**Course Code: CSC824**

**Course Title: Advanced System Analysis And Design**

**Assignment On**

**Agile Methodoloy**

**By**

**Group Three**

**AGILE METHODOLOGY**

1. **What is Agile?**
2. **How Does Agile Works?**
3. **How Can Agile Be Used For Requirement Gathering?**
4. **Implementation of Agile.**

**What is Agile?**

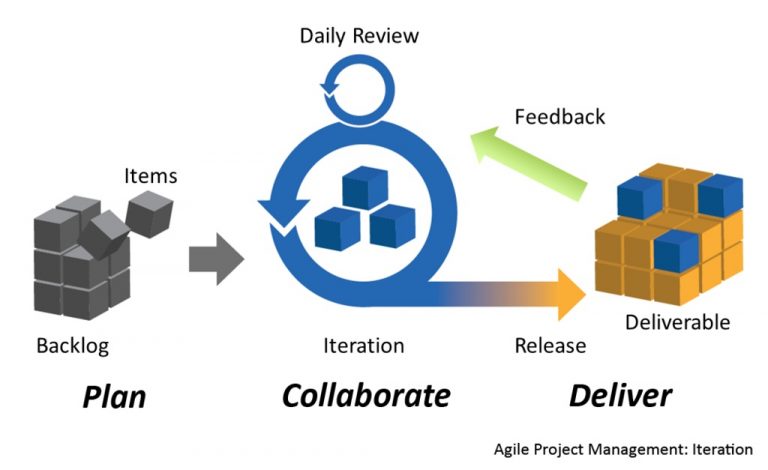
To be agile is to be able to adapt to change quickly and easily.

The Agile approach is an approach that breaks a project (software project) into services (Microservices). It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and evaluating. Continuous collaboration is vital, both with team members and project stakeholders.

Agile approach is an adaptive approach, which is based on the following philosophy: (**The Manifesto for Agile Software Development**)

Individuals and interactions over processes and tools  
Working software over comprehensive documentation  
Customer collaboration over contract negotiation  
Responding to change over following a plan.

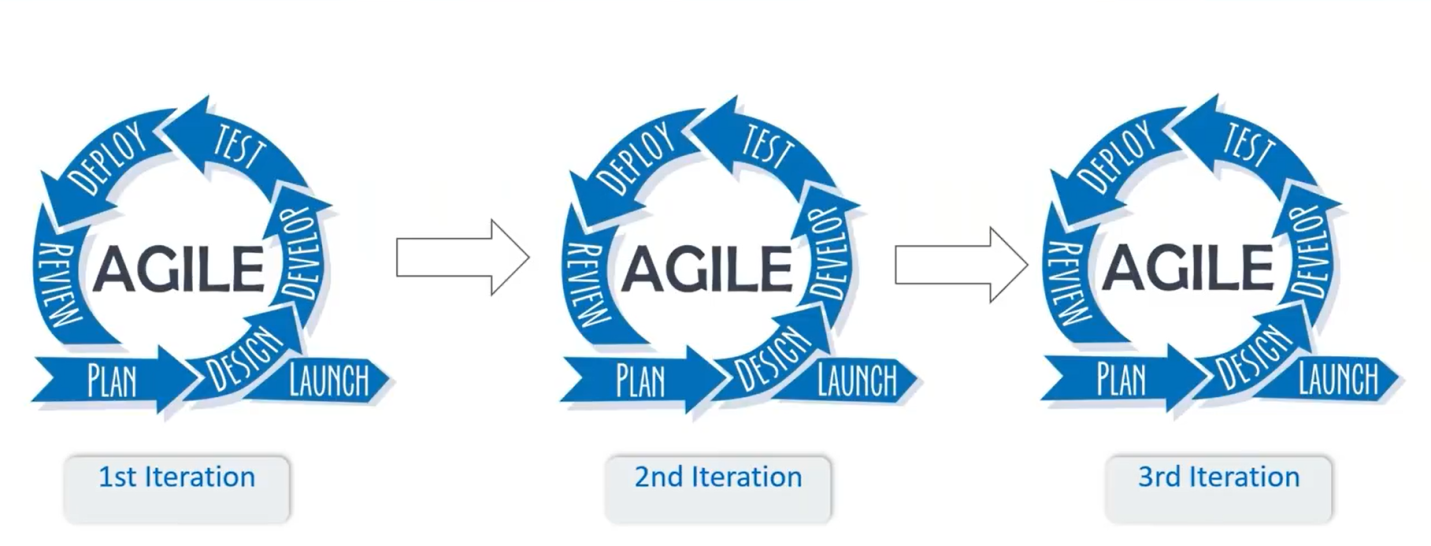
**How Does Agile Works?**

****

**Services**

**(prioritised)**

*Fig. 1.0, Agile approach.*



*Fig. 1.1, Agile iteration process*

The figure 1.0 above shows how Agile divides a project into services (microservices). Each one of these services is a desired feature the user wants in the software. Developers work through these services as you might a to-do list, working out which to prioritise and grouping them into iterations, with estimated deadlines for each iteration (usually around two weeks).

Once an iteration is complete, developers should have a potentially shippable product users can test. This means agile projects create something simple that they can then iterate on based on users’ feedback, making the software better suited to users’ needs while minimising complexity.

It means that developers don’t often start work with a full set of requirements, but instead discover new requirements through user feedback that they can then adapt their software to meet.

The sprint duration is always fixed. That ensures developers and users can regularly review the project’s direction and keep it on track. However, it also means that a project can run over schedule, unless developers decide to reduce its scope and ambition.

Fig. 1.1 above show in detail the iteration process in agile approach, which are:

Planning

Design

Develop

Test

Deploy

Review

**Planning Process**

Agile planning is a project planning method that estimates work using self-contained work units called iterations or sprints. Sprints are periods of 1-2 weeks in which a team focuses on a small set of work items and aims to complete them.

The planning process is done in the following manner:

1. **An agile project plan is divided into releases and sprints.**

Agile planners define a release as creating a new product or substantially updating an existing product. Each release is broken down into several iterations, also called sprints. Each sprint has a fixed duration, typically 1-2 weeks, and the team has a predefined list of work items to work through in each sprint. The work items are called user stories.

User stories are descriptions of a feature told from the perspective of the user.

1. **Planning is based on user stories**

A user story briefly describes a need experienced by your users.

Unlike in traditional project management methodologies like waterfall, in which teams would create detailed technical specifications of exactly what they would build, in agile planning, the team only documents what the user needs. Throughout the sprint, the team figures out together how to address that specific need in the best way possible.

1. **Planning is iterative and incremental**

All sprints are of equal length, and an agile team repeats the same process over and over again in every sprint. Each sprint should result in working features that can be rolled out to end-users.

An iterative process allows the team to learn what they are capable of, estimate how many user stories they can complete in a given timeframe (the team’s velocity), and learn about problems that impede their progress. These problems can be taken care of in subsequent sprints.

1. **Estimation is done by team members themselves**

A core ethic of agile planning is that development teams should participate in planning and estimation, and not have the work scope “dictated” to them by management.

**Design Process**

Design process involves multiple stages: Understand, Research, Sketch, Design, Prototype, Test, Refine.

An iterative design process starts from first stage and moves towards last stage. Then the output is analyzed, and multiple iterations are run to further improve it.

In this process, your goal is not to create a deliverable, but to improve the design in each iteration.

This process is done by dividing your required functionality into small (deliverable) iterations so that by end of an iteration, you are sharing your design to customer to get his feedback.

This process helps ensure:

**Quick Feedback**: It becomes easier to know timely that what customer wants, instead of providing him a complete prototype and then ask for feedback.

**Change Management**: Changes in design becomes easier and cheaper. No waste of time and money.

**Faster Development**: Delivering small chunks of design to development team also enables faster implementation. Development team does not wait for the complete design to start its implementation.

**Development process**

Development process in the iteration is the process of building or updating the services or the project in small dependable bits or parts according to their priorites. During the iteration, the team completes the service by building and testing the new functionality. Teams deliver services incrementally, depoying their work to the Product Owner as soon as they are done, enabling teams to arrive at the iteration review ready to show their completed work.

**Testing Process**

Unlike the waterfall model, in an agile model, a test process is written and updated for every release.

The agile testing process is separated into four parts called the agile testing quadrants.

1. Quadrant I
2. Quadrant II
3. Quadrant III
4. Quadrant IV
5. **Agile Quadrant I** – The internal code quality is the main focus in this quadrant, and it consists of test cases which are technology driven and are implemented to support the team, it includes:
6. Unit Tests
7. Component Tests
8. **Agile Quadrant II** – It contains test cases that are business driven and are implemented to support the team. This Quadrant focuses on the requirements. The kind of test performed in this phase are:
9. Testing of examples of possible scenarios and workflows.
10. Testing of User experience such as prototypes.
11. Pair testing
12. **Agile Quadrant III** – This quadrant provides feedback to quadrants one and two. The test cases can be used as the basis to perform automation testing. In this quadrant, many rounds of iteration reviews are carried out which builds confidence in the product. The kind of testing done in this quadrant are:
13. Usability Testing.
14. Exploratory Testing.
15. Pair testing with customers.
16. Collaborative testing.
17. User acceptance testing
18. **Agile Quadrant IV** – This quadrant concentrates on the non-functional requirements such as performance, security, stability, etc. With the help of this quadrant, the application is made to deliver the non-functional qualities and expected value.
19. Non-functional tests such as stress and performance testing.
20. Security testing with respect to authentication and hacking.
21. Infrastructure testing.
22. Data migration testing.
23. Scalability testing.
24. Load testing.

**Deployment Process**

At this point the completed or updated service(s) is deployed into the production environment, where they are accessed by the users and get feedback from users.

This process of deploying completed or updated services and getting feedback is also known as the **Continuous Deployment,** the continuous deployment process involves the following activities:

**Deploy to production** describes the practices necessary to deploy a solution to a production environment.

**Verify the solution** describes the practices needed to make sure the changes operate in production as intended before they are released to customers.

**Monitor for problems** describes the practices to monitor and report on any issues that may arise in production.

**Respond and recover** describes the practices to rapidly address any problems that happen during deployment.

**Review Process**

The Iteration Review is a cadence-based event, where each team inspects the increment at the end of every Iteration to assess progress, and then adjusts its backlog for the next iteration.

During the Iteration review, each Agile Team measures and then demonstrates its progress by showing working stories to the Product Owner and other stakeholders to get their feedback. Teams demonstrate the significant new behavior and knowledge gained from the iteration’s Stories.

The iteration review provides a way to gather immediate, contextual feedback from the team’s stakeholders on a regular cadence. The purpose of the iteration review is to measure the team’s progress by showing working stories to the Product Owner and other stakeholders to get their feedback. The iteration review serves three important functions:

1. It brings closure to the iteration timebox, to which many individuals have contributed to provide new value to the business.
2. It allows team members to demonstrate the contributions they have made and to take some satisfaction and pride in their work.
3. It provides an opportunity for the team to receive feedback to improve the solution under development.

**How Can Agile Be Used For Requirement Gathering?**

Agile requirements gathering is a practice teams often perform on the go. For example, developers update requirements between iterations.

Agile nurtures the requirements and solution so that they evolve into the final delivery. At the center of this nurturing process is the user story, brought into life at the very start, growing and maturing through each iteration and release, gathering more detail for the requirements of the end product throughout the whole process.

Requirements on an Agile project are gathered in the product backlog and detailed through user stories. A user story is a short, simple description of a feature told from the perspective of the user of the system.

**Some techniques used for gathering user stories:**

**User Interviews**

This very common approach to gathering user stories is certainly more effective if the interviews are conducted across a wide selection of users and roles. Try defining a set of questions asked of a selected set of users on a one-to-one basis. The keys to the success of this type of information gathering are the selection of the right interviewees, whether they are users or proxy-users (managers, IT staff, etc.), and asking the right kind of question. Having domain expertise and technical knowledge will certainly help in pulling the stories together.

**Prototyping**

Prototyping, an excellent way to present ideas and nurture communication, can highlight areas that had been overlooked and provide valuable insight into the user’s approach to work. You might be surprised how quickly users realize that what they have asked for is not necessarily what they want. The way you present the prototype is very flexible and it can be done by wire framing, Powerpoint or even sticky notes–as long as you can present the ideas and see the work flow, then anything is acceptable.

It is important to note: At the end of any prototyping session you must destroy the prototype. It is not a permanent project deliverable.

**Questionnaires**

Questionnaires are highly effective in gathering more information and prioritizing the user stories that are already in hand and have a large user population.

The questionnaire should not be the primary means of getting information from users. It neither allows you to follow the user down a path nor lends itself to follow-up questions. The ideal situation for using the questionnaire is when you require further detail about a specific area or you want information in order to prioritize stories.

**Observations**

The opportunity to observe how the users use an application can be rare, but the information obtained (by observation) during a meeting with user(s) can certainly improve their productivity and experience. This makes it an invaluable tool in the development of the user stories. That provides a developer with far greater insight when pulling together the user stories.

**Story-Writing Workshops**

Story-writing workshops are probably by far the best technique for identifying and formulating user stories. The technique, similar to brainstorming, brings together the users, Product Owner, Scrum Team, etc., who then discuss and identify as many user stories as possible. This workshop does not prioritize and estimate the user stories but solely identifies requirements. The workshop must be managed well; it can easily slip into design- and problem-solving side tracks. Remember that the purpose of the workshop to identify as many user stories as possible at a very high level.

**Agile Implementation**

A good Agile implementation requires the decentralization of control to deliver value in the shortest and sustainable lead time. In traditional project management, every decision must go through several "high-authority" approval steps before its realization. This results in massive project delays and reduces the team's morale.

Agile has a list of Framework that can be use for the implementation of agile. These frameworks are methods for applying the principles of Agile to a project. Think of Agile as the overall philosophy, and these frameworks as tools you use to carry out that philosophy.

**Agile Frameworks List**

1. **Scrum**

It is a team-centric framework, utilising clearly defined team roles and responsibilities to implement the responsive style of Agile project management.

It focuses on Team-led projects.

1. **Kanban**

Kanban is similar to Scrum, in that it aims to support teams working at top-level efficiency together.

It zooms in on the workflow aspect of a project, streamlining what is in progress and avoiding bottlenecks in productivity. Kanban usually involves use of a Kanban board or flowchart.

It focuses on Streamlining workflow.

1. **Extreme Programming**

Extreme Programming (XP) is popular in the software development sector due to its goal of frequent releases.

This approach is ideal for projects where continuous value delivery is a high priority.

It focuses on Value Delivery.

1. **Feature-Driven Development**

Feature-Driven Development is not a million miles from XP. It also seeks to deliver value to clients regularly throughout the lifecycle of a project.

This particular framework is client-centric, paying particular attention to stakeholder engagement.

It focuses on Stakeholder Management

1. **Crystal**

Crystal also dedicates a high level of focus to the project team, and places a lot of value on their ability to make key decisions on what's best for the project's success.

It’s a great option for smaller teams who prefer a “lightweight” approach to their project work – less documentation, reporting and micro-management.

It focuses on Self-Managing Teams

1. **Dynamic Systems Development Method**

Dynamic Systems Development Method (DSDM) Like XP and FDD respectively, it aims for regular value delivery and clear communication with stakeholders.

DSDM concentrates on delivering the project goals on time and within budget.

It focuses on Time & Budget Control.

**Implement Agile using the Scrum Framework**

Scrum is a team-centric framework, utilising clearly defined team roles and responsibilities to implement the responsive style of Agile project management.

The following steps are guidelines to implement Scrum.

1. **Define your first Scrum Team**

The team is comprised of of 5-9 members. These members all have a combination of competencies and can include developers, testers, support, designers, business analysis, etc. All the members continuously work closely together. The team itself is in charge of delivering shippable product increments by the end of each sprint.

1. **Define your Sprint length**

A sprint is a time-box that lasts between 7 and 30 days, and typically it remains the same length for the duration of a project. A sprint planning meeting proceeds each sprint where the work for the sprint is planned, and the team commits to completing this work. At the end of a sprint a review/meeting with a demonstration of the completed work is held. Here the improvements are reviewed and work for the next sprint is planned. If you don't have a clue of how long the time-box should be start with 2 weeks.

1. **Appoint a Scrum Master**

The Scrum Master is the catalyst of the scrum group. They ensure that the scrum group is effective and progressive. In the event of any impediment, the Scrum Master follows up plus resolves the issues for the group.

You can think of this as the Project Manager for the team, except the person shouldn't dictate what the team works on and shouldn't overly try to micro-manage anything. The Scrum Master will assist the team in planning the work for the coming sprints.

1. **Appoint the Product Owner**

The Product Owner should be a person that can be in charge of making sure the team produces value from the project to the business, client or whoever wants the project (the end buyer). The Product Owner typically writes the client-centric requirements in the form of stories, prioritizes them, and provides them to the backlog.

1. **Create the Initial Product Backlog**

The Product backlog is a wish list of all of the user stories (requirements) that is expected to be completed in the project. The most important story should be in the top of the list, so the entire backlog is continuously ranked in order based on story importance.

1. **Plan and Start your First Sprint**

Based on the backlog prioritization, the team now picks items from the list (typically from the top). The team brainstorms and decides on what and how much they can complete in the upcoming sprint. This is called the sprint planning meeting.

Once the team agrees, the sprint is started and the team starts working on the stories.

1. **Close the Current and Start the Next Sprint**

When the end of the time-box is reached, the end of the current sprint, all planned work should hopefully be done. If this is not the case it's up to the team to decide if the remaining work should transfer to the next sprint or be put back into the backlog.

The team now does a retrospective where they discuss what went well and what could be improved for the next sprint. After that, the sprint planning meeting for the next sprint starts and the process is repeated.

There's no limit for the amount of sprints, except if they are set by a deadline (based on budget or time), or the entire backlog is completed. If none of these criteria are met, the sprints just keep going indefinitely.

**References;**

<https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/>

<https://agilemanifesto.org/>

<https://monday.com/blog/project-management/agile-planning/>

<https://uxdworld.com/2019/01/31/agile-design-process/>

<https://www.guru99.com/agile-testing-a-beginner-s-guide.html>

<https://www.forecast.app/blog/implementation-of-scrum-7-steps>

<https://www.professionaldevelopment.ie/what-are-agile-frameworks>

<https://www.brighthubpm.com/agile/123103-assembling-a-toolbox-for-agile-requirements/>

https://www.scaledagileframework.com/continuous-deployment/