VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Module – 1

Agile Project Management

Dr. Rajesh M
School of Computer Science and Engineering
Vellore Institute of Technology [VIT]
Chennai, India.

Traditional Project Management Practices *can* Lead to....

- Chaos – Junior Project Managers tend to either:
- allow too much uncontrolled changes to take place (to ensure customer satisfaction)
- or are too strict in allowing for change (resulting in irate customers).

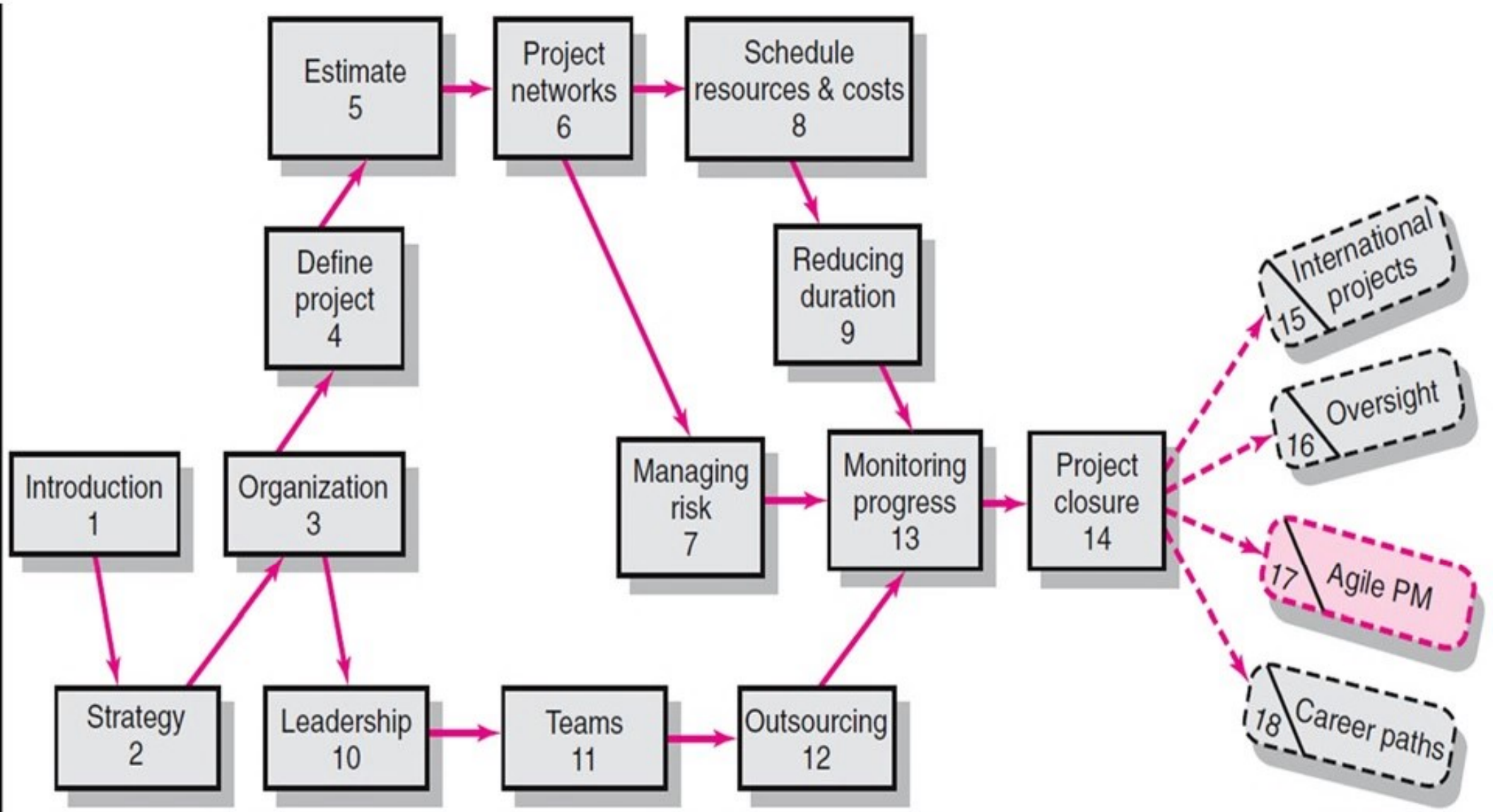
Traditional Project Management Practices *can* Lead to....

- Dramatic Project Underperformance – According to the Standish Group's *Chaos Reports*, only 16 percent of IT projects are successful, the remainder are:
  - Late
  - Over Budget
  - Deliver only a fraction of original scope in order to meet budget restrictions
  - Cancelled

What Is Different About Agile Methods?

- They are all about **managing the impact of change on a project**.
- They allow change to be introduced into a project in a orderly way that attempts to **maximize the benefits for the sponsor.**
- They **control the risks that the change introduces**.

Where we are now???



Traditional PM Vs Agile Methods

- **Traditional PM Approach**
  - **Concentrates on thorough, upfront planning** of the **entire project.**
  - **Requires a high degree of predictability** to be effective.

- **Agile Project Management (Agile PM)**
  - **Relies on incremental, iterative development cycles** to complete less-predictable projects.
  - Is ideal for exploratory projects in which requirements need to be discovered and new technology tested.
  - **Focuses on active collaboration between the project team and customer** representatives.

Traditional PM Vs Agile Methods

Traditional

- Design up front Fixed scope Deliverables
- Freeze design as early as possible Low uncertainty
- Avoid change
- Low customer interaction Conventional project teams

Agile

- Continuous design Flexible Features/requirements
- Freeze design as late as possible High uncertainty
- Embrace change
- High customer interaction Self-organized project teams

Agile PM Principles

Focus on customer value

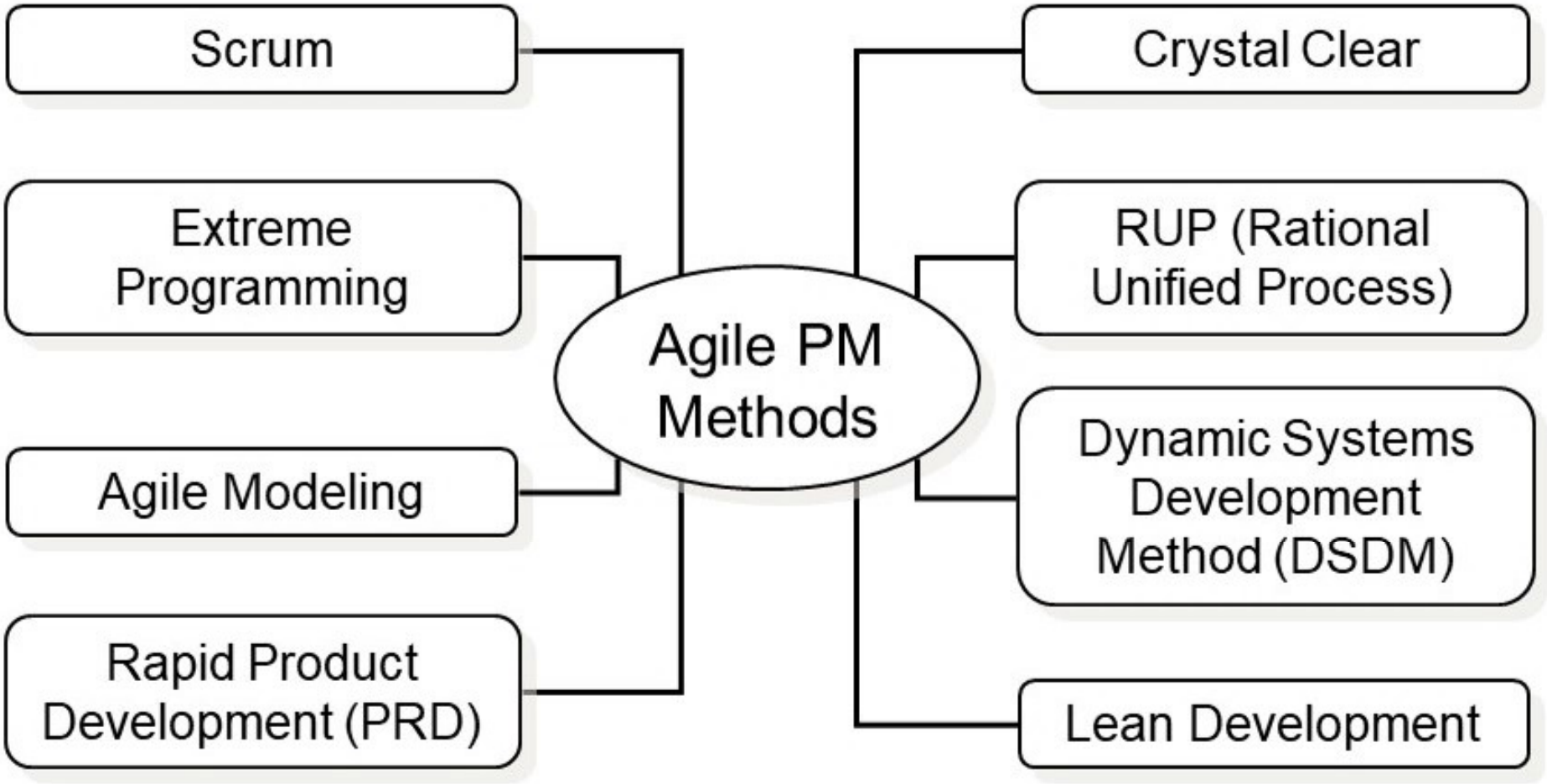
Iterative and incremental delivery

Experimentation and adaptation

Self-organization

Continuous improvement

Popular Agile PM Methods



Agile PM in Action: Scrum

Scrum Methodology

- Is a holistic approach for use by a cross-functional team collaborating to develop a new product.
- Defines product features as deliverables and prioritizes them by their perceived highest value to the customer.
- Re-evaluates priorities after each iteration (sprint) to produce fully functional features.
- Has four phases: analysis, design, build, test

Key Roles and Responsibilities in the Scrum Process

Product Owner

- Acts on behalf of customers to represent their interests.

Development Team

- Is a team of five-nine people with cross-functional skill sets is responsible for delivering the product.

Scrum Master (aka Project Manager)

- Facilitates scrum process and resolves impediments at the team and organization level by acting as a buffer between the team and outside interference.

Applying Agile to Large Projects

Scaling

- Is using several teams to work on different features of a large scale project at the same time.

Staging

- Requires significant up-front planning to manage the interdependences of different features to be developed.
- Involves developing protocols and defining roles to coordinate efforts and assure compatibility and harmony.

Limitations and Concerns of Agile PM

- It does not satisfy top management's need for budget, scope, and schedule control.
- Its principles of self-organization and close collaboration can be incompatible with corporate cultures.
- Its methods appear to work best on small projects that require only five-nine dedicated team members to complete the work.
- It requires active customer involvement and cooperation.

Key Terms

Feature

Iterative Incremental Development (IID)

Scrum meeting

Scrum Master

Sprint backlog

Product Backlog

Product Owner,

Burn-down Chart, Scaling

Agile PM

Self Organizing Team

Estimation and Velocity

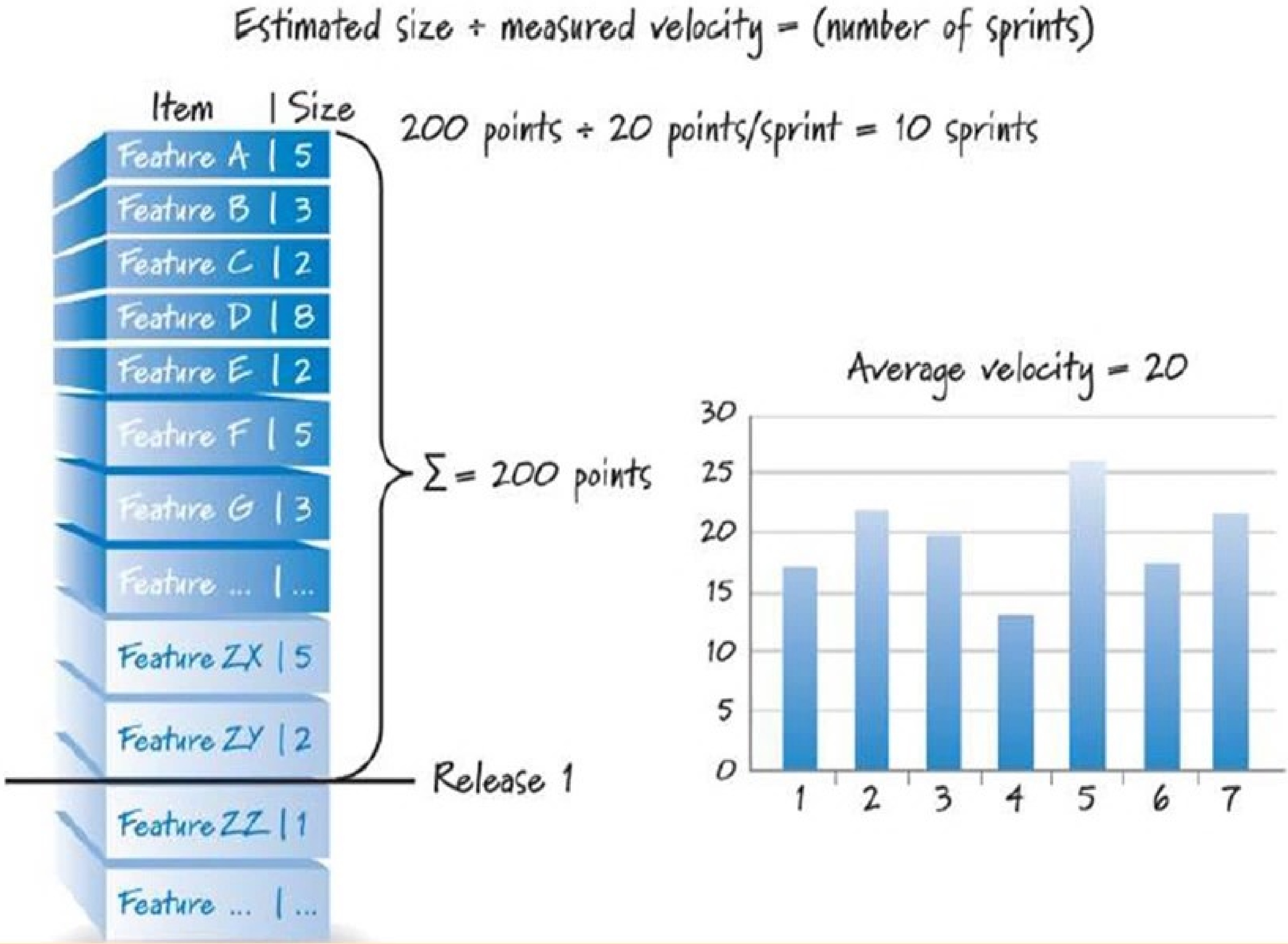
Estimation and Velocity

- When planning and managing the development of a product, we need to answer important questions such as:
  - "How many features will be completed?"
  - "When will we be done?"
  - "How much will this cost?"
- To answer these questions, we need to estimate the size of what we are building and measure the velocity at which we can get it done.
- With that information, we can derive the likely product development duration (and the corresponding cost) by dividing the estimated size of a set of features by the team's velocity.

Relationship Among Size, Velocity, and Duration

- Basic Question: How much time do we need to create the features in Release 1?
  - Answer:
    1. Gauge the **size** of Release 1 by adding the individual size estimates for the PBIs targeted for Release 1.
    2. Estimate the team's **velocity**: How much work the team typically gets done each sprint.
       1. At the end of each sprint, add the size estimates of the BPIs that were completed in the sprint; this sum is the team's velocity for that sprint.
       2. Calculate an average velocity for the sprints that have been completed.
    3. Now that we have estimated size and measured average velocity, calculate the **duration** by dividing the size by the velocity.

Relationship Among Size, Velocity, and Duration

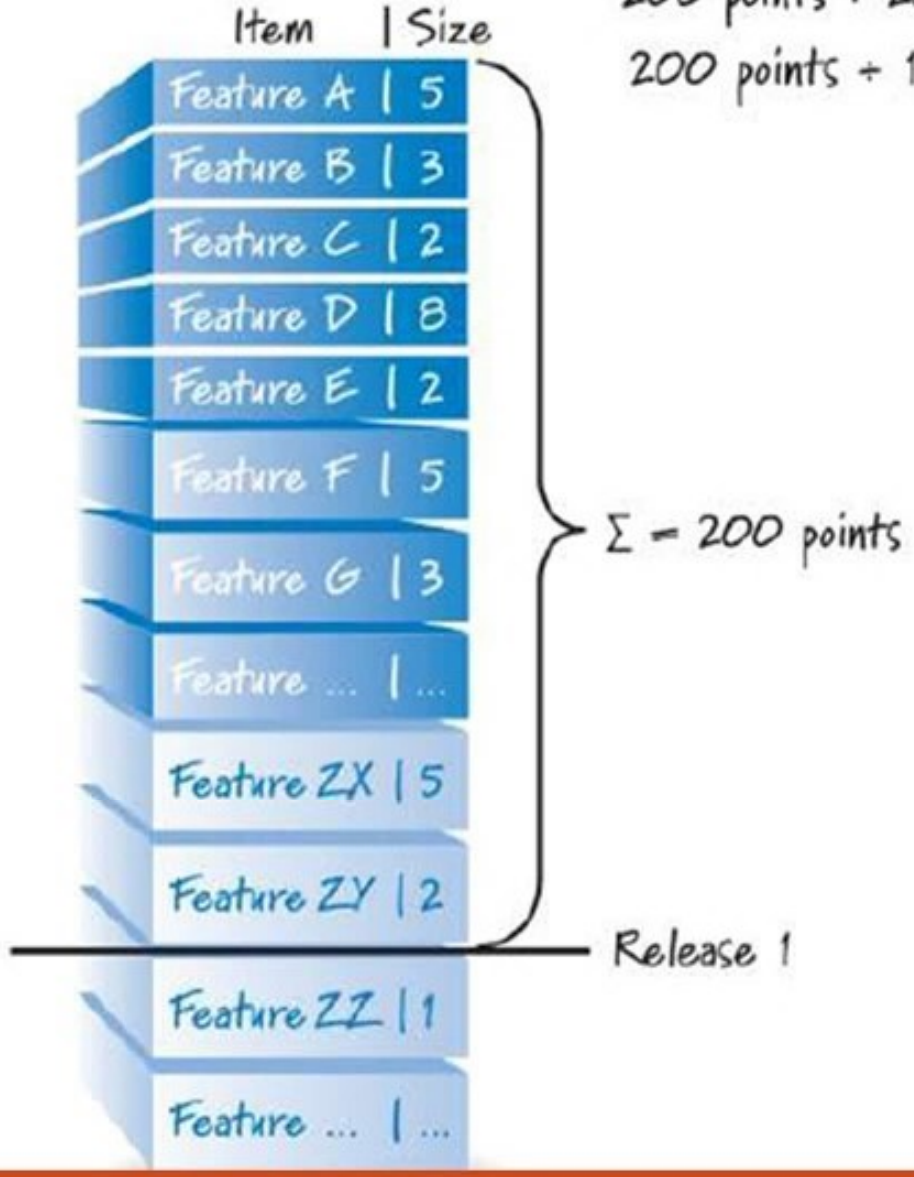


Velocity Range: Calculation and Use

Release 1 will need 10 to 12 sprints to complete

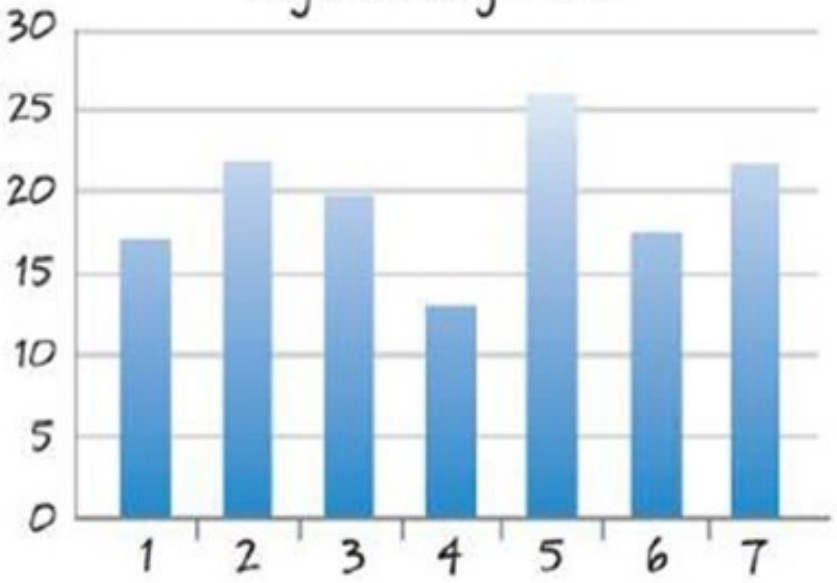
200 points ÷ 20 points/sprint = 10 sprints

200 points ÷ 17 points/sprint = 12 sprints



Low velocity = 17

High velocity = 20

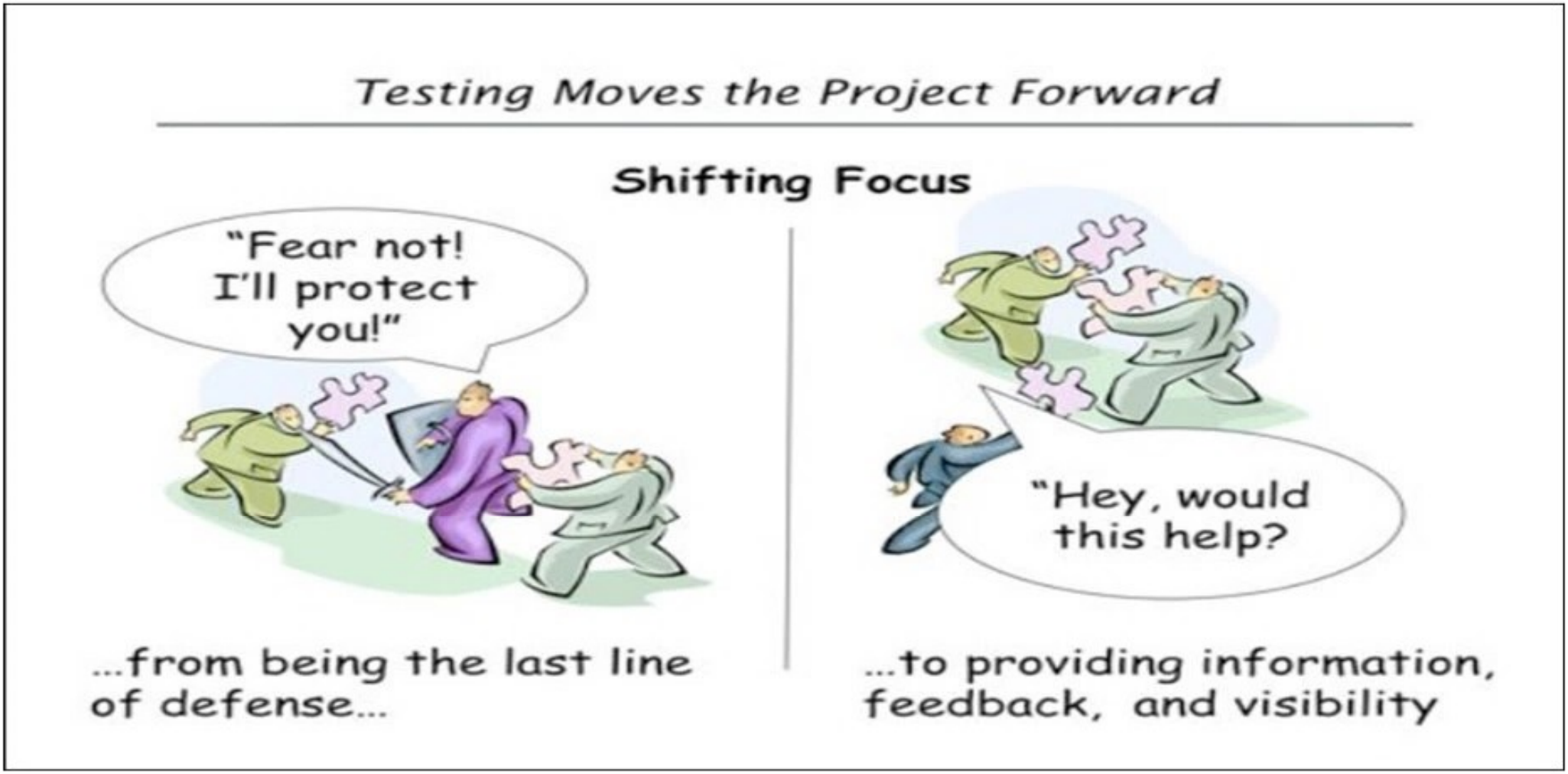


Limitations and Concerns of Agile PM

- It does not satisfy top management's need for **budget, scope, and schedule**.
- Its principles of **self-organization and close collaboration** can be incompatible with corporate cultures.
- Its methods appear to work best on small projects that **require only five-nine dedicated team members** to complete the work.
- It **requires active customer involvement and cooperation**.

Agile Testing

What is Agile Testing?



What is Agile Testing?

- **A software testing practice** that **follows the principles of agile** software development is called **Agile Testing.**

- **Agile** is an **iterative development methodology**, where requirements evolve **through collaboration** between the **customer** and **self-organizing teams** and agile aligns development with customer needs.

Advantages of Agile Testing

- Agile Testing **Saves Time and Money**
- **Less Documentation**
- **Regular feedback** from the end user
- **Daily meetings** can **help to determine the issues** well in advance.

Agile Testing Principles

- **Testing moves the project forward:**
  - Testing as **quality gate**;
  - **Discuss about bug** priority, not importance;

Agile: Build product from beginning, using testing to

- **provide feedback on continuous basis**;

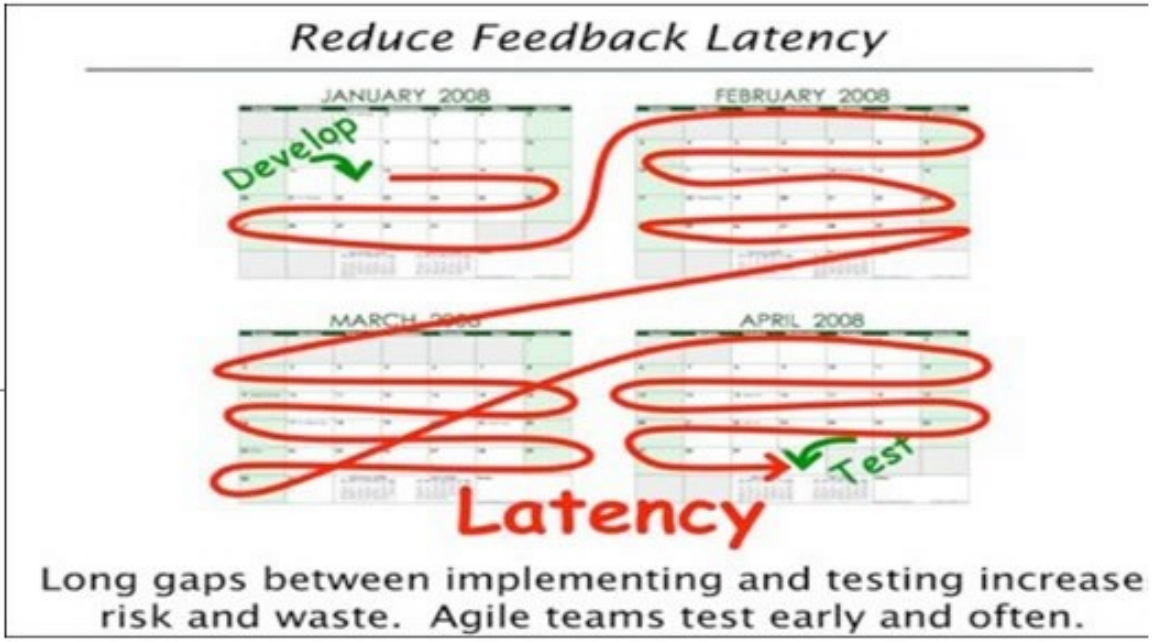
- **Testing is not a Phase:**

  Agile teams test continuously;

- ***"Continuous testing is the only way to ensure continuous progress".***

Everyone Tests:

- **Independent testers** vs **whole team**;
- Testers execute tests; Developers too;



Shortening Feedback Loops:

- How the software is behaving?
- What is Feedback loop? **Automated unit test** – checks behavior of methods;

Run often **Acceptance test** – checks end- to-end behavior of system;

- **Manual Regression Test:** Takes longer time to execute;
- Feedback time increases to days / weeks;

❑Keep the code clean:

- Example of discipline that Agile team have;
- Bugs are fixed within the iteration;

Lightweight Documentation:

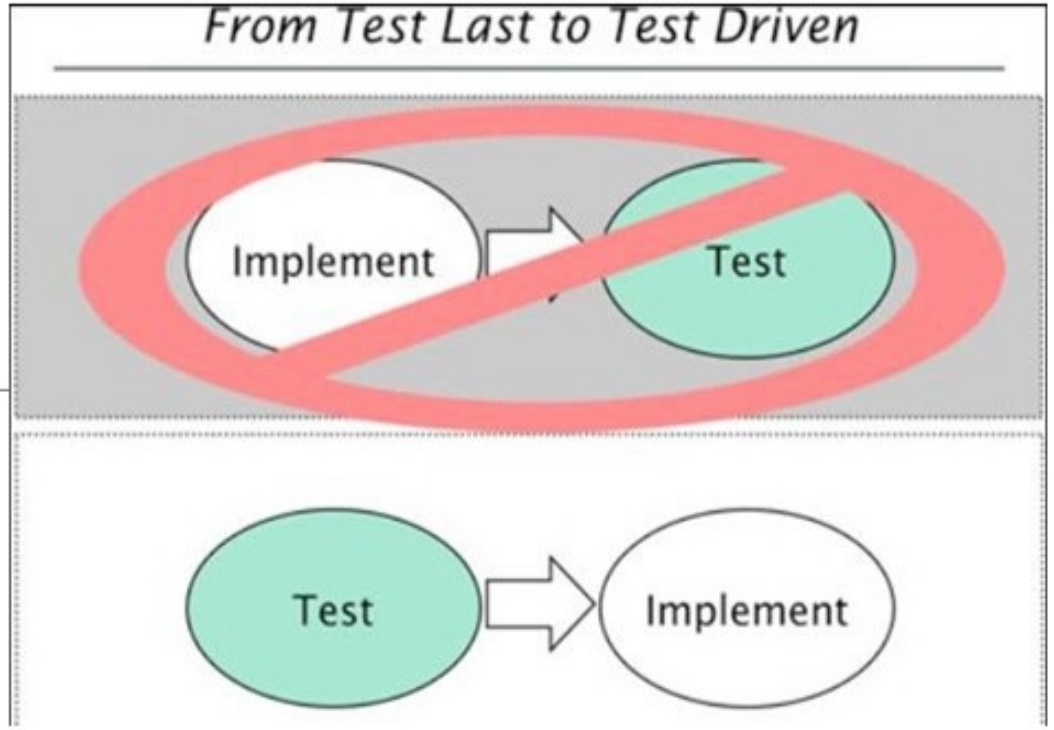
- Instead of writing comprehensive test documentation,

Agile testers:

- Use reusable checklists to suggest tests
- Use lightweight documentation styles/tools
- Leverage documents for multiple purpose

❑ "Done Done" Not Just Done:

- Strict division between **development & testing phase.**
- Feature isn't done until it's been tested;
- Last 10% of the effort takes 90% of the time.

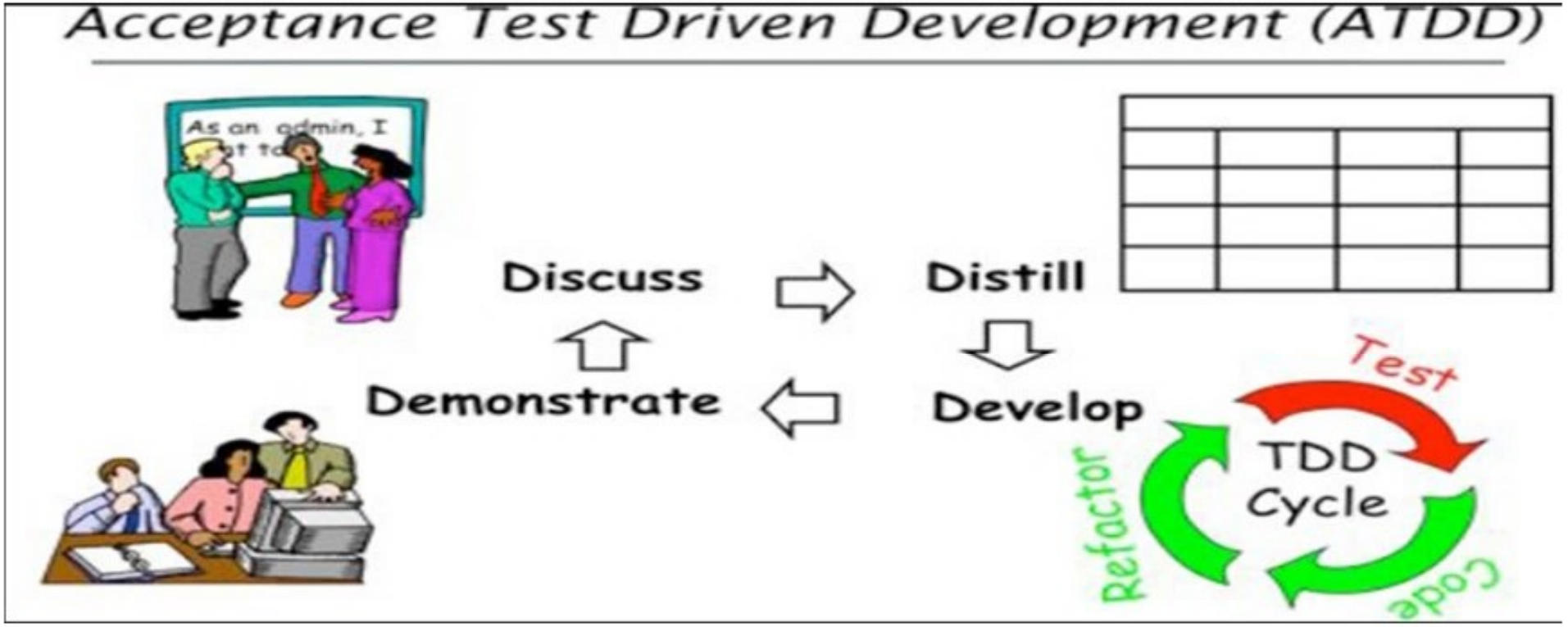


❑ Test-Last vs Test-Driven:

- In traditional environments, tests are derived from project artifacts such as requirements documents.
- In Agile, defining the **tests with the requirements**, and using those tests to drive the development effort, gives us much more shared focus on the goal.

Six Concrete Test Practices suitable for Agile Projects

- **A**utomated **U**nit / **I**ntegration **T**ests;
- **T**est **D**riven **D**evelopment;
- **A**utomated system level **r**egression **t**ests;
- **A**cceptance **T**est **D**riven **D**evelopment (**ATDD**)



Exploratory Testing:

Exploratory Testing

Simultaneously…

…learning about the software

…designing tests

…executing tests

using feedback from the last test to inform the next

Collaborative Testing:

Effective Testing on Agile Projects...

...ultimately becomes a whole team effort integrated throughout the development cycle.

Agile Testing Activities

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

- 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
- Acceptance Testing
- Status Reporting on Progress of Testing
- Defect Tracking

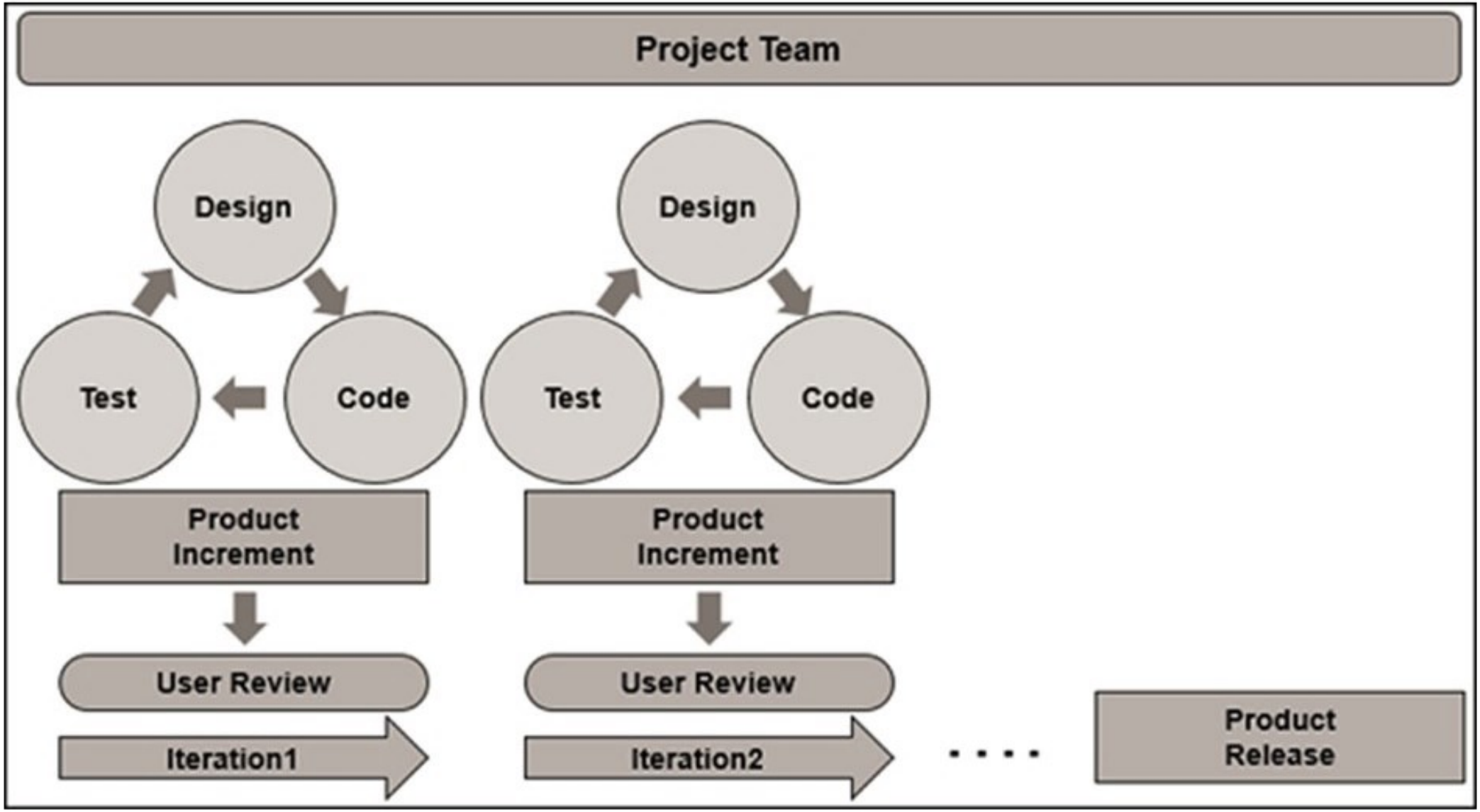
Agile Testing Vs. Waterfall Testing

Agile Testing	Waterfall Testing
Testing is not a separate phase and occurs concurrently with development.	**Testing is a separate phase**. All levels and types of testing can begin only after the completion of development.
Testers and developers work together.	**Testers work separately** from developers.
Testers are involved in coming up with requirements. This helps in requirements mapping to the behaviors in the real world scenario and also framing the acceptance criteria. Also, logical Acceptance Test Cases would be ready along with the requirements.	**Testers may not be involved in the requirements phase.**
Acceptance Testing is done after every iteration and customer feedback is sought.	**Acceptance Testing** is done only at the **end of the project.**
Every iteration completes its own testing thus allowing **regression testing to be implemented every time new functions or logic are released.**	**Regression Testing** can be implemented **only after the completion of development.**
No time delays between coding and testing.	**Usual time delays** between coding and testing.
Continuous testing with overlapping test levels.	**Testing is a timed activity** and test levels cannot overlap.
Testing is a best practice.	Testing is often overlooked.

Agile Testing - Methodologies

- Agile is an **iterative development methodology**, where the **entire project team participates in all the activities**.
- The requirements evolve as the **iterations progress through collaboration between the customer and the self-organizing teams**.
- As **Coding and Testing are done interactively and incrementally** during the course of development, the end-product would be of quality and ensures customer requirements.
- **Every iteration results in an integrated working product increment** and is delivered for User Acceptance Testing. The customer feedback thus obtained would be an input to the next / subsequent Iterations.

Agile Testing - Methodologies



Agile Testing Methodologies

Test-Driven Development (TDD) – Test-Driven Development (TDD) is based on coding guided by tests.

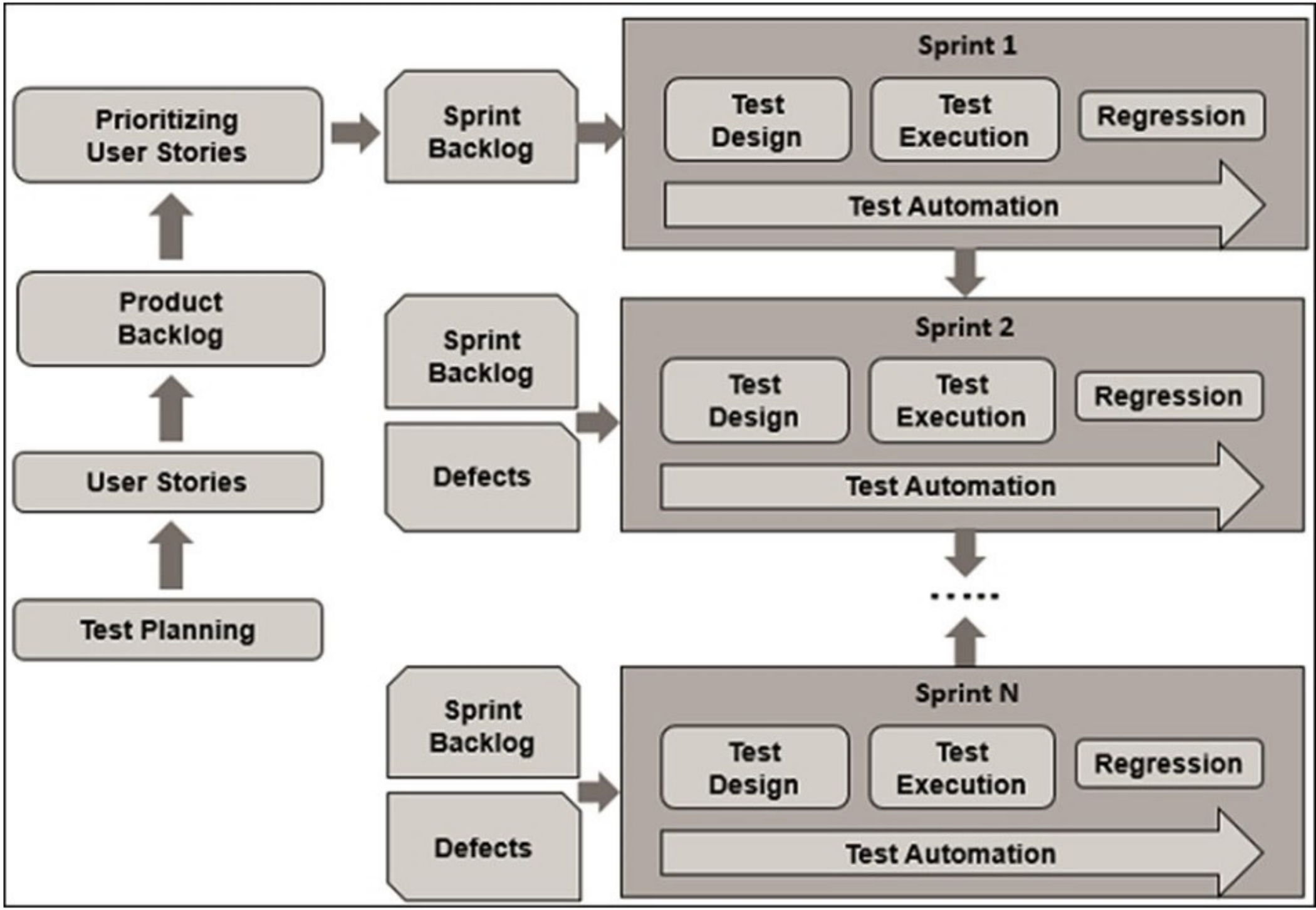
Acceptance Test-Driven Development (ATDD) – Acceptance Test-Driven Development (ATDD) is **based on communication between the customers, developers and testers and driven by pre-defined Acceptance Criteria** and Acceptance Test Cases.

Behavior-Driven Development (BDD) – In Behavior-Driven Development (BDD) testing is **based on the expected behavior of the software being developed**.

Agile Testing Lifecycle

In **Scrum**, the **Testing activities** include –

- **Contributing to User Stories** based on the expected behavior of the System depicted as Test Cases
- **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**


Prioritizing User Stories
Sprint Backlog
Sprint 1
Test Design
Test Execution
Regression
Test Automation
Product Backlog
Sprint Backlog
Defects
Sprint 2
Test Design
Test Execution
Regression
Test Automation
User Stories
Test Planning
Sprint Backlog
Defects
Sprint N
Test Design
Test Execution
Regression
Test Automation

Test Driven Development

- In the **Test Driven Development (TDD)** method, the code is developed based on the **Test first approach** directed by Automated Test Cases.
- A **test case is written first to fail, code is developed based on that to ensure that the test passes.**
- Method is repeated, **refactoring is done** through the development of code.

TDD can be understood with the help of the following steps –

- **Step 1 – Write a Test case** to reflect the expected behavior of the functionality of the code that needs to be written.
- **Step 2 – Run the test**. The test fails as the code is still not developed.
- **Step 3 – Develop code** based on the test case.
- **Step 4 – Run the test again.** This time, the test has to pass as the functionality is coded. **Repeat Step (3) and Step (4)** till the test passes.
- **Step 5 – Refactor the code**.
- **Step 6 – Run the test again** to ensure it passes.

Repeat **Step 1 – Step 6** adding test cases to **add functionality**. The added tests and the earlier tests are run every time to ensure the code is running as expected. To make this process fast, tests are automated.

The tests can be at **unit, integration or system level**. Constant communication between testers and developers needs to be ensured.


Improve code quality by REFACTORING
Re-factor
Red
Write the Test that FAILs
REPEAT
Green
Make the code PASS
Test Driven Development

References

- K.S. Rubin, Essential Scrum: A Practical Guide to the Most Popular Agile Process, Addison-Wesley, 2012.
- M. Cohn, Succeeding with Agile: Software Development Using Scrum, Addison-Wesley, 2009
- S.W. Ambler, M. Lines, Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise, IBM Press, 2012.
- Chetankumar Patel, Muthu Ramachandran, Story Card Maturity Model (SMM): A Process Improvement Framework for Agile Requirements Engineering Practices, Journal of Software,Academy Publishers, Vol 4, No 5 (2009), 422-435, Jul 2009.
- Kevin C. Desouza, Agile information systems: conceptualization, construction, and management, Butterworth-Heinemann, 2007
- K. Beck, C. Andres, Extreme Programming Explained: Embrace Change, 2nd Edition, Addison-Wesley, 2004.

Thank you!..