EMERGING PROGRAMMING PARADIGMS
ASSESSMENT ASSIGNMENT
GURSHAAN SINGH BAJAJ
2017UCO1627

Write short note on following:

1. Scrum

The Scrum framework is a light weight framework for developing, managing and maintaining complex products. Foundation of Scrum is Lean Thinking and Empirical process control, that is, transparency, inspection, and adaptation.
Scrum is a set of rules and guidelines that provides important structure to the work-flow to help the team cut down on the technical, business and interpersonal complexity of software development as a whole and empowers the team to achieve large and long term goals. Scrum's approach to product development is iterative and incremental with multiple feedback loops – also called inspect and adapt loops, inbuilt with the process.

Scrum is based on three pillars. Transparency, Inspection, and Adaptation.

Transparency: Transparency allows everyone on the Scrum Team and outside to align and optimize the goals of the Scrum Team, self-organize based on the available information and adapt and plan ahead on the basis of the feedback.
Inspection: In the presence of information available transparently, the members of the Scrum Team are able to inspect any difference between actual results and the expected results.
Adaptation: Due to regularly and fast moving goals, customer needs, competition, etc. there is an ongoing need to adjust the direction of the products/services, organizations offer. Scrum events offer opportunities for regular Inspection and Adaptation.

Scrum recognizes and describes 3 roles. The Product Owner, the Development Team, and the Scrum Master.

Product Owner: The PO is responsible for maximizing the Return on investment (ROI). The Product Owner does so by collaborating with the customer to understand business needs and setting the direction of the product.
Development Team: The Dev Team follows product vision and direction set by the Product Owner and helps translate it into a valuable software product. The development team members share the ownership and contributions to build, deliver and support the product.
Scrum Master: The Scrum Master serves the Product Owner to achieve the product goals, the development team to become a high-performance team and the organization to become a learning organization. The Scrum Master acts as a servant leader and team coach.

Scrum Values
Focus: Focus on delighting the customer by delivering highest value features first.
Openness: Being open to all stakeholders with information about the work.
Respect: Respect for people and their freedom to express.
Courage: Being courageous to challenge mediocrity, the status quo, top-down command, and hierarchy.
Commitment: Committing to serving customers through valuable and quality products.

2. Lean Development

Lean development is the way to apply Lean principles to software development. The objective of this method is to optimize the efficiency of the entire development process by eliminating unnecessary practices that does not have any positive impact on the outcome of the project.

Here are 5 practices of lean software development.

Eliminate waste
The key element of practicing Lean is to eliminate that does not add value. Development of unnecessary code or functionality delays the delivery time to customer and also slows down feedback loop.

Create knowledge
Another basic principle of Lean Development is to Create Knowledge. This requires discipline and focus to implement. Creation of knowledge encourages Lean teams to offer the infrastructure to property document the code.

Deliver Fast
Every software development team wants to deliver fast and put the final product into hands of the customer. Tech teams spend a generous amount of time in figuring out why software development teams cannot deliver fast. One of the most common roadblocks are unclear requirements, and over-engineering of simple things.

Respect People
People working in a development team is often the most neglected aspect in software development. Respecting people working in a development team is the essential to work effectively as a team.

Optimize the whole
Optimization is a serious issue in software development teams. When developers feel pressured to deliver, they release the kind of code that often does not meet quality requirements. Suboptimization is a process of optimizing the whole.

3. Extreme Programming (XP)

Extreme Programming is a very popular set of practices and are a part of the Agile family. Extreme Programming fits very well within other processes like Scrum. It is a set of values, principles and practices that are used to breed quality from and within the development team.
Extreme Programming differs from Scrum in that Scrum is a framework for delivery software and XP is a set of practices that the team does to reach excellence. For this reason they work very well together.

Values of Extreme Programming
- Individual values on how we see our work, the things that make us happy with our work, the way we are perceived and the progress we make.
- Team values about what the shared team produces, how the team interacts. Definition of a team for a particular set of stories, discipline, features etc.
- Organizational values. What are the values we expect our organization to have. Both internally and externally.

Principles
Principles are a bridge to turn our values into specific concepts related to software development. The principles can help decision making and can be translated into practices which we as an organization and individuals can practice each day.

The fundamental principles of XP are:
- Rapid feedback
- Assume Simplicity
- Incremental Change
- Embracing Change
- Quality Work

Practices
Practices come from principles.The core practices are:

1. Sit Together - More discussion and better working happens when the team sits together.
2. Whole team - All the skills must be present on the team to deliver the product.
3. Informative workspace - The environment that the team works in should be visual and this allows total transparency of what everyone is working on and any issues that arise.

4. Energized work - Teams do not work longer than 40 hours per week. Anymore and productivity rapidly decreases.
5. Pair Programming - Developers sit together and program in a pair. One person drives and the other navigates.
6. Stories - Stories are used as place holders to conversation, they are flexible and different than requirements.
7. Weekly Cycle - The development iteration is a week long.
8. Quarterly cycle - Larger amounts of work, such as epics, releases etc. should be release every quarter.
9. Slack - The team needs to build in Slack. A highly productive and energized team needs time out to maintain their usual pace.
10. Ten Minute Build - The code base should be able to be automatically built and the tests run in 10 minutes or less.
11. Continuous Integration - All the integration tests should be run automatically and frequently in order to ensure the entire system integrity is sound.
12. Test First Programming - Tests should be written that match the acceptance criteria for the stories and then the code should then be written to make the tests pass. This is TDD (test driven development).
13. Incremental Design - The design of the system should evolve as development unfolds, rather than a huge upfront detailed design.

4. Adaptive Software Development (ASD)

Adaptive Software Development (ASD) is a method for the creation and development of software systems. It focuses on rapid creation and evolution of software systems. ASD is a part of rapid application development.
Adaptive software development life cycle is mission focused, feature based, iterative, time-boxed, risk driven, and change tolerant. In adaptive software development, the developer has a basic idea in mind and they go to work. The focus is in the computer code.

In ASD, there are no pre-planned steps. The software is made very quickly. New versions can come out very quickly as the development cycle is very short. The structure of adaptive software development and rapid application development are similar, difference lies in the fact that:

- adaptive software development does not allow the time when the project is finished, rapid application development does.
- adaptive software development does not have a real end point whereas rapid application development allows the end of project.

Adaptive Software Development life cycle comprises of three phases:

- Speculation
- Collaboration
- Learning

In speculation, user requirements are understood and an adaptive cycle planning is conducted. It depends on bug and user reports.
Effective collaboration with customer is very important. Communication, teamwork, individual creativity are a part of effective collaboration. The individual developers combine their portions of work.
Learning cycles are based on the short iterations with design, build and testing. ASD teams can learn through focus groups, formal technical reviews and postmortems.

5. Feature Driven Development

The FDD (Feature-driven development) propose:

- short and large iterations
- to have better communication throughout the development
- to make frequent deliveries with real done works
- accurate progress information
- to have processes appreciated by customers, developers and managers

An iteration in FDD (Feature-driven development) consists of 5 different steps:

1. Create the system model with a UML class diagram

2. List the features to achieve
3. Assign features to developers
4. Create the template for each of the features
5. Develop each of the features

Feature Driven Development (FDD) Features

Let's work by feature
The FDD (Feature-driven development) strongly favors the feature-based development. A "feature" must be possible to develop in two weeks like a user-story. If not, split the feature into two separate features.
The FDD (Feature-driven development) teams are defined in Feature Team; they will be responsible for their parts of development; the feature team will also be chosen in the future for developments concerning their scope.

Developers are responsible
This a practice that is totally in opposition to the agile methods which is currently being used within companies is the "Class Ownership". Basically, a class in the code is owned by a developer (or two developers) as long as he is in the company. The responsive developer will work on this code in the future, if it's necessary.

Regular builds
The FDD (Feature-driven development) imposes the fact of having regular builds in order to regularly test the product advancement. The feedback recovery is an essential point in agile products.

Progress reporting tool
The FDD (Feature-driven development) imposes the fact of making progress reporting as in Scrum. It is very important in the projects to have transparency on the progress of the project in order to be able to manage the obstacles as soon as possible.