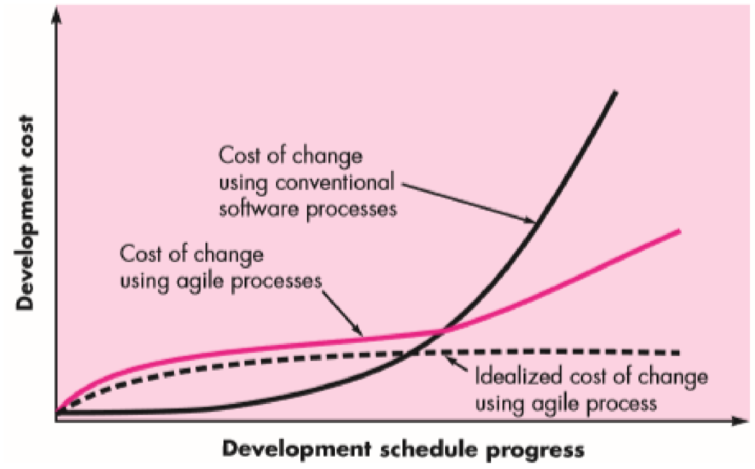
Chapter-3

INTRODUCTION TO AGILITY:

What is “Agility-

- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed
- Rapid, incremental delivery of software

An Agility and Cost of Change:



The conventional wisdom in software development is that the cost of change increases nonlinearly as a project progresses (Figure 3.1, solid black curve). A usage scenario might have to be modified, a list of functions may be extended. The change requires a modification to the architectural design of the software, the design and construction of three new components, modifications to another five components, the design of new tests, and so on. Costs escalate quickly, and the time and cost required to ensure that the change is made without unintended side effects is nontrivial.

Agility argue that a well-designed agile process “flattens” the cost of change curve (Figure 3.1, shaded, solid curve), allowing a software team to accommodate changes late in a software project without dramatic cost and time impact.

AGILITY PRINCIPLES:

1. Our highest priority is to satisfy the customer through early and continuous delivery of software.

2. Welcome changing requirements, even late in development.

3. Deliver working software frequently,with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need.

6. The most efficient and effective method of conveying information is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity—the art of maximizing the amount of work not done—is essential.

11. The best architectures, requirements, and designs emerge from self– organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Characterestics of software engineer or Human Factor in Agile Development process

If members of the software team are to drive the characteristics of the process that is applied to build software, a number of key traits must exist among the people on an agile team and the team itself:

Competence-In an agile development (as well as software engineering) context, "competence" encompasses innate talent, specific software-related skills, and overall knowledge of the process that the team has chosen to apply. Skill and knowledge of process can and should be taught to all people who serve as agile team members.

Common focus-Although members of the agile team may perform different tasks and bring different skills to the project, all should be focused on one goal—to deliver a working software increment to the customer within the time promised. To achieve this goal, the team will also focus on continual adaptations (small and large) that will make the process fit the needs of the team.

Collaboration-Software engineering (regardless of process) is about assessing, analyzing, and using information that is communicated to the software team; creating information that will help all stakeholders understand the work of the team; and building information (computer software and relevant databases) that provides business value for the customer. To accomplish these tasks, team members must collaborate—with one another and all other stakeholders.

Decision-making ability- Any good software team (including agile teams) must be allowed the freedom to control its own destiny. This implies that the team is given autonomy—decision-making authority for both technical and project issues.

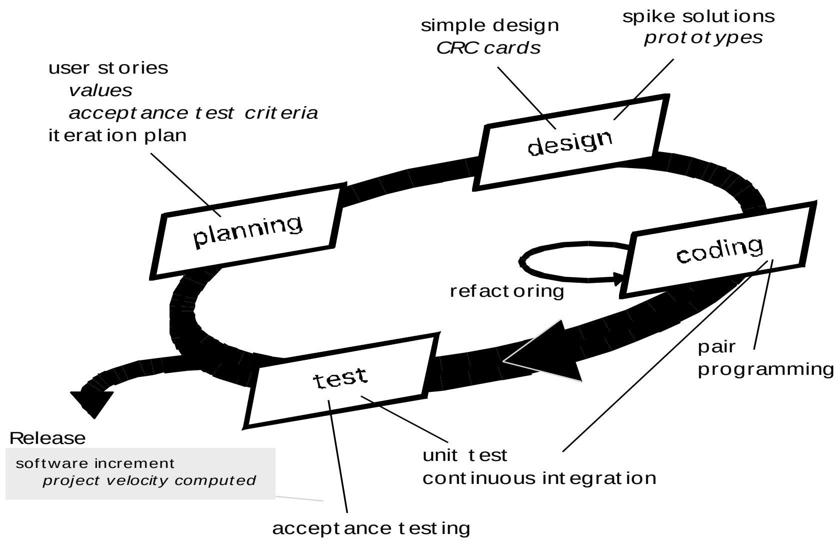
Fuzzy problem-solving ability- Software managers must recognize that the agile team will continually have to deal with ambiguity and will continually be buffeted by change. In some cases, the team must accept the fact that the problem they are solving today may not be the problem that needs to be solved tomorrow. However, lessons learned from any problem-solving activity (including those that solve the wrong problem) may be of benefit to the team later in the project.

Mutual trust and respect-The agile team must become what DeMarco and Lister call a "jelled" team . A jelled team exhibits the trust and respect that are necessary to make them "so strongly knit that the whole is greater than the sum of the parts."

Self-organization- In the context of agile development, self-organization implies three things: (1) the agile team organizes itself for the work to be done, (2) the team organizes the process to best accommodate its local environment, (3) the team organizes the work schedule to best achieve delivery of the software increment. Self-organization has a number of technical benefits, but more importantly, it serves to improve collaboration and boost team morale. In essence, the team serves as its own management. Ken Schwaber [Sch02] addresses these issues when he writes: "The team selects how much work it believes it can perform within the iteration, and the team commits to the work. Nothing demotivates a team as much as someone else making commitments for it. Nothing motivates a team as much as accepting the responsibility for fulfilling commitments that it made itself.

Extreme Programming (XP)

The Extreme Programming is commonly used agile process model. It uses the concept of object-oriented programming. A developer focuses on the framework activities like planning, design, coding and testing. XP has a set of rules and practices.



The XP Process:

XP Planning - Begins with the creation of "user stories" by listening. After that customer assigns a value and cost to the story and stories are grouped to for a deliverable increment. A commitment is made on delivery date. After the first increment define subsequent delivery dates for other increments.

XP Design - The XP design follows the 'keep it simple' principle. A simple design always prefers the more difficult representation. For difficult design problems, suggests the creation of "spike solutions. Encourages "refactoring"—an iterative refinement of the internal program design.

XP Coding - The developer is focused on what must be implemented. After the initial design work is done, the team creates a set of unit tests which can test each story or part. Also Encourages "pair programming" where two people work together for programming.

XP Testing - All unit tests are executed daily (whenever code is modified). "Acceptance tests" are defined by the customer and executed to assess customer visible functionality

TEAM STRUCTURE :

Team structure addresses the issue of organization of the individual project teams. There are some possible ways in which the individual project teams can be organized.

There are mainly three formal team structures:

1) Chief programmer

2) Democratic

3) Mixed team organizations

1) **Chief Programmer Team -** In this team organization, a senior engineer provides the technical leadership and is designated as the chief programmer. The chief programmer partitions the task into small activities and assigns them to the team members. He also verifies and integrates the products developed by different team members. The structure of the chief programmer team is shown in below fig.

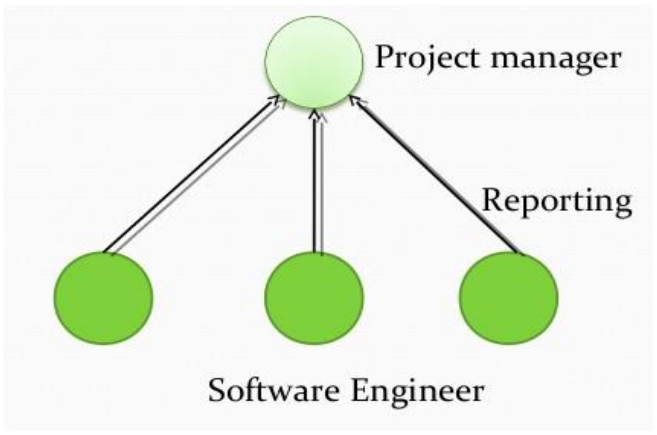
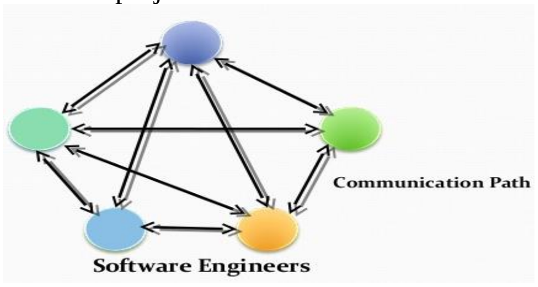


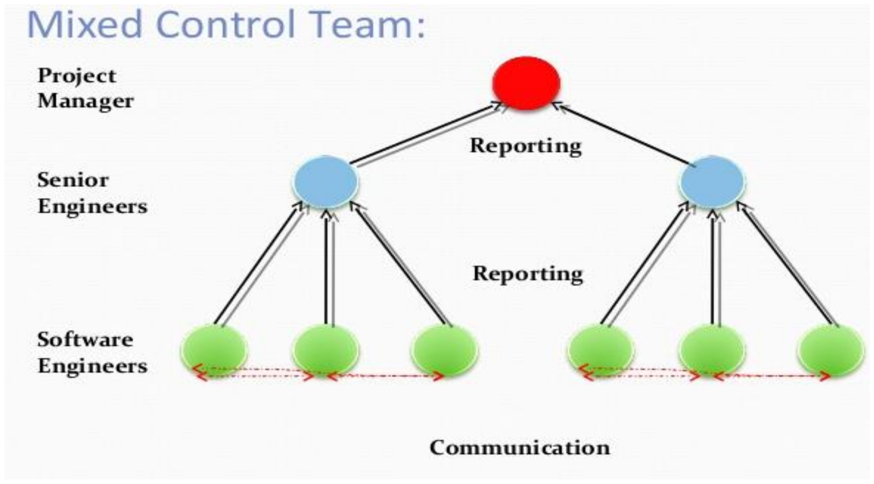
Fig. Chief Programmer Team

However, the chief programmer team leads to lower team morale, since team-members work under the constant supervision of the chief programmer. This also inhibits their original thinking.

2) **Democratic Team Structure -** Typically, a manager provides the administrative leadership. At different times, different members of the group provide technical leadership. The democratic team organization encourages egoless programming as programmers can share and review one another's work. A democratic team structure is suitable for projects requiring less than five or six engineers and for research-oriented projects.



3) **Mixed Control Team Organization –** The mixed team organization combines the ideas from both the democratic organization and the chief-programmer organization. The mixed control team organization is shown in below fig.



The democratic connections are shown as dashed lines and the reporting structure is shown using solid arrows. The mixed control team organization is suitable for large team sizes.