Subject: Roll No: 2213194, 2213196, 2113208

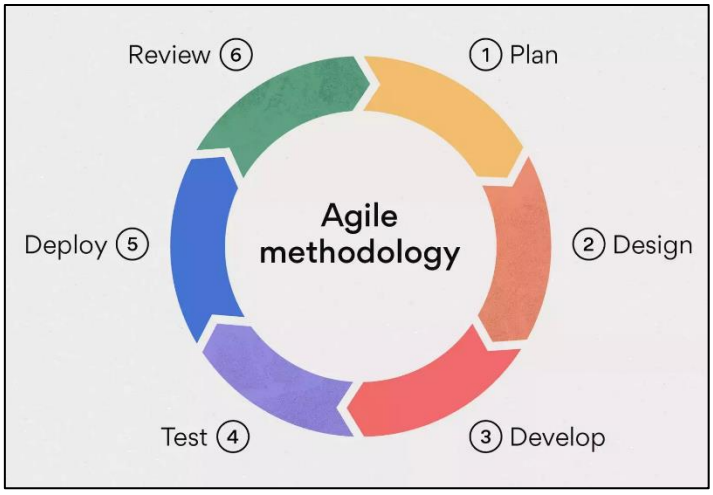
Experiment 2

Aim: Application of Agile Process Model on the project (JIRA)

Theory:

- What is Agile Process Model

  Agile is a methodology which consists of several iterative and incremental software development methodologies. The word agile refers to the ability to move quickly and easily. Therefore, an Agile approach enables project teams to adapt faster and easier compared to other project methodologies. The Agile methodology involves continuous planning, testing, integration and feedback from the stakeholders or clients throughout the life cycle of the software. The main objective of these models is to increase team involvement, and make quick decisions as per the situations. Agile methodology is mainly designed and developed to avoid common development issues during the software development life cycle and increase the overall efficiency of the development team. So, in a single sentence, the developments performed using Agile methodologies are normally developed and built iteratively and incrementally. In this methodology, a company can produce or deliver a quality product in less time and improve customer satisfaction at the end.



Need of Agile Process Model: -

a) Lower Cost
b) Enables clients to be happier with the end product by making improvements and involving clients with development decisions throughout the process.
c) Encourages open communication among team members, and clients.
d) Providing teams with a competitive advantage by catching defects and making changes throughout the development process, instead of at the end.
e) It keeps each project transparent by having regular consistent meetings with the clients and systems that allow everyone involved to access the project data and progress.

- Agile Process Model Types: -

  Each Agie method varies in the way it defines the steps of software development and the goal is to adapt the change while working on the software.

  1) Scrum: -

     Scrum, a dynamic Agile methodology, embodies a "inspect and adapt" philosophy. Unlike traditional linear approaches, Scrum emphasizes iterative progress and embraces change as a driving force. The heart of Scrum lies in its structured ceremonies:

     a) Sprint Planning: A collaborative session where the team selects a set of prioritized user stories for the upcoming sprint and outlines the tasks required to complete them.
     b) Daily Stand-up: A concise daily meeting where team members share updates on their work, discuss impediments, and align their efforts.
     c) Sprint Review: At the end of each sprint, the team showcases the completed features to stakeholders, eliciting feedback that fuels improvements and informs the product backlog.
     d) Sprint Retrospective: A reflective gathering where the team assesses their processes, celebrates achievements, and fine-tunes strategies for enhanced efficiency.

     Scrum's roles, including the Product Owner, who champions the customer's needs, and the Scrum Master, who safeguards the process, create a supportive ecosystem. By fostering transparent communication, encouraging self-organization, and embracing change, Scrum not only accelerates product development but makes it a powerful and unique Agile model.

  2) Kanban: -

     Kanban, a distinctive Agile methodology, revolves around visualizing and optimizing workflow. Unlike fixed sprint durations, Kanban emphasizes a continuous flow of work items, responding to real-time demands.

     Central to Kanban is the "kanban board," a visual representation of tasks progressing through different stages. This provides teams with clear visibility into their work, making bottlenecks and inefficiencies evident.

     Key features of Kanban include:

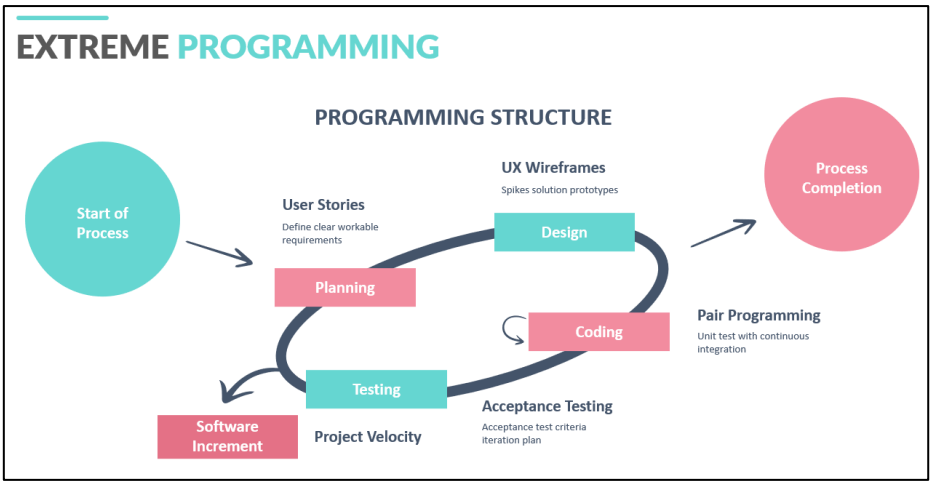
     1. Work-in-Progress Limits: Setting limits on tasks in each stage prevents overload and encourages focused effort, promoting smoother workflow.
     2. Continuous Improvement: Regular retrospectives empower teams to refine processes, enhance collaboration, and streamline performance.
     3. Metrics-Driven: Kanban employs quantitative data to measure lead time, cycle time, and other metrics, enabling informed decisions and strategic enhancements.

     Kanban's adaptability is particularly suited for support and maintenance teams, where incoming tasks vary. It thrives in environments where change is constant, enabling teams to remain responsive and effectively manage evolving priorities.

3) Extreme Programming (XP): -
Extreme Programming (XP), a cutting-edge Agile approach, is like a software craftsmanship workshop. It emphasizes creating top-notch code through Test-Driven Development (TDD), where tests are written before coding to ensure reliability. Pair Programming, like a collaborative duet, enhances code quality by sharing knowledge effectively.
The quick cycles allow rapid adjustments to changing requirements, akin to an artist refining their work. Continuous feedback loops, maintain alignment with customer needs. Frequent releases transform software development into a dynamic process, adapting to evolving expectations.
By embracing change, teamwork, and technical skill, Extreme Programming composes a remarkable performance in the software development world, where code becomes a work of art.

- Agile Model used in Existing model/system: -

Extreme Programming is well-suited for projects that require rapid iterations, close collaboration with customers, and a focus on delivering high-quality software. It provides a structured yet flexible approach that promotes responsiveness to change and continuous improvement throughout the development process.



Extreme Programming (XP) Agile model used to develop a hotel management system offered a tailored approach that addressed the unique demands of the hospitality industry: -

1. Test-Driven Development (TDD): XP's TDD ensures a dependable and thoroughly tested hotel management system. By writing tests before code, the system can maintain reliability and accuracy, critical for managing guest reservations, check-ins, and financial transactions.

2. Customer-Centric Iterations: XP's iterative approach aligns with the evolving needs of hotel staff and guests. Regular interactions with stakeholders facilitate rapid adjustments to accommodate changing requirements, whether it is integrating new services or optimizing the guest experience.
3. Pair Programming for Quality: In implementing core features, XP's pair programming method fosters collaboration between developers. This practice can result in higher-quality code for essential functions like room assignment algorithms, ensuring efficient guest accommodation.
4. Frequent Releases: XP's emphasis on small, frequent releases could mean the hotel management system can be deployed in stages. This allows the hotel to benefit from early functionality, such as reservations, and gradually incorporate additional modules like billing or housekeeping.
5. Refactoring for Adaptability: The dynamic nature of the hospitality industry requires system adaptability. XP's refactoring principle helps maintain code flexibility, making it easier to integrate new services, manage peak seasons, and adjust to changing market trends.
6. Continuous Improvement: XP's regular retrospectives encourage the hotel management team to continuously refine processes. This practice ensures that the system's performance and user experience consistently improve, enhancing guest satisfaction and operational efficiency.

Incorporating Extreme Programming into the development of a hotel management system fosters a collaborative, customer-focused, and adaptable approach, aligning technology with the ever-changing needs of the hospitality sector.