

# INTRODUCTION TO AGILE METHODOLOGY AND SOFTWARE TESTING

PMIT 6111 Software Testing & Quality Assurance

## SOFTWARE DEVELOPMENT PHASES

Requirement Analysis

Design

Coding

Testing

Maintenance

# WATERFALL VS AGILE

## PROJECT SUCCESS RATES AGILE VS WATERFALL


METHOD	SUCCESSFUL	CHALLENGED	FAILED
AGILE	42%	47%	11%
WATERFALL	13%	59%	28%

WWW.VITALITYCHICAGO.COM

Source: Standish Group Report 2020

# WHAT IS AGILE METHODOLOGY?

- The word 'agile' means –
  - Able to move your body quickly and easily.
  - Able to think quickly and clearly.
- **Agile** is an iterative development methodology, where both development and testing activities are concurrent.
- Testing is not a separate phase; Coding and Testing are done interactively and incrementally, resulting in quality end product, which the meets customer requirements. Further, continuous integration results in early defect removal and hence time, effort and cost savings.

The background of the slide is a painting of a group of people in a meeting. The painting is in a soft, painterly style with visible brushstrokes. It depicts several individuals in business attire, some standing and some seated, engaged in a discussion. The lighting is warm and somewhat diffused, creating a sense of a collaborative environment. The text is overlaid on this background, with the title at the top and several paragraphs of text in the center and bottom.

# Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.  
Through this work we have come to value:

**Individuals and interactions** over processes and tools

**Working software** over comprehensive documentation

**Customer collaboration** over contract negotiation

**Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

# CHARACTERISTICS OF AGILE SOFTWARE DEVELOPMENT

- Light Weighted methodology
- Small to medium sized teams
- vague and/or changing requirements
- vague and/or changing techniques
- Simple design
- Minimal system into production

## EXISTING AGILE METHODS

- Extreme Programming
- Scrum
- Crystal Methods
- Feature Driven Development
- Lean Development
- Dynamic Systems Development Methodology (DSDM)

## EXTREME PROGRAMMING -- XP

- Most prominent Agile Software development method
- Prescribes a set of daily stakeholder practices
- “Extreme” levels of practicing leads to more responsive software.
- Changes are more realistic, natural, inescapable.



## EXTREME...

Taking proven practices to the extreme

- If testing is good, let everybody test all the time
- If code reviews are good, review all the time
- If design is good, refactor all the time
- If integration testing is good, integrate all the time
- If simplicity is good, do the simplest thing that could possibly work
- If short iterations are good, make them really, really short

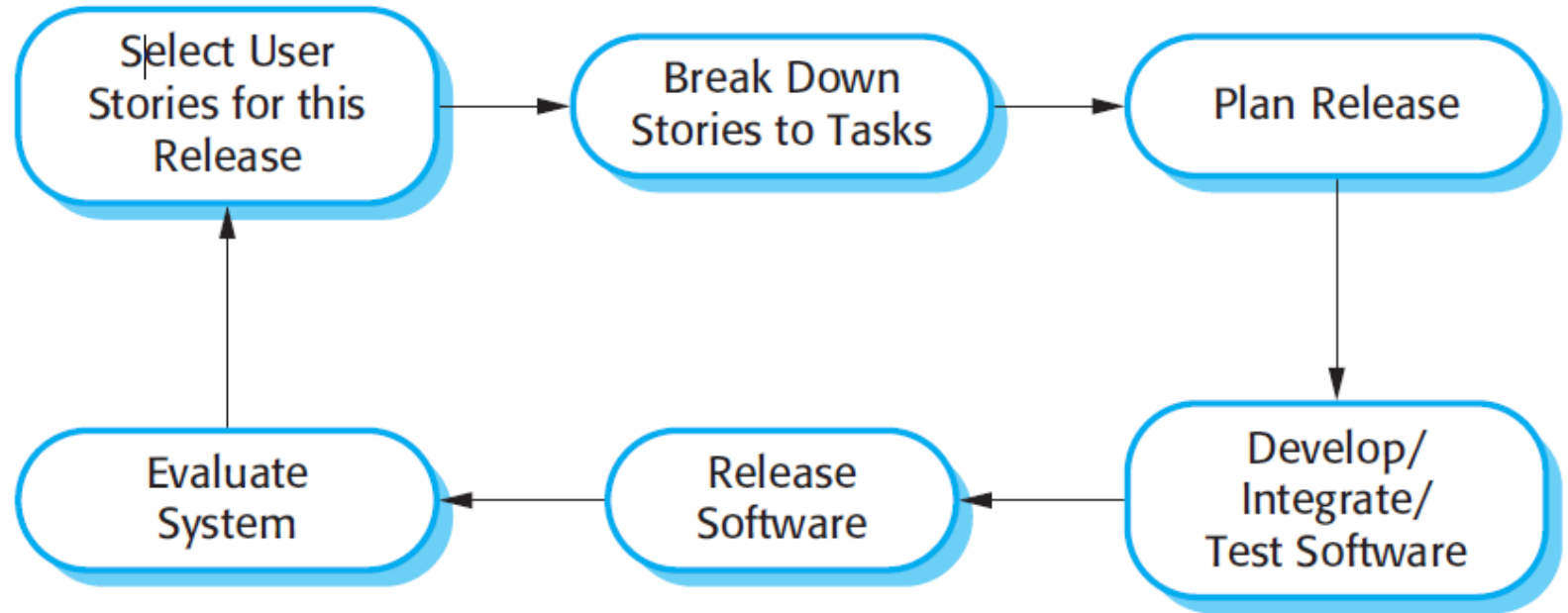
# EXTREME PROGRAMMING PRACTICES (A)

Principle or practice	Description
Incremental planning	Requirements are recorded on story cards and the stories to be included in a release are determined by the time available and their relative priority. The developers break these stories into development 'Tasks'. (As example we we'll see Figures 3.5 and 3.6 later.)
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

## EXTREME PROGRAMMING PRACTICES (B)

Principle or practice	Description
Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

# XP RELEASE CYCLE

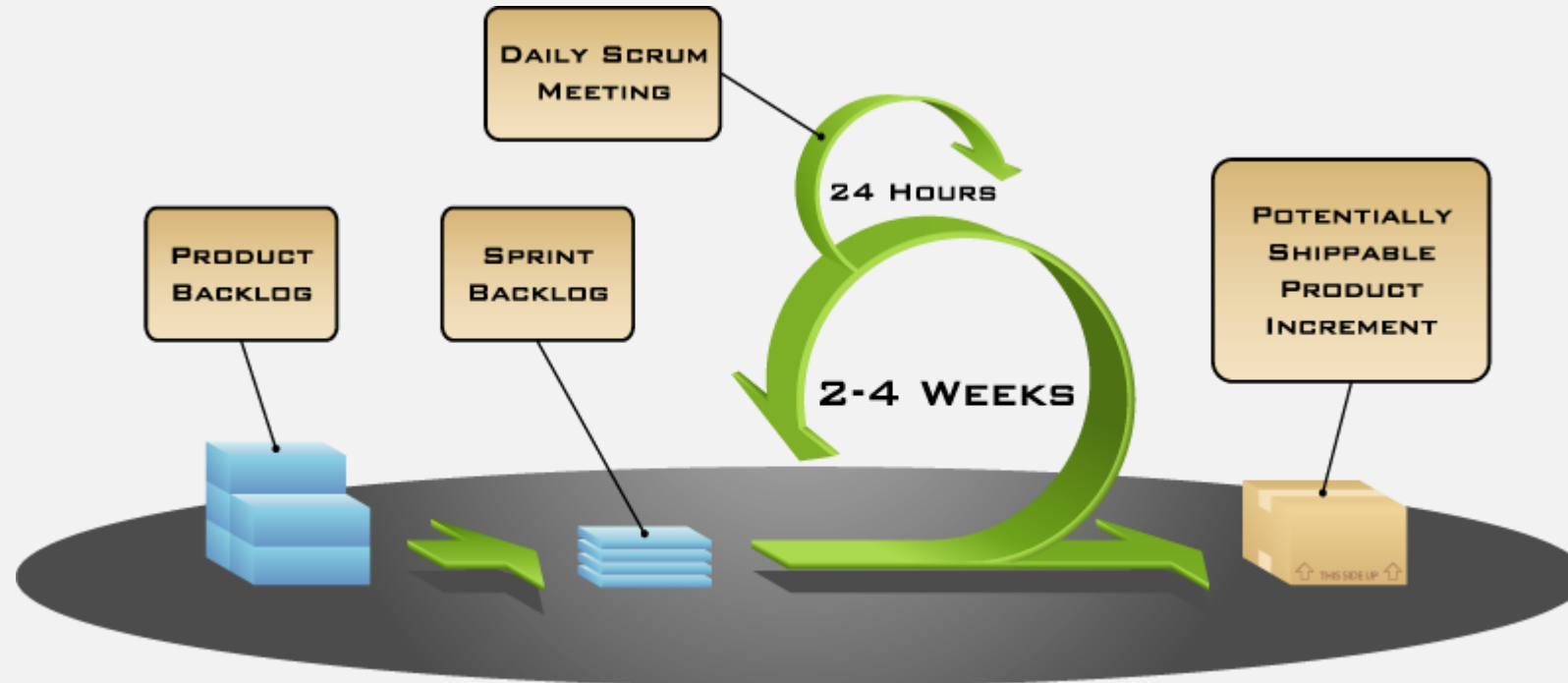


**Figure 3.3** The extreme programming release cycle

## SCRUM -- AN AGILE PROCESS

- The Scrum approach is a general agile method but its focus is on managing iterative development rather than specific agile practices.
- There are three phases in Scrum.
  - The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.
  - This is followed by a series of sprint cycles, where each cycle develops an increment of the system.
  - The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

# FUNCTIONALITY OF SCRUM



COPYRIGHT © 2005, MOUNTAIN GOAT SOFTWARE

# BASIC PARTS OF SCRUM

- Role
  - Scrum Master
  - Product Owner
  - Team
- Artifacts
  - Product Backlog
  - Sprint Backlog
  - Burn-down Charts
- Meetings
  - Sprint Planning
  - Sprint Review
  - Sprint Retrospectives
  - Daily Scrum
- The Sprint cycle
  - Sprints are fixed length, normally 2–4 weeks. They correspond to the development of a release of the system in XP.
  - The starting point for planning is the product backlog, which is the list of work to be done on the project.
  - The selection phase involves all of the project team who work with the customer to select the features and functionality to be developed during the sprint.
  - Once these are agreed, the team organize themselves to develop the software. During this stage the team is isolated from the customer and the organization, with all communications channelled through the so-called 'Scrum master'.
  - The role of the Scrum master is to protect the development team from external distractions.
  - At the end of the sprint, the work done is reviewed and presented to stakeholders. The next sprint cycle then begins.

## WHAT IS AGILE TESTING?

- Agile Testing is a software testing practice that follows the principles of agile software development.
- Agile Testing involves all members of the project team, with special expertise contributed by testers. Testing is not a separate phase and is interwoven with all the development phases such as requirements, design and coding and test case generation. Testing takes place simultaneously through the Development Life Cycle.
- Furthermore, with testers participating in the entire Development Lifecycle in conjunction with cross-functional team members, the contribution of testers towards building the software as per the customer requirements, with better design and code would become possible.
- Agile Testing covers all the levels of testing and all types of testing.



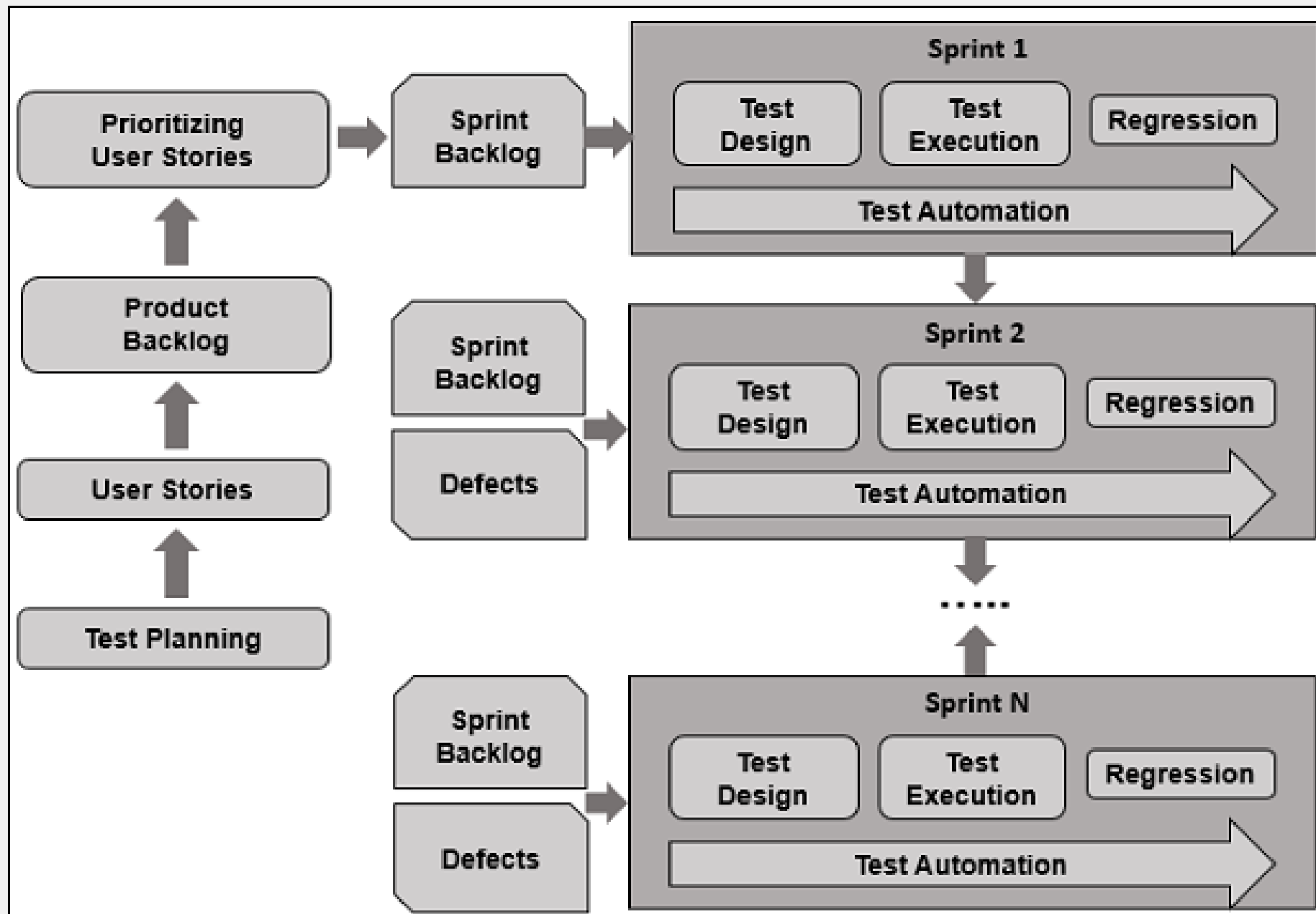
# AGILE TESTING ACTIVITIES

- The Agile Testing Activities at Project Level are –
  - Release Planning (Test Plan)
    - For every Iteration,
    - Agile Testing Activities during an Iteration
  - Regression Testing
  - Release Activities (Test Related)
- The Agile Testing Activities during an iteration include –
  - Participating in iteration planning
  - Estimating tasks from the view of testing
  - Writing test cases using the feature descriptions
  - Unit Testing
  - Integration Testing
  - Feature Testing
  - Defect Fixing
  - Integration Testing
  - Acceptance Testing
  - Status Reporting on Progress of Testing
  - Defect Tracking

## SCRUM BASED AGILE TESTING LIFECYCLE

- In Scrum, the Testing activities include –
  - Identify expected behavior of the system to derive test cases from User Stories
  - Release Planning based on Test Effort and Defects
  - Sprint Planning based on User Stories and Defects
  - Sprint Execution with Continuous Testing
  - Regression Testing after the completion of Sprint
  - Reporting Test Results
  - Automation Testing

# SPRINT BASED LIFECYCLE



# A 'PRESCRIBING MEDICATION' STORY

## Prescribing medication

The record of the patient must be open for input. Click on the medication field and select either 'current medication', 'new medication' or 'formulary'.

If you select 'current medication', you will be asked to check the dose; If you wish to change the dose, enter the new dose then confirm the prescription.

If you choose, 'new medication', the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

If you choose 'formulary', you will be presented with a search box for the approved formulary. Search for the drug required then select it. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

In all cases, the system will check that the dose is within the approved range and will ask you to change it if it is outside the range of recommended doses.

After you have confirmed the prescription, it will be displayed for checking. Either click 'OK' or 'Change'. If you click 'OK', your prescription will be recorded on the audit database. If you click 'Change', you reenter the 'Prescribing medication' process.

## EXAMPLES OF TASK CARDS FOR PRESCRIBING MEDICATION

### **Task 1: Change dose of prescribed drug**

### **Task 2: Formulary selection**

### **Task 3: Dose checking**

Dose checking is a safety precaution to check that the doctor has not prescribed a dangerously small or large dose.

Using the formulary id for the generic drug name, lookup the formulary and retrieve the recommended maximum and minimum dose.

Check the prescribed dose against the minimum and maximum. If outside the range, issue an error message saying that the dose is too high or too low. If within the range, enable the 'Confirm' button.

# TEST CASE DESCRIPTION FOR DOSE CHECKING

## Test 4: Dose checking

### Input:

1. A number in mg representing a single dose of the drug.
2. A number representing the number of single doses per day.

### Tests:

1. Test for inputs where the single dose is correct but the frequency is too high.
2. Test for inputs where the single dose is too high and too low.
3. Test for inputs where the single dose \* frequency is too high and too low.
4. Test for inputs where single dose \* frequency is in the permitted range.

### Output:

OK or error message indicating that the dose is outside the safe range.

## TESTING IN EXTREME PROGRAMMING

- Testing is central to XP and XP has developed an approach where the program is tested after every change has been made.
  - Unit Tests and Functional Tests
  - Test a little, code a little...“Test-first programming”
  - Tests become the specification
  - Tests give confidence in the system as user is involved
  - Tests give courage to change the system
  - Automated test harnesses are used to run all component tests each time that a new release is built.

## TEST-FIRST DEVELOPMENT

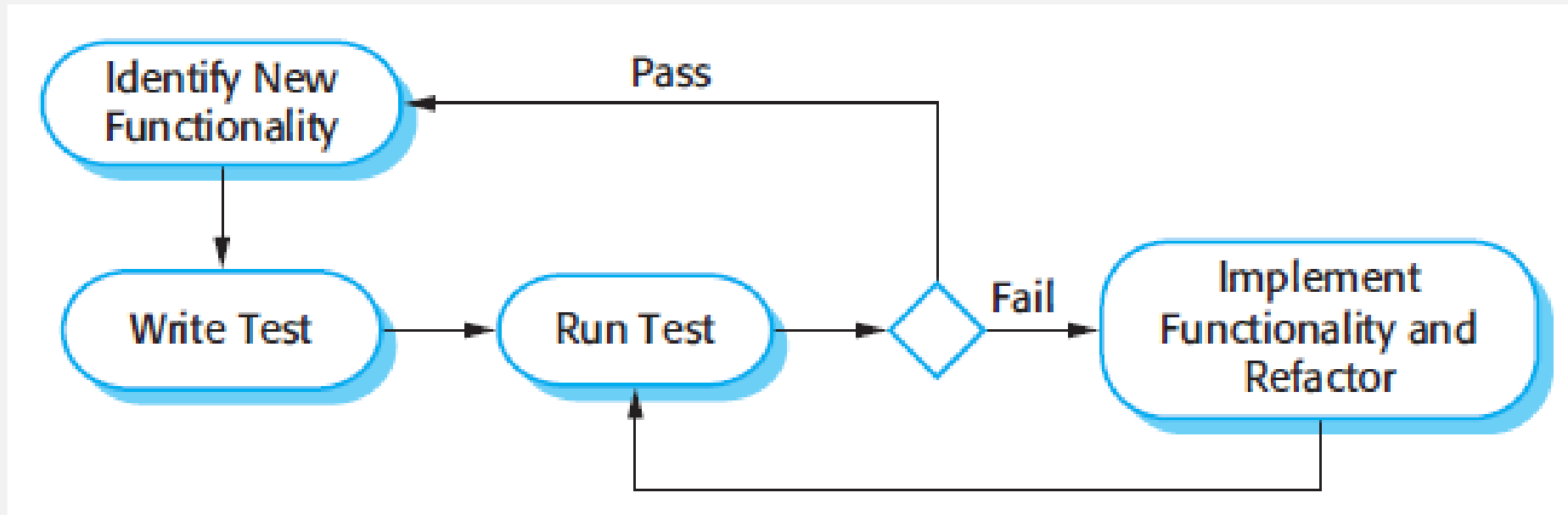
- Writing tests before code clarifies the requirements to be implemented.
- Tests are written as programs rather than data so that they can be executed automatically. The test includes a check that it has executed correctly.
  - Usually relies on a testing framework such as JUNIT.
- All previous and new tests are run automatically when new functionality is added, thus checking that the new functionality has not introduced errors.



# TEST-DRIVEN DEVELOPMENT

- Test-driven development (TDD) is an approach to program development in which you inter-leave testing and code development.
- Tests are written before code and ‘passing’ the tests is the critical driver of development.
- You develop code incrementally, along with a test for that increment. You don’t move on to the next increment until the code that you have developed passes its test.
- TDD was introduced as part of agile methods such as Extreme Programming. However, it can also be used in plan-driven development processes.

# TEST-DRIVEN DEVELOPMENT

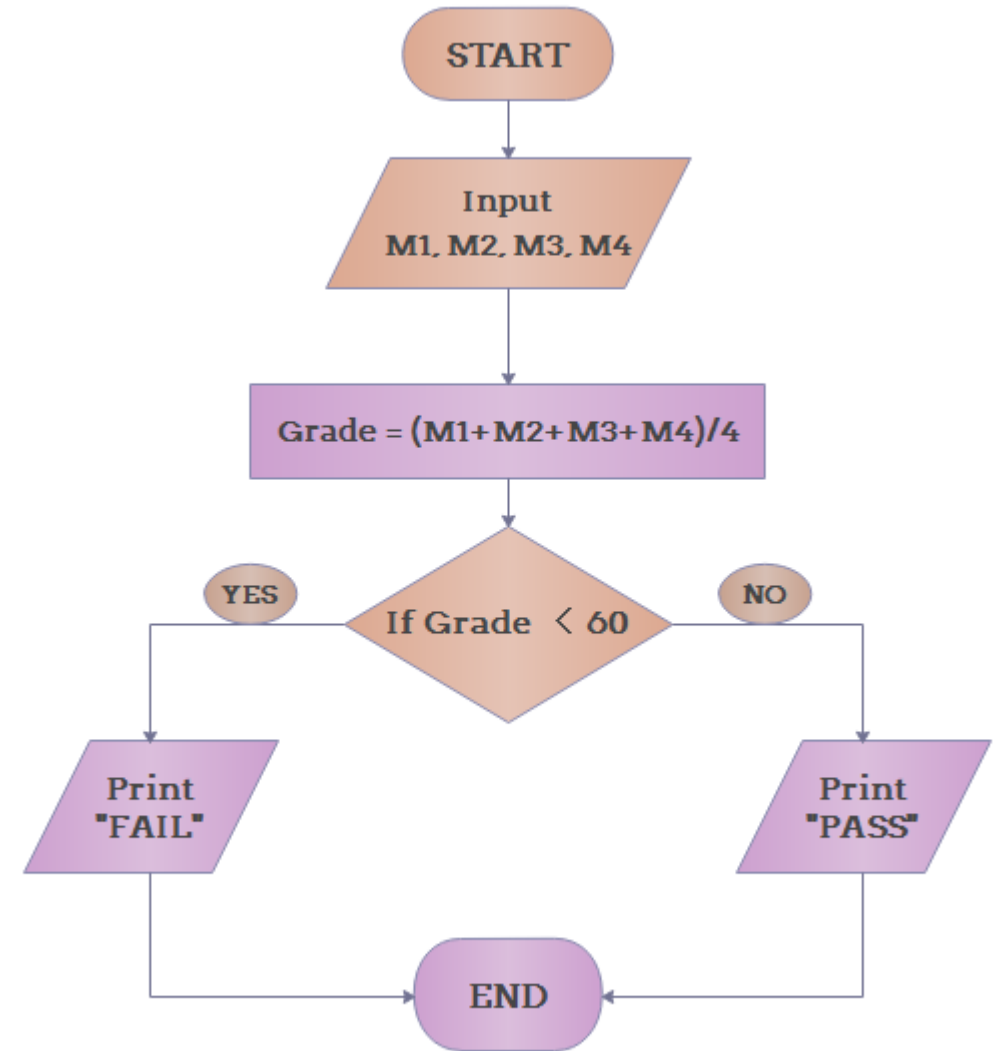


## TDD PROCESS ACTIVITIES

- Start by identifying the increment of functionality that is required. This should normally be small and implementable in a few lines of code.
- Write a test for this functionality and implement this as an automated test.
- Run the test, along with all other tests that have been implemented. Initially, you have not implemented the functionality so the new test will fail.
- Implement the functionality and re-run the test.
- Once all tests run successfully, you move on to implementing the next chunk of functionality.

# TDD EXAMPLE

Input	Expected Output	Actual Output	Validity
72, 77, 75, 10	Pass	Fail	O/P: Invalid I/P: Valid
12, 25, 50, 24	Fail	Fail	O/P: valid I/P: Valid
120, 200, 225, -20	Wrong Input	Pass	O/P: Invalid I/P: Invalid
80, 82, 75, 60	Pass	Pass	O/P: valid I/P: valid



## CUSTOMER INVOLVEMENT

- The role of the customer in the testing process is to help develop acceptance tests for the stories that are to be implemented in the next release of the system.
- The customer who is part of the team writes tests as development proceeds. All new code is therefore validated to ensure that it is what the customer needs.
- However, people adopting the customer role have limited time available and so cannot work full-time with the development team. They may feel that providing the requirements was enough of a contribution and so may be reluctant to get involved in the testing process.

# TEST AUTOMATION

- Test automation means that tests are written as executable components before the task is implemented
  - These testing components should be stand-alone, should simulate the submission of input to be tested and should check that the result meets the output specification. An automated test framework (e.g. JUnit) is a system that makes it easy to write executable tests and submit a set of tests for execution.
- As testing is automated, there is always a set of tests that can be quickly and easily executed
  - Whenever any functionality is added to the system, the tests can be run and problems that the new code has introduced can be caught immediately.

## XP TESTING DIFFICULTIES

- Programmers prefer programming to testing and sometimes they take short cuts when writing tests.
  - For example, they may write incomplete tests that do not check for all possible exceptions that may occur.
- Some tests can be very difficult to write incrementally.
  - For example, in a complex user interface, it is often difficult to write unit tests for the code that implements the 'display logic' and workflow between screens.
- It difficult to judge the completeness of a set of tests. Although you may have a lot of system tests, your test set may not provide complete coverage.

# BENEFITS OF TEST-DRIVEN DEVELOPMENT

- Code coverage
  - Every code segment that you write has at least one associated test so all code written has at least one test.
- Regression testing
  - A regression test suite is developed incrementally as a program is developed.
- Simplified debugging
  - When a test fails, it should be obvious where the problem lies. The newly written code needs to be checked and modified.
- System documentation
  - The tests themselves are a form of documentation that describe what the code should be doing.



## ROLE OF TESTER IN AGILE TEAM

- In the Agile Lifecycle, a tester plays a significant Role in –
  - Teamwork
  - Test Planning
  - Sprint Zero
  - Integration
  - Agile Testing Practices

# TEAMWORK

- **Collaborative Approach** – Working with cross-functional team members on Test Strategy, Test Planning, Test Specification, Test Execution, Test Evaluation, and Test Results Reporting. Contributing the testing expertise in conjunction with other team activities.
- **Self-organizing** – Planning and organizing well within the sprints to achieve the targets of testing by amalgamating expertise from other team members as well.
- **Empowerment** – Making appropriate technical decisions in achieving the team's goals.
- **Commitment** – Committing to understanding and evaluating the product's behavior and characteristics as required by the customers and stakeholders.
- **Transparent** – Open, Communicating and Accountable.
- **Credibility** – Ensuring the credibility of the test strategy, its implementation, and execution. Keeping the customers and stakeholders informed on the test strategy.
- **Open to Feedback** – Participating in sprint retrospectives to learn from both successes and failures. Seeking customer feedback and acting quickly and appropriately to ensure quality deliverables.
- **Resilient** – Responding to changes.

# TEST PLANNING

- Testing Scope
- New functionalities which are being tested
- Level or Types of testing based on the features complexity
- Load and Performance Testing
- Infrastructure Consideration
- Mitigation or Risks Plan
- Resourcing
- Deliverables and Milestones

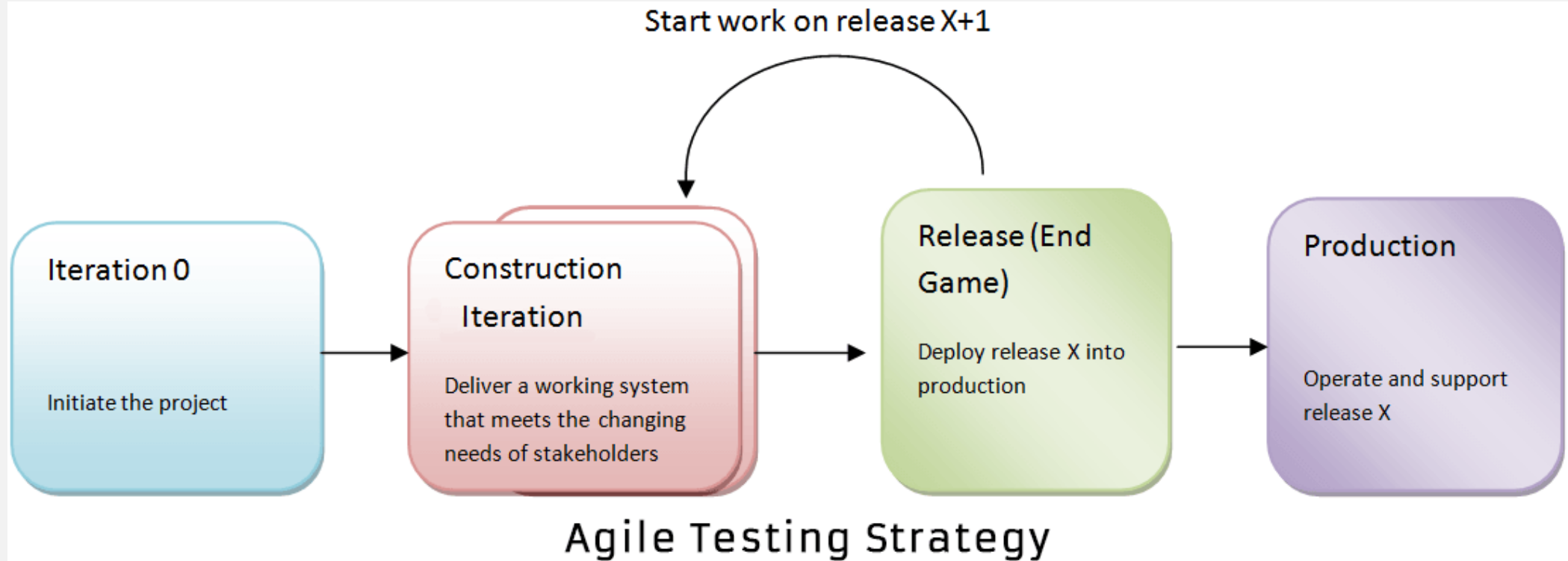
# SPRINT ZERO

- Sprint Zero involves preparation activities before the first sprint. A tester needs to collaborate with the team on the following activities –
  - Establishing a business case for the project
  - Establish the boundary conditions and the project scope
  - Outline the key requirements and use cases that will drive the design trade-offs
  - Outline one or more candidate architectures
  - Identifying the risk
  - Cost estimation and prepare a preliminary project

# AGILE TESTING PRACTICES

- An Agile tester needs to adapt Agile practices for testing in an agile project.
  - **Pairing** – Two team members work together at the same keyboard. As one of them tests, the other reviews/analyzes testing. The two team members can be
    - One tester and one developer
    - One tester and one business analyst
    - Two testers
  - **Incremental Test Design** – Test cases are built from user stories, starting with simple tests and moving to more complex tests.
  - **Mind Mapping** – A mind map is a diagram to organize the information visually. Mind mapping can be used as an effective tool in Agile testing, using which information regarding the necessary test sessions, test strategies and test data can be organized.

# AGILE TESTING STRATEGIES



## REFERENCE

1. Ian Sommerville, “Software Engineering”, 9th Edition, Chapter 3 – Agile Software Development
2. <https://www.guru99.com/agile-testing-course.html>
3. [https://www.tutorialspoint.com/agile\\_testing/index.htm](https://www.tutorialspoint.com/agile_testing/index.htm)