ASSIGNMENT-01

Experiment Name: Agile Development Methodology Process

Course Title: CSE 326

Course Code: System Analysis and Design

Submitted To:

SUPTA RICHARD PHILIP

SENIOR LECTURER

DEPARTMENT OF CSE

CITY UNIVERSITY DHAKA

Submitted By:

Name: Md. Harun Or Rashid

ID: 171442552

Batch: 44th

AGILE METHODOLOGY

Agile methodology has taken the software development world by storm and rapidly cemented its place as "the gold standard." Agile methodologies all started based on four core principles as outlined in the Agile Manifesto. These methodologies are rooted in adaptive planning, early delivery and continuous improvement, all with an eye toward being able to respond to change quickly and easily. As a result, it's no surprise that 88% of respondents in VersionOne's 2017 State of Agile Report ranked "ability to adapt to change" as the number one benefit of embracing agile.

INRODUCTION TO THE PROJECT

The most widely-used Agile methodologies include:

- Lean and Kanban Software Development
- Extreme Programming (XP)
- Crystal
- Agile Scrum Methodology
- Dynamic Systems Development Method (DSDM)

Lean and Kanban Software Development

Lean Software Development is an iterative Agile methodology that focuses the team on delivering value to the customer and on the efficiency of the "Value Stream," which is the mechanism that delivers that value. It is a highly flexible, evolving methodology without rigid guidelines, rules, or methods.

The main principles of the Lean methodology include:

- Eliminating Waste
- Amplifying Learning
- Deciding as Late as Possible
- Delivering as Fast as Possible
- Empowering the Team
- Building Integrity In
- Seeing the Whole

Lean development eliminates waste by asking users to select only the truly valuable features for a system, prioritize those features, and then work to deliver them in small batches. It relies on rapid and reliable feedback between programmers and customers, emphasizing the speed and efficiency of development workflows. Lean uses the idea of a work product being "pulled" via customer request. It gives decision-making authority to individuals and small teams since this has been proven to be a faster and more efficient method than a hierarchical flow of control. Lean also concentrates on the efficient use

of team resources, trying to ensure that everyone is as productive as possible for the maximum amount of time. It strongly recommends that automated unit tests be written at the same time the code is written.

Kanban

Kanban is a highly visual workflow management method that is popular among Lean teams. In fact, 83% of teams practicing Lean use Kanban to visualize and actively manage the creation of products with an emphasis on continual delivery, while not overburdening the development team. Like Scrum, Kanban is a process designed to help teams work together more effectively.

Kanban is based on 3 basic principles:

- **Visualize what you'll do today (workflow):** Seeing all the items within the context of each other can be very informative
- **Limit the amount of work in progress (WIP):** This helps balance the flow-based approach so teams don't start and commit to too much work at once
- **Enhance flow:** When something is finished, the next highest priority item from the backlog is pulled into play

Kanban promotes continuous collaboration and encourages active, ongoing learning and improvement by defining the best possible team workflow.

Extreme Programming (XP)

Extreme Programming (XP), originally described by Kent Beck, has emerged as one of the most popular and controversial Agile methodologies. XP is a disciplined approach to delivering high-quality software quickly and continuously. It is intended to improve software quality and responsiveness in the face of changing customer requirements. It promotes high customer involvement, rapid feedback loops, continuous testing, continuous planning, and close teamwork to deliver working software at very frequent intervals, typically every 1-3 weeks.

The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels. As an example, code reviews are considered a beneficial practice. Taken to the extreme, code can be reviewed continuously through the practice of pair programming.

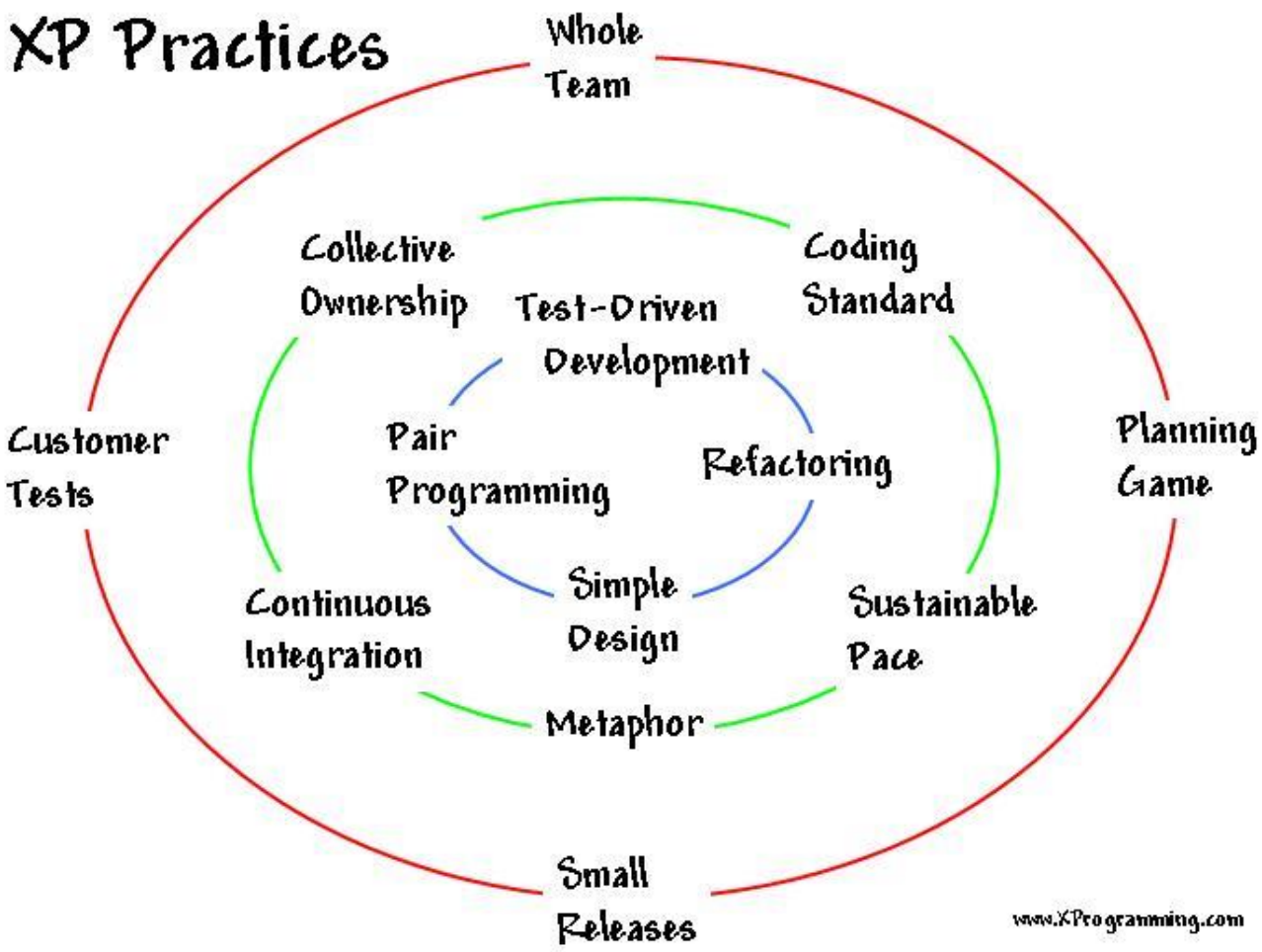
The original XP method is based on four simple values -€ " simplicity, communication, feedback, and courage.

It also has twelve supporting practices:

- Planning Game
- Small Releases
- Customer Acceptance Tests
- Simple Design
- Pair Programming

- Test-Driven Development
- Refactoring
- Continuous Integration
- Collective Code Ownership
- Coding Standards
- Metaphor
- Sustainable Pace

Don Wells has depicted the XP process in this popular diagram:



In XP, the customer works closely with the development team to define and prioritize user stories. The development team estimates, plans, and delivers the highest priority user stories in the form of working, tested software on an iteration-by-iteration basis. In order to maximize productivity, the practices provide a supportive, lightweight framework to guide a team and ensure high-quality software.

Crystal

The Crystal methodology is one of the most lightweight, adaptable approaches to software development.

Crystal is actually comprised of a family of Agile methodologies, including Crystal Clear, Crystal Yellow, Crystal Orange and others. Each has unique characteristics driven by several factors, such as team size, system criticality, and project priorities. This Crystal family addresses the realization that each project may require a slightly tailored set of policies, practices, and processes in order to meet the product ‘s unique characteristics.

Introduced by Alistair Cockburn, Crystal focuses primarily on people and the interaction among them while they work on a software development project. There is also a focus on business-criticality and business-priority of the system under development.

Unlike traditional development methods, Crystal doesn’t try to fix the tools and techniques of development but keeps people and processes at the core of the process. However, it is not only the people or the processes that are important, rather the interaction between them that is most important.

Several key tenets of Crystal include teamwork, communication, and simplicity, as well as reflection to frequently adjust and improve the process. Like other Agile methodologies, Crystal promotes early, frequent delivery of working software, high user involvement, adaptability, and the removal of bureaucracy or distractions.

DSDM’s Eight Key Principles:

1. Focus on the business need
2. Deliver on time
3. Collaborate
4. Never compromise quality
5. Build incrementally from firm foundations
6. Develop iteratively
7. Communicate continuously and clearly
8. Demonstrate control

Requirements are baselined at a high level early on in the project. Rework is built into the process, and all development changes must be reversible. Requirements are planned and delivered in short, fixed-length time-boxes – also known as sprints or iterations – and prioritized using MoSCoW Rules.

Feature Driven Development (FDD)

Feature Driven Development is a model-driven, short-iteration process that was built around software engineering best practices such as domain object modeling, developing by feature, and code ownership. The blending of these practices that resulted in a cohesive whole is the best characteristic of FDD.

Feature Driven Development consists of five basic activities:

- Development of an overall model
- Building a feature list
- Planning by feature
- Designing by feature
- Building by feature

FDD begins by establishing an overall model shape, which will result in a feature list. It then continues with a series of two-week "plan by feature, design by feature, build by feature" iterations. The features are small, "useful in the eyes of the client" results. If they will take more than two weeks to build, then they will have to be broken down into smaller features.

FDD's main purpose is to deliver tangible, working software in a timely manner, repeatedly. The advantage of using FDD is that it is scalable even to large teams due to the concept of 'just enough design initially' (JEDI). Because of its feature-centric process, FDD is a great solution to maintain control over incremental and inherently complex Agile projects.

Conclusion:

Agile models are based on iterative software development. An independent working module is built after the completion of iteration. Iteration should not consume more than two weeks to complete a code. Agile methodologies invite the developers to get involved in testing, rather than a separate quality assurance team.

Agile methodologies are suitable in changing environments because of new practices and principles that enable a team to develop a product in short duration.

Reference: