Emerging Programming Paradigms

Assignment

(Shradha Khapra - 2017UCO1664)

Q. Write short notes on following:

1. Lean Development
2. Extreme programming (XP)
3. Adaptive Software Development (ASD)
4. Feature Driven Development
5. Scrum

Ans. 1.

Lean Development

Lean development is the application of Lean principles to software development. Lean principles were initially developed as a way to optimize the production line to minimize waste and maximize value to the customer. Primarily, lean development is based on 7 principles:

1. Waste Elimination - Waste is defined as anything that is capable of reducing code quality, hindering time and effort, or reducing delivered business value. The examples of such waste are unnecessary code or functionality, programming delays, unclear requirements, and insufficient testing. Lean development focuses on eliminating these factors.
2. Amplifying Knowledge - Lean development also focuses on learning required technologies and gaining understanding about what the user really requires.
3. Delaying Commitment (or late decision making) - Lean development requires ample delay in finalizing decisions as it leaves for minimal space for changes at the end, hence reducing cost of changes.

4. Delivering Fast - Iterative development is utilized to deliver new applications or enhancements as quickly as possible.
   Eg - Engineers using Lean development came up with the concept of MVP (minimum viable product). This approach aids to enhance a software incrementally, and on the basis of the feedback provided by end users, and discards parts of no value.
5. Team Empowerment - Lean software development is a system that is directly aimed at empowering members of the team. An approach of such type is necessary to keep the developers motivated and contributes to a faster application of changes to software products that are needed to reflect the changes in the environment.
6. Built-in Integrity -A built-in integrity is also a guiding principle in lean development methodology, as various issues in this area lead to waste of different types. Repetitive testing of the code, mistakes in logging and resolving them tends to take time and thus can potentially increase cost of development. Therefore, lean addresses such nuances before they can even take place.
7. View application as a whole - Lean software development opposes breaking application into components and stands for focusing on value stream as a whole. This helps in ensuring that there aren't disruptions or miscommunications, which can surely occur if members were divided across various sub-teams.

Thus, it can be said that Lean principles do allow optimization of the team's workflow and creation of a sense of unity among everyone involved in the project. This shared responsibility leads to translation of work into higher performance.

2.

Extreme Programming (XP)

Extreme Programming (XP) is a lightweight, efficient, low-risk, flexible, predictable and scientific way to develop a software. eXtreme Programming (XP) was initiated and developed to address the targeted needs of development of software by small teams. These teams may have requirements that are vague and changing.

Extreme Programming is a form of the Agile software development methodologies. XP provides principles that help guide the behavior of the team. The team is expected to self-organize. XP’s core values have following properties −

- Each practice is simple and self-complete.
- It is a combination of practices that produces more complex and emergent behavior.

A key assumption of Extreme Programming is that the cost of changing a program can be held mostly constant over time.

This can be achieved with −

- Emphasis on continuous feedback from the customer
- Short iterations
- Design and redesign
- Coding and testing frequently
- Eliminating defects early, thus reducing costs
- Keeping the customer involved throughout the development
- Delivering working product to the customer

Extreme Programming involves −

- Writing unit tests before programming and keeping all of the tests running at all times. The unit tests are automated and eliminate defects early, thus reducing the costs.
- Starting with a simple design just enough to code the features at hand and redesigning when required.
- Programming in pairs (called pair programming), with two programmers at one screen, taking turns to use the keyboard. While one of them is at the keyboard, the other constantly reviews and provides inputs.
- Integrating and testing the whole system several times a day.
- Putting a minimal working system into the production quickly and upgrading it whenever required.
- Keeping the customer involved all the time and obtaining constant feedback.

Iterating facilitates the accommodating changes as the software evolves with the changing requirements.

3.

Adaptive Software Development (ASD)

Adaptive Software Development is a method to build complex software and system. ASD focuses on human collaboration and self-organisation. Adaptive Software Development is cyclical like the Evolutionary model, with the phase names reflecting the unpredictability in the complex systems. The phases in the Adaptive development life cycle are −

1. Speculate

   The term plan is too deterministic and indicates a reasonably high degree of certainty about the desired result. The implicit and explicit goal of conformance to plan, restricts the manager's ability to steer the project in innovative directions.

   In Adaptive Software Development, the term plan is replaced by the term speculate. While speculating, the team does not abandon planning, but it acknowledges the reality of uncertainty in complex problems. Speculate encourages exploration and experimentation. Iterations with short cycles are encouraged.

2. Collaborate

   Complex applications are not built, they evolve. Complex applications require that a large volume of information be collected, analyzed, and applied to the problem. Turbulent environments have high rates of information flow. Hence, complex applications require that a large volume of information be collected, analyzed, and applied to the problem. This results in diverse Knowledge requirements that can only be handled by team collaboration.

Collaboration would require the ability to work jointly to produce results, share knowledge or make decisions.

In the context of project management, Collaboration portrays a balance between managing with traditional management techniques and creating and maintaining the collaborative environment needed for emergence.

3. Learn

   The Learn part of the Lifecycle is vital for the success of the project. Team has to enhance their knowledge constantly, using practices such as −

   - Technical Reviews
   - Project Retrospectives
   - Customer Focus Groups

   Reviews should be done after each iteration. Both, the developers and customers examine their assumptions and use the results of each development cycle to learn the direction of the next. The team learns −

   - About product changes
   - More fundamental changes in underlying assumptions about how the products are being developed

   The iterations need to be short, so that the team can learn from small rather than large mistakes.

These three phases reflect the dynamic nature of Adaptive Software Development. The Adaptive Development explicitly replaces Determinism with Emergence. It goes beyond a mere change in lifecycle to a deeper change in management style. Adaptive Software Development has a dynamic Speculate-Collaborate-Learn Lifecycle.

The Adaptive Software Development Lifecycle focuses on results, not tasks, and the results are identified as application features.

4.

Feature Driven Development

Feature-Driven Development (FDD) is a client-centric, architecture-centric, and pragmatic software process. Features are an important aspect of FDD. A feature is a small, client-values function such as - calculating the total sales, validating the user password or authorizing the sales transaction of a customer.

Typically used in large-scale development projects, five basic activities exist during FDD:

- Develop overall model

  The client and the development team make an overall model. Detailed domain models are created and then these models are progressively merged into the overall model. Guided by a chief architect, team members get a good understanding of the complete model.
- Build a Features List

  Information gathered in the 1st step is now deduced to make a list of required features. A feature is a small, client valued output. The whole project is thus divided into features. A feature needs to be delivered every two weeks. Therefore the feature the team decides to work on must take less than two weeks to be implemented.

- Plan By Feature

  Now the development of features is planned. It is all about in which order the features will be implemented. Teams are selected and assigned feature sets.

- Design By Feature

  The chief programmer chooses the features and the domain classes that will be involved in designing the feature. Sequence diagrams are drawn. General designs of the features are also finalized. Class and method prologues are written. It is all followed by a design inspection.

- Build By Feature

  After the design inspection, the domain expert explains the specifics, class owners start building and implementing all the items necessary to support the design. Code is developed, unit tested and inspected and approved by the Chief Programmer who then gives an ok and the completed feature is added to the main build.

5.

Scrum

Scrum is an agile project management methodology or framework used primarily for software development projects with the goal of delivering new software capability every 2-4 weeks. Although developed for agile software development, agile Scrum became the preferred framework for agile project management in general and is sometimes simply referred to as Scrum project management or Scrum development.

The components of Agile Scrum management team are:

1. The Product Owner - The Product Owner is the project's key stakeholder - usually an internal or external customer, or a spokesperson for the customer.
2. The ScrumMaster - The ScrumMaster is the servant leader to the Product Owner, Development Team and Organization. With no hierarchical authority over the team but rather more of a facilitator, the ScrumMaster ensures that the team adheres to Scrum theory, practices, and rules.
3. The Development Team - The Development Team is a self-organizing, cross-functional group armed with all of the skills to deliver shippable increments at the completion of each sprint.

A sprint in Scrum is a time-boxed period during which specific work is completed and made ready for review. Sprints are usually 2-4 weeks long but can be as short as one week.

Three pillars uphold every implementation of empirical process control: transparency, inspection, and adaptation.

1. Transparency
2. Inspection
3. Adaptation