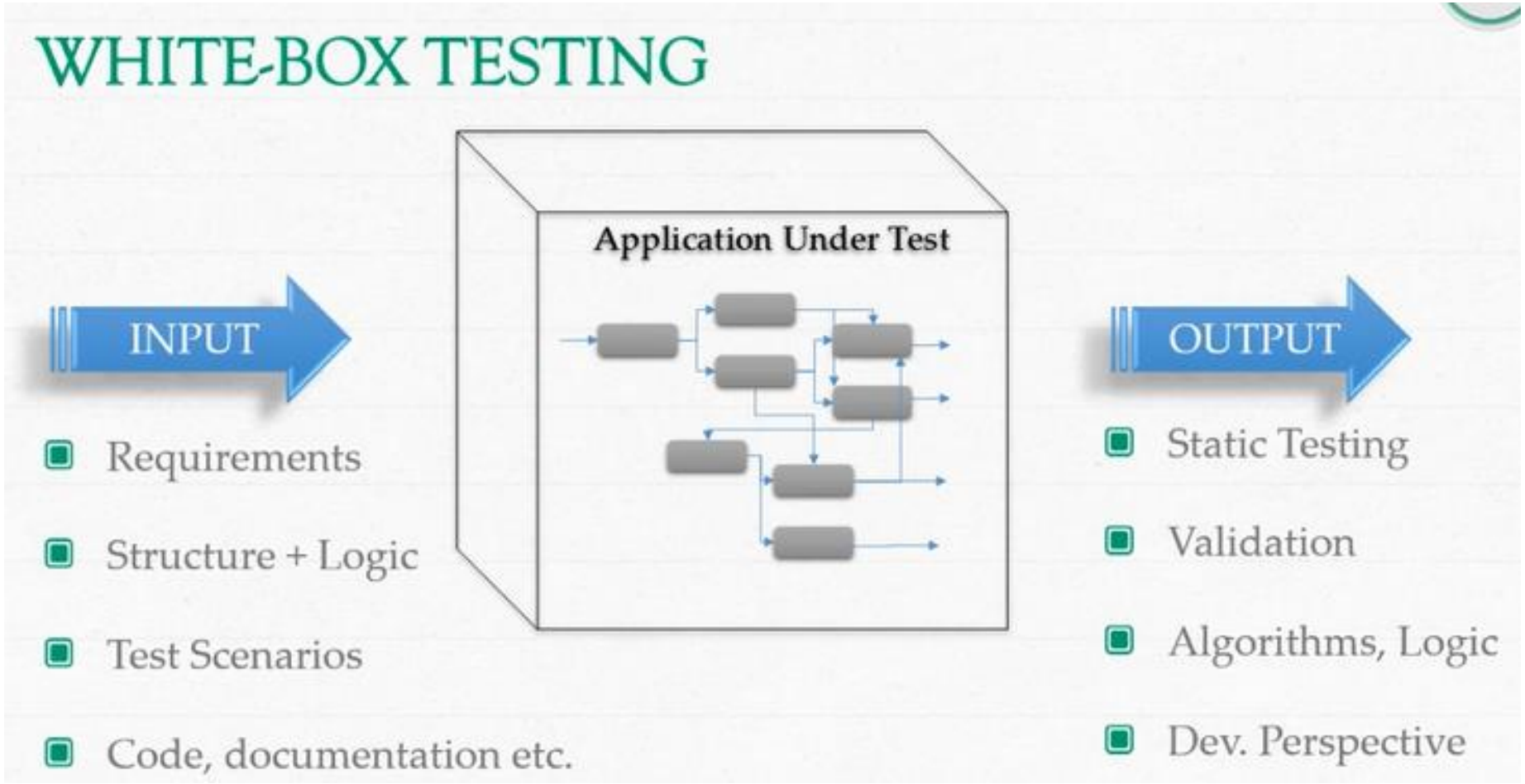
Forms of testing – Dynamic testing



Forms of testing – Dynamic testing

ADVANTAGES	DISADVANTAGES
Efficient for large code segments	Limited Coverage
Code Access is not required	Inefficient testing
Separates user's perspective from developer's perspective	Blind Coverage
Moderately skilled testers can also test	Test cases are difficult to design

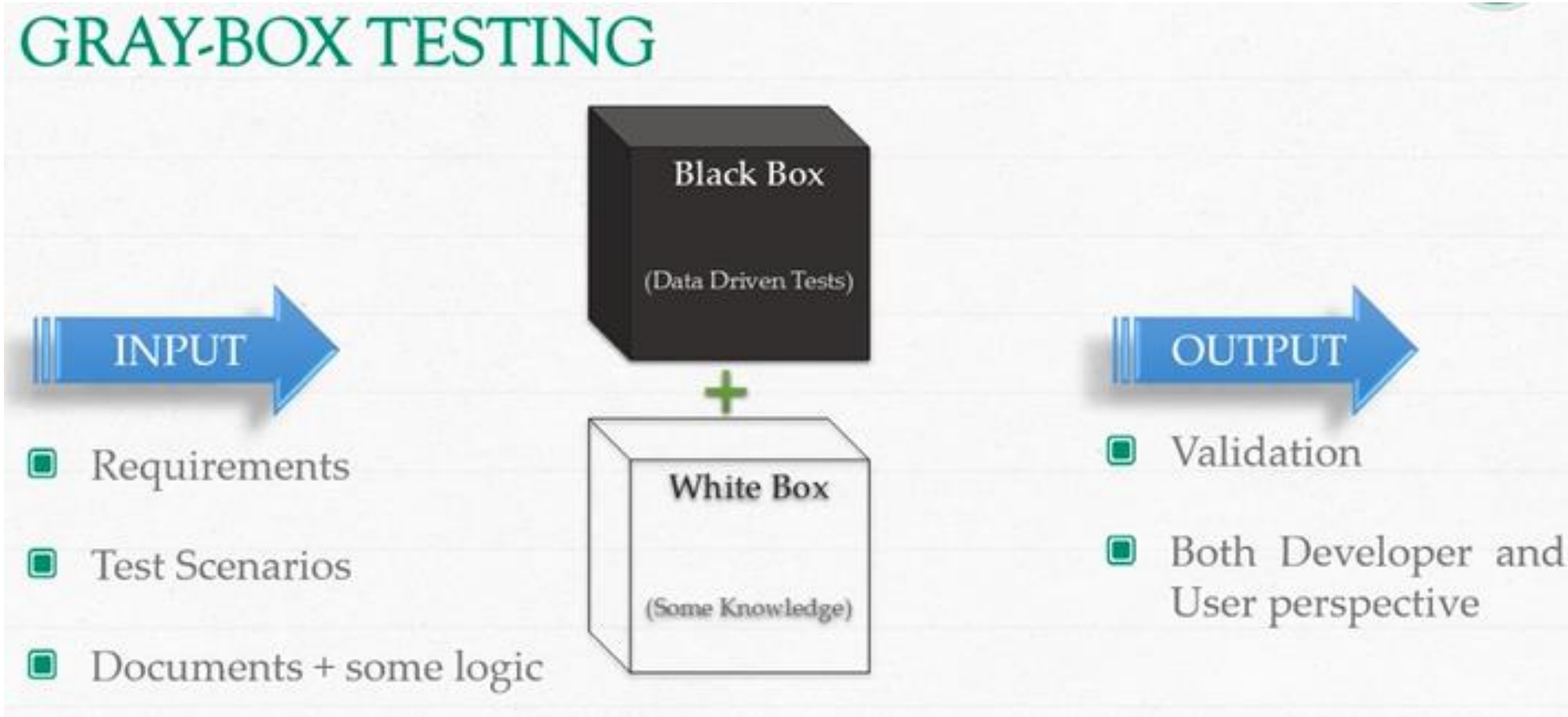
Forms of testing – Dynamic testing



Forms of testing – Dynamic testing

ADVANTAGES	DISADVANTAGES
Verification + Validation	Costs are relatively higher
Algorithms and logic can be checked	Exhaustive and Time consuming
Helps in optimization of the code	Maintaining whitebox testing is difficult
Maximum coverage	Skilled Testers are needed.

Forms of testing – Dynamic testing



Forms of testing – Dynamic testing

ADVANTAGES	DISADVANTAGES
Offers combined effects	Partial Code Coverage
Non Intrusive	Defect Identification is difficult
Intelligent Test Authoring	Not suited for Algorithm Testing
Unbiased Testing	Partly time consuming and exhaustive

Forms of testing – Dynamic testing

EQUIVALENCE PARTITIONING

- Randomly providing input and testing takes lot of time
- No surety about coverage of all valid and invalid input
- Divide test input data into equivalence classes

EQUIVALENCE CLASSES

- Software application responds similarly to every input in a class
- Test only one input in each partition
- If one condition passes then other conditions will also pass

EQUIVALENCE PARTITIONING

- Identify the equivalence classes
- Write the test cases to cover these classes
- Expected results for invalid partitions too

Example – Fixed number of valid values

Forms of testing – Dynamic testing

BOUNDARY VALUE ANALYSIS

- Refinement of Equivalence partitioning technique
- Range checking
- Values at the extreme ends of partition
- Boundaries are good place to look for defects

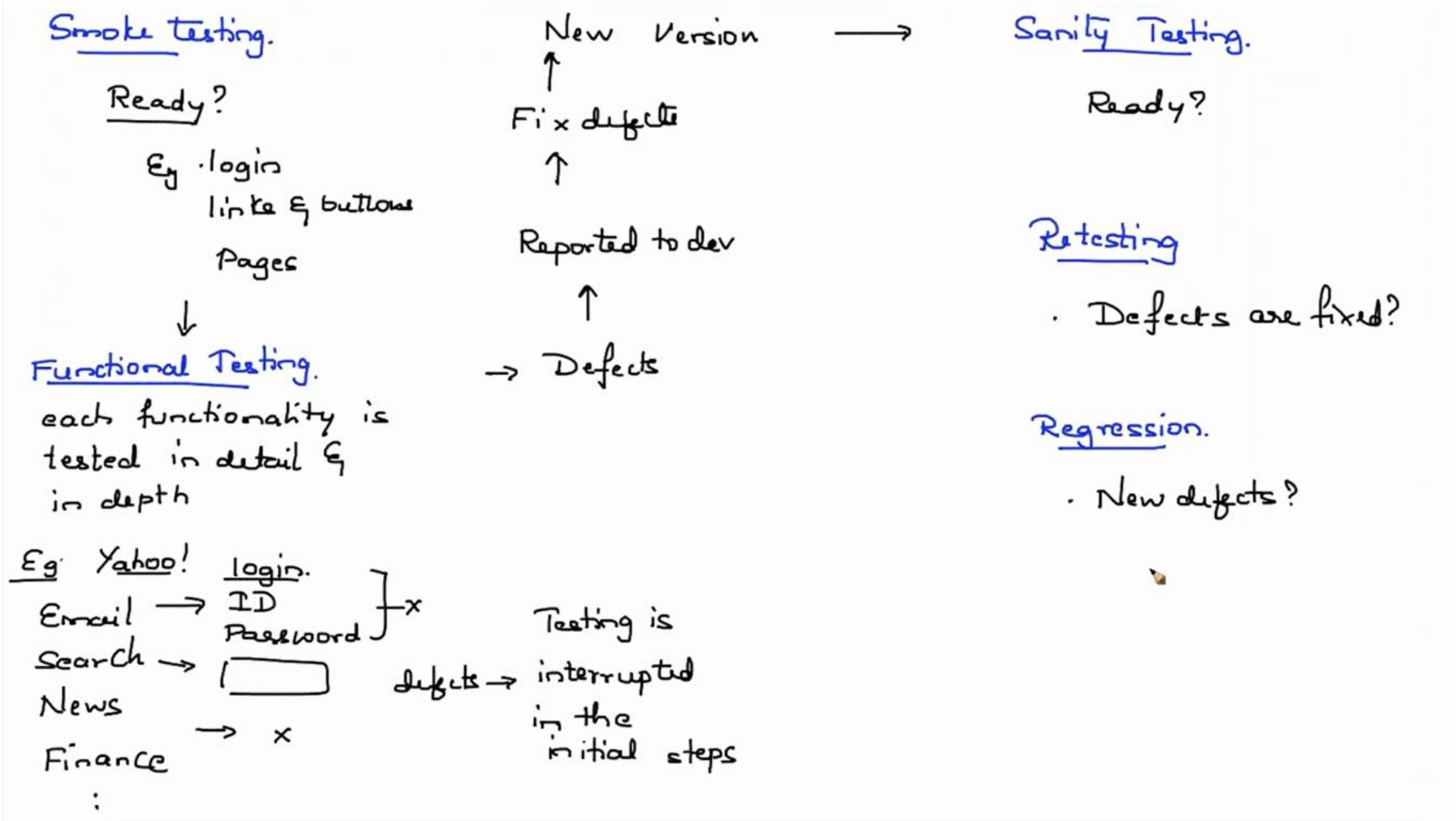
Example – Fixed number of valid values

Software QA Testing and related terminologies

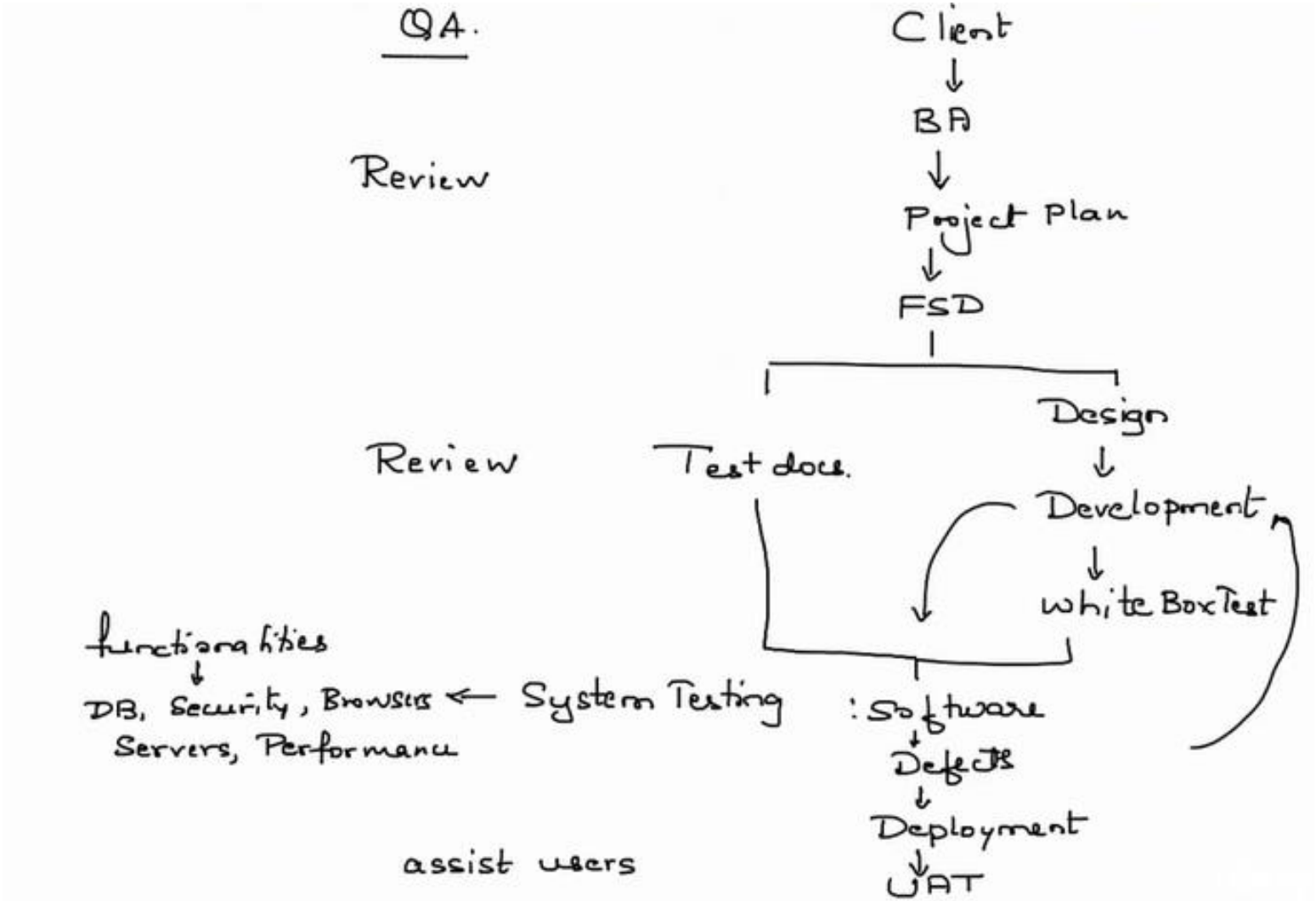
- Types of System Testing by QA

  - Exploratory testing
  - Initial level testing- Smoke testing
  - Functional testing
  - Initial level testing- Sanity testing
  - Retesting
  - Regression testing
  - Cross Browser Testing
  - External interface/ Intersystem testing
  - Security testing
  - High availability testing
  - Database Testing

Software QA Testing and related terminologies



Software QA Testing and related terminologies



Basic Terms and Definitions

- **Test plan (Test Plan)** - a document that describes the entire scope of work on testing, starting with a description of the object, strategy, timetables, criteria start and end of testing, to the required process of the equipment, expertise and risk assessment with options to resolve them .
- **Design Test (Test Design)** - this stage of the testing process, which are designed and created test cases (test cases), in accordance with certain criteria before quality and testing purposes.
- **Test case (Test Case)** - an artifact that describes a set of steps, the specific conditions and parameters necessary to verify the implementation of the test function or part of it:
  The test suite (Test suite) - Complete test kits for the test component or system, which is usually one postcondition test is used as a precondition for later.

Basic Terms and Definitions

- A **bug** is an error in a piece of code. It may not have any consequences, for example, it may trigger only in cases that do not occur in any product that uses that code.
- A **defect** is something wrong that causes unexpected behavior or makes something less suitable for its actual uses.
- A defect is often caused by a bug but not always. For example, if the specification says software should do something and the software does that, it's not a bug. But if that makes the software unsuitable for its intended use, it's a defect.
- A bug may cause a defect but not always. For example, if the specification says a certain error case should be handled a certain way and the code doesn't do that, it's a bug in that code. But if the project that contains that code does can be shown to be incapable of producing that case, it's not a defect in that project.
- **Bug / Defect MRC (Bug Report)** - a document that describes a situation or sequence of actions, which led to incorrect operation of the testing facility, stating the reasons and the expected result.

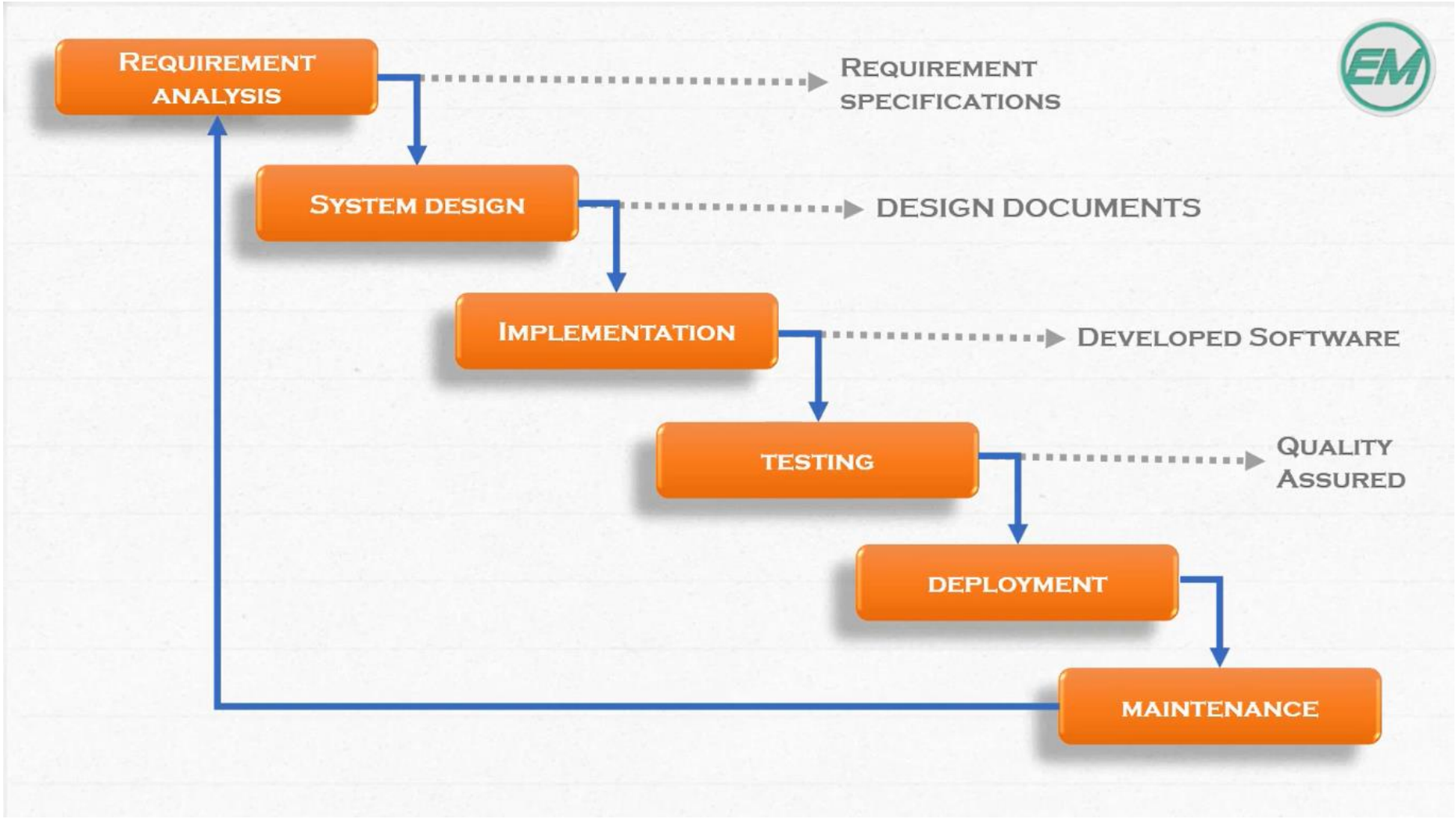
Requirement

- Horoskop
- Meteorološki podaci
- Dnevne novine
- Recepti
- Bioskop
- Akcijski katalozi
- Koncertna dvorana

- Minimum 7 - maksimum 10 zahteva za web aplikaciju
- Ukoliko je potrebno, ubaciti specifikaciju
- Pisati na engleskom
- Zabranjeno korišćenje računara i mobilnih telefona

Different ways of developing a Software

Waterfall

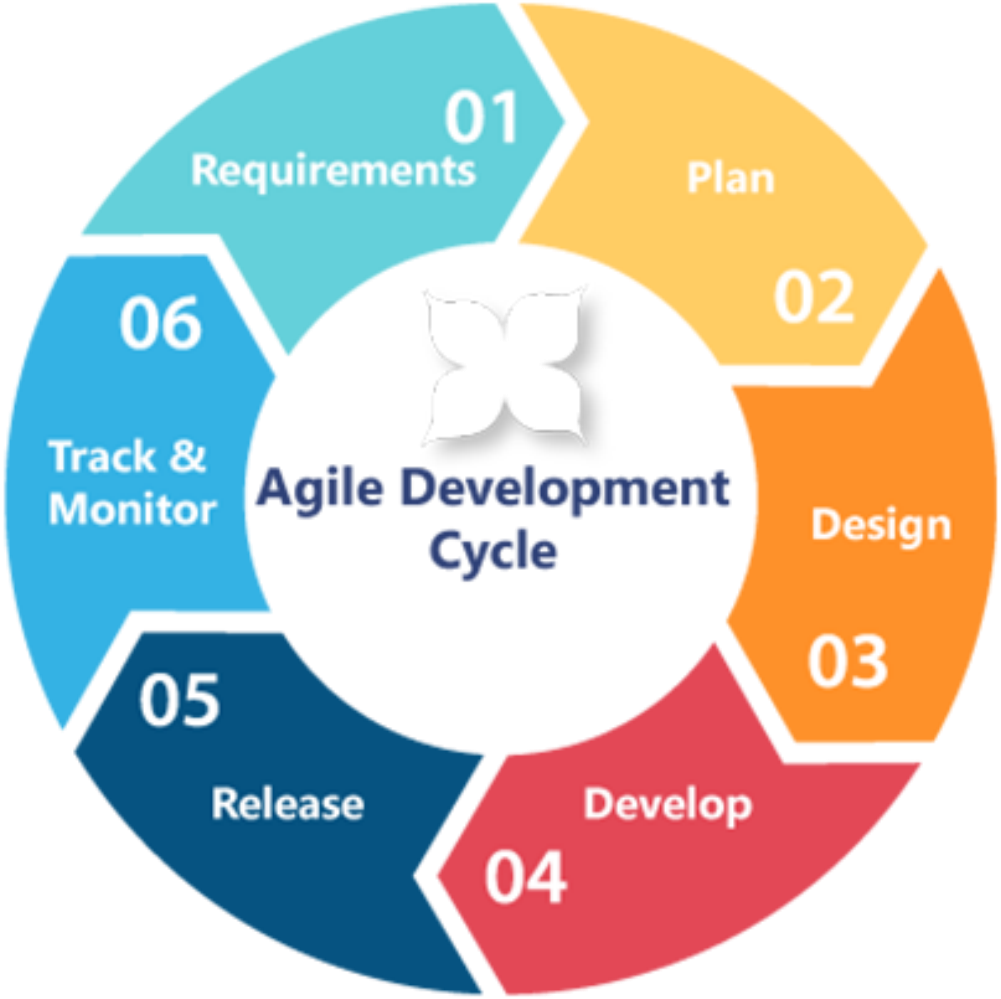


Different ways of developing a Software

ADVANTAGES	DISADVANTAGES
Simple & Easy to Understand	Difficult to go back and change something
Easy to Manage	No working software until late stages of SDLC
Phases don't Overlap	Risk and Uncertainty
Best for Small Projects	Not suitable for complex and object oriented projects

Different ways of developing a Software

- Agile

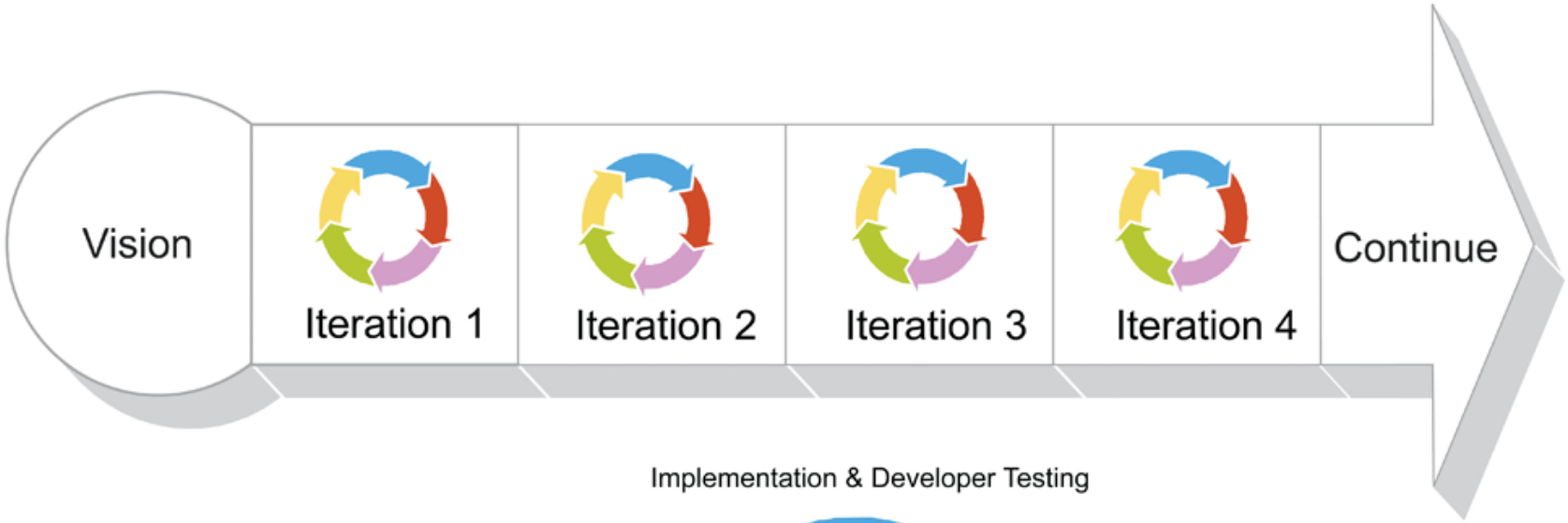


Different ways of developing a Software

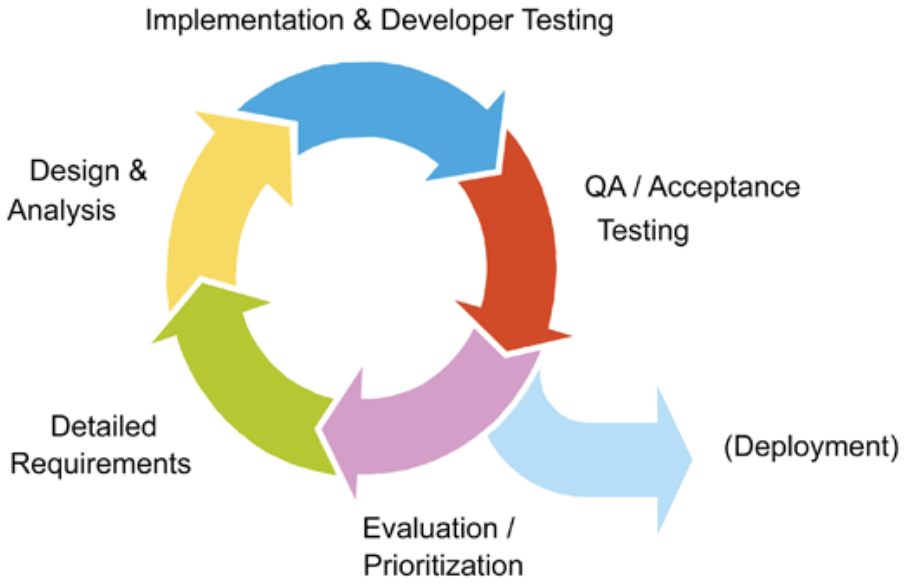
Advantages of Agile

- Agile understand all the requirements can't be collected in the beginning of the project.
- So, requirements could be changed or collected thru-out the development cycle.
- Customers are involved basically in throughout the development cycle.
- Agile doesn't need to much of doc; Focus on frequent deliver of the project.
- Better visibility and communication
- Test early and often.
- Higher quality
- Reduced Risk and earlier release of software.

Agile SCRUM methodology

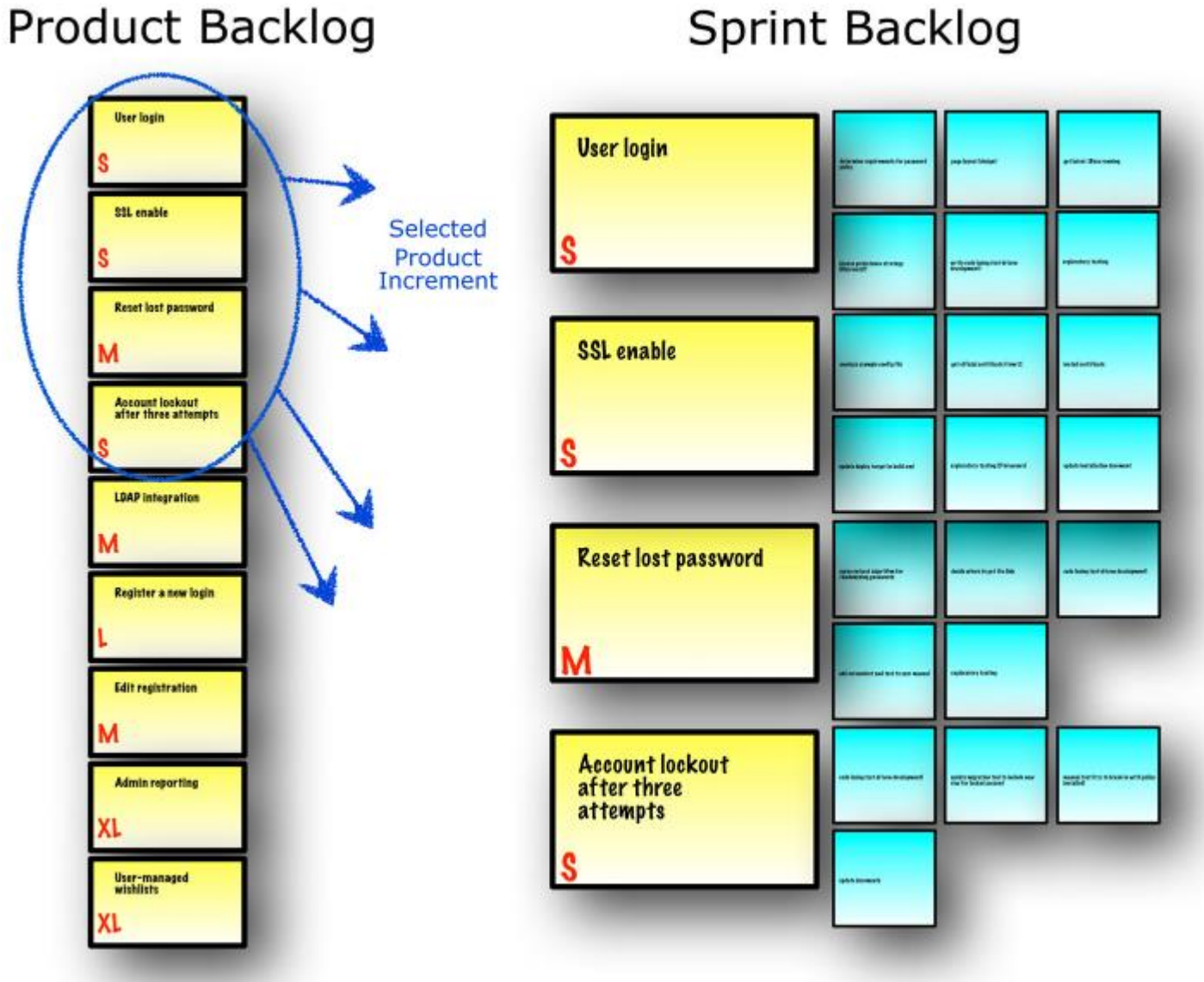


Iteration Detail



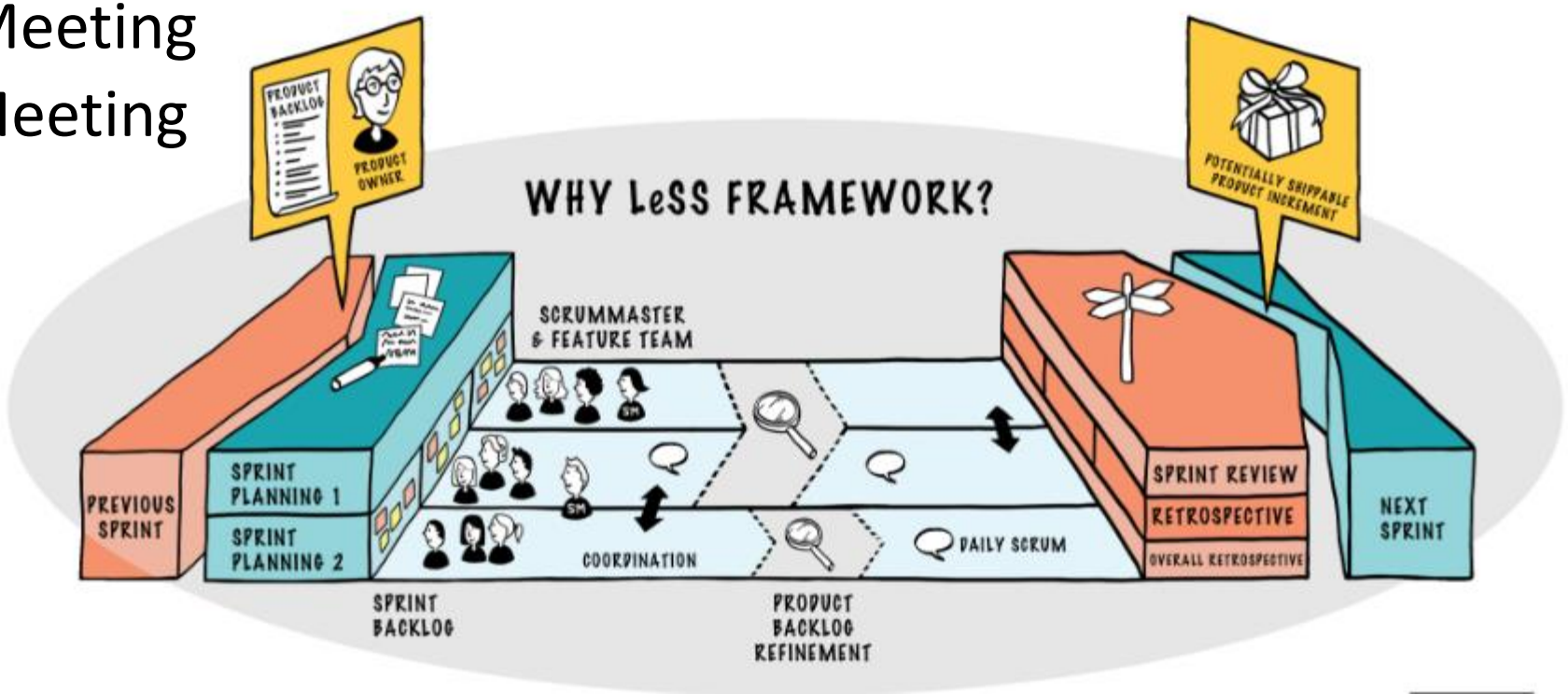
Agile SCRUM methodology

- **Scrum Artifacts**
  - Product Backlog
  - Product Backlog Item (PBI)
  - Sprint Backlog
  - Sprint Task
  - Sprint Burndown Chart
  - Product / Release Burndown Chart
- **SCRUM Roles**
  - Product Owner
  - Scrum Development Team
  - ScrumMaster



Agile SCRUM methodology

- **Scrum Meetings**
  - Sprint Planning Meeting
  - Daily Scrum and Sprint Execution
  - Sprint Review Meeting
  - Sprint Retrospective Meeting
  - Backlog Refinement Meeting



Behavior Driven Development (BDD)

1. **Test Driven Development (TDD)**—Tests are written before code and passing the tests is the critical driver of development.
2. **Acceptance Test Driven Development (ATDD)**—Team members with different perspectives (customer, business analyst, tester, developer) collaborate and write acceptance tests in advance of implementing the corresponding functionality.
3. **Behavior Driven Development (BDD)**—Tests are written in a non-technical language that everyone can understand (e.g. a domain-specific language like Gherkin). BDD combines the principles of TDD and ATDD and forms an approach for building a **shared understanding** on what kind of software to build **by discussing examples.**

Behavior Driven Development (BDD)

- BDD brings customers, end-users, BAs, QAs, and SEs of the software product into one table for effective sharing of knowledge on the system and its testing requirements.
- BDD consists of cycles of a set of steps to follow:
  - Identify business **feature**.
  - Identify **scenarios** under the selected feature.
  - Define **steps** for each scenario.
  - **Run** feature and **fail**.
  - Write code to **make steps pass**.
  - **Refactor** code, Create **reusable automation library**.
  - Run feature and **pass**.
  - Generate **test reports**.

```
As [role]
I want [feature]
So that [benefit/business reason]
```

```
Given Exact context
When Action/Event
Then Outcomes
And/But More of the same…
```

Verification vs Validation

VERIFICATION vs VALI

- Are we building the product in a right way?
- Meets Requirements
- Done by QA team
- Static activities
- Objective process

VALIDATION

- Are we building right product?
- Evaluate the software
- Done by testing team
- Subjective process

Fundamental Test Activities

TEST ACTIVITIES

- Test planning and control
- Test analysis and design
- Test implementation and execution
- Evaluate exit criteria
- Test closure

Fundamental Test Activities

1. **Planning and Control activities**
   - Concerned with planning activities that are part of the testing.
   - Understand the customer expectations, the goals of testing, and the risks associated with the project.
   - Create test plans documents, test strategy documents, risk analysis and mitigation plans etc.
   - Define the approach that has to be used to test the AUT
   - Define the schedule of the testing phase and the exit criteria for completing the test execution.
2. **Analysis and Design activities**
   - Focus will be on converting user expectations into test designs and test cases.
   - Analyse the goals and requirements of the testing phase which have been identified in the earlier activities
   - Identify the test conditions based on the analysis of test items, their specifications and our knowledge about the structure and behaviour of the system. From these test conditions, start designing test cases and test suites.

Fundamental Test Activities

3. The implementation and Execution activities
    - Running the designed test cases on the application under test and record the behaviour of the AUT.

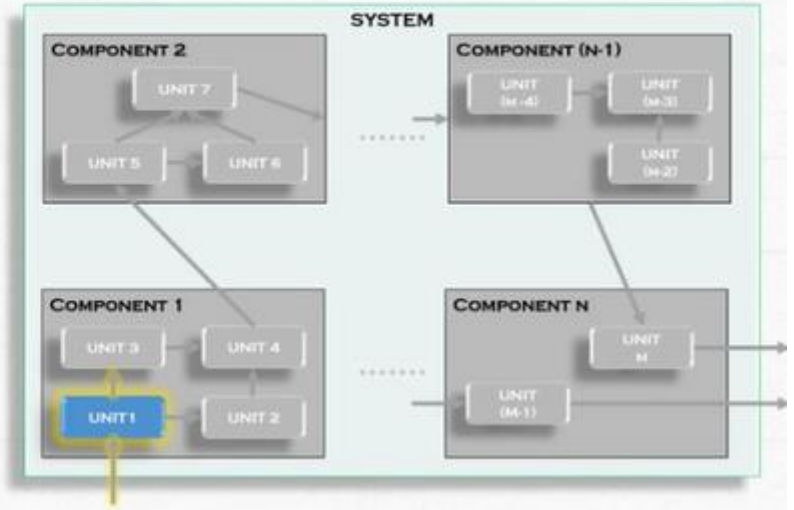
4. Stop testing based on Exit criteria
    - Stop testing and write a test summary report.

5. Test closure activities
    - Sign off the testing for the current project and collect data from the completed activities and use it for further projects.

Levels of Testing

UNIT

- Smallest individual module
- Program, Procedure, Method
- Discover discrepancies between specification and actual behavior
- Performed by developers
- Test a module in isolation
- Correct in terms of functionality and requirements



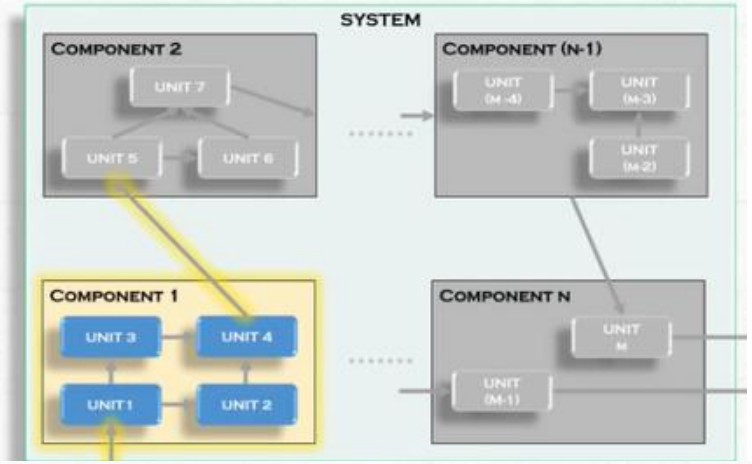
Levels of Testing

BENEFITS	LIMITATIONS
Finds problems early	Difficult to catch every bug
Facilitates change	Every execution path can't be evaluated
Simplifies Integration	Limit on number of test scenarios
Helps with documentation	

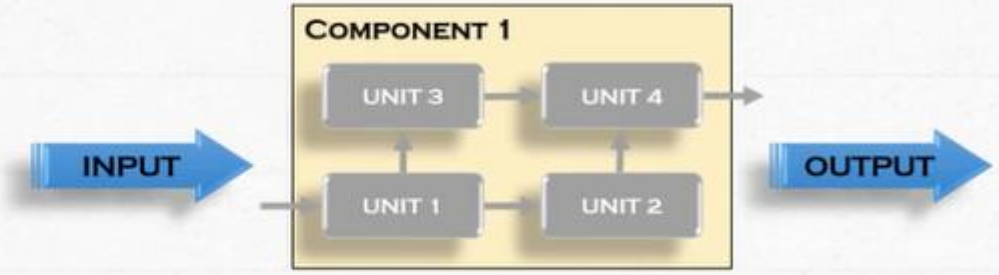
Levels of Testing

WHAT IS A COMPONENT?

- Set of logically related modules
- Performs one or more functionality
- Ensures modules work correctly when put together



- Performed by the testers
- Higher level of integration
- Can be done in isolation from rest of the system



Levels of Testing

Levels of Testing

Levels of Testing

SYSTEM TESTING

- End to end flows
- Test the system as a whole
- Customer requirements & Quality standards
- Specialized testing team

NEED FOR SYSTEM TESTING

- Functional & technical specifications
- Application architecture and business requirements are validated
- Creativity

TYPES OF SYSTEM TESTING

- Volume Testing
- Load/Stress Testing
- Security Testing
- Usability Testing
- Performance Testing
- Resource Usage Testing
- Recovery Testing
- Reliability Testing

Levels of Testing

ALPHA TESTING

- Form of Acceptance Testing
- Customized product
- Commercial product
- Features expected

WHO PERFORMS ALPHA TESTING

- Built under contract
  - Client
- Commercial Product
  - Independent test team

ALPHA TESTING

- Not just about fixing the bugs
- Developer ideas and process
- Understand the users' behaviour

Levels of Testing

BETA TESTING

- Beta – "a nearly complete prototype of a product"
- Done after Alpha testing
- Pre-release testing
- Sample of intended audience
- Feedback to the project team
- Typos, confusing flow, crashes
- Quality of final release - high

TYPES OF BETA TESTING

- Open beta test
- Closed beta test

STEPS FOR BETA TEST

- Time frame
- Type of beta test
- Mechanism to obtain feedback
- Analyse the feedback
- Release the changes
- Restart the beta test
- Deem the software as 'feature complete'