Anatomies of Test Plans & Test Cases

What is a Test Plan?

- A **Test Plan** describes the testing scope and the approach. **Test Plan is a dynamic document**.
- Defines the objectives of the testing clearly.
- It **summarizes** the testing scope and specifies all aspects of the testing including the testing **roadmap**.
- It sets the **conditions for testing** by defining the testing activities and the testing dependencies.
- It identifies the tools, the resources, the timelines, the risks and the entrance/exit criteria for testing.
- The **Test Plan is a testing contract** that will have to be reviewed and **approved** by all of the project stake holders.
- The **Test Plan** goes through many iterations before completion.
- The Test Plan **starts early** in the project and gets completed before the testing starts.
- There are many different **test plan formats**. The **Test Plan** outline should be flexible and should fit the project.
- The **test plan reviews** allow more fact findings and yields great results.
- The **test plan** should be **clear, concise** and to the point.
- The **test plan** is one of the important **testing artifacts**

Test Plan Sections

1. Test Plan/ Test Strategy is a document that outlines the:
    1. **Scope of testing**: This will be a **high level description** of what will be covered in the testing. It also includes the functions that will be tested and the areas that will not be tested.
    2. **Testing Objectives**: This will include the **goals and the objectives** of the testing. It helps the QA manager and the testers to focus on the areas that are needed for testing.
    3. **Testing Schedules**: This section confirms the testing schedules for all of the subset of testing. E.g Functional, Integrations, End to End, Performance, and UAT timelines.
    4. **Roles and Responsibilities:**It is important to specify who is responsible for which aspect of testing and the support associated with the testing.
    5. **Testing Phases**
        1. **Re-iterate** the testing phases and the timelines associated with the entire testing life cycle.
        2. **Show** the sequence of testing and the dependency with other projects.
        3. **State the requirements** for Performance and load testing and the required SLAs.
    6. **Features to be Tested (in scope), Features not to be tested (out of scope): Specify** the features that will be part of testing and the features that will not be in the scope of testing. Explain the reasoning why for both.
    7. **QA Entrance/Exist Criteria:**
        1. Each phase of testing should have an **entrance and exit criteria** to ensure that the objectives are met before the start or ending each phase of the testing.
        2. A complete **Code Freeze prior to start of Functional Testing** (Entrance Criteria)
        3. **No Open major defects** open upon completion of the End to End testing (Exit Criteria)
    8. **QA Exist Criteria**
    9. **Dependencies: State** any testing **dependencies to any external systems**, or any other **test equipment**, or resources in other groups creating input/output data for the testing.
    10. **Test Environments**:
        1. **State any hardware requirements** or any peripheral test equipment needed in the test environment.
        2. State any requirements for the **test configuration** or any particular test data needed for testing.
        3. State the **Release Management process** during testing and how each release should get deployed in QA for testing.
    11. **Risks & Assumptions**:
        1. Enter **any risks that could jeopardize the time lines** or any assumptions that is taken into considerations during the testing.
        2. It includes any assumptions about the testing or the **development resources**, testing tools and/or the downstream test environments availability.
    12. **Defect Severities/Processes**
        1. Re-iterate the Defect Tracking processes and the Defect Severities.

        2. Responsibilities of the Development, Test Lead and QA in regards to the defect management.
    13. **Document Version Control:** The **Test Plan** document should have **version numbers, Author, and dates** associated with the last updates to the document
    14. **Appendices**
        1. References to Test Case documentations must exist in the **appendices**
        2. Include any user roles list, and technical details information in the Appendices as well
2. Developed by the QA or the Lead QA, reviewed by all members on the project team. Approved by the managers.

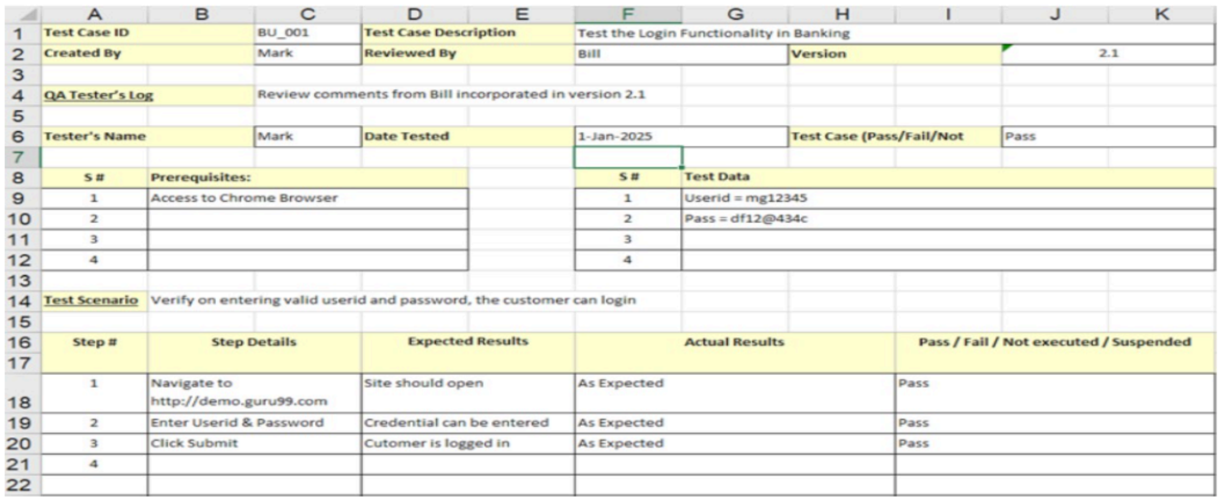
What are Test Scenarios & Test Cases

- A **Test Scenario** is a high level description of the Test Case(s).
  - It covers a wide range of possibilities.
  - It is a high level description of the test area.
- A **Test Case** is a detailed description of a test condition.
  - There are **One to One or One to Many** relationships between a Test Scenario and TestCases.
- Example:
  - TestScenario:
    - TestScenario1: Check the Login Functionality of google.Com.
  - TestCases:
    - TestCase1: Check for Valid user id and password (Positive Test Case).
    - TestCase2: Check for invalid user id and password (Negative Test Case).
    - TestCase3: Check for Null values for User id and Password (NegativeTestCase).

Anatomy of a Test Case

1. A Test Case contains the following:
    1. **Test Case Description**: A clear test description in one sentence.
    2. **Pre-Conditions:** Any assumptions or pre-requisites needed for this test case.
    3. **Test Steps:** 1 to 7 Test Steps
    4. **Expected Results**: What the tester should experience or see during the execution of the steps:
        1. If all test steps yield the correct Expected Results ->The Test Case Passes
        2. If not all of the test steps yield the correct Expected Results, two conditions exist:
            1. The Test Case Fails with a Medium/High/Major defect
            2. The Test Case Passed with a minor defects
        3. Actual Result is what the tester observes, and compares with the Expected Results
2. The **Test Cases** should be written in such away that other testers can execute without any questions.
    1. The Test Case description should be very clear.
        - Example: A Registered User should be able to successfully login to the site using valid user id & pw.
    2. The test steps need to be clear and be 'Action Driven' using 'Action Words', like 'Click on', 'Navigate', 'Enter', 'Verify' and, etc.
    3. The **Expected Results** should clearly describe what the tester should **expect** during the step execution.
3. A Test Case Example:
    1. **Description:** A registered user should be able to successfully login to the gmail.com website.
    2. **Precondition:** the user must already be registered with an email address and password.
    3. **Assumption:** a supported browser is being used.
    4. TestSteps:
        1. Step1: Navigate to gmail.com
        2. Step2: In the 'email' field, enter the email address of the registered user.
        3. Step3: Click the 'Next' button.
        4. Step4: Enter the password of the registered user
        5. Step5: Click'SignIn'

5. **Expected Result:** A page displaying the users ‘inbox email should load, showing any new messages.



	A	B	C	D	E	F	G	H	I	J	K
1	Test Case ID		BU_001	Test Case Description		Test the Login Functionality in Banking					
2	Created By		Mark	Reviewed By		Bill		Version		2.1	
3											
4	QA Tester's Log		Review comments from Bill incorporated in version 2.1								
5											
6	Tester's Name		Mark	Date Tested		1-Jan-2025		Test Case (Pass/Fail/Not		Pass	
7											
8	S #	Prerequisites:				S #	Test Data				
9	1	Access to Chrome Browser				1	Userid = mg12345				
10	2					2	Pass = df12@434c				
11	3					3					
12	4					4					
13											
14	Test Scenario	Verify on entering valid userid and password, the customer can login									
15											
16	Step #	Step Details		Expected Results		Actual Results			Pass / Fail / Not executed / Suspended		
17											
18	1	Navigate to http://demo.guru99.com		Site should open		As Expected			Pass		
19	2	Enter Userid & Password		Credential can be entered		As Expected			Pass		
20	3	Click Submit		Cutomer is logged in		As Expected			Pass		
21	4										
22											

Testing, Test Results and the Testing Artifacts

- During testing **screen shots** may be captured and recorded.
- During testing the database may have to get **restored multiple times**.
- **Test Stubs** may have to be written to be able to execute more test cases.
- **Multiple rounds of regression** testing may have to be performed.
- **Automation test scripts** should be written to speed up the testing.
- All test cases have to be in the **Test Management System** before execution.
- **Test Reports** should show the clear execution status. Passed, Failed, Executed.

Test Scenarios & Test Cases Where to start? The Loan Application Example

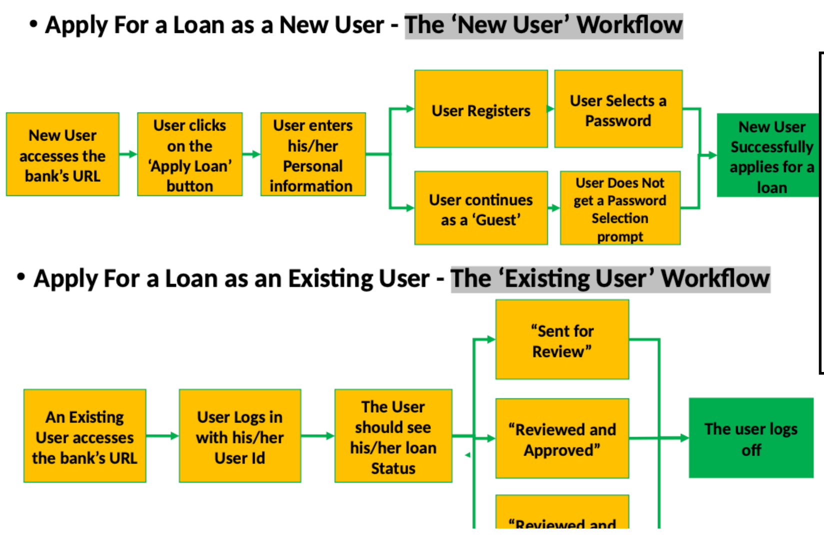
---

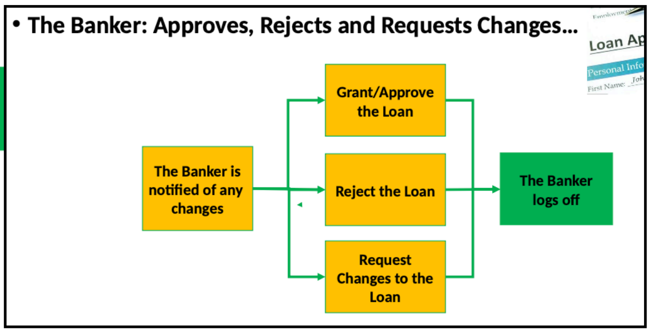
Steps to Writing Test Cases?

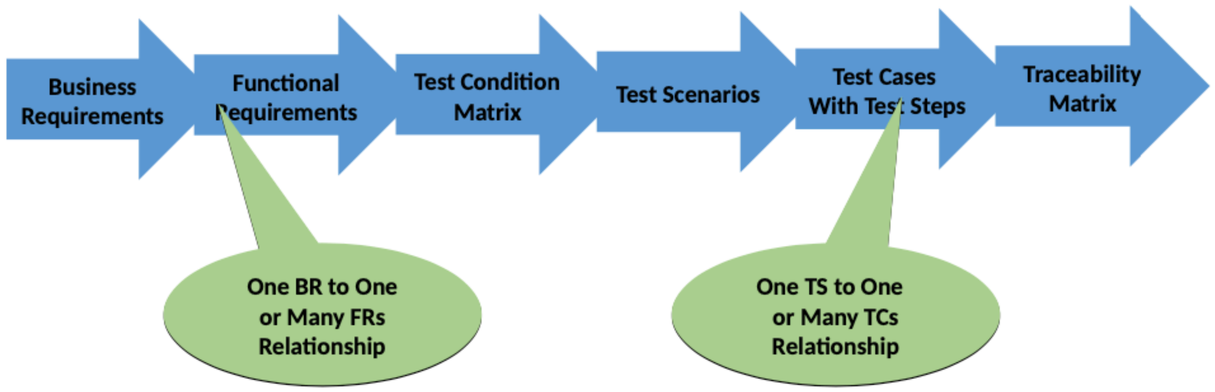
1. Read the Business Requirements (BRD)
2. Read and analyze the Functional Requirements(FRD)
3. Write the Test Plan / Test Strategy document
4. Create the Test Condition Matrix and come up with the test permutations
5. Perform the Equivalence Class Partitioning to reduce the test permutations
6. Create the test scenarios and link them to the Functional Requirements
7. Create the test cases with the test steps / expected results and actual results
8. Create the Traceability Matrix

Example of BRD/FSD/Test Condition Matrix

- The Loan Application
  - **The Business:** Delivered the Business Requirements.
  - **The BSA:** Completed the Functional Requirements (Requirements or the Use Cases)
  - **QA:** Created the Master Test Plan/Test Strategy document.
  - **QA:** Created the Test Condition Matrix.
  - **QA:** Created the Test Scenarios and the Test Cases.
  - **QA and the BSA:** Created the Traceability Matrix.


• Apply For a Loan as a New User - The 'New User' Workflow
New User accesses the bank's URL
User clicks on the 'Apply Loan' button
User enters his/her Personal information
User Registers
User Selects a Password
User continues as a 'Guest'
User Does Not get a Password Selection prompt
New User Successfully applies for a loan
• Apply For a Loan as an Existing User - The 'Existing User' Workflow
An Existing User accesses the bank's URL
User Logs in with his/her User Id
The User should see his/her loan Status
"Sent for Review"
"Reviewed and Approved"
"Reviewed and
The user logs off


• The Banker: Approves, Rejects and Requests Changes...
Loan Ap
Personal Info
First Name: Joh
The Banker is notified of any changes
Grant/Approve the Loan
Reject the Loan
Request Changes to the Loan
The Banker logs off


Business Requirements
Functional Requirements
Test Condition Matrix
Test Scenarios
Test Cases With Test Steps
Traceability Matrix
One BR to One or Many FRs Relationship
One TS to One or Many TCs Relationship

Path to Creating Test Cases

P25 - P30有具体的sample

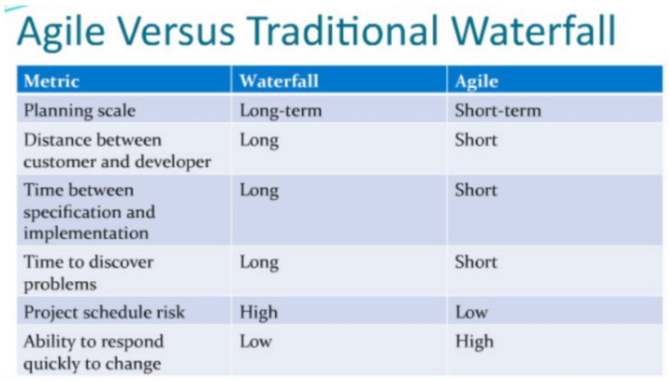
Agile

Agile Development

1. Adaptive rather than predictive
2. People oriented than process oriented
3. Project is divided into milestones
4. Minimum viable product (MVP)
5. Scrum teams
6. Milestones have sprints
7. Each sprint has multiple stories

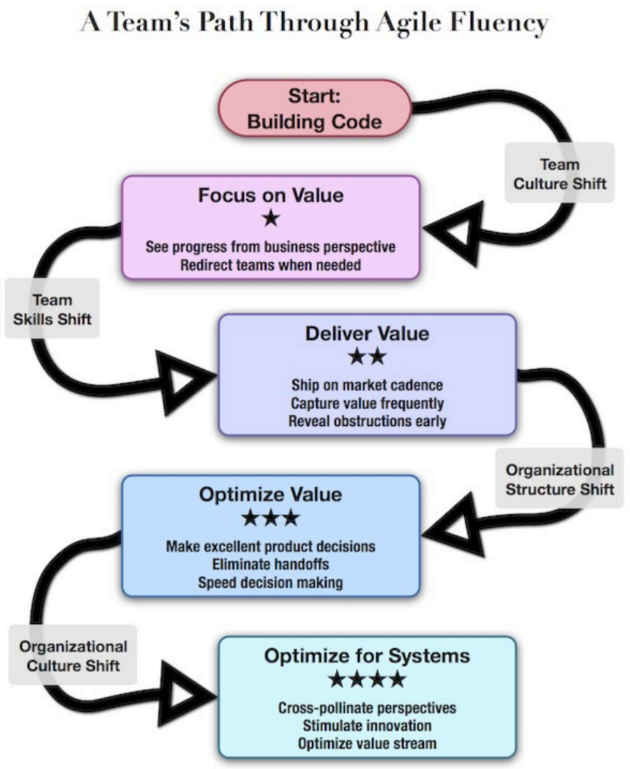
Automated Testing in Agile World

1. Each Sprint adds **new features** and refactoring as the design might evolve Previous code should continue to work



Agile Versus Traditional Waterfall

Metric	Waterfall	Agile
Planning scale	Long-term	Short-term
Distance between customer and developer	Long	Short
Time between specification and implementation	Long	Short
Time to discover problems	Long	Short
Project schedule risk	High	Low
Ability to respond quickly to change	Low	High


A Team's Path Through Agile Fluency
Start: Building Code
Team Culture Shift
Focus on Value ★
See progress from business perspective
Redirect teams when needed
Team Skills Shift
Deliver Value ★★
Ship on market cadence
Capture value frequently
Reveal obstructions early
Organizational Structure Shift
Optimize Value ★★★
Make excellent product decisions
Eliminate handoffs
Speed decision making
Organizational Culture Shift
Optimize for Systems ★★★★
Cross-pollinate perspectives
Stimulate innovation
Optimize value stream

2. **Automated testing** framework provides the firepower for this continuous development
3. Automated tests are **added incrementally** and developing a prefect testing framework is not the intention at the start
4. Meaningful tests **add more value** than automated everything

Principles behind the Agile Manifesto （12 Principles: ）

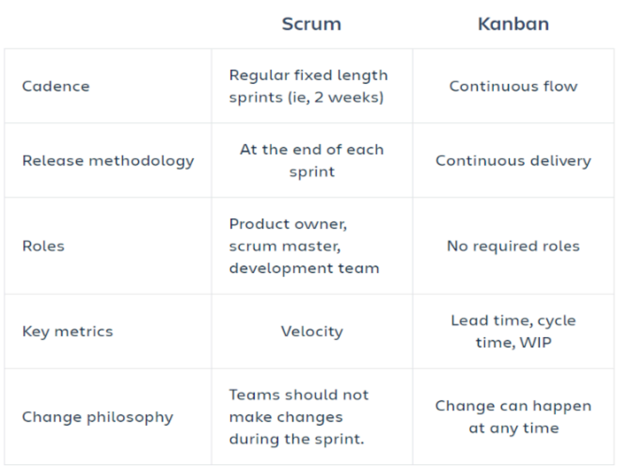
1. Our highest **priority is to satisfy the customer** through early and continuous delivery of valuable software.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver** working software **frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must **work together** daily throughout the project.
5. Build projects **around motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
6. The **most efficient and effective method** of conveying information to and within a development team is **face-to-face** conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a **constant pace** indefinitely.
9. Continuous attention to **technical excellence** and good design enhances agility.
10. **Simplicity**--the art of maximizing the amount of work not done--**is essential**.
    1. Make the design as simple as possible and don't incorporate every potential future requirement in the initial design.
11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
12. At regular intervals, the **team reflects on how to become more effective**, then tunes and adjusts its behavior accordingly

Agile – Scrum Framework

There are several Agile Frameworks: **Scrum, Kanban, XP and Others**

1. **Scrum:** A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.
    1. Scrum is **lightweight** and simple to understand.

    2. Scrum emphasizes **self-organizing**, cross functional teams.
    3. Scrum does **not prescribe any technical practices**, rather it **emphasizes** management practice such as the role of Scrum master.
2. **Kanban**
    1. **Kanban** is a **popular framework** used to implement agile software development. It requires **real-time communication** of capacity and full transparency of work.
    2. Work items are represented visually on a **kanban board**, allowing team members to see the state of every piece of **work at any time**. A **Kanban Board is also used to visualize work** and optimize the flow of the work among the team.
    3. While physical boards are popular among some teams, **virtual boards** are a crucial feature in any agile software development tool for their **traceability, easier collaboration, and accessibility** from multiple locations.
3. **Scrum vs. Kanban**
    1. **Kanban** is primarily concerned with **process improvements,** while **Scrum** is concerned with getting more **work done faster**.



	Scrum	Kanban
Cadence	Regular fixed length sprints (ie, 2 weeks)	Continuous flow
Release methodology	At the end of each sprint	Continuous delivery
Roles	Product owner, scrum master, development team	No required roles
Key metrics	Velocity	Lead time, cycle time, WIP
Change philosophy	Teams should not make changes during the sprint.	Change can happen at any time

Scrum Factors and Core Team values

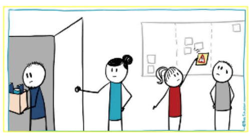
1. 3 Important **Factors of Scrum**:
    1. Transparency of the Process.
    2. Frequent Inspection of the Artifacts.
    3. Adaptation, or making adjustments after inspection.
2. Core **Team Values of Scrum**:
    1. Team Commitment to meet the goals.
    2. Value Focus to deliver better quality, faster.
    3. Openness with the team members to adapt better.
    4. Respect the team to allow them voicing concerns.
    5. Value Courage to allow greater challenge.
3. **Scrum Roles**
    1. Product Owner
    2. Scrum Master
    3. “The Team” • QA • Developers • BSA

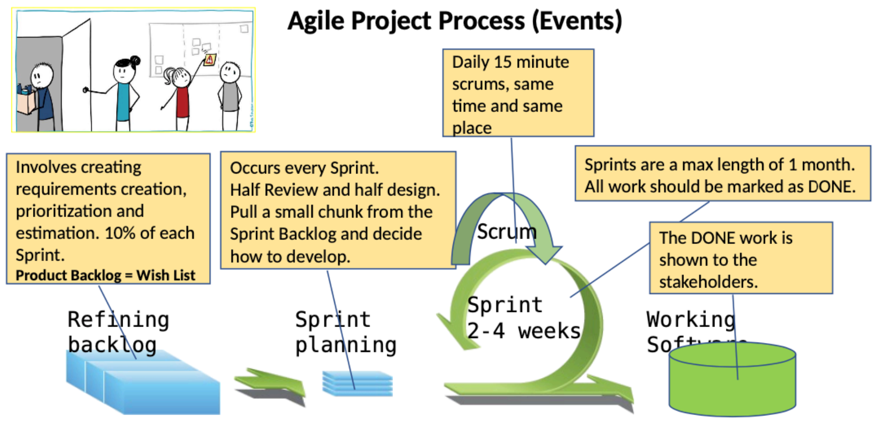
The Complete Agile Team

1. The Whole-Team Approach – The team works together on all projects:
    1. Project Manager
    2. Tester
    3. Business Analyst
    4. Business Representative
    5. Database Analyst
    6. Networking/Infrastructure
    7. User Experience
    8. Developer
2. The **Scrum Master** is responsible for the project’s success
    1. Role of the Scrum Master
        1. **Represents** management and the project
        2. **Responsible** for enacting Scrum values and practices
        3. **Remove** impediments

        4. **Ensures** that the team is productive
        5. **Enable** close cooperation across all roles and functions
        6. **Shields** the team for external influences
        7. Is a **Servant Leader**
    2. Don’ts
        1. Does not own the product decisions on product
        2. Does not make estimation on team’s behalf
        3. Does not assign the task to the team members
        4. does not manage the team
    3. What is a Servant Leadership?
        1. Change from a traditional leadership (workers need to be monitored closely)
        2. It is based on **trust** (the team members to be self- motivated)
        3. The Servant Leader:
            1. **Protects** the team from outside **distractions**
            2. **Removes** impediments
            3. **Facilitates** the team to address the tasks and resolve problems
            4. **Removes obstacles** that would prevent the team providing business values

3. Daily Scrums
    1. The scrum meetings criteria: • **Daily** • SameTime • SamePlace • Standup
    2. Everyone is invited, however only the team members and the scrum master can talk.
    3. 3 Questions are asked:
        1. What did you do yesterday?
        2. What will you do today?
        3. Is anything blocking you??
4. **The Product owner** is responsible for the product success
    1. The Product Owner role is **filled by one person** who has a vested interest in the product:
    2. End user and customer advocate : describe the product with understanding of the users and the functionality of the product
    3. business advocate : understand the needs of the organization for the product
    4. communicator: capable of the communicating vision and intent
    5. decision making: be the final decision maker for hard product decisions
    6. Product Owner Roles
        1. **Defines** the features of the product
        2. **Decides** on Release dates and contents
        3. Is responsible for the Total Cost of Ownership and the Profitability of the Product
        4. **Orders** features according to Market value
        5. **Adjusts** features and priority every sprint as needed
        6. **Available** daily to answer questions
        7. **Inspects** product progress
    7. The Product Owner Does Not
        1. **Choose** how much work will be accomplished in the Sprint.
        2. **Change** anything within the Sprint once it has started.
5. **The “Team”**
    1. The team’s Dos and Don’ts
        1. **Self organizing** (Self empowered, Responsible with no individual egos)
        2. Should include developers, testers, UI designers, etc.
        3. FullTime (FTEs)
        4. Should not change during Sprints
        5. Should not skip meetings
        6. **Should not stop** working when they hit roadblock




Agile Project Process (Events)
Daily 15 minute scrums, same time and same place
Involves creating requirements creation, prioritization and estimation. 10% of each Sprint.
Product Backlog = Wish List
Occurs every Sprint. Half Review and half design. Pull a small chunk from the Sprint Backlog and decide how to develop.
Sprints are a max length of 1 month. All work should be marked as DONE.
The DONE work is shown to the stakeholders.
Scrum
Refining backlog
Sprint planning
Sprint 2-4 weeks
Working
Soft

Sprints

1. Scrum projects progress via a series of Sprints
    1. During a **Sprint**: **the Scrum team takes a small set of features from Idea -> Coded -> Tested**
    2. At the end of the **Sprint** the team has gone through: Idea -> Coded -> Tested -> Integrated -> Delivered
    3. There should be no change during the print
2. Sprint Durations, Pros and Cons
    1. **Four-Week Sprint:**
        1. **Pros:** easy to roadmap, low process load
        2. **Cons**: Long turnaround and mini Waterfall
    2. One Week Sprint:
        1. **Pros:** Fast Turnaround, High Energy
        2. **Cons:** Minimal customer feedback, Lack of a roadmap, Relatively heavy or no process.
3. **The Scrum Events**
    1. **Sprint Planning:** 2 hours per week on Requirements & Design
    2. **Sprints:** One Month Max
    3. **Daily Scrums:** 15 minutes Daily
    4. **Backlog Refinement (grooming):** 2 hours/week
    5. **Sprint Review:** a couple of hours a week
    6. **Sprint Retrospectives:** a few hours for a one month Sprint
4. Agile Terminology
    1. **User Story –** A defined feature that the user desires.
    2. **Theme –** User Stories that have something in common**.**
    3. **Epic –** A large User Story that may be able to be broken down into smaller stories as they get ready for development.
    4. **Spike –** A Special user story used for investigation activity.
    5. **Product Backlog** – A list of desired work on the project
    6. **The Sprint Backlog** – The Product Owner works with the Development team to select stories based on a Sprint goal
    7. **Burndown Charts** – Tracks work remaining. Story Points vs. Days.
    8. **Sprint Burndown Chart** – Show work remaining
    9. **Release Burndown Chart** – Show work remaining for the entire release
    10. **Velocity** – Rate of Progress.
5. **Agile Planning**
    1. **Product Vision** – is key to success
    2. **Product Roadmap** – shows progress toward strategy
    3. **Release Plan** – Occurs once the vision and roadmap are complete
    4. **Sprint Plan** – Sprint Planning
    5. **Daily Plan** – Scrums
    6. **Minimum Viable Product:**
        1. The highest return on Investment.
        2. To test an idea by exposing an early version to target users

Problems with the Agile Implementation

1. It is **difficult to change** the culture of an organization.
2. **Waterfall** is in place and is hard to change.
3. Not many people like **"Change".**
4. **The Leadership** has not bought into it.
5. Agile can be implemented in any organization by: **Education,** Execution, Continuous Improvement