EPP Assignment Final Evaluation

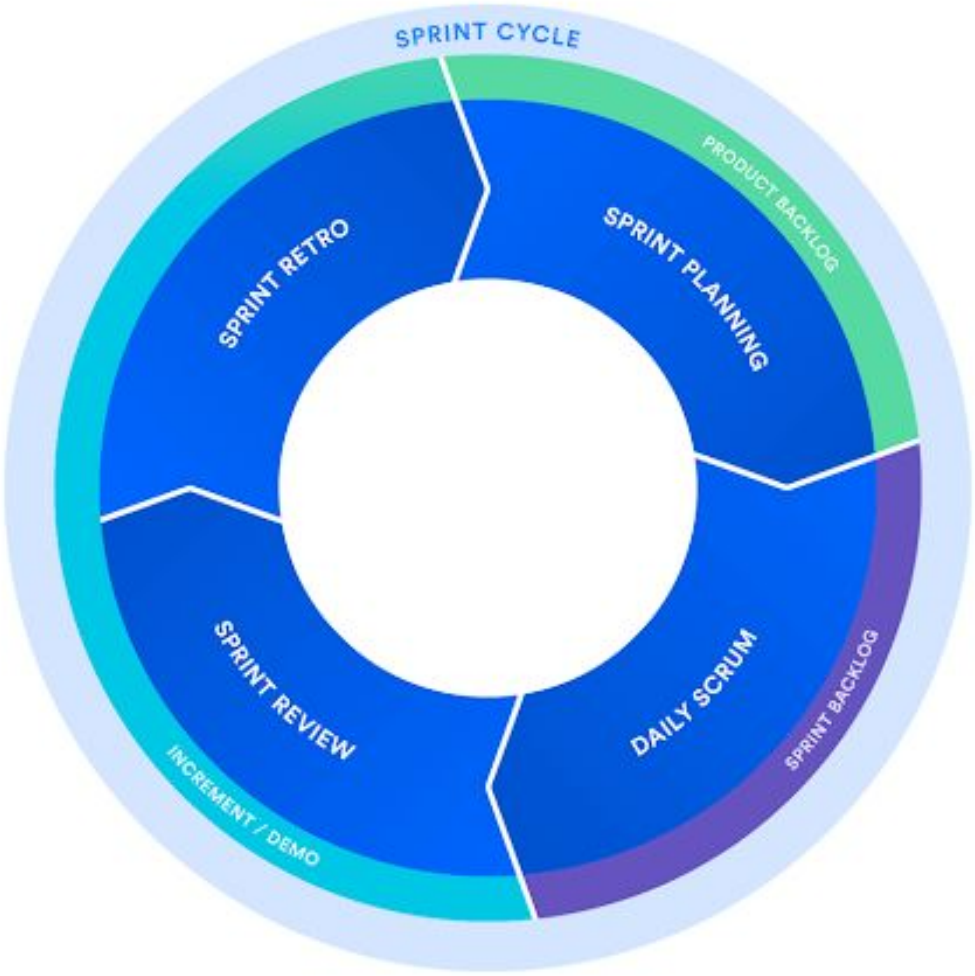
Mitul Hirna
2017UCO1689
COE-3

Scrum

Scrum is a framework used by companies to help teams work together. Scrum encourages teams to learn dynamically through experiences, taking onus of organisations themselves, and reflect on their wins and losses to continuously improve.

Scrum is often used due to its simple framework structure. The rules, artefacts, events, and roles are easy to understand. The self learning and functioning approach helps remove ambiguities in the development process, while giving sufficient space for companies to introduce their individualities in the functioning.

The clear demarcation of roles and planned events ensure that there is transparency and collective ownership throughout the development cycle. Quick releases keep the team motivated and the users and stakeholders happy as they can see the progress in a short amount of time.



Scrum Artefacts: tools to solve a problem

- **Product Backlog**: it is a to do list that is maintained by the product owner or product manager. The is a dynamic list of features, requirements, enhancements, and fixes that acts as the input for the sprint backlog. This list is dynamic in nature and keeps changing because the problem may not be relevant or solved through other ways, thus enhancing the utilisation of time.
- **Sprint Backlog:** This is the list of items, bug fixes and other reports that are selected by the development team for implementation in the current sprint cycle.

There is a sprint planning meeting and the team chooses which items it will work on for the sprint from the product backlog. There may be flexibility in the current sprint but the fundamentals sprint goal - what the team wants to achieve from the current sprint - cannot be compromised.

- **Sprint Goal:** This is the usable end product from a sprint. The duration of the sprint varies and so does its nature. The definition of these sprint is defined by the team.

Scrum Ceremonies or events

These are events and ceremonies that the team holds when following the scrum framework. These events and ceremonies are done to make checks and keep track of progress and re-evaluate the previous held sprints.

The following key ceremonies a scrum team might partake in are:

- **Organise the backlog:** This is the job of the product owner. It involves steering the project in the right vision and keeping in touch with the market need. Thus the product owner keeps track of the product progress and has a constant pulse of the market and the customer.

- **Sprint Planning:** This event is to decide on what objective the team has for the next sprint. Specific stories are used to align the team with the goal and are also agreed upon by the scrum team to be feasible to implement during the sprint. At the end of the sprint planning session, each member of the team needs to be clear on what can be delivered in the sprint and how the increment can be delivered.
- **Sprint:** This is the time during which the team comes together and works on the objective. The most common time for a sprint is usually two week. The sprint duration varies with each sprint session, depending on the complexity of the objective and the time available for development.
- **Daily Scrum (Stand up):** The goal of this meeting is to get the day's objectives and goals set. This is usually a 15 min meeting that is used to address the team, to get a focussed 24 hour results.
  This usually involves answering three questions from each team member: What did I do yesterday ?, What do I plan to do today ? Are there any obstacles ?

- **Sprint Review:** At the end of the sprint, the team gets together for an informal session to view a demo, or inspect, the sprint goal. The team is now open to feedback and reviews for refining the product. This is usually conducted by the Product Manager.

- **Sprint retrospective:** The retrospective is where the team comes together to document and discuss what worked and what didn't work in a sprint, a project, people or even ceremonies.

There are three important roles for scrum success:
Product Owner, Scrum master, and the development team.

- **Product Owner:** They are the owner and the leader of the project.. Closely partner with the business and the team to ensure everyone understands the work items in the product backlog. They have control over the shipment of the product and the final say in the product.
- **Scrum master:** The scrum masters are champions within the team and advise on how to fine tune the several aspects of the project. An effective scrum master deeply understands the work being done by the team and can help the team optimize their transparency and delivery flow. The facilitate all team related activities.
- **Development team:** The development team get the work done and are often small enough to eat two pizzas. The team consists of different members of different skill set and they work with the scrum master and the product owner to bring the project to life.

Lean Development

Lean software development is a translation of lean manufacturing principles and practices to the software development domain, primarily adapted from the Toyota Production System, it is emerging with the support of a pro-lean subculture within the Agile community.

It offers a solid conceptual framework, values and conceptual framework, values and principles, as well as good practices derived from experience, that support agile organisations.

The principles of lean development are

1. **Eliminate waste:** With respect to software projects and other such domains, the waste include: Building the wrong feature or product, Mismanaging the backlog, Rework, Unnecessarily complex solutions, Extraneous cognitive load, Psychological distress, Waiting/multitasking, knowledge loss, ineffective communication. A system is deployed in order to identify the waste, called a value stream mapping.

2. **Amplifying Learning:** Due to the nature of software development, the measure of the software is by the fitness for use and not in conformance of the requirement. And so excess importance is placed on the exploration of ideas than adding more documentation or detailed planning. The code written is immediately tested by running tests. The learning cycles are short and frequent, with feedback sessions. There is relevant customer and developer feedback taken into account before beginning the next short cycle.
3. **Decide as late as possible:** Since software development is always associated with some uncertainty, better result are achieved with a set of set-based or option-based approach, delaying decisions as much as possible and predictions. With the number of approaches and possible solutions that can be discussed , this brainstorming produces the best possible solutions. This also helps in bringing about the best decision regarding the crucial decisions. The market demand is also carefully analysed in this.
4. **Deliver as fast possible:** In software development, the fastest and the software that is in the market is the one that survives and not necessarily the best one. Keeping this in mind, the team release the updates as soon

as possible and get customer and developer feedback, and the changes are incorporated in the stand up meeting and the next sprint.

5. **Empower the team:** Software development is more than just mental grind, it requires a higher purpose and high motivation to produce appreciable results. This is why it is essential to support and empower the team of developers. This is a turnaround where the manager is taught and learns how the work is done so as to better facilitate the team and better manage and in the bigger picture bring better results.
6. **Build integrity in:** The product or the system must be an overall experience for the customer. That is how integrity is perceived. As more features are added to the original code base, the harder it becomes to add further improvements. The verified system is how integrity is perceived and which is a priority for the customer.
7. **Optimize the whole:** The modern software system are not simply the sum of their parts, but also the product of their interactions. Defects in software tend to accumulate during the production of the code. Due to the integration of the code, the part is made to be highly optimised. The entire system should be made a priority when it comes to the optimisation of the system as compared to a part.

Extreme Programming

Extreme programming is a type of agile software development methodology, which is intended to improve software quality and responsiveness to changing customer requirements. It focuses on having fast releases which is intended to increase productivity and introduce checkpoints at which new customer requirements can be adopted.

- **Goals:** XP attempts to reduce the cost of changes in requirements by having multiple short development cycles instead of long cycles. The changes during these cycles are inescapable and hence are planned for and this contributes to the original goal of XP of increasing productivity.

- **Activities:** XP consists of four basic activities that are performed within the software development process: coding, testing, listening, and designing. They are as follows:
    - **Coding:** Coding is the language that is spoken in an XP environment. The most efficient form considered is code, the developers can best communicate in code, they give feedback by writing and editing code.
    - **Testing:** Testing is essential in the world of XP. More the testing, more flaws can be solved. XP utilises Unit Tests and Acceptance tests with system-wide integration testing being encouraged. System wide testing varies as per schedules and stability.
    - **Listening:** Listening is essential and better communication must be established with the customer regarding the technical aspects of the problem will be solved. This is called planning game.
    - **Designing:** System designs are continuously tested and redesigned to remove dependencies and keep the system as independent as possible.

- **Values:** XP runs on five values which are as follows:
    - **Communication:** Formal software development requires a lot of communication and institutional communication is broken down in XP. This is replaced by verbal communication and feedback.
    - **Simplicity:** The simplest solutions are pushed forward in extreme programming.
    - **Feedback:** Feedback is essential in XP is quite essential and is often observed in terms of customer feedback, unit tests and team reviews.

- **Courage:** Courage is essential in teams that use XP so as to be confident in the code produced as XP focuses on quick releases and thus is an essential attribute to have.
- **Respect:** Programmers should have respect for others and self-respect. This is because they should honor each other's work.

Adaptive Software Development

Adaptive software development (ASD) is a software development process that embodies the principle that continuous adaptation of the process to the work at hand is the normal state affairs.

It is a development process that replaces the traditional waterfall cycle with a repeating series of speculate, collaborate and learn cycles.

- **Speculate:** Speculation is essential in this case as planning can be too definitive, contrary to the philosophy of the development model.
- **Collaborate:** Due to the adaptive nature of the development model, several features are added at the current time and knowledge may not be with one team, hence collaboration is required.
- **Learn:** The reviews, feedbacks, mistakes help the team to understand the market and customer feedback. This allows the team to understand the requirements better.

The characteristics of an ASD life cycle are that it is mission focused, feature based, iterative, timeboxed, risk driven and change tolerant.

During the iterations of ASD, the knowledge gathered by making small mistakes based on false assumptions and correcting those mistakes, thus leading to greater experience and eventually mastery in the problem domain.

Feature Driven Development

Feature driven development is an agile framework that, as its name suggests, organises software development around making progress on features. Features in the FDD context, are not necessarily product features in the commonly understood sense.

The project life cycle has the following constituents:

- **Develop an overall model:** Developing a high level design and then through detailed domain models are presented by small groups for peer reviews. The specifics once decided are proceeded with.
- **Build a feature list:** This is a list of features that is obtained from the model development and then functionally decomposing each domain into subject areas, which consist of features. The features are not too long, else are broken down into smaller features.
- **Plan by feature:** The obtained feature set is planned down in the form of development plan and the programmers are set to work.
- **Design by feature:** A team leader selects the set of features that are to be developed in a small period of time. After the details are written down, a design inspection takes place.
- **Build by feature:** Class owners develop codes, which are integrated into the main system after they build pass all the unit and integration.

Feature completion is monitored by the completion of milestones. Each feature has multiple milestones to be completed sequentially. The first half is done in the design by feature part, and rest are completed in the latter part of the Build by features.